University of St. Gallen

School of Computer Science

to obtain the title of

Master of Science in Computer Science

Master Thesis on

Explainable Multimodal-based Depression Awareness at the Edge

---

Submitted by:

Tibor Haller

20-474-045

Approved on the application by:

Prof. Dr. Bruno Rodrigues

Date of Submission:

April 24, 2026

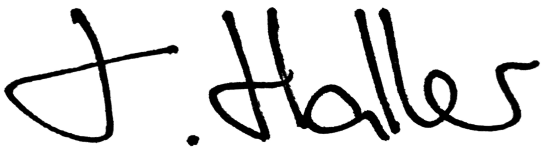
Acknowledgements

I would like to express my sincere gratitude to the people who supported me throughout the course of this thesis.

First and foremost, I wish to thank my supervisor, Prof. Dr. Bruno Rodrigues, for his continued guidance, constructive feedback, and confidence in the direction of this work. His expertise and encouragement were instrumental in shaping both the research and the final outcome.

I am equally grateful to Dr. Bernhard Bermeitinger for co-supervising my master's thesis.

On a personal note, I would like to thank my family for their unwavering support and encouragement. Finally, I owe a heartfelt thank you to my partner, whose patience, understanding, and belief in me carried me through the most demanding phases of this work.



St. Gallen, April 24, 2026 Tibor Haller

Abstract

Depression is among the most prevalent and debilitating mental health conditions worldwide, with more than 230 million people suffering from major depressive disorder globally, making it a major contributor to the overall burden of disease. While automated diagnostic approaches have already been investigated to support this depressive burden, current research focuses mainly on black-box models that provide post-hoc explainability at best. Existing research into white-box approaches often considers only single modalities or does not account for the role of context at the time of data collection. Today's state of depression monitoring thus still lacks *explainable* approaches that also consider multimodal and context-aware dimensions for household environments.

This thesis investigates the design and implementation of an explainable white-box system that supports multimodal and contextual data for depression monitoring in IoT-enabled households based on DSM-5. Consequently, it proposes the *CML Framework* as a novel approach with three main contributions: First, it demonstrates how multiple sensor modalities can be combined within an explainable, white-box monitoring framework grounded in DSM-5 criteria. Second, it introduces a context model that enriches automated assessments with environmental information, thereby improving the reliability of analysis. Third, it provides an end-to-end system architecture together with a proof-of-concept implementation that validates the feasibility of the proposed design.

The *CML Framework* and its implementation demonstrate that multimodal and context-aware depression monitoring is technically viable while preserving a white-box explainability design, and under the constraint that the system must be deployable at the edge, such as at a household environment. It advances the system toward real-world deployment and opens the path for clinical validation and richer calibration through domain experts.

Table of Contents

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
biLSTM	Bidirectional Long Short-Term Memory
CML	Contextual Multimodal Linkage
CNN	Convolutional Neural Network
CRUD	Create, Read, Update, Delete
DSM	Diagnostic and Statistical Manual of Mental Disorders
DSM-5	Diagnostic and Statistical Manual of Mental Disorders, Fifth Edition
DSR	Design Science Research
EMA	Exponential Moving Average
FASL	Fuzzy Aggregation by Symmetric Laws
FDA	Food and Drug Administration
FK	Foreign Key
GMLP	Good Machine Learning Practice
GPS	Global Positioning System
HAIM	Healthcare AI Multimodal
IoT	Internet of Things
JSON	JavaScript Object Notation
LIME	Local Interpretable Model-agnostic Explanations
LLM	Large Language Model

MDD Major Depressive Disorder

MDHAN Multimodal Depressive Hierarchical Attention Network

ML Machine Learning

NFR Non-Functional Requirement

ORM Object-Relational Mapping

PHQ-9 Patient Health Questionnaire-9

PK Primary Key

PoC Proof of Concept

RADAR-MDD Remote Assessment of Disease and Relapse for Major Depressive Disorder

REST Representational State Transfer

SHAP SHapley Additive exPlanations

SMA Simple Moving Average

UUID Universally Unique Identifier

XGBoost Extreme Gradient Boosting

YAML YAML Ain't Markup Language

List of Algorithms

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Major depressive disorder is among the most prevalent and debilitating mental health conditions worldwide. According to the Institute for Health Metrics and Evaluation, approximately 236 million people suffer from this depressive disorder globally, making it a leading cause of disability and a major contributor to the overall burden of disease [14]. The disorder is characterized by persistent sadness, loss of interest, and a range of other cognitive and physical symptoms that severely impair daily functioning [3]. Left untreated, depression can lead to chronic disability, deterioration of interpersonal relationships, substance abuse, and, in severe cases, suicide [3, 14]. Early detection is therefore critical, as timely intervention has been shown to substantially improve treatment outcomes and reduce the long-term burden on both individuals and healthcare systems [20].

Despite its prevalence, depression remains difficult to diagnose reliably. Traditional diagnostic procedures depend heavily on self-report questionnaires and clinical interviews, both of which are inherently subjective and constrained to episodic encounters in clinical settings [8]. These methods often fail to capture the continuous, fluctuating nature of depressive symptoms as they manifest in everyday life. As a result, depressive episodes may go undetected for extended periods, delaying treatment and worsening prognosis. This limitation is particularly acute in the home environment, where individuals spend the majority of their time and where depressive symptoms most directly affect daily routines, yet where clinical observation is entirely absent.

Recent advances in digital phenotyping—the quantification of human behaviour through personal device data—and ubiquitous sensing have opened promising avenues for continuous, passive monitoring of mental health [7, 38]. Smartphone-based studies

such as StudentLife have demonstrated robust associations between passively sensed behavioural patterns and mental health outcomes [41], while large-scale remote monitoring programs like RADAR-MDD have shown that multimodal traces collected over several months can effectively track the course of depressive illness [26, 32]. In household settings, Internet-of-Things devices further expand the range of observable signals, enabling passive collection of behavioral cues—such as speech patterns, sleep routines, and network activity—without requiring active user participation, such as in questionnaires.

However, the majority of actively researched depression detection systems rely on black-box machine learning models whose internal reasoning is opaque to clinicians and patients [33, 34]. In clinical settings, where decisions carry significant consequences, such opacity undermines trust and hinders adoption [11]. Assessments grounded in standard diagnostic references used by clinicians—such as the Diagnostic and Statistical Manual of Mental Disorders, Fifth Edition (DSM-5) [2]—are particularly difficult to justify when the underlying model cannot inherently explain how it arrived at its conclusions. Recent work has therefore begun to explore explainable, white-box approaches that transparently link sensor observations to established diagnostic criteria. Notably, through a Linkage Framework proposed by Länzlinger [23] and a privacy-preserving architecture by Nef [27].

Yet these contributions each address a single data modality in isolation—speech data and network traffic, respectively. A home environment naturally produces a rich variety of behavioural signals that, taken together, could yield a far more comprehensive picture of an individual's mental state than any single source alone. Further, a home environment produces contextual information that can be used to improve accuracy of results. Bridging this gap—fusing heterogeneous sensor streams and incorporating contextual information within an explainable framework that clinicians can trust and act upon—remains an open challenge and forms the motivation for the present work.

1.2 Problem Statement

As established in the preceding section, the Linkage Framework [23] and the network-traffic architecture by Nef [27] have each demonstrated the feasibility of explainable, white-box depression monitoring for a single data modality. However, neither contribution addresses how heterogeneous sensor streams can be robustly combined within such a framework, nor how contextual information, such as the physical environment at the

time of data collection, can be incorporated to improve the reliability of automated assessments.

It is not known how the Linkage Framework can be extended to fuse multiple data modalities and contextual information into a coherent, explainable monitoring system for major depressive disorder in IoT-enabled home environments. This problem carries direct consequences for two groups. Individuals with depression currently lack continuous, passive monitoring tools that operate unobtrusively in their home environment, where symptoms most acutely affect daily life [8, 34]. Clinicians, in turn, require transparent and interpretable outputs grounded in DSM-5 criteria before they can responsibly act on automated assessments [11, 33]. A requirement increasingly reinforced by regulatory guidance on transparency in medical AI systems [39]. Without a principled multimodal and context-aware extension of the existing white-box approach, sensor-based depression monitoring risks falling short of the clinical robustness needed for real-world deployment.

1.3 Research Question

Building on the identified gap, this thesis addresses the following research question:

How can the linkage framework be used to design an explainable white-box system that supports multimodal and contextual data for monitoring of major depressive disorder based on DSM-5 factors in IoT-enabled household environments?

This question targets two capabilities that the existing Linkage Framework lacks: the ability to robustly fuse heterogeneous sensor modalities from different sources into a single analysis pipeline, and the ability to incorporate contextual information about the environment in which observations are collected. Addressing both capabilities within a white-box design is essential to producing assessments that clinicians can trace from sensor data through to a DSM-5-grounded conclusion. Figure 1.1 illustrates the envisioned pipeline at a high level.

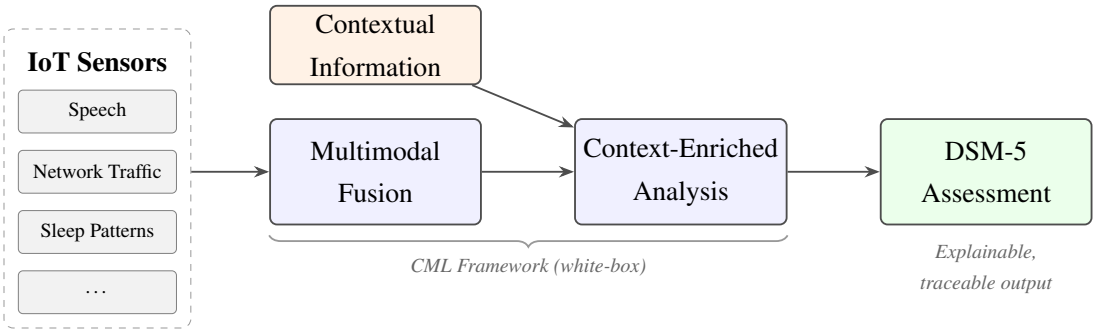


Figure 1.1: Conceptual overview of the envisioned system.

To answer the research question, this thesis pursues the five objectives summarised in Table 1.1. Objective 1 extends the Linkage Framework with a conceptual approach to support heterogeneous sensor modalities. Objective 2 further extends the framework to incorporate contextual information about the environment in which observations are collected, enabling the same sensor reading to be interpreted differently depending on the context in which data was collected. Objective 3 integrates these conceptual extensions into an end-to-end system architecture that orchestrates data ingestion, context evaluation, multimodal analysis, and explainability within a cohesive design. Objective 4 realises this architecture as a proof-of-concept prototype, demonstrating that the design can be implemented with concrete technology choices. Finally, Objective 5 evaluates the prototype to determine whether the system satisfies the design goals established by the preceding objectives.

Table 1.1: Overview of research objectives.

ID	Objective
Obj-1	Extend the Linkage Framework to support multimodal data input.
Obj-2	Extend the Linkage Framework to support contextual information.
Obj-3	Design a system architecture that integrates the CML Framework to support explainable, multimodal, and context-enriched automated health assessments.
Obj-4	Implement a proof-of-concept prototype that realizes the proposed architecture.
Obj-5	Evaluate the prototype.

1.4 Contributions

The primary contributions of this thesis are threefold. First, it demonstrates how multiple sensor modalities can be combined within an explainable, white-box monitoring framework grounded in DSM-5 criteria. Second, it introduces a context model that enriches automated assessments with environmental information, thereby improving the reliability of analysis. Third, it provides an end-to-end system architecture together with a proof-of-concept implementation that validates the feasibility of the proposed design.

1.5 Scope and Delimitations

This thesis focuses on the design and prototypical implementation of an explainable, multimodal, and context-aware monitoring system for major depressive disorder within an IoT-enabled household environment. The work extends the existing Linkage Frame-

work [23] with multimodal fusion and context integration capabilities, and validates these extensions through a proof-of-concept prototype.

Several aspects are explicitly outside the scope of this work. The prototype operates on simulated sensor data rather than on recordings from real patients; clinical validation with human subjects would require ethical approval and longitudinal study designs that exceed the boundaries of a single master thesis. Likewise, the system is not intended for production deployment: it serves as a research prototype to demonstrate the feasibility of the proposed design, not as a certified medical device. The evaluation assesses functional correctness, system behaviour under defined test scenarios, and non-functional properties such as performance, but does not include diagnostic accuracy metrics against clinical ground truth, as no real patient data is available. Finally, while the architecture is designed to accommodate additional modalities, this thesis integrates only the modalities for which prior work within the Linkage Framework has established biomarker definitions; extending the system with entirely new modalities is left to future work.

1.6 Methodology

This thesis follows the Design Science Research (DSR) methodology as formalised by Hevner et al. [15] and operationalised through the process model of Peffers et al. [29]. DSR is an established paradigm for information systems research whose primary contribution is a purposeful artifact. DSR provides the appropriate foundation as the central objective of this work is to extend an existing conceptual framework and to realise this extension as a functioning software prototype. Continuity with the predecessor thesis by Länzlinger [23], which also adopts DSR, further supports this choice.

Peffers et al. [29] define six DSR activities: (1) problem identification and motivation, (2) definition of objectives, (3) design and development, (4) demonstration, (5) evaluation, and (6) communication. This thesis organises these activities into four phases, each aligned with one or more research objectives. The first phase conducts a foundational literature review to establish the knowledge base required for Objectives 1 through 3. It also includes Peffers activities 1 and 2, which have already been discussed as part of chapter 1. The second phase produces the conceptual artifacts: an extended Linkage Framework with multimodal support (Obj-1) and context integration (Obj-2), together with an end-to-end system architecture that unifies both extensions (Obj-3). This phase corresponds to Peffers activity 3. The third phase translates the design into a proof-of-concept prototype (Obj-4), serving as both development completion and demonstration

(Peffers activities 3 and 4). The fourth and final phase evaluates the prototype (Obj-5) through functional requirement verification, scenario-based system behaviour assessment, and non-functional analysis (Peffers activity 5). Communication (activity 6) is realised through this thesis document itself.

1.7 Thesis Outline

The thesis is organised as follows. Chapter 2 introduces the theoretical and technical foundations, including major depressive disorder and DSM-5 diagnostic criteria, multimodality, explainability concepts, and the existing Linkage Framework, as well as a review of related work in multimodal depression detection, explainable health monitoring, and context-aware systems. Chapter 3 presents the core conceptual contribution: the extension of the Linkage Framework with multimodal and context-aware capabilities, and the system architecture that integrates these extensions into a cohesive design. Chapter 4 details the proof-of-concept prototype that realises the proposed architecture, covering the technology stack, data persistence, analysis pipeline, context evaluation engine, and dashboard. Chapter 5 assesses the prototype against the defined objectives through functional and non-functional evaluation, discusses system behaviour under representative test scenarios, and concludes with an analysis of limitations and avenues for future research. Chapter 6 summarises the contributions and outlines avenues for future research.

Chapter 2

Fundamentals

2.1 Background

This section establishes the conceptual vocabulary required to follow the remainder of this thesis, covering major depressive disorder and its DSM-5 diagnostic criteria, explainability in automated decision systems, multimodal sensing and fusion, and context-aware computing.

2.1.1 Depression

Depressive disorders constitute a broad class of mental health conditions characterised by disturbances in mood, cognition, and psychomotor function that impair an individual's ability to carry out everyday activities [2]. The class encompasses several distinct diagnoses, including persistent depressive disorder, premenstrual dysphoric disorder, and substance-induced depressive disorder, but major depressive disorder (MDD) is the most prevalent form and the primary focus of clinical research on depression [3]. This thesis concentrates exclusively on MDD.

Diagnostic criteria. The Diagnostic and Statistical Manual of Mental Disorders, Fifth Edition (DSM-5), published by the American Psychiatric Association [2], is the standard diagnostic reference used in clinical practice and research. It defines MDD through nine symptom criteria, listed in Table 2.1. A diagnosis of MDD requires that at least five of these nine symptoms are present during the same two-week period, representing a change from previous functioning. At least one of the five must be either depressed mood (C1) or diminished interest (C2), which are considered the core symptoms of the disorder [2].

Table 2.1: DSM-5 symptom criteria for major depressive disorder.

ID	Symptom domain	Description
C1	Depressed mood	Depressed mood most of the day, nearly every day, as indicated by subjective report or observation by others
C2	Diminished interest	Markedly diminished interest or pleasure in all, or almost all, activities most of the day, nearly every day
C3	Weight/appetite change	Significant weight loss or gain, or decrease or increase in appetite nearly every day
C4	Sleep disturbance	Insomnia or hypersomnia nearly every day
C5	Psychomotor change	Psychomotor agitation or retardation nearly every day, observable by others
C6	Fatigue	Fatigue or loss of energy nearly every day
C7	Worthlessness/guilt	Feelings of worthlessness or excessive or inappropriate guilt nearly every day
C8	Cognitive impairment	Diminished ability to think or concentrate, or indecisiveness, nearly every day
C9	Suicidal ideation	Recurrent thoughts of death, recurrent suicidal ideation, or a suicide attempt

Relevance to this thesis. The nine DSM-5 criteria provide the structural foundation on which the system presented in this thesis is built. Throughout the remainder of this work, the term *indicator* refers to exactly one DSM-5 criterion, and the diagnostic threshold described above is operationalised as a configurable rule in the system's assessment layer (Chapter 3).

2.1.2 Explainability in Automated Decision Systems

In the context of automated decision systems, *explainability* refers to the degree to which a human observer can understand the cause of a system's output [33]. This notion applies equally to AI models (e.g., neural networks) and to traditional systems (e.g., rule-based decisions): in both cases, a stakeholder may need to trace how inputs were transformed into a particular decision or score. The related terms *interpretability* and *transparency* are often used interchangeably in the literature; this thesis follows the convention of Rudin [33] and treats them as synonymous, using *explainability* throughout.

Black-box and white-box systems. Automated decision systems can be positioned on a spectrum according to the visibility of their internal reasoning. At one end, *black-box* systems, such as deep neural networks, learn complex input–output mappings whose internal representations are not directly interpretable by humans. The system may achieve high predictive accuracy, yet a domain expert cannot inspect the individual steps that led to a particular output. At the opposite end, *white-box* systems, expose every computational step in a form that a human can follow, verify, and audit [33].

Inherent interpretability vs. post-hoc explanation. A central distinction in the explainability literature separates two strategies for making system outputs understandable [33]. *Inherent interpretability* constrains the system to transparent operations from the outset, so that the reasoning path is readable by construction. *Post-hoc explanation*, by contrast, first builds an opaque system and then applies a separate technique to approximate its reasoning after the fact.

Post-hoc methods are arguably less suitable for high-stakes decision domains as they explain a *surrogate* of the model, not the model itself, and can therefore produce explanations that diverge from the model's actual behaviour [33]. This concern is reinforced in healthcare applications, where misleading explanations may erode clinical trust rather than build it [11].

Relevance to this thesis. This thesis adopts the inherent-interpretability approach. Every operation in the processing pipeline is closed-form, parameterised by expert-set values, and decomposable into individually inspectable contributions. The specific design choices that realise this property are presented in Chapter 3.

2.1.3 Multimodal Sensing

Multimodal sensing refers to the use of multiple, heterogeneous sensor types to observe a phenomenon from complementary perspectives. In health monitoring, common modality categories include acoustic sensors (e.g., speech analysis), physiological sensors (e.g., heart rate, skin conductance), behavioural sensors (e.g., actigraphy, smartphone usage patterns), and environmental sensors (e.g., network traffic, ambient light). Each modality captures a different facet of an individual's behaviour; the observable characteristics derived from these signals are referred to as *biomarkers*: qualitative behavioural features that can serve as evidence for a clinical construct. Throughout this thesis, the term *biomarker* is used in this sense.

Multimodal fusion. When multiple modalities are available, their information must be combined into a unified output through a process known as *multimodal fusion*. The literature distinguishes three canonical strategies [34]:

- *Early fusion* (feature-level): raw features from all modalities are concatenated into a single representation before processing.
- *Late fusion* (decision-level): each modality is processed independently and the resulting per-modality outputs are combined at the decision stage.
- *Hybrid fusion*: information is combined at one or more intermediate processing stages, blending elements of both early and late fusion.

Relevance to this thesis. The system presented in this thesis integrates multiple sensor modalities to derive biomarkers for the DSM-5 indicators introduced in the preceding subsection. The specific fusion strategy adopted and its rationale are presented in Chapter 3.

2.1.4 Context-Aware Computing

Dey [9] defines *context* as any information that can be used to characterise the situation of an entity, where an entity is a person, place, or object relevant to the interaction between a user and a system. Common context categories identified in the literature include identity, location, activity, time, and environmental conditions [4, 30]. In a household monitoring setting, for instance, relevant context may include who is present, what activity is occurring, or whether the individual is alone.

Context-aware systems. A system is *context-aware* if it uses context to provide relevant information or services to the user, where relevancy depends on the user's task [9]. The literature identifies two broad roles that context plays in such systems [30]:

- *Filtering* — selecting what information is relevant. In multimodal systems, this role manifests as *multimodal arbitration*: dynamically selecting or prioritising which modality to rely on based on the current situation. For example, during a social gathering, speech-derived biomarkers are highly informative; when a person is home alone, speech data is largely absent and network activity or movement patterns become the primary source of evidence.
- *Adaptation* — adjusting how information is processed or interpreted. In multimodal systems, this role manifests as *conditional feature weighting*: the significance of a

biomarker observation changes depending on the situation in which it was recorded. For example, reduced speech activity may be noteworthy when other people are present in the household, but unremarkable when the person is home alone.

Relevance to this thesis. This thesis employs context in both roles: arbitration determines which modalities and biomarkers are relevant in a given situation, while conditional weighting adjusts how strongly each biomarker contributes to an indicator. The concrete context model and its terminology are introduced in Chapter 3.

2.2 Mathematical Foundations

This section serves as a self-contained reference for the algorithmic components of the system by introducing the key mathematical concepts used in Chapter 3.

2.2.1 Fuzzy Set Theory and Membership Functions

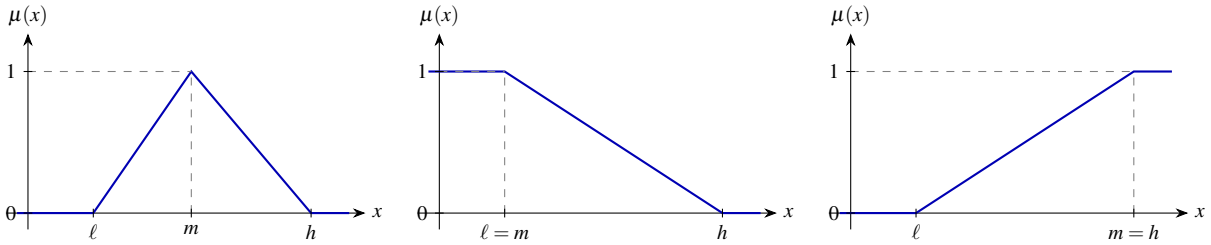
Classical set theory assigns elements a binary membership status: an element either belongs to a set or it does not. Fuzzy set theory, introduced by Zadeh [43], generalises this notion by allowing graded membership. Formally, a *fuzzy set* A over a domain X is defined by a *membership function* $\mu_A : X \to [0, 1]$, where 0 indicates no affiliation, 1 denotes full affiliation, and intermediate values express the *grade of membership*.

This continuous-valued representation is particularly relevant when modelling quantities derived from sensor data or real-world measurements, where sharp boundaries between categories (e.g., “low” and “medium”) rarely exist in practice. Instead of forcing a binary classification at an arbitrary threshold, fuzzy membership captures the gradual transition between categories.

Linguistic variables and sets. A *linguistic variable* is a variable whose values are not numbers but words or sentences in a natural language [44]. Each such value, called a *linguistic value*, is associated with a fuzzy set that defines its meaning over a numerical base variable. The collection of all linguistic values of a variable forms its *term set*. For example, a linguistic variable “temperature” might have the term set $\{low, medium, high\}$, where each value is characterised by its own membership function over the base variable (e.g., degrees Celsius). Given a numeric reading, every membership function of that variable is evaluated simultaneously, producing one grade of membership per linguistic value. The resulting vector of grades captures how strongly the observed reading fits each qualitative category, rather than committing to a single label.

Triangular membership function. Among the various families of membership functions, triangular and trapezoidal shapes are the most common in practice [19]. The triangular membership function is defined by three parameters (ℓ, m, h), where ℓ is the left foot, m is the peak, and h is the right foot. The function returns 0 outside the support interval $[\ell, h]$, rises linearly from 0 at ℓ to 1 at m, and falls linearly from 1 at m to 0 at h:

$$\mu_{\text{tri}}(x;\ell,m,h) = \begin{cases} 0 & \text{if } x \le \ell, \\ \dfrac{x-\ell}{m-\ell} & \text{if } \ell < x \le m, \\ \dfrac{h-x}{h-m} & \text{if } m < x < h, \\ 0 & \text{if } x \ge h. \end{cases} \tag{2.1}$$



(a) Standard ($\ell < m < h$) **(b)** Left-shoulder ($\ell = m$) **(c)** Right-shoulder ($m = h$)

Figure 2.1: Three types of triangular membership functions visualized.

Two degenerate cases arise in practice, illustrated alongside the standard shape in Figure 2.1. When $\ell = m$ (left-shoulder, Figure 2.1b), the rising edge collapses and the function equals 1 at the left boundary, producing a shape suitable for terms such as *low* where the smallest value in the domain should receive full membership. Conversely, when $m = h$ (right-shoulder, Figure 2.1c), the falling edge collapses, producing a shape suitable for open-ended terms such as *high*. In both cases, the division by zero that would arise from a zero-width edge is resolved by convention: membership is defined as 1 at the degenerate boundary.

Trapezoidal membership function. The trapezoidal function generalises the triangle by introducing a flat plateau between two interior points, as illustrated in Figure 2.2. It is defined by four parameters (ℓ, ℓ_p, h_p, h), where ℓ and h are the outer feet and ℓ_p and h_p delimit the plateau on which membership equals 1:

$$\mu_{\text{trap}}(x;\,\ell,\ell_p,h_p,h) = \begin{cases} 0 & \text{if } x \le \ell, \\ \dfrac{x-\ell}{\ell_p-\ell} & \text{if } \ell < x \le \ell_p, \\ 1 & \text{if } \ell_p < x \le h_p, \\ \dfrac{h-x}{h-h_p} & \text{if } h_p < x < h, \\ 0 & \text{if } x \ge h. \end{cases} \tag{2.2}$$

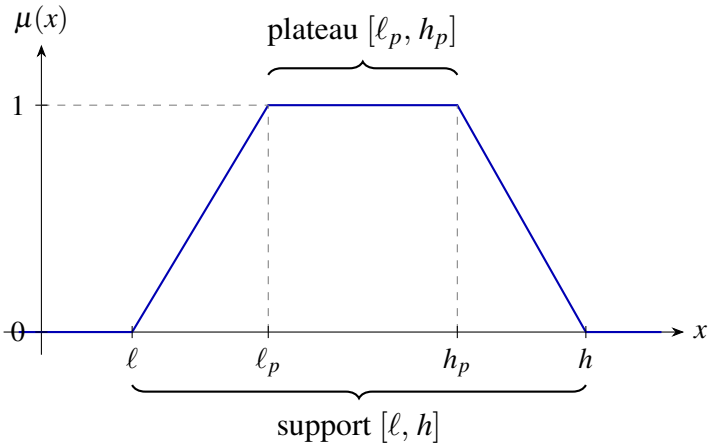
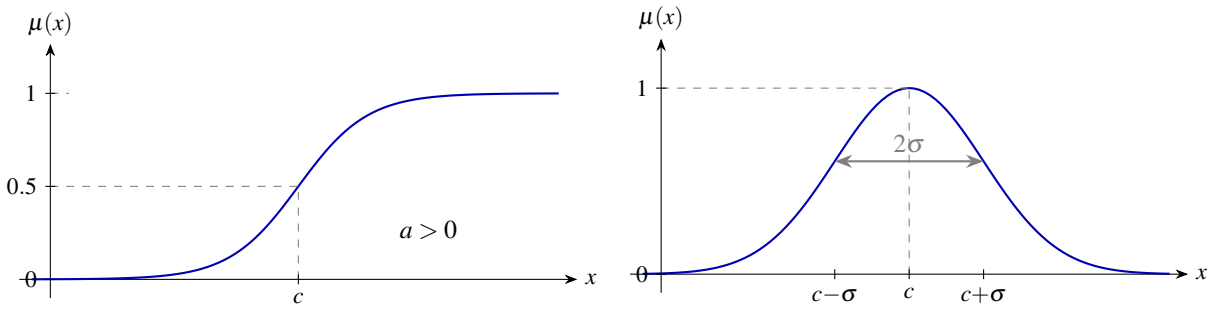


Figure 2.2: Trapezoidal membership function visualized.

Trapezoidal functions are particularly useful for linguistic terms where a range of values should receive full membership rather than a single peak.

Sigmoid and Gaussian membership functions. Beyond piecewise-linear shapes, two smooth families appear frequently in fuzzy modelling [17], both shown in Figure 2.3.



(a) Sigmoid $\mu_{\text{sig}}(x;\,a,c)$ **(b)** Gaussian $\mu_{\text{gauss}}(x;\,c,\sigma)$

Figure 2.3: Sigmoid and Gaussian membership functions visualized.

A *sigmoid* (logistic) membership function (Figure 2.3a) provides a smooth S-shaped transition controlled by a midpoint c and a steepness parameter a. The sign of a determines the direction of the transition: positive values produce a rising curve (left-open), negative values a falling curve (right-open).

$$\mu_{\text{sig}}(x; a, c) = \frac{1}{1 + e^{-a(x-c)}}. \tag{2.3}$$

A *Gaussian* membership function (Figure 2.3b) centres full membership at a target value c and decays symmetrically according to a width parameter σ:

$$\mu_{\text{gauss}}(x; c, \sigma) = e^{-\frac{(x-c)^2}{2\sigma^2}}. \tag{2.4}$$

Unlike the triangular and trapezoidal families, neither function has finite support: membership approaches but never exactly reaches 0. Both families are revisited in Section 2.2.1, where their practical selection criteria are discussed alongside additional alternatives.

Practical selection of membership functions. Nef [27] applied membership functions to network-based behavioural interpretation for depression detection and employed three of the families defined above: triangular as the default, sigmoid for smooth single-sided transitions, and Gaussian for situations where evidence is strongest near a target value and decays symmetrically.

Nef selected the triangular function as the default for three reasons: its parameters are directly interpretable, its finite support $[\ell, h]$ confines influence to a defined range, and its shape is simple to tune. The sigmoid is preferred when hard cut-offs at the support boundaries are undesirable, for example, when modelling a gradual increase in risk without an abrupt threshold. The Gaussian is suited to cases where a known target operating point exists and deviations in either direction should reduce membership equally.

The trapezoidal function, while not employed by Nef, fills a gap in the above selection. Where the triangular function assigns full membership to exactly one value, the trapezoidal plateau grants full membership to a continuous range $[\ell_p, h_p]$. This is advantageous when small variations in the input should not alter the membership grade — a property neither the triangular nor the smooth families provide.

2.2.2 Fuzzy Logic Operators

The preceding subsection defined membership functions that map a single input value to a grade of membership in $[0, 1]$. In practice, however, systems must combine several such grades into a single composite value — for instance, to determine whether a set of conditions is jointly satisfied or whether at least one condition holds. Fuzzy logic provides standard operators for this purpose, analogous to Boolean AND and OR in classical logic [19, 43].

Fuzzy conjunction — t-norms. A *triangular norm* (t-norm) generalises the Boolean AND to continuous-valued logic. Formally, a t-norm is a function $T\colon [0,1]^2 \to [0,1]$ that is commutative, associative, monotone, and satisfies the boundary condition $T(a,1) = a$ [19]. The most widely used t-norm is the *minimum* operator proposed by Zadeh [43]:

$$T_{\min}(\mu_1, \ldots, \mu_n) = \min_i \mu_i. \tag{2.5}$$

Its defining characteristic is that the weakest input determines the output: all conditions must be satisfied, and the least satisfied condition governs the composite score.

Fuzzy disjunction — t-conorms. The dual of the t-norm is the *triangular conorm* (t-conorm), which generalises Boolean OR. A t-conorm $S\colon [0,1]^2 \to [0,1]$ satisfies the same axioms as a t-norm except that the boundary condition becomes $S(a,0) = a$ [19]. The standard t-conorm is the *maximum* operator [43]:

$$S_{\max}(\mu_1, \ldots, \mu_n) = \max_i \mu_i. \tag{2.6}$$

Here the strongest input determines the output: at least one condition must be satisfied, and the most satisfied condition governs the composite score.

Weighted mean as a compensatory operator. The minimum and maximum operators are *extremes*: in each case, a single input dominates the output entirely. Zimmermann and Zysno [46] observed that human decision makers often reason in a compensatory manner, where strong satisfaction of one condition partially offsets weak satisfaction of another. The *weighted arithmetic mean* captures this behaviour and is the most common

compensatory aggregation operator in fuzzy systems [19]:

$$M_w(\mu_1, \ldots, \mu_n) = \frac{\sum_{i=1}^{n} w_i \mu_i}{\sum_{i=1}^{n} w_i}, \tag{2.7}$$

where $w_i > 0$ are importance weights. When the weights are normalised to sum to 1, the denominator equals 1 and the operator simplifies to $\sum_i w_i \mu_i$. The weighted mean exhibits four properties that are relevant for the aggregation tasks in this thesis:

- *Bounded output:* if all $\mu_i \in [0, 1]$ and $\sum w_i = 1$, then $M_w \in [0, 1]$.
- *Monotonicity:* increasing any μ_i (while holding the others fixed) increases M_w.
- *Interpolation:* $\min_i \mu_i \leq M_w \leq \max_i \mu_i$ — the result always lies between the t-norm and t-conorm values.
- *Decomposability:* each term $w_i \mu_i$ is individually inspectable, making the contribution of every input transparent.

Despite these favourable properties, the weighted mean is *not* a t-norm. A t-norm requires that whenever any single input equals 0, the entire output collapses to 0, regardless of how high the remaining inputs are. The weighted mean does not satisfy this requirement. Consider two equally weighted inputs where $\mu_1 = 0.8$ and $\mu_2 = 0$: the minimum operator returns 0 (one condition is entirely unsatisfied, so the conjunction fails), whereas the weighted mean returns $0.5 \times 0.8 + 0.5 \times 0 = 0.4$ (the strong first input partially compensates for the unsatisfied second input). The weighted mean therefore does not model strict conjunction but occupies a middle position on the strictness spectrum between conjunction and disjunction [46].

Trade-off summary. Table 2.2 summarises the three operators introduced above. The minimum is maximally strict: a single weak input collapses the composite score, making it suitable when all conditions are genuinely required. The maximum is maximally permissive: a single strong input suffices, making it suitable when any one condition is sufficient. The weighted mean is compensatory: all inputs contribute proportionally, and none dominates, making it suitable when partial satisfaction should be reflected and domain knowledge can express relative importance through weights.

Table 2.2: Comparison of fuzzy aggregation operators.

Operator	Formula	Strictness	Suitable when ...
Minimum (AND)	$\min_i \mu_i$	Strict	all conditions must hold; the weakest governs
Maximum (OR)	$\max_i \mu_i$	Permissive	any single condition suffices; the strongest governs
Weighted mean	$\sum_i w_i \mu_i$	Compensatory	partial satisfaction matters; importance is weighted

This trade-off is revisited in Section 2.2.3, where the weighted mean is adopted as the aggregation kernel of the FASL operator, and applied in Section 3.5.1.5, where context evaluation selects between conjunctive and disjunctive semantics depending on the nature of the context assumption.

2.2.3 Fuzzy Additive Symptom Likelihood as an Aggregation Primitive

Nef [27] introduced *Fuzzy Additive Symptom Likelihood* (FASL) as a mechanism for aggregating individual biomarker assessments into composite indicator scores. The core idea is to assign each biomarker a fuzzy membership function that maps raw observations to a degree of membership $\mu \in [0,1]$, and then combine these degrees through a transparent, weighted additive model. Formally, the likelihood score L_k for indicator k is defined as

$$L_k = \frac{\sum_i w_{k,i}\, \mu_{k,i}(x_i)}{\sum_i w_{k,i}}, \tag{2.8}$$

where $\mu_{k,i}(x_i) \in [0,1]$ denotes the membership degree of biomarker i evaluated against the fuzzy set associated with indicator k, and $w_{k,i}$ represents the indicator-local weight assigned to that biomarker. The weights are typically normalized to $\sum_i w_{k,i} = 1$, meaning that the expression simplifies to a normalised weighted sum $L_k = \sum_i w_{k,i}\, \mu_{k,i}(x_i)$, which is bounded in $[0,1]$.

Because FASL adopts the weighted mean as its aggregation kernel, it inherits the properties established in Section 2.2.2 — bounded output, monotonicity, interpolation, and decomposability. Beyond these mathematical guarantees, three operational qualities make the operator particularly suitable as a reusable aggregation primitive:

- *Robustness:* every input contributes to the composite score in proportion to its weight, meaning that the weighted sum distributes influence across all available inputs. This ensures that no single piece of evidence is discarded or allowed

to dominate. The resulting score therefore reflects the collective strength of the evidence rather than being governed by a single extreme value.

- *Tunability:* the weights $w_{k,i}$ can be adjusted by experts without altering the aggregation mechanism itself.
- *Graceful degradation:* if certain inputs are unavailable, the remaining weights can be renormalised so that the operator continues to produce a meaningful score from the available evidence.

Nef applied this operator specifically to the domain of depression monitoring, where network-derived biomarkers are aggregated per DSM-5 criterion to obtain indicator likelihoods [27]. However, the core mechanism, a normalised weighted sum of $[0,1]$-valued membership degrees, is not specific to biomarker aggregation. Any many-to-one mapping from fuzzy memberships to a composite score can employ the same operator, provided the inputs are bounded in $[0,1]$ and relative importance can be expressed through weights. This generality is exploited in Chapter 3, where the weighted-mean primitive is applied at two distinct aggregation levels: once to combine context marker memberships into a context assumption score (Section 3.5.1.5), and once to aggregate directed biomarker memberships into an indicator score per time window (Section 3.5.2).

2.2.4 Exponential Moving Average as a Temporal Smoothing Technique

Signals derived from sensor data are inherently noisy: individual readings may fluctuate due to measurement artefacts, environmental interference, or transient events rather than genuine changes in the underlying quantity of interest. When such a signal is used as input to a decision process, raw fluctuations can propagate through downstream computations and cause unstable outputs. A smoothing technique is therefore needed that dampens transient noise while remaining responsive to real shifts in the underlying signal.

Definition. Simple exponential smoothing computes a recursively weighted running average of a time series in which recent observations receive exponentially greater weight than older ones [10]. The smoothed value S_t at time t is defined by the recurrence relation

$$S_t = \alpha\, x_t + (1-\alpha)\, S_{t-1}, \tag{2.9}$$

where x_t is the raw observation at time t and $\alpha \in (0, 1]$ is the *smoothing factor*. The recursion is initialised by setting $S_0 = x_0$, using the first observation as the initial smoothed value.

Role of the smoothing factor. The parameter α governs the trade-off between responsiveness and stability. When α is close to 1, the smoothed value tracks the raw signal closely, providing fast response to changes but offering little noise reduction; at the limit $\alpha = 1$, the filter degenerates to the raw signal itself. Conversely, when α is close to 0, the smoothed value is dominated by its own history, producing strong noise suppression at the cost of slower response to genuine shifts.

Because each update blends the new observation with the previous smoothed value, older observations are never fully discarded but their influence diminishes with each successive step. This can be seen by expanding the recurrence. Substituting S_{t-1} with its own definition yields

$$\begin{aligned} S_t &= \alpha x_t + (1-\alpha) S_{t-1} \\ &= \alpha x_t + (1-\alpha)\left[\alpha x_{t-1} + (1-\alpha) S_{t-2}\right] \\ &= \alpha x_t + \alpha(1-\alpha) x_{t-1} + (1-\alpha)^2 S_{t-2} \\ &= \boldsymbol{\alpha} x_t + \boldsymbol{\alpha(1-\alpha)} x_{t-1} + \boldsymbol{\alpha(1-\alpha)^2} x_{t-2} + (1-\alpha)^3 S_{t-3}. \end{aligned}$$

Continuing the substitution reveals that each past observation x_{t-k} contributes with weight $\alpha(1-\alpha)^k$. Since $(1-\alpha) < 1$, this weight shrinks exponentially with increasing k — which is why the technique is called *exponential* smoothing.

Properties relevant to this thesis. Four properties make simple exponential smoothing suitable for temporal stabilisation of $[0, 1]$-valued scores:

- *Recency weighting:* unlike a simple moving average, which assigns equal weight to all observations within a fixed window, the exponential decay continuously down-weights older observations, producing a smooth transition that naturally prioritises the most recent evidence.
- *Single-parameter tuning:* the entire smoothing behaviour is governed by the single parameter α, making the filter straightforward to configure by a domain expert without requiring additional structural decisions (such as choosing a window size).

- *Bounded output:* if all inputs satisfy $x_t \in [0, 1]$, then $S_t \in [0, 1]$. This follows directly from the convex combination in Equation (2.9) and ensures that the smoothed value inherits the input bounds without additional clamping — a property that is important when smoothing normalised scores.
- *Computational simplicity:* each update requires one multiplication and one addition, with $O(1)$ memory, since only the previous smoothed value needs to be stored.

Comparison with simple moving average. The simple moving average (SMA) computes the unweighted arithmetic mean over a fixed window of the last n observations. Two limitations motivate the preference for exponential smoothing in this context. First, the SMA assigns equal weight to every observation in the window, meaning that an outlier at the boundary of the window exerts the same influence as the most recent reading. Second, the SMA exhibits a discontinuity whenever the oldest observation exits the window: the smoothed value can jump abruptly even when the new observation is unremarkable, simply because a distant value was dropped. Exponential smoothing avoids both issues: its exponential decay naturally prioritises recent data, and because older observations are down-weighted rather than discarded at a sharp boundary, the output is continuous and free of boundary effects [10]. Figure 2.4 visualises the weight that each method assigns to past observations.

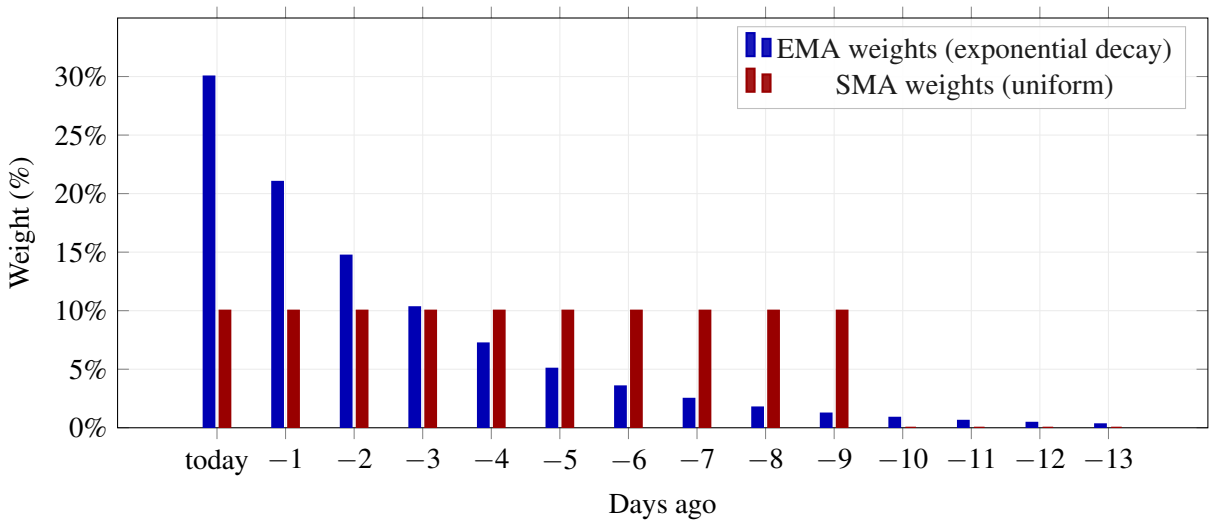


Figure 2.4: Difference in assigned weights across 14 days between EMA and SMA.

The EMA weights decay exponentially — the most recent reading dominates, and older readings contribute progressively less. The SMA, by contrast, assigns identical

weight to every observation within its 10-day window and zero weight outside. Observations older than 10 days are not considered. The EMA technique is applied in Section 3.5.1.6, where it serves as the temporal stabilisation mechanism in the context evaluation pipeline. The choice of α and its effect on context detection responsiveness are discussed there.

2.3 Related Work

This section surveys four strands of prior work that frame the thesis: multimodal depression detection, explainability, context-aware health monitoring, and white-box predecessors. A consolidated table (Table 2.3) then summarises the findings and highlights the gap this thesis fills.

2.3.1 Multimodal Approaches to Depression Detection

Multimodal fusion has emerged as the dominant methodological direction for automated depression detection, driven by consistent evidence that combining complementary signals improves on unimodal baselines. Soenksen et al. [36] evaluated 14,324 models under the HAIM framework across twelve clinical tasks and reported multimodal configurations outperforming the best single-source counterparts by 6–33%. Within depression specifically, Lin et al.'s SenseMood [24] couples a CNN over user-posted images with a BERT encoder over text, Sadeghi et al. [34] combine a large language model applied to interview transcripts with a biLSTM over facial-expression features, and Liu et al. [25] fuse facial-expression and body-posture streams recorded under emotional stimulation to reach 90.4% accuracy. In each case, the multimodal configuration outperforms its own unimodal ablations.

This prevailing fusion paradigm is, however, deep and opaque. Zhao et al. [45] organise contemporary strategies along an early–middle–late axis, and the depression-detection literature populates all three cells with neural machinery: SenseMood [24] fuses CNN and BERT embeddings at an intermediate layer, Sadeghi et al. [34] concatenate LLM and Bi-LSTM features at the intermediate level before a joint SVR, and Liu et al. [25] adopt a decision-level ResNet50 ensemble. Across these architectures, the fusion step is a learned transformation whose contribution cannot be traced back to a clinically named construct: explanations, when present, are appended post-hoc rather than derived from the model's own reasoning.

A small body of work pairs multimodal fusion with explanations. For instance, Kumar et al.'s MultiDepNet [21] integrates visual, physiological, audio, and textual streams and reports per-modality contributions via post-hoc LIME explanations; Khan et al. [18] combine dynamic gating, multi-head attention, and SHAP attributions to identify salient modalities and features. The direction is forming, but the explanations remain post-hoc overlays on deep architectures. The concern raised by Pahud de Mortanges et al. [28] therefore persists at the system level: multimodal depression systems that attempt explainability still do so after the fact, and none expose a reasoning path a clinician could audit step by step.

Taken together, these works establish that multimodal fusion is the empirically validated direction, but that it is, at present, an opaque one. While the notion of multimodality paired with explainability is forming, it it primarily focused on post-hoc mechanisms. This thesis retains the multimodal ambition while pursuing it within an inherently interpretable framework: each modality's contribution remains individually traceable, and fusion is expressed as a transparent decision-level combination rather than a learned embedding, directly addressing the interpretability tension surveyed above.

2.3.2 Explainability in Depression Detection Systems

The explainability landscape of supervised depression detection has been surveyed most recently by Lam et al. [22], whose meta-analysis of twenty studies published between 2014 and 2025 finds that SHAP serves as the primary explanatory tool in 70% of works, LIME in 15%, and 5% combine both; the remaining 10% provide no interpretive technique at all. The surveyed classifiers are opaque architectures, so their explanations are reconstructed after the fact rather than derived from the model's own reasoning.

Two recent works exemplify this dominant paradigm. Ahmed et al. [1] train an XGBoost classifier on wearable actigraphy from the Depresjon dataset and apply SHAP and LIME to identify circadian-disruption features as top predictors, achieving 84.9% binary accuracy on a single behavioural modality. Zogan et al. [47] instead propose MDHAN, a hierarchical attention network for social-media depression detection that treats tweet-level and word-level attention weights as explanatory scores. In both systems the explanation is an overlay onto an otherwise opaque pipeline, and the detection task is restricted to one modality.

When multiple modalities are fused, the problem compounds. Pahud de Mortanges et al. [28], surveying XAI for multimodal and longitudinal data in medical imaging, observe

that only a small number of multimodal medical AI studies incorporate explainability at all, and that early-fusion architectures (where heterogeneous inputs are concatenated before prediction) obscure how each modality contributes to the final decision. Taken together, these surveys document a depression-detection literature in which explainability is almost exclusively post-hoc and concrete systems operate on a single modality, while multimodal XAI remains rare even in adjacent medical domains.

The two direct white-box predecessors to this thesis, Nef [27] and Länzlinger [23], demonstrate that inherent interpretability is attainable for depression monitoring on a single modality (home-router network traffic and speech, respectively). Extending that property to a multimodal setting, where each modality's contribution remains individually traceable, is the gap this thesis addresses.

2.3.3 Context-Aware Health Monitoring

Context-awareness has matured into an established discipline spanning decades of research in ubiquitous and pervasive computing. Seminal work by Perera [30] and Baldauf [4] surveys the breadth of context-aware architectures, middleware platforms, and concrete systems deployed across IoT and ubiquitous computing environments. Early demonstrations of context-driven adaptation, such as the location-tracking systems presented by Harter et al. [13], established the technical feasibility of collecting, interpreting, and acting upon contextual signals in real-world settings.

Context-aware principles have been tentatively applied to depression monitoring, though fragmented across a small number of systems. Burns et al. [5] introduced Mobilyze!, the first explicit context-aware depression intervention, leveraging over 38 smartphone sensors to predict mood, activity, and social context; however, the system remained a proof-of-concept and did not scale to broader clinical use. Subsequent work by Saeb et al. [35] demonstrated that GPS-derived contextual features including circadian movement patterns and location entropy correlate with PHQ-9 severity. Canzian and Musolesi [6] further illustrated the depression-signal potential of contextual modalities, using smartphone mobility traces alone to detect depressive trajectories and earning a UbiComp 10-Year Impact Award, albeit using a single modality and opaque methods. More recently, Wang et al. [42] mapped phone and wearable contextual features to five DSM-5 symptoms, achieving 81.5% recall, thereby bridging context-awareness with clinical grounding.

However, while contextual signals improve prediction accuracy, they are commonly fed into opaque models that constitute these context-aware depression systems. How context shapes assessment remains hidden from clinicians. Further, no system combined three properties at once yet: context-aware reasoning in explicit form with multimodal sensor fusion and inherent explainability. This thesis bridges this gap by making context an explicit, interpretable component of multimodal sensor fusion within a DSM-5-grounded framework.

2.3.4 White-Box Predecessors

The preceding overview established that while multimodal fusion is the empirically validated direction for depression detection, there is a need for more research on inherently explainable architectures, and that while context-awareness has been applied to depression monitoring, it is typically fed into opaque models. A small body of work has explored explainable white-box architectures for depression assessment with explicit clinical grounding.

Länzlinger [23] proposed the *Linkage Framework*, a five-layer mapping that decomposes a DSM-5 diagnosis into indicators, biomarkers, metrics, and raw measurements, instantiated on acoustic speech features for major depressive disorder. Nef [27] followed a conceptually similar pipeline on home-router network-traffic features and introduced *Fuzzy Additive Symptom Likelihood* as a transparent, weight-parameterised aggregation operator that composes biomarker memberships into DSM-5 indicator scores. Both systems deliver what post-hoc XAI cannot: reasoning emerges from the model's own structure rather than being reconstructed after prediction. However, each operates on a single modality (speech and network traffic, respectively) and neither incorporates contextual information about the circumstances under which measurements were recorded.

2.3.5 Research Gap

Table 2.3 consolidates the discussed results into a comparative overview along the three dimensions of multimodality, explainability, and context-awareness.

Table 2.3: Comparative overview of related work for depression-monitoring systems.

Work	Modalities	Fusion Level	Explainability	Context
Lin et al. 2020 [24]	Image + text	Hybrid	–	no

Table 2.3 (continued)

Work	Modalities	Fusion Level	Explainability	Context
Sadeghi et al. 2024 [34]	Text + facial	Hybrid	Post-hoc (SHAP)	no
Liu et al. 2025 [25]	Facial + posture	Late	–	no
Kumar et al. 2025 [21]	Visual + physio. + audio + text	Early + Late	Post-hoc (LIME)	no
Khan et al. 2026 [18]	Video + audio + text	Hybrid	Post-hoc (SHAP + attn.)	no
Ahmed et al. 2025 [1]	Actigraphy	–	Post-hoc (SHAP/LIME)	no
Zogan et al. 2022 [47]	Social-media text	–	Post-hoc (attention)	no
Burns et al. 2011 [5]	Phone sensors	Early	–	yes
Canzian & Musolesi 2015 [6]	Phone mobility	–	–	yes
Wang et al. 2018 [42]	Phone + wearable	Early	–	yes
Länzlinger 2025 [23]	Speech (acoustic)	Late	Inherent	no
Nef 2025 [27]	Network traffic	–	Inherent	no

Notably, while these systems individually investigate one or few these properties, there is no depression-monitoring system that fully unites multimodal fusion, inherent explainability, and context-awareness within a DSM-5-grounded framework. Thus, this thesis proposes a novel approach for depression monitoring that integrates all three properties (multimodal fusion, inherent explainability, and context-awareness) within a DSM-5-grounded framework, thereby advancing the state-of-the-art in automated depression assessment systems.

Chapter 3

Design

This chapter presents the design contributions of the thesis in three parts. First, the *Contextual Multimodal Linkage (CML) Framework* is introduced as a conceptual extension of the Linkage Framework by Länzlinger [23], adding explicit support for multimodal data and contextual information and thereby addressing *Objectives 1* and *2* (see Table 1.1). Second, a system architecture is proposed that operationalises the framework into a deployable pipeline, addressing *Objective 3*. Finally, a set of functional and non-functional requirements is derived to guide the proof-of-concept implementation presented in Chapter 4.

3.1 The CML Framework

This section introduces the *Contextual Multimodal Linkage (CML) Framework*, which addresses two structural limitations of the original Linkage Framework [23]: the lack of explicit multimodal support and the absence of contextual information. The section opens with the *CML Mapping Pyramid* as a new illustration of the mapping mechanism, then details each of its two extensions (multimodal support and context sensitivity) in the subsequent subsections.

3.1.1 The CML Mapping Pyramid

The *CML Mapping Pyramid* shown in Figure 3.1 extends the original Linkage Framework with three changes:

Multimodal Extension: First, Layers 1–3 are replicated into n parallel modality streams, each maintaining an independent path from raw measurements through metrics to biomarkers. These streams converge at the boundary to Layer 4, where per-modality biomarkers are aggregated into shared indicator scores, realising a late-fusion as detailed in Section 3.1.2.

Context Extension: Second, an auxiliary context input feeds into Layer 4, adjusting the weight that individual biomarker contributions carry within the indicator aggregation depending on the prevailing situational circumstances, as detailed in Section 3.1.3.

Rename Layer 5: Third, Layer 5 is renamed from *Diagnosis* to *Assessment* to reflect the system's supportive role as it is not designed to perform diagnosis by itself; a responsibility that lies with the specialists.

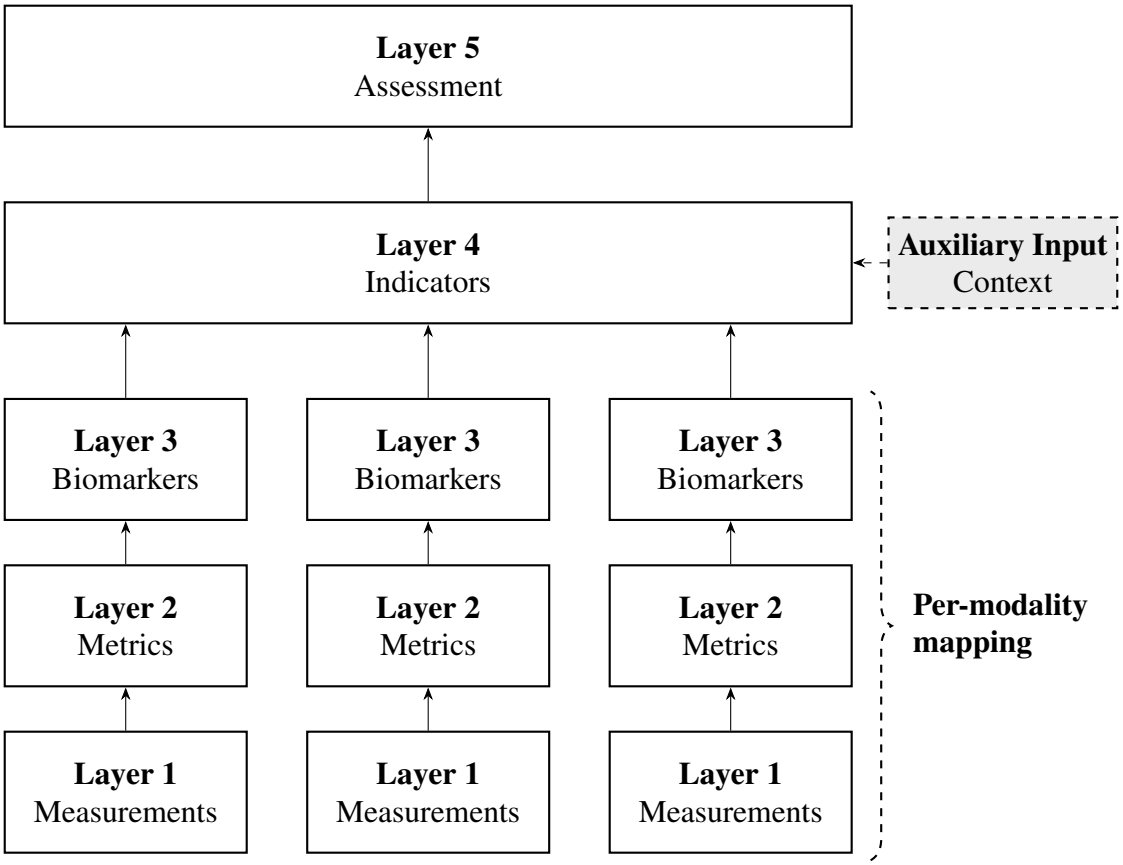


Figure 3.1: *CML Framework*, conceptual *Mapping Pyramid*.

Unchanged core structure. The five-layer hierarchy (with Layers: Measurement → Metric → Biomarker → Indicator → Assessment) remains the backbone of the extended model. No layer is removed, reordered, or redefined; the extensions are strictly additive. As a consequence, the three properties interpretability, uniformity, and interchangeability, from the original framework, carry over to the extended version, because the layer-by-layer traceability chain through which they are realised is preserved.

Independence and composability. The two structural extensions operate orthogonally. The multimodal extension adds parallel pathways that converge at Layer 4; the context extension modifies the relative weight of contributions within that same aggregation

step. Neither extension presupposes the other: a single-modality deployment benefits from context adjustment, and a multimodal deployment without context data simply applies uniform weights. When both are active, modality streams converge at Layer 4 and the contributions through which they converge are context-adjusted: the mechanisms compose naturally.

Alignment with research objectives. The multimodal extension addresses Objective 1 (*extend the Linkage Framework to support multimodal data input*) by making the convergence point and fusion boundary explicit in the conceptual model. The context extension addresses Objective 2 (*extend the Linkage Framework to support contextual information*) by introducing a systematic mechanism through which situational factors influence indicator computation. Together, the CML Framework constitutes the conceptual foundation upon which the remaining design sections build.

3.1.2 Multimodal Extension

The original Mapping Pyramid depicts a single vertical stream from measurement to diagnosis and leaves the point at which data from different modalities converges unspecified. This subsection makes the multimodal mechanism explicit in the conceptual model by defining where and how modalities are combined.

Design decision: late fusion at the indicator boundary. The extended framework adopts a late-fusion architecture in which each modality maintains its own independent path through Layers 1–3 (Measurement $\rightarrow$ Metric $\rightarrow$ Biomarker). Convergence occurs exclusively at Layer 4 (Indicator), where per-modality biomarkers are combined into indicators through a transparent aggregation mechanism. The fusion boundary therefore sits between Layer 3 and Layer 4.

Four properties of the Linkage Framework motivate this placement of the fusion boundary; the underlying fusion types are discussed in Section 2.1.

First, *modality independence*. Layers 1–3 typically have large differences across modalities: for instance, speech analysis produces prosodic metrics such as pitch variability and pause ratio, whereas network traffic analysis produces behavioural metrics such as session frequency and data volume. Fusing at Layer 2 would require defining explicit cross-modal relationships between metrics that are semantically unrelated, thereby coupling the framework more strongly to specific modality pairings.

Second, *interchangeability*. Any modality can be added or removed without structural changes, provided it delivers biomarkers that map to the shared Layer 4 indicators.

Concretely, a new modality—for example, network traffic analysis—can be integrated by defining its own Layer 1–3 mapping and connecting its biomarkers to existing indicators; no existing modality stream or indicator logic requires modification. Conversely, under an early- or intermediate-fusion strategy, adding a modality would require defining new cross-modal metric relationships with existing modalities, introducing coupling between streams that are otherwise independent.

Third, *explainability*. Per-modality biomarkers remain individually inspectable before and after fusion, preserving the traceability chain that constitutes a core design goal of the framework (Section 3.2). A clinician reviewing an elevated *psychomotor retardation* indicator can trace which modality contributed what evidence and how strongly—per-modality attribution that would be obscured if fusion occurred earlier, at the metric level. Furthermore, the fusion step itself must remain interpretable: combining biomarkers through a transparent formula is straightforward, whereas combining heterogeneous metrics from different modalities into an interpretable joint representation is considerably more difficult.

Fourth, *practical complexity*. Fusing modalities at the metric level is impractical for typical use due to two issues. First, it is non-obvious what combination of metrics of different modalities constitutes a biomarker. Although fusing at the biomarker-level introduced a similar practical problem (ie., what biomarkers constitute an indicator), the deeper granularity-level of metric-level fusion perturbates the issue. Second, the combinatorial space grows rapidly with an increasing number of modalities. The theoretical approach is similar to fusion at the biomarker-level—but with an increased complexity. Late fusion at the biomarker level inherits the same structural challenge at a coarser granularity, where the mapping to clinical indicators is better-defined and the combinatorial burden remains manageable.

Cross-modal indicator composition. Indicators aggregate evidence from biomarkers to produce the indicator-level scores used for assessment. Because convergence occurs at the indicator level, a single DSM-5 indicator can draw on biomarkers from *multiple* modalities simultaneously. For instance, a single indicator may be computed using biomarkers from two different modalities (speech and network). This mechanism makes the multimodal extension more than merely additive: modalities reinforce each other at the indicator level, improving assessment robustness.

Modality-agnostic system boundary. The extended framework deliberately does not prescribe or constrain how external analysis tools organise their modalities internally. The

system boundary is set at the biomarker layer (Layer 3): once biomarkers are delivered to the framework, they are treated uniformly regardless of how many raw sensor modalities contributed to their computation.

Consequently, *modality* in the context of the extended pyramid refers to a *biomarker source* rather than necessarily a single physical sensor type. A tool that internally fuses audio and video data still appears as one modality stream from the framework's perspective, and the late-fusion architecture remains unaffected. This distinction is relevant because certain clinically meaningful signals can only be detected through cross-modal analysis *within* a tool. Verbal–facial affect mismatch, for instance, requires temporally aligned audio and video data that cannot be recovered by fusing separate per-modality indicator scores after the fact. The framework accommodates such cases by placing no constraints below Layer 3: tools are free to employ whatever internal fusion strategy best serves their biomarker extraction. Each source is responsible for how it provides the necessary biomarkers that are mapped to the DSM-5 indicators.

The practical benefit is that the framework can freely map any available biomarker to any indicator, regardless of its origin. If a new tool appears that derives the same biomarker from a different sensor, it slots into the existing indicator mapping without structural changes.

3.1.3 Context Extension

The original framework did not consider yet how situational circumstances under which an observation was recorded influence how it should interpreted. To address this gap, the extended framework introduces *context* as an auxiliary input, referring with context to external situational factors such as environmental and social conditions surrounding the observation, rather than internal states such as mood or fatigue, which are what the biomarkers themselves capture.

Design decision: context as auxiliary input. The extended framework models context as an *auxiliary input to Layer 4* (Indicator), where it adjusts the weight that individual biomarker contributions carry within the indicator aggregation depending on the prevailing situation. Two properties the Linkage Framework motivate this placement of the auxiliary context.

First, *explainability*. Because context modifies the weight of a biomarker-to-indicator link rather than the biomarker value itself, the influence of context on the assessment remains individually inspectable. A clinician reviewing an elevated indicator can trace

not only which biomarkers contributed but also how the prevailing context adjusted each contribution — attribution that would be obscured if context were applied earlier, at the metric or measurement level.

Second, *practical complexity* Applying context at Layers 1–3 would require defining context-dependent transformations for each metric or measurement individually, across every modality stream. The combinatorial space grows rapidly with the number of modalities and metrics. At the indicator boundary, context operates on a smaller, clinically defined set of biomarker-to-indicator weights, keeping the configuration space manageable and the rules interpretable.

Context-to-biomarker contribution. The active context at the time of data collection influences the contribution of a biomarker to an indicator. This mechanism is orthogonal to the multimodal extension. While multimodality handles how biomarkers are combined, context simply influences the relevance of each biomarker given the contextual situation at hand. When both are active, modality streams converge at Layer 4 and the contributions through which they converge are context-adjusted.

Modality-agnostic context sources. The extended framework deliberately does not prescribe or constrain what specific context sources are available. New context markers can be introduced without redefining existing situational categories, and new categories can combine existing markers. This mirrors the modality-agnostic system boundary established for biomarkers : just as the framework accepts any biomarker regardless of its sensor origin, it accepts any context marker regardless of its source.

3.2 Explainability by Design

Explainability is an inherent property of the *Linkage Framework*. The design therefore treats explainability not as a post-hoc add-on but as a first-class architectural constraint. This commitment directly realises Objective-3, which requires an architecture that supports *explainable*, multimodal, and context-enriched health assessments (Table 1.1), and underpins the first contribution of this thesis: demonstrating that multiple sensor modalities can be combined within an explainable, white-box framework (Section 1.4).

Consequently, the *CML Framework* preserves the interpretability property of the original model: the layer-by-layer mapping from measurements to assessment remains transparent and traceable. This means that every computational stage in the pipeline, from raw data to DSM-5-aligned assessment, must use operations whose inputs, parameters, and outputs are individually inspectable by a human observer.

3.2.1 Design Principles for Explainability

The explainability constraint translates into three concrete design principles that govern the pipeline architecture presented in the subsequent sections.

Principle 1: End-to-end traceability. For any output of the system—whether an indicator score, a context evaluation, or an episode decision—a user must be able to trace the complete derivation path back to the raw input data. This principle requires that every intermediate result is preserved and attributable. The traceability chain spans the full pipeline. No step in this chain involves an operation that discards attribution or collapses distinct inputs into an unrecoverable aggregate.

Principle 2: No black-box operations. Every computational operation in the pipeline is constrained to closed-form, parameterised functions with no learned or opaque components. The system contains no neural networks, no gradient-based optimisation, and no opaque embeddings. Computations use configurable parameters (no neural-network-based transformations) and explicit weights and operators. The constraint provides interpretability at every stage.

Principle 3: Deterministic reproducibility. Given identical inputs and an identical configuration, the system must produce identical outputs. No computation in the pipeline involves stochastic sampling, non-deterministic ordering, or random initialisation. This property is essential for auditability: any result—whether an indicator score, a context evaluation, or an episode decision—can be reproduced and independently verified after the fact by re-running the same inputs through the same configuration. Without determinism, traceability (Principle 1) would be undermined, because a traced derivation could not be confirmed by re-execution.

3.2.2 Limitations

The white-box design carries inherent trade-offs that must be acknowledged. First, interpretability does not imply clinical validity. The fact that every computation step can be inspected and understood does not establish that the system's outputs are diagnostically accurate. Interpretability enables trust-building and supports clinical scrutiny, but it does not substitute for validation against clinical ground truth—which remains outside the scope of this thesis (Section 1.5). Second, the no-black-box constraint implies that configurable parameters must also be set through a transparent process, possibly with the help of domain experts. Otherwise, it becomes unclear how results are achieved.

While this ensures full transparency over parameter values, it also means that the system cannot automatically adapt to data patterns. While some approaches might exist that automatically calibrate parameters while preserving traceability, this investigation remains out of scope.

3.3 System Architecture Overview

This section translates the *CML Framework* introduced in Section 3.1 into a concrete system architecture. Figure 3.2 presents the system architecture. A dashed boundary separates the components under design from the external sources that feed data into the system and the human actors that interact with it. The remainder of this section introduces each component within the proposed system architecture and summarises the end-to-end data flow.

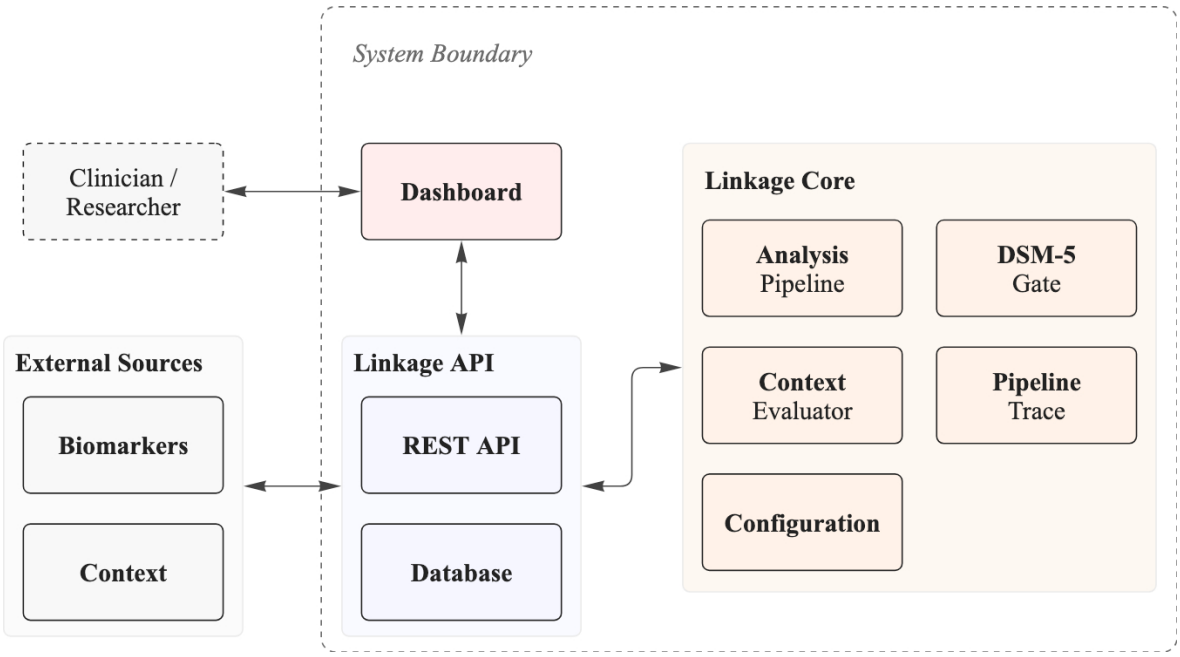


Figure 3.2: High-level system architecture.

External sources. Biomarker and context data originate from modality-specific tools that reside outside the system boundary. Speech analysis tools, for instance, extract prosodic and linguistic biomarkers from audio recordings, while network analysis tools derive behavioural metrics from packet-level traffic data. These tools operate at Layers 1–3 of the Mapping Pyramid and submit their results to the system at the Layer 3 boundary; using the modality-agnostic interface established by the multimodal extension

(Section 3.1.2). Context sources follow the same submission path, providing situational markers such as ambient noise levels or the number of people present via the modality-agnostic Linkage API.

Linkage API. The Linkage API serves as the single entry point for all external data. It exposes a REST interface through which biomarker and context sources submit observations in a uniform data format, regardless of modality. The API validates incoming data, authenticates the submitting source, and persists all observations to the database. By consolidating ingestion into one component with a modality-agnostic contract, the architecture ensures that adding a new data source requires no structural changes to the system as a new external tool only needs to conform to the existing interface. Section 3.4.1 specifies the API in detail.

Database. A relational database acts as the central persistence layer and the primary decoupling mechanism between components. The API writes raw observations; the Linkage Core reads them for analysis and writes computed results back; the Dashboard queries both raw and computed data for visualisation. Because all inter-component data exchange passes through shared storage rather than direct runtime calls, each component can operate, evolve, and be tested independently. The database also stores configuration snapshots alongside each analysis run, enabling reproducibility of past results. Section 3.4.2 specifies the persistence in detail.

Linkage Core. The Linkage Core constitutes the computational centre of the system. It comprises four subcomponents: a *Context Evaluator* that derives the active situational context from raw markers, an *Analysis Pipeline* that computes indicator scores from normalised biomarker data, a *DSM-5 Gate* that applies decision logic to infer whether a depressive episode is likely, and a *Pipeline Trace* that records intermediate results for full traceability. All subcomponents are governed by external configuration files, allowing researchers to adjust parameters without modifying application code. Section 3.5 specifies the design of the linkage core in detail.

Dashboard. The Dashboard provides a web-based interface through which clinicians and researchers interact with the system. It enables users to trigger analysis runs, inspect the full pipeline trace from raw biomarker to episode decision, and adjust configuration parameters. The Dashboard invokes Linkage Core components directly to trigger analysis and context evaluation runs, and queries the database for both raw observations and computed results. Section 3.6 specifies the dashboard in detail.

Pipeline Trace. Data moves through the system in a linear, traceable sequence. External tools submit biomarker and context observations to the Linkage API, which persists them to the database. When a clinician or researcher triggers an analysis run via the Dashboard, the Linkage Core reads the relevant observations from the database, evaluates context, computes indicators, applies DSM-5 gate logic, and writes all intermediate and final results back to the database. The Dashboard then queries these results for visualisation. At every stage, the pipeline trace captures the inputs, parameters, and outputs of each computation, ensuring that any indicator or episode decision can be traced back through the full chain of transformations to its originating biomarker readings.

3.4 Linkage API

This chapter presents in detail the *Linkage API*, which is one of the principal components presented in the overview of the system architecture in Section 3.3. The component governs how data enters, traverses, and exits the system. Figure 3.3 presents the Linkage API as a component-view, including the *external* and *internal boundary*. These two distinct boundaries of the Linkage API structure all data exchange.

External Boundary. The first boundary is the *external boundary*, which is governed through an API through which all external sources must go. All external systems must submit data through the API, which enforces authentication and validates every incoming payload. Direct access from outside to the database is not possible. By consolidating all external access into a single, well-defined interface, the API boundary ensures that the internal state of the system remains protected from uncontrolled modification.

Database Boundary. The second boundary is the *database boundary*: the system's internal integration layer. Inside the system, the three components External Boundary API, Linkage Core, and Dashboard never invoke one another directly. Instead, they communicate exclusively by reading from and writing to the shared database. The API writes raw observations; the Linkage Core reads those observations, performs its analysis, and writes computed results back; the Dashboard queries both raw and computed data for visualisation. Because all inter-component data exchange passes through shared storage rather than direct runtime calls, each component can be developed, deployed, and tested independently of the others.

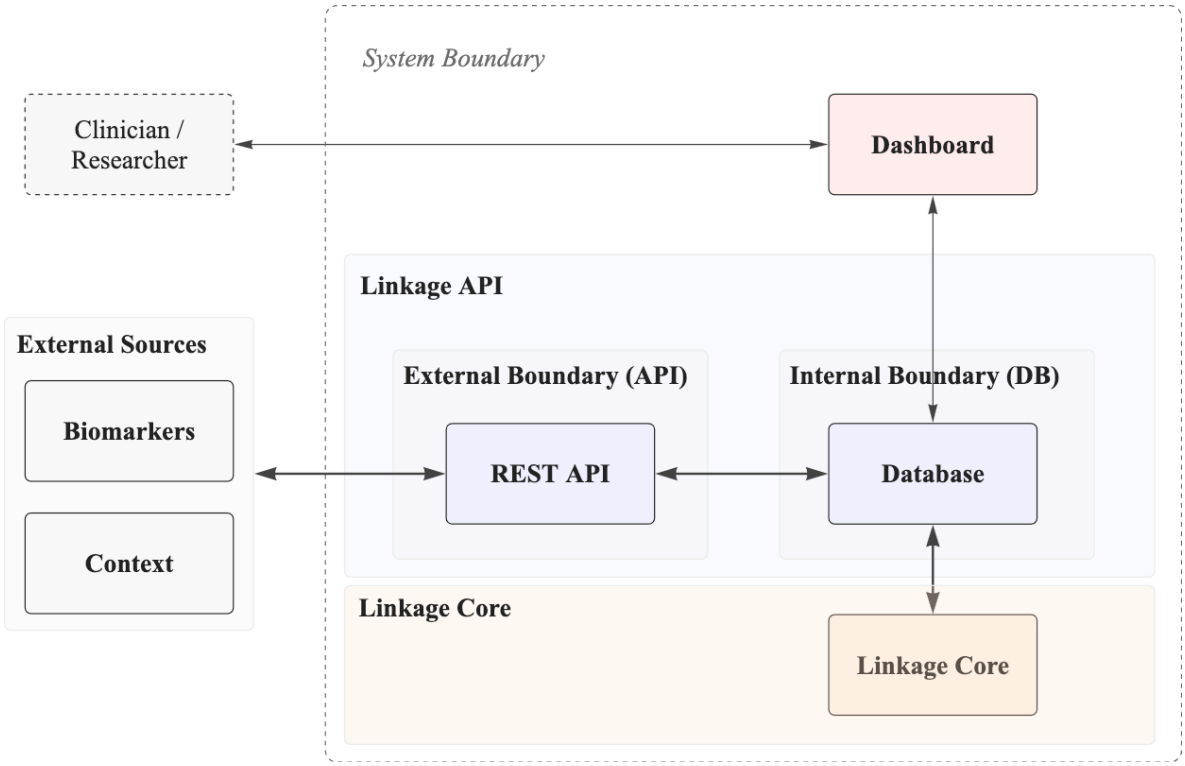


Figure 3.3: Linkage API: The external and internal boundaries mediate dataflow.

Together, the two boundaries establish three architectural guarantees that the remainder of this chapter relies on:

- *Data contract*: external tools must conform to the API's modality-agnostic schema, providing a uniform ingestion format regardless of the originating sensor modality.
- *Separation of concerns*: each internal component owns a clearly delineated responsibility and interacts with the others only through the database, eliminating tight coupling.
- *Extensibility*: adding a new data source or a new internal component requires no structural changes to the existing system—the new participant simply conforms to the established boundaries.

The following subsections specify the two boundaries in detail: the API and its data contract (Section 3.4.1), and the database as the internal integration layer (Section 3.4.2).

3.4.1 API

The Linkage API contains the system's *external boundary*, which is the sole point of entry for data from outside the system. External, modality-specific tools are responsible for data collection and biomarker computation (Layers 1–3 of the CML Mapping Pyramid, Figure 3.1). This system consumes the resulting biomarker values at the Layer 3 output. The decision to place the system boundary at Layer 3 is deliberate and was established in the modality-agnostic system boundary discussion of Section 3.1.2. The Linkage API is the concrete architectural design of that boundary.

Asymmetric data flow. The API enforces a deliberate asymmetry between inbound and outbound data. On the *write path*, external sources submit biomarker readings and context markers through the API. These are observational datapoints entering the system for the first time. On the *read path*, computed indicators—the products of the analysis pipeline—are exposed as read-only resources. Indicators are produced exclusively by the Linkage Core, which writes them directly to the database; no external actor can create, modify, or delete an indicator through the API. This asymmetry protects the integrity of computed results: every indicator is the deterministic output of a traceable analysis run, and it cannot be compromised by external writes. Retrieval of raw biomarker and context data is also supported, enabling external tools to verify what was ingested.

Additionally, the API requires authentication to identify the submitting source, enabling per-source traceability. Any tool conforming to the data contract can participate in the system, independent of its internal modality handling.

Design decision: modality-agnostic data contract. The central design decision of the API is its modality-agnostic data contract. All biomarkers—regardless of their modality source—share a uniform envelope structure: a user identifier, a timestamp, a type field identifying the modality, a flexible value field carrying the modality-specific payload, and optional metadata. Context markers follow the same structural pattern with their own type taxonomy; the only difference is the type discriminator field name (`biomarker_type` versus `context_type`). Biomarkers and context markers are separate resources with separate endpoints, but the envelope is otherwise identical. Table 3.1 illustrates three payloads conforming to this contract.

Table 3.1: Example of modality-agnostic data contracts.

Resource	Type Field	Type Value	Value Payload
Biomarker	`biomarker_type`	`speech`	`{pitch_mean: 120.5, speech_rate: 3.2}`
Biomarker	`biomarker_type`	`network`	`{session_count: 14, avg_duration: 42.1}`
Context	`context_type`	`location`	`{setting: "home", people_present: 0}`

This design has a critical extensibility property: adding a new modality requires no schema changes, no new endpoints, and no modifications to existing code. The new modality is simply a new value in the type field. This directly realises the modality-agnostic system boundary defined in Section 3.1.2 and is the mechanism through which the Extended Linkage Framework achieves its modality-independence at the system level.

Design decision: REST as interface style. The API follows REST, which remains the dominant API paradigm with 93% of responding organisations using it in 2025 [31]. Beyond prevalence, REST is a natural fit for this system's interaction pattern: external tools submit timestamped observations and retrieve computed results. The data model maps directly to resources (biomarkers, context markers, indicators), and operations are exclusively CRUD, a scenario where REST's resource-oriented semantics align without overhead.

3.4.2 Database

The Linkage API contains the system's *internal boundary* in the form of the database. This subsection focuses on the design decisions that shape the data layer and the conceptual categories of information it manages. As established above, the database serves as the sole communication channel between the three internal components. For instance, if one component wants to send data to another, it must store the data in the database for the other component to read, it cannot send the data directly to the other component. Data always flows through the database to the respective components.

In a typical dataflow, the *Linkage API* writes raw observations; the *Linkage Core* reads those observations, performs its analysis, and writes computed results back; the

Dashboard queries both raw and computed data for visualisation. This architecture provides three properties:

- *Temporal decoupling*: data can be ingested at one time and analysed later, as real-time connectivity between components is not required.
- *Independent evolution*: each component can be modified, replaced, or scaled without affecting the others, as long as the database schema contract is maintained.
- *Single source of truth*: there is exactly one authoritative location for every piece of data, eliminating synchronisation concerns between components.

Three design decisions shape the data layer. Together, they ensure that the database fulfils its role as an integration backbone while preserving the traceability and extensibility guarantees established at the system boundary level.

Relational model. The first decision is the adoption of a *relational model*. Temporal biomarker data has a natural tabular structure: each reading is a row with a user, a timestamp, a type, and a value. More importantly, relational integrity constraints, such as foreign keys from indicators to analysis runs, or from analysis runs to configuration snapshots, enforce the traceability chain that is a core design goal of this system (Section 3.2). Every computed result is linked to the run that produced it and the configuration that governed that run.

Flexible value storage. The second decision concerns *flexible value storage*. Biomarker and context payloads are stored as structured data (JSON) rather than fixed columns. This is the persistence-layer counterpart of the modality-agnostic API contract (Section 3.4.1): a speech biomarker with fields such as `pitch_mean` and `speech_rate` coexists in the same table as a network biomarker with fields such as `session_count` and `packet_loss`, without requiring schema migration when a new modality is introduced. The trade-off, namely reduced query-time type safety and indexing capability on payload fields, is acceptable because the analysis pipeline accesses these values programmatically through typed application code, not through ad-hoc SQL queries.

Configuration snapshots. The third decision mandates *configuration snapshots*. Every analysis run persists the full configuration (weights, thresholds, membership function parameters) that was active at execution time. This ensures reproducibility: a past analysis can be understood in terms of the parameters that produced it, even if the configuration has since changed.

The database organises information into five conceptual categories that collectively span the full data lifecycle, from initial sensor ingestion through intermediate evaluation to final clinical output:

Raw input data: Biomarker readings and context markers as ingested through the API; the unprocessed observations that enter the system.

Evaluated context: The output of the context evaluation pipeline (Section 3.5.1): context history entries recording the dominant context, confidence score, and evaluation metadata for each evaluation window.

Computed results: The output of the analysis pipeline (Section 3.5.2): window-level indicator scores, daily aggregated summaries, and episode-level clinical decisions.

Run metadata: Analysis runs and context evaluation runs, each linked to a configuration snapshot and a pipeline trace that captures the inputs, parameters, and outputs of every computation step.

User baselines: Per-user, per-biomarker statistical profiles (mean, standard deviation) used for z-score normalisation in the analysis pipeline.

3.5 Linkage Core

This section introduces the computational core of the system, illustrated in Figure 3.4 and comprises four principal components: the *Context Evaluator*, which determines the dominant situational context from raw context markers; the *Analysis Pipeline*, which fuses biomarker scores into indicator-level assessments using context-adjusted weights; the *DSM-5 Gate*, which maps aggregated indicator scores to clinical episode decisions; and the *Configuration*, which governs every tuneable parameter across the preceding three components. Data flows through these components sequentially: context evaluation precedes analysis so that context-adjusted weights are available when the pipeline aggregates biomarker contributions, and the DSM-5 Gate operates on the indicator scores that the pipeline produces. The following subsections detail each component in turn.

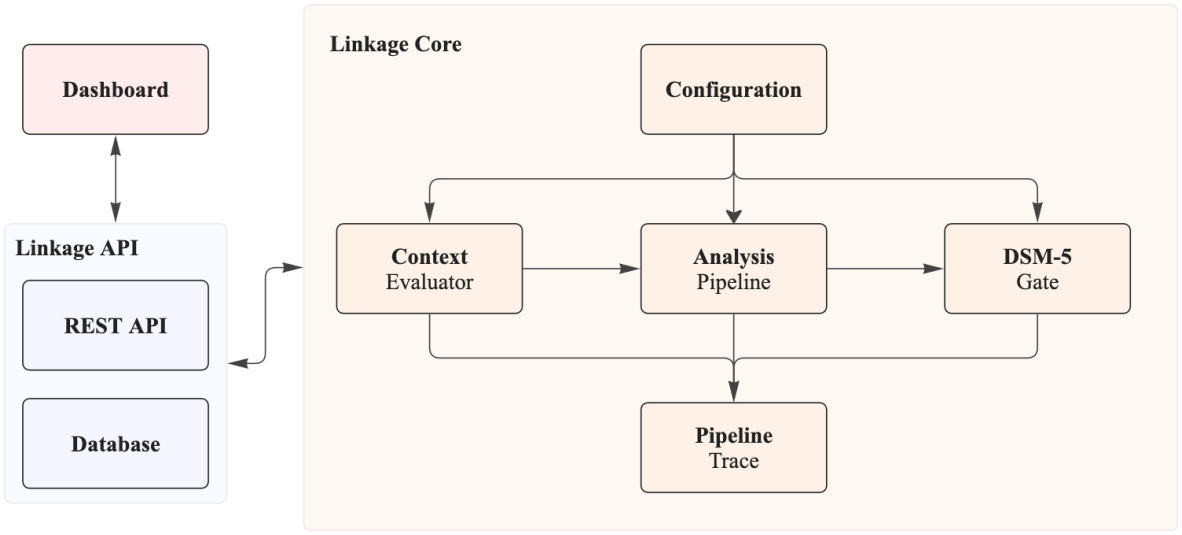


Figure 3.4: Linkage Core: Overview of components.

3.5.1 Context Evaluator

The thesis established in Section 3.1.3 that context enters the Extended Mapping Pyramid as an auxiliary input to Layer 4, where it adjusts the weight that individual biomarker contributions carry within the indicator aggregation. This subsection defines *how* the the *Context Evaluator* determines the context that serves as auxiliary input.

3.5.1.1 Representation of context

The context component operates on two distinct levels of representation.

Context markers are raw numeric sensor inputs that describe individual facets of the environment. For example, the number of people in the room, the ambient noise level, or the level of network activity. They are gathered alongside biomarkers, through sensors such as a presence sensor.

Context types, by contrast, are derived interpretive states that characterise the overall situational category. Consider the situation *"young adult alone in bed browsing social media at night"*. Arguably, no single marker can capture this context type by itself. Instead, a combination of markers is used. For example, this context type could be characterized by the combination of:

- the number of people in the room must be low,
- network activity must be high,
- ambient noise must be low,

- and the number of door-open events must be low.

Each context type is therefore defined by an *assumption*, a set of context markers that characterise the context type. Table 3.2 illustrates this structure for the example above.

Table 3.2: Example of context assumptions for a context type.

Context type	Context marker	Expected value	Weight
Young adult alone in bed browsing social media at night	People in room	low	0.30
	Network activity	high	0.30
	Ambient noise	low	0.20
	Door-open events	low	0.20

Given sufficient context markers, any contextual situation can be represented. Further, this separation also preserves extensibility: new context markers can be introduced without redefining existing context types, and new context types can combine existing markers in novel ways, all through configuration rather than code changes.

3.5.1.2 Context evaluation pipeline

The context evaluation pipeline, see Figure 3.5, performs six steps to transform raw context markers into an active context with a confidence score.

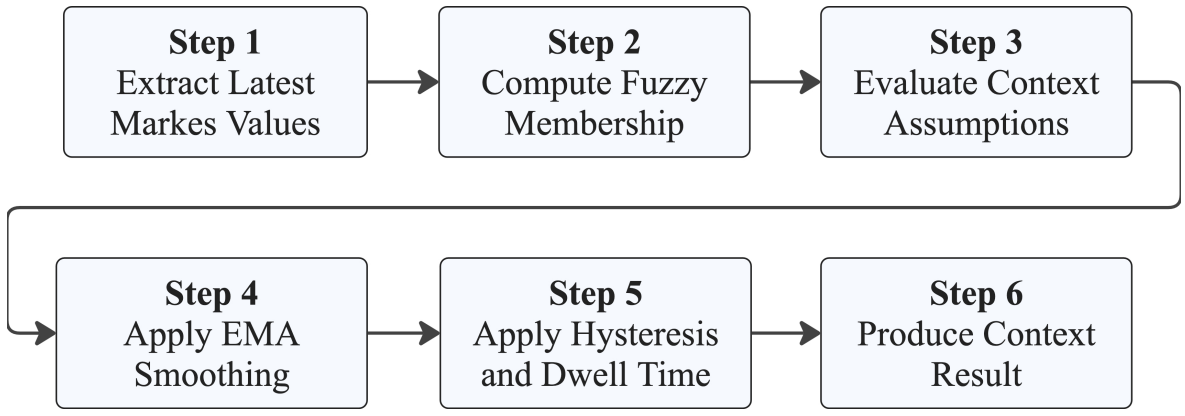


Figure 3.5: Overview of the six-step context evaluation pipeline.

The six-step pipeline is executed repeatedly at a configurable interval. Regular re-evaluation is a necessary constraint because a person's context can change substantially throughout the day. The context evaluation must be executed often enough to capture

all contextual changes. If context is not evaluated with a sufficient frequency, the system would either lack sufficient context information or, arguably worse, use outdated information to wrongly influence the analysis.

Subsequent subsections explain each step in detail, beginning with the extraction of the latest marker values and concluding with the production of the final context result.

3.5.1.3 Step 1: Extract Latest Marker Values

At each evaluation interval, the evaluator receives all context marker readings that fall within the interval's time window. Because data sources may report values more frequently than the evaluation interval, multiple readings can exist for the same marker within a single interval. For instance, a presence sensor might have reported *people_in_room* = 3 at 10:00 and *people_in_room* = 7 at 10:05, both falling within the same fifteen-minute interval. Similarly, not every sensor will provide readings at the same frequency. The first step of the pipeline resolves this multiplicity by extracting the most recent reading per marker name, producing a single snapshot: one value per marker.

Given the example from Table 3.2, Step 1 would yield the latest values of all relevant sensors. Table 3.3 illustrates a possible output.

Table 3.3: Example of output for single interval.

Context marker	Value
people_in_room	1
network_activity_level	0.9
ambient_noise	0.45
door_open_events	1

Selecting the latest value rather than aggregating historical readings is a deliberate design choice: context evaluation should describe the situation *as it is* at the time of evaluation, not as it was on average over the preceding interval. Averaging across multiple readings would produce a blended value that may correspond to no real situation: a room that held three people ten minutes ago and seven people now is not well described by an average of five. By restricting the input to the most recent observation, the pipeline ensures that downstream steps operate on the best available approximation of the current environment.

3.5.1.4 Step 2: Compute Fuzzy Membership

The marker snapshot produced by Step 1 contains raw numeric values on heterogeneous scales: some are integer counts (e.g., *people_in_room* ranging from 0 to 10), others are normalised ratios on $[0, 1]$ (e.g., *ambient_noise*). These values cannot be compared or combined directly, because a count of 1 and a ratio of 0.9 carry no comparable meaning without additional interpretation. Step 2 resolves this by translating each raw value into *fuzzy membership degrees* (Section 2.2.1).

For every marker and each of its configured linguistic sets, a membership function $\mu : X \to [0, 1]$ maps the raw value to a degree that expresses how strongly the observed reading satisfies the corresponding qualitative category. Table 3.4 illustrates this output format for a single marker. A marker configured with three linguistic sets (*low*, *medium*, *high*) yields three membership degrees per evaluation interval, one for each set.

Table 3.4: Example of membership degrees produced by step 2. Specific results depend on the actual configuration of memberships.

Marker	***low***	***medium***	***high***
ambient_noise $(x = 0.45)$	0.17	0.50	0.00

All sets are evaluated unconditionally, because different context types may reference different linguistic sets for the same marker: one context assumption might query whether *ambient_noise* is *low*, while another queries whether the same marker is *high*. By computing every degree upfront, Step 2 provides a complete membership profile that Step 3 can index selectively without requiring re-evaluation.

Available membership function families. The computation of membership degrees requires selecting a function family whose shape encodes how gradually a raw value transitions between full and no affiliation with a linguistic set. This thesis employs four families: triangular, trapezoidal, sigmoid and Gaussian. Practical selection criteria that guide the choice of family for a given marker are discussed in Section 2.2.1.

Running example. Returning to the marker snapshot from Step 1, Step 2 evaluates every marker against each of its configured linguistic sets. Table 3.5 lists the membership function type and parameters configured for each marker in the running example. All functions follow the triangular or trapezoidal definitions.

To illustrate how the degrees are obtained, consider the marker *ambient_noise* with value $x = 0.45$. The *low* set has support up to $h = 0.5$, while the *medium* set begins

Table 3.5: Example of membership configuration for the four context markers used in the running example.

Context marker	Domain	*low*	*medium*	*high*
people_in_room	[0, 10] count	tri (0, 1, 2)	tri (1, 3, 5)	trap (4, 7, 10, 10)
network_activity_level	[0, 1] ratio	tri (0, 0.2, 0.5)	tri (0.4, 0.5, 0.7)	tri (0.6, 1.0, 1.0)
ambient_noise	[0, 1] ratio	tri (0, 0.2, 0.5)	tri (0.4, 0.5, 0.7)	tri (0.6, 1.0, 1.0)
door_open_events	[0, 10] count	tri (0, 0, 2)	tri (1, 3, 5)	trap (4, 6, 10, 10)

at $\ell = 0.4$; consequently, the interval $[0.4, 0.5]$ is an overlap region in which both sets have non-zero membership. Because $x = 0.45$ lies within this interval, it receives a membership degree in both sets. Figure 3.6 visualises the three linguistic sets together with the resulting membership degrees at $x = 0.45$.

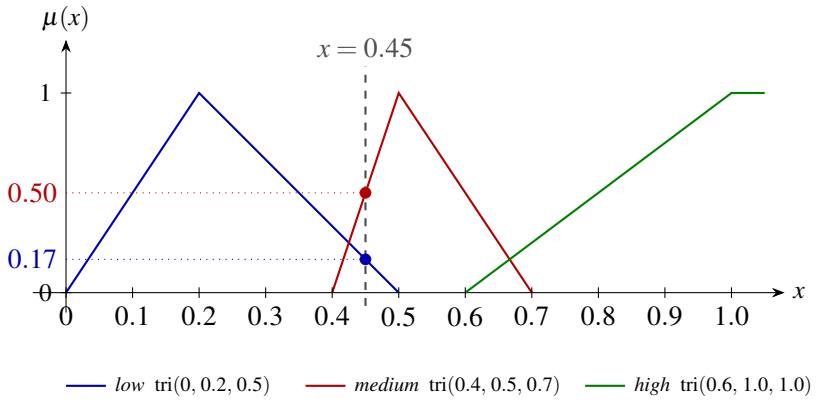


Figure 3.6: Applied membership function for context marker *ambient noise*.

Consider that the *low* set is configured as a triangular function with $(\ell, m, h) = (0, 0.2, 0.5)$. Since $m < x < h$, only the descending branch of Equation (2.1) applies:

$$\mu_{\text{tri}}(0.45; 0, 0.2, 0.5) = \frac{h - x}{h - m} = \frac{0.5 - 0.45}{0.5 - 0.2} = 0.17.$$

Consider that the *medium* set is configured with $(\ell, m, h) = (0.4, 0.5, 0.7)$. Since $\ell < x < m$, the ascending branch applies:

$$\mu_{\text{tri}}(0.45; 0.4, 0.5, 0.7) = \frac{x - \ell}{m - \ell} = \frac{0.45 - 0.4}{0.5 - 0.4} = 0.50.$$

The *high* set with $(\ell, m, h) = (0.6, 1.0, 1.0)$ receives $\mu = 0.00$ because $x < \ell$. A noise level of 0.45 thus belongs partially to both *low* and *medium*, illustrating the core property

of fuzzy membership: a single observation can carry graded affiliation with multiple linguistic sets simultaneously, rather than being forced into exactly one category. Table 3.6 shows the resulting membership degrees for all markers, each evaluated against the parameters listed in Table 3.5.

Table 3.6: Example of computed fuzzy membership degrees.

Context marker	Value	μ_{low}	μ_{medium}	μ_{high}
people_in_room	1	**1.00**	0.00	0.00
network_activity_level	0.9	0.00	0.00	**0.75**
ambient_noise	0.45	**0.17**	0.50	0.00
door_open_events	1	**0.50**	0.00	0.00

Design rationale: fuzzy over crisp classification. An alternative to fuzzy membership would be crisp threshold classification, in which each marker value is assigned to exactly one linguistic set through hard boundaries (e.g., *people_in_room* $\leq 2 \Rightarrow$ *low*, otherwise *not low*). The general limitations of crisp classification—discontinuous decisions at arbitrary boundaries and the loss of gradual transitions—are discussed in Section 2.2.1. Within this pipeline, however, crisp classification would have an additional, structural consequence: the downstream steps operate on continuous $[0, 1]$ signals. Step 3 computes a weighted aggregation of membership degrees to produce a context assumption score, Step 4 applies exponential moving average smoothing to stabilise that score over time, and Step 5 uses hysteresis thresholds and dwell times to prevent oscillation between competing contexts. Each of these mechanisms relies on the ability to distinguish strong support from marginal support—a distinction that binary inputs (0 or 1) would collapse entirely. A marker reading just inside a crisp boundary would contribute identically to one well within it, and the pipeline would lose the granularity needed for smooth temporal transitions and meaningful confidence scores. Fuzzy membership preserves the inherent measurement uncertainty and propagates it through every stage of the pipeline, consistent with the transparency and auditability principles that guided the original membership function design by Nef [27].

3.5.1.5 Step 3: Evaluate Context Assumptions

Step 3 uses the initial context assumption to select the relevant context markers and aggregate their membership degrees into a single raw score per context type. The score quantifies the degree to which the current sensor readings support the assumed context.

As established in Table 3.2, each context assumption comprises a set of conditions, where each condition specifies a context marker, the expected linguistic set (e.g., *low* or *high*), and a weight reflecting the marker's relative importance. The weights within an assumption are normalised to sum to 1.0. The aggregation applies the weighted mean across all conditions:

$$s_{ctx} = \frac{\sum_i w_i \mu_i}{\sum_i w_i} = \sum_i w_i \mu_i, \tag{3.1}$$

where μ_i denotes the membership degree of the i-th marker in its expected linguistic set (from Step 2) and w_i the corresponding condition weight. Since the weights sum to 1.0, the denominator equals 1 and can be omitted (Section 2.2.3).

A dedicated *neutral* context type serves as the system fallback. Unlike other context types, the neutral context carries no conditions and therefore no weighted-mean computation. Instead, the system defaults to the neutral context whenever no non-neutral assumption produces a score exceeding a configurable threshold (default 0.3). This mechanism ensures that the pipeline always yields a valid context result, even when sensor readings do not support any specific situational interpretation. The practical consequence of a neutral context is that no contextual weights are applied to influence the computation.

The weighted-mean formulation also provides graceful degradation in the presence of missing data. If a context marker is temporarily unavailable, for instance, because a sensor is offline, its condition is excluded from the summation and the remaining weights are renormalised to maintain a valid $[0, 1]$ output. Should all markers referenced by an assumption become unavailable, the assumption score defaults to 0.0, effectively deferring to the neutral fallback.

Running example. Table 3.7 demonstrates the aggregation for the running example. The membership degrees are taken from Table 3.6, and the weights from Table 3.2.

Table 3.7: Example of weighted contribution for the four context markers.

Cond.	Context marker	Set	w_i	μ_i	$w_i \times \mu_i$
C_1	people_in_room	*low*	0.30	1.00	0.300
C_2	network_activity_level	*high*	0.30	0.75	0.225
C_3	ambient_noise	*low*	0.20	0.17	0.034
C_4	door_open_events	*low*	0.20	0.50	0.100
			1.00		**0.659**

The resulting score of 0.659 exceeds the neutral threshold of 0.3, indicating substantial support for the assumed context type. Notably, the contribution of individual conditions varies considerably: the *people_in_room* marker contributes its full weight (0.300), whereas *ambient_noise* contributes only 0.034 due to its low membership degree of 0.17 in the *low* set. The weighted mean accommodates this variation by allowing strong markers to compensate for weaker ones, rather than collapsing the entire score to the weakest element.

If multiple context types (each with its own context assumption) exist, they are all evaluated and one score per context type is propagated to Step 4.

Design rationale: weighted mean as aggregation operator. The choice of the weighted mean for Step 3 over the alternative operators introduced in Table 2.2 is motivated by the following three requirements:

- Partial satisfaction should be reflected proportionally: a context type should be considered plausible when most of its markers align, even if one marker provides only marginal support.
- Markers differ in importance, and the operator should incorporate that domain knowledge through weights—an unweighted arithmetic mean would treat all markers as equally relevant.
- Each term $w_i \times \mu_i$ remains individually inspectable, preserving the traceability inherited from the FASL primitive (Section 2.2.3).

3.5.1.6 Step 4: Apply EMA Smoothing

Step 4 receives as input a single score per context type, quantifying how strongly the current sensor readings support that context. However, these scores are derived from sensor data that is inherently noisy: transient measurement artefacts, momentary environmental fluctuations, or brief anomalies in a single marker can cause the raw score to shift substantially between consecutive evaluations. If the pipeline were to select the dominant context directly from these raw scores, the result would oscillate with every minor fluctuation, propagating instability into the downstream weight adjustments of Steps 5 and 6. Step 4 addresses this by applying EMA smoothing to each context type's score sequence, producing temporally stabilised scores that reflect sustained trends rather than isolated readings.

As established in Section 2.2.4, the EMA computes a recursively weighted running average in which recent observations receive exponentially greater weight than older ones. Applied to the context evaluation pipeline, the smoothed score $\tilde{s}_t$ for a given context type at evaluation cycle t is

$$\tilde{s}_t = \alpha\, s_t + (1-\alpha)\, \tilde{s}_{t-1}, \tag{3.2}$$

where s_t denotes the raw weighted-mean score produced by Step 3 and $\tilde{s}_{t-1}$ is the smoothed score from the previous evaluation cycle. The smoothing factor α is exposed as a configurable parameter of the pipeline, since its optimal value depends on the characteristics of the deployment environment and should be tuned by a domain expert. Generally, systems with highly volatile sensor data benefit from lower α values that suppress noise more aggressively, whereas systems with stable but infrequent readings may require higher values to remain responsive to genuine shifts.

Running example. To illustrate the effect of EMA smoothing, Table 3.8 continues the running example from Step 3. Two effects are visible: First, the transient anomaly at cycle $t = 2$ does not destabilise the smoothed score as it decreases by only 0.073 (from 0.665 to 0.592). However, once a genuine shift starts at cycle $t = 4$, the smoothed score starts to follow (from 0.615 at $t = 3$ to 0.451 at $t = 5$). EMA does not suppress sustained change; it tracks it with a delay governed by $(1-\alpha)$.

Table 3.8: Example of applied EMA smoothing to the running example ($\alpha = 0.3$). Bold values indicate deviations from baseline.

Cycle t	Raw score s_t	EMA computation	Smoothed $\tilde{s}_t$
0	0.659	$\tilde{s}_0 = s_0$ (initialisation)	0.659
1	0.680	$0.3 \times 0.680 + 0.7 \times 0.659$	0.665
2	**0.420**	$0.3 \times 0.420 + 0.7 \times 0.665$	0.592
3	0.670	$0.3 \times 0.670 + 0.7 \times 0.592$	0.615
4	**0.310**	$0.3 \times 0.310 + 0.7 \times 0.615$	0.524
5	**0.280**	$0.3 \times 0.280 + 0.7 \times 0.524$	0.451

Design rationale: EMA over alternative smoothing methods. Two alternatives to simple exponential smoothing were considered for this pipeline stage.

The first alternative is the simple moving average (SMA), which by design assigns equal weight to every reading within the window and drops observations abruptly when

they exit (Section 2.2.4). In the context evaluation pipeline, this boundary discontinuity is particularly problematic: an unremarkable new reading can cause the smoothed score to jump simply because an old reading was discarded, potentially triggering a false context switch in Step 5. Additionally, the SMA requires choosing a window size n, introducing a second parameter that interacts non-trivially with the evaluation frequency.

The baseline alternative is to apply no smoothing at all, passing the raw weighted-mean scores directly to Step 5. However, without smoothing, the hysteresis and dwell-time mechanisms in Step 5 would need to absorb all temporal variability on their own, requiring either very high thresholds, which would delay detection of genuine shifts, or accepting frequent false switches.

Simple exponential smoothing avoids these limitations. It provides a single configurable parameter α, produces continuous output free of boundary effects, and is computationally minimal—properties that align with the pipeline's requirements for stability, responsiveness, and operational simplicity.

3.5.1.7 Step 5: Apply Hysteresis and Dwell Time

Step 4 produces temporally smoothed scores that dampen transient noise, yet smoothing alone does not prevent oscillation when two context types have similar scores near a decision boundary.

Each cycle, the context type with the marginally higher score would be declared dominant, causing the active context to flip back and forth despite neither score reflecting a convincing change in the underlying situation. Step 5 addresses this instability by imposing two complementary stabilisation gates: hysteresis and dwell time.

Hysteresis: the magnitude barrier. Hysteresis introduces a minimum score difference that a candidate context must exceed relative to the currently active context before a switch is considered. Let $\tilde{s}_{\text{cand}}$ denote the smoothed score of the candidate and $\tilde{s}_{\text{curr}}$ the smoothed score of the currently active context. The hysteresis gate is satisfied if and only if

$$\tilde{s}_{\text{cand}} - \tilde{s}_{\text{curr}} > h, \tag{3.3}$$

where $h \geq 0$ is the configurable hysteresis threshold (default 0.1).

Dwell time: the temporal barrier. Even after passing the hysteresis gate, a single evaluation cycle in which the candidate leads by a sufficient margin may reflect a transient spike rather than a genuine context shift. Dwell time addresses this by requiring the

candidate to sustain its advantage across d consecutive evaluation cycles (default $d = 2$). The pipeline maintains a counter c for each candidate context type. At each evaluation cycle, if the candidate passes the hysteresis gate, its counter is incremented; if it fails, the counter resets to 0. The switch is permitted when $c \geq d$.

Running example. Table 3.9 illustrates both mechanisms. At cycle $t = 2$, a genuine environmental shift causes the alternative context to exceed the hysteresis threshold of $h = 0.1$. However, the dwell counter below the required threshold of $d = 2$; the switch is blocked and the incumbent context persists. At cycle $t = 3$, the alternative context continues to exceed the hysteresis threshold, incrementing the dwell counter to $c = 2 = d$. Both gates are now satisfied, and the switch is committed.

Table 3.9: Example of hysteresis and dwell time applied to the running example.

t	$\tilde{s}_{\text{LNB}}$	$\tilde{s}_{\text{AC}}$	Diff.	Hysteresis	Dwell	Active
0	0.659	0.200	—	—	—	LNB
1	0.665	0.220	—	—	—	LNB
2	0.524	0.680	0.156	pass	$c = 1 < 2$	LNB
3	0.451	0.710	0.259	pass	$c = 2 = d$	AC

Design rationale: why both mechanisms are needed. Hysteresis and dwell time address complementary failure modes: hysteresis sets the bar for *how much* the candidate must lead, and dwell time verifies *for how long* that lead persists. Hysteresis alone prevents switches caused by marginal score differences but cannot distinguish a single strong spike from a sustained shift: a transient anomaly that momentarily exceeds the threshold by a wide margin would trigger an immediate, potentially incorrect switch. Dwell time alone prevents transient spikes but does not address the case in which two context types have genuinely similar scores: without a magnitude barrier, the context would swap repeatedly.

Parameter tuning. Both mechanisms must be tuned by domain experts to balance stability and responsiveness based on the characteristics of the deployment environment.

The dwell time mechanism must be considered together with the configured evaluation interval in Section 3.5.1.2. Because the dwell counter increments once per evaluation cycle, the real-world duration that a candidate must sustain its advantage equals d times

the evaluation interval. For example, with an interval of fifteen minutes, $d = 2$ requires the candidate to lead for at least thirty minutes before a context switch is committed.

3.5.1.8 Step 6: Produce Context Result

Step 6 concludes the context evaluation pipeline by storing the outcome into a *ContextResult*, as illustrated in Table 3.10. A ContextResult comprises two key elements: the *active context type* and a dictionary of *confidence scores* that maps every configured context type to its smoothed confidence value.

Table 3.10: Example of ContextResult from step 6 of the context evaluation pipeline. LNB denotes the *late-night browsing* context; AC denotes the alternative context.

		Confidence scores	
Timestamp	**Active context**	**LNB**	**AC**
22:00	LNB	0.665	0.200
22:15	AC	0.592	0.680
22:30	AC	0.451	0.710

The result is stored in the database and used in the analysis pipeline, see Section 3.5.2, to adjust the contribution of individual biomarkers based on the active context.

3.5.2 Analysis Pipeline

The thesis established in Section 3.1.2 that multimodal biomarkers enter the CML Mapping Pyramid in Layer 3 and are transformed into indicators for Layer 4. This subsection defines each step of the Analysis Pipeline, see Figure 3.7.

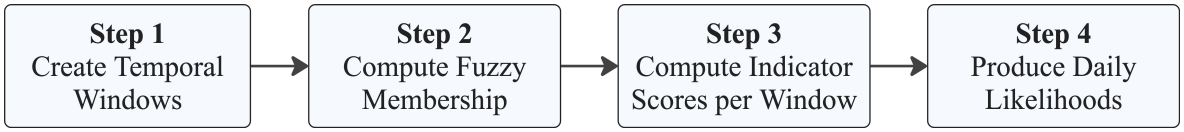


Figure 3.7: Overview of the four-step analysis pipeline.

The pipeline constitutes the concrete realisation of two design decisions established in the CML Framework (Section 3.1): late fusion at the indicator boundary, where biomarkers from multiple modalities converge into shared indicator scores, and context-adjusted weighting, where the active situational context influences the contribution of individual biomarkers.

Each stage is governed by configurable parameters externalised to the configuration system, enabling domain experts to adjust the analytical behaviour without modifying application code and ensuring that each analysis run can be reproduced under its exact parametrisation (Section 3.5.4). All intermediate results are recorded in the pipeline trace, preserving the end-to-end traceability chain required by the explainability principles established in Section 3.2.

3.5.2.1 Step 1: Create Temporal Windows

Sensor data from different modalities arrives at different rates and with different timestamps. For example, a speech-related sensor may produce a reading every time the user speaks, while a router-based network traffic sensor may produce readings at fixed intervals (e.g., every five minutes).

Before these heterogeneous streams can be compared or combined, they must be aligned to a common temporal reference. Within each window, multiple readings of the same biomarker are reduced to a single representative value through a configurable aggregation method. The arithmetic mean is selected as the default method due to its simplicity and interpretability.

Consequently, Step 1 groups all biomarker readings into fixed-size temporal *tumbling* windows, producing a single value per window. As a result, any given day is represented as a sequence of windows, each containing one value per biomarkers, which can be directly compared and combined in the subsequent steps. Table 3.11 illustrates a possible output.

Table 3.11: Example of result of Step 1 for one temporal window.

Biomarker	Type	Aggr. Value
whispering	speech	0.72
prolonged_pauses	speech	0.58
reduced_social_interaction	network	0.81
passive_media_binge	network	0.65

Selecting the right window size. The choice of window size represents a trade-off between temporal sensitivity and aggregation robustness, and is externalised to the configuration system (Section 3.5.4) so that domain experts can tune it without modifying application logic.

Smaller windows capture finer-grained temporal dynamics, such as a brief spike in speech rate during a stressful phone call, but risk containing too few readings per biomarker, particularly for modalities with low sampling rates. Larger windows provide greater statistical mass per aggregation and are more tolerant of irregular reporting intervals, but smooth over short-lived patterns that may be clinically relevant.

3.5.2.2 Step 2: Compute Fuzzy Membership

The output of Step 1 contains aggregated, but heterogenous, biomarker values per window. These biomarker values are not directly comparable across modalities: they span different physical units, different numeric ranges, and different per-user baselines. A value that is unremarkable for one individual may represent a significant deviation for another. Step 2 resolves this by transforming the values into *directed membership degrees* with values between $[0,1]$, which reflect *the level of clinical concern* associated with the observed value relative to the individual's baseline.

Stage (a): Z-score relative to user baseline. Given the user's baseline (what a typical value is for that biomarker for that user), the first stage computes a z-score that quantifies how much the observed value deviates from the user's norm. Consequently, the z-score answers the question: *How unusual is this value compared to this user's baseline?*

The z-score is computed using the standard formula:

$$z = \frac{x - \mu_b}{\sigma_b}, \tag{3.4}$$

where x is the aggregated biomarker value of a window from Step 1, and μ_b and σ_b are the user's personal baseline mean and standard deviation for that biomarker.

The user's baseline is established in an initial calibration phase, which produces parameters μ_b and σ_b for each biomarker.

Stage (b): Sigmoid membership mapping. The second stage maps the z-score to a membership degree $\mu \in [0,1]$, which answers the question: *How clinically concerning is this deviation?*

The membership degree is computed using the sigmoid membership function (Section 2.2.1):

$$\mu = \frac{1}{1 + e^{-z}}. \tag{3.5}$$

The output μ has a clear semantic interpretation: a value of $\mu = 0.5$ indicates that the biomarker is exactly at the user's baseline (neutral), values approaching $\mu = 1$ indicate

that the observed value lies well above the baseline, and values approaching $\mu = 0$ indicate that it lies well below.

The choice of sigmoid over alternative mapping functions is motivated by three considerations:

- *Generality.* The sigmoid generalises across heterogeneous biomarker types without requiring per-biomarker function design. Triangular functions, as employed by Nef [27], require individually tuned parameters (ℓ, m, h) for each metric. An approach that becomes impractical as the set of biomarkers grows with new modalities.
- *Semantic anchor.* At $z = 0$ (i.e., the user's baseline), the membership degree evaluates to $\mu = 0.5$, representing neutral clinical concern. This provides a meaningful midpoint for the entire pipeline.
- *Smoothness.* Linear clipping or piecewise mappings introduce discontinuities at their boundaries, whereas the sigmoid produces smooth, continuous output across the entire input range.

Directionality correction. Each biomarker carries a configured direction that indicates whether higher values signify increased clinical concern (*higher_is_worse*) or decreased concern (*lower_is_worse*). For biomarkers where lower values indicate greater severity, the membership degree is inverted:

$$\mu_{\text{dir}} = 1 - \mu \,. \tag{3.6}$$

After this correction, $\mu_{\text{dir}} \to 1$ always signals high clinical concern and $\mu_{\text{dir}} = 0.5$ signals baseline behaviour.

3.5.2.3 Step 3: Compute Indicator Scores per Window

After Step 2, each window contains directed membership degrees for individual biomarkers, but these per-biomarker scores do not yet correspond to the DSM-5 indicators that the system must ultimately assess. Step 3 defines the aggregation from biomarkers to indicators, and in doing so, serves as the concrete late-fusion point where biomarkers from different modalities converge into a single indicator score per window (Section 3.1.2).

Biomarker-to-indicator mapping. Which biomarkers contribute to which indicator, and with what base weight and direction, is entirely configurable (Section 3.5.4). Each

indicator declares its set of contributing biomarkers. Table 3.12 illustrates the mapping for a single indicator using the four biomarkers introduced in the Step 1 running example. Each row specifies a contributing biomarker, its modality origin, its base weight within the indicator, and the direction of clinical concern.

Table 3.12: Example biomarker-to-indicator mapping for indicator C1 (depressed mood).

Indicator	Biomarker	Modality	Base weight	Direction
C1: depressed mood	whispering	speech	0.30	higher_is_worse
	prolonged_pauses	speech	0.20	higher_is_worse
	reduced_social_interaction	network	0.30	higher_is_worse
	passive_media_binge	network	0.20	higher_is_worse

Aggregation. Building on the FASL primitive (Section 2.2.3), the indicator score for indicator k is computed as:

$$L_k = \frac{\sum_i \hat{w}_i \, \mu_{\text{dir},i}}{\sum_i \hat{w}_i}, \tag{3.7}$$

where $\mu_{\text{dir},i}$ is the directed membership degree of biomarker i from Step 2, $\hat{w}_i$ is its context-adjusted weight, and $L_k \in [0, 1]$ expresses how strongly the window's evidence supports indicator k.

Context-adjusted weight. To compute $\hat{w}_i$, the active context from Section 3.5.1 is applied to adjust the biomarker's configured base weight w_i:

$$\hat{c}_i = 1 + (c_i - 1) \times conf, \qquad \hat{w}_i = w_i \times \hat{c}_i, \tag{3.8}$$

where c_i is the context weight multiplier configured for biomarker i under the active context type, and $conf$ is the confidence score of the context result (Section 3.5.1.8).

In practice, when $conf = 1$, the full context weight is applied. When $conf = 0$, $\hat{w}_i$ reduces to w_i and the formula collapses to providing a non-context-adjusted result.

Multimodal fusion realisation. Step 3 constitutes the concrete late-fusion point established in Section 3.1.2: biomarkers from heterogeneous modalities converge into a single indicator score through Equation (3.7). Each modality contributes independently computed directed membership degrees; no cross-modal interaction occurs before this aggregation. The formula itself is modality-agnostic: it does not distinguish the origin of a biomarker. This realises the interchangeability property of the CML Mapping Pyramid:

adding a new modality requires only registering new biomarkers and their weights in the configuration; no modification to the aggregation logic is needed.

Running example. Table 3.13 traces the full computation for indicator C1 (depressed mood) across a single window, using the four biomarkers introduced in Step 1 and the context result from the context evaluation pipeline (LNB active, $conf = 0.665$).

Table 3.13: Example of computed indicator score for C1 (depressed mood) in one window. Context multipliers c_i are configured per biomarker under the active context type LNB.

Biomarker	Mod.	μ_{dir}	w_i	c_i	$\hat{w}_i$	$\hat{w}_i \times \mu_{\text{dir}}$
whispering	speech	0.82	0.30	1.20	0.340	0.279
prolonged_pauses	speech	0.64	0.20	1.00	0.200	0.128
reduced_social_interaction	network	0.88	0.30	0.80	0.260	0.229
passive_media_binge	network	0.73	0.20	1.10	0.213	0.156
				Sum	**1.013**	**0.792**

Applying Equation (3.7): $L_{\text{C1}} = 0.792 / 1.013 = 0.782$. The result is a single, interpretable score in $[0, 1]$ for every indicator and window that uses multimodal biomarker data and context-adjusted weighting.

3.5.2.4 Step 4: Produce Daily Likelihoods

The DSM-5 Gate (Section 3.5.3) operates on daily granularity, yet the preceding steps produce one indicator score per temporal window. Step 4 bridges this gap by aggregating window-level scores into a single daily likelihood $L_k(d) \in [0, 1]$ for each indicator k and day d.

The concept of daily aggregation as an intermediate between window-level scores and the gate logic originates in the work of Nef [27]. This thesis adopts the same bridging role but replaces the aggregation mechanism with a *consecutive-window peak mean* that is designed to reward sustained symptomatic periods over isolated spikes. Compared to a plain daily mean, the consecutive-window peak mean is more sensitive to symptomatic periods: a few hours of pronounced symptoms surrounded by an otherwise unremarkable day are still detected.

Consecutive-window peak mean. A single elevated window may reflect transient noise, whereas a sustained elevation over multiple consecutive windows is more likely to signal a genuine symptomatic period. To capture this, the daily likelihood is defined as the maximum mean of any k consecutive windows within the day:

$$L_k(d) = \max_j \frac{1}{k} \sum_{i=j}^{j+k-1} s_i \,, \tag{3.9}$$

where $s_1, \ldots, s_n$ are the chronologically ordered indicator scores from Step 3 for indicator k on day d, and k is configurable (Section 3.5.4).

Running example. Continuing the C1 (depressed mood) example from Step 3, assume a day with six windows produces the indicator scores shown in Table 3.14, and $k = 3$.

Table 3.14: Example of computed daily likelihood for C1 (depressed mood) with $k = 3$. The shaded group marks the consecutive triple with the highest mean.

Window	s_i
08:00	0.52
08:15	0.61
08:30	0.78
08:45	0.82
09:00	0.74
09:15	0.59

The sliding window of size $k = 3$ yields the consecutive means $\{0.637, 0.737, \mathbf{0.780}, 0.717\}$. Applying Equation (3.9): $L_{C1}(d) = 0.780$, reflecting the peak sustained period rather than the single highest window ($s_4 = 0.82$) or the overall daily mean (0.677).

Final output The final output of the analysis pipeline is one daily likelihood score $L_k(d) \in [0, 1]$ *per indicator and day*. The output is provided to the DSM-5 Gate, described next, which determines whether the multi-day pattern satisfies the diagnostic criteria for a depressive episode. Table 3.15 illustrates the structure for a single day.

Table 3.15: Exemplary result of the analysis pipeline.

Day	Indicator	$L_k(d)$
2025-04-01	C1: depressed mood	0.780
2025-04-01	C2: loss of interest	0.610
...	...	...

3.5.3 DSM-5 Gate

The analysis pipeline described in the preceding subsection produces a daily likelihood score $L_k(d) \in [0, 1]$ for each of the nine DSM-5 indicators. These scores express how strongly a given day's data points toward a particular symptom, but they do not yet answer the clinical question: is a depressive episode indicated? The *DSM-5 Gate* translates the clinical definition of a Major Depressive Episode into a computable, rule-based decision layer. To recap the definition [2]:

> *"Five (or more) of the following symptoms have been present [nearly every day] during the same 2-week period and represent a change from previous functioning; at least one of the symptoms is either (1) depressed mood or (2) loss of interest or pleasure."*

Adapted from the gate design introduced by Nef [27], the algorithm applies the diagnostic counting rule in two stages: first, it determines for each indicator whether it is *present* over a recent time window; second, it evaluates whether enough indicators are present, including at least one core symptom, to flag a depressive episode.

The gate logic and the formulas presented below are largely adopted from Nef's design as the DSM-Gate logic itself did not change [27]. Although the preceding pipelines differ substantially—notably through the addition of multimodality and contextual information—the rule-based logic to determine the presence of an episode continues to follow the definition of DSM-5.

Stage 1: Per-indicator presence. For each indicator k on each day d, the gate converts the continuous daily likelihood $L_k(d)$ into a binary presence flag by comparing it against a configurable threshold θ (default $\theta = 0.6$):

$$I_k(d) = \begin{cases} 1 & \text{if } L_k(d) \geq \theta, \\ 0 & \text{otherwise.} \end{cases} \tag{3.10}$$

A day on which $I_k(d) = 1$ is called a *positive day* for indicator k. To operationalise the DSM-5 phrase "nearly every day," the gate counts positive days over a sliding window of the last M days (default $M = 14$, matching the two-week criterion). If at least N of those days are positive (default $N = 10$), the indicator is considered *present*:

$$\text{present}(k) = \begin{cases} \text{true} & \text{if } \sum_{d=t-M+1}^{t} I_k(d) \geq N, \\ \text{false} & \text{otherwise.} \end{cases} \tag{3.11}$$

Figure 3.8 illustrates this two-step process of Stage 1: each cell represents one day within the sliding window, coloured according to whether the daily likelihood exceeded the threshold θ as formalised in Equation (3.10); the indicator is marked as present when the count of positive days reaches N as formalised in Equation (3.11).

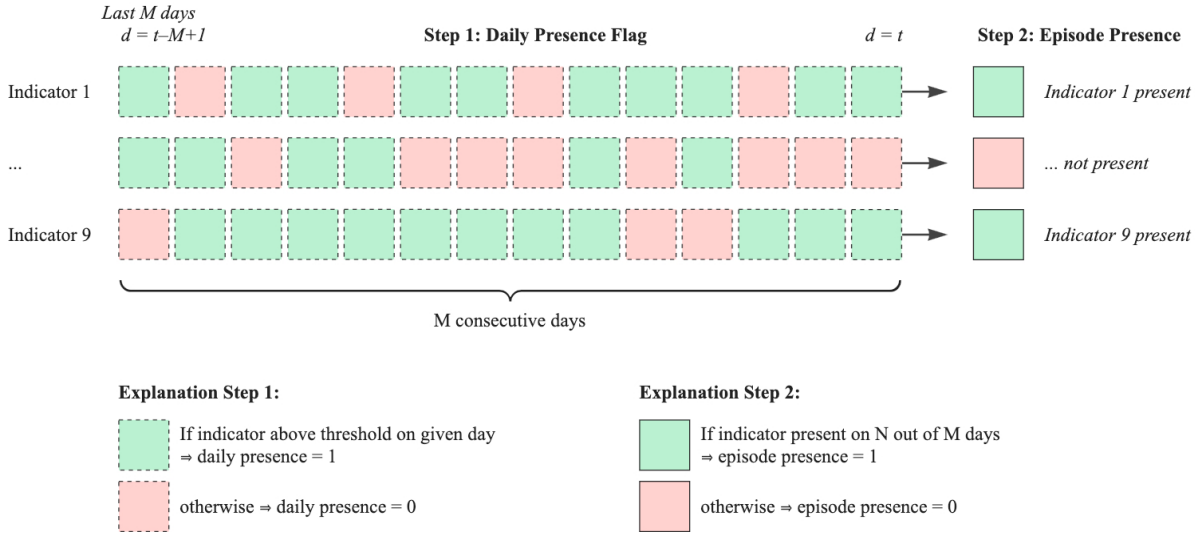


Figure 3.8: Illustration of the DSM gate logic for stage 1.

Stage 2: Episode decision. After evaluating all nine indicators, the gate aggregates the per-indicator presence flags into a single episode-level assessment.

Let $\mathscr{C}$ denote the set of core indicators as defined by the DSM-5:

$$\mathscr{C} = \{\text{C1: depressed mood}, \text{C2: loss of interest}\}.$$

The function episode(t) returns 1 when at least five of the nine indicators are present and at least one of them belongs to $\mathscr{C}$:

$$\text{episode}(t) = \begin{cases} 1, & \text{if } \sum_{k=1}^{9} \text{present}_k(t) \geq 5 \text{ and } \exists c \in \mathscr{C} : \text{present}_c(t) = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{3.12}$$

Configurable parameters. All gate parameters are externalised to the configuration summarized in Table 3.16.

Table 3.16: Configurable parameters of the DSM-5 Gate.

Symbol	Config key	Description
θ	`theta`	Daily likelihood threshold for indicator presence
M	`m_window`	Sliding-window size in days (DSM-5 two-week period)
N	`gate_need`	Required positive days within the window
—	`min_indicators`	Minimum number of present indicators for an episode decision
—	`core_indicators`	Set of core indicators (depressed mood, loss of interest)

3.5.4 Configuration

The preceding subsections introduced numerous parameters at different stages of the pipeline. A deliberate design decision is to *externalise* all such parameters into configuration rather than embedding them in source code. Three principles motivate this separation:

Separation of domain knowledge. The configuration constitutes a declarative encoding of clinical knowledge, meaning which biomarkers map to which indicators, how context modifies interpretation, what the DSM-5 diagnostic thresholds are. The pipeline code, by contrast, encodes computational procedures. Externalising the former from the latter ensures that changes to clinical understanding do not require changes to algorithmic code, and vice versa.

Accessibility. Clinicians or researchers can tune the system without requiring software development expertise. This aligns with the white-box, interpretable philosophy established in the framework extension: the analytical behaviour of the system should be transparent and adjustable.

Parametrisation. Because configuration is externalised and self-contained, each analysis run can be associated with its exact parametrisation. This enables both reproducibility through the possibility to retracte any result to the configuration that produced it, and

systematic experimentation, for example to assess how the system's output responds to parameter variation. This is especially useful for a research prototype whose parameters still lack empirically validated defaults.

All configurable parameters have been summarized in Appendix D. It includes configurable parameters for the principal components: Analysis Pipeline, Context Evaluator, DSM-5 Gate.

3.6 Dashboard

This section describes what the dashboard must offer at a design level and why each capability exists as a distinct concern. The dashboard serves two complementary purposes: first, to provide the interface through which users can work with the data; and second, to make the system's computational reasoning interpretable.

In practice, a user must be able to run the entire pipeline through the dashboard (from raw data to episode decision). Further, a user must be able to inspect any computational step, enabling interpretation without detailed knowledge of the underlying code. This constraint aligns with the system's approach to explainability, which is detailed in Section 3.2.

3.6.1 Capabilities

The dashboard is organised around five capabilities, each addressing a distinct design concern.

Experiment. Experiments allow for the same data to be analysed under different parameter sets without modification to the system, supporting iterative calibration and comparative evaluation.

Mock Data. The system is designed for longitudinal biomarker data from real users, but during development and evaluation, real clinical data is unavailable. A dedicated mock data facility ensures the system can be validated end-to-end against controlled, reproducible scenarios with known expected outcomes. Configurable settings, such as data interval; random seed; and selected modalities, allow users to construct targeted test cases that exercise specific aspects of the pipeline.

Context. Context evaluation is a multi-stage pipeline and its intermediate states must be exposed to align with the system's white box design principle which requires that

every intermediate step is inspectable. Users need to verify that the inferred context is plausible before trusting analysis results that depend on it.

Analysis. The analysis pipeline produces a single episode decision, but the clinical value lies in understanding how that decision was reached. The pipeline must expose its full computation chain because the system's white-box design principle requires that every intermediate step is inspectable. This is what distinguishes the system from black-box approaches identified in the literature (Section 2.1).

Data. Raw data is exposed as a validation mechanism independent of the pipeline. Users can verify that the data entering and being produced by the system matches expectations.

3.7 Requirements

This section specifies the functional and non-functional requirements that define the scope of the proof-of-concept implementation [16, 37]. The implementation of the requirements is described in Chapter 4 and their fulfilment is subsequently assessed in Chapter 5.

3.7.1 Functional Requirements

Table 3.17 summarises and describes the eight epics that guide the requirements of the architectural implementation. The complete list of functional requirements is provided in Section B.1.

Table 3.17: Epics that guide the architectural implementation.

Epic	Description	FRs
Epic 1: Infrastructure & Deployment	The system must be deployed an infrastructure compatible for a household environment.	FR-1–2
Epic 2: API	The system must provide a REST API for data ingestion and retrieval.	FR-3–14
Epic 3: Database & Persistence	The system must persist all ingested and computed data with configuration snapshots to enable reproducibility.	FR-15–22
Epic 4: Context Evaluation	The system must execute context evaluation for a specified user and time range.	FR-23–35

Epic	Description	FRs
Epic 5: Analysis Pipeline	The system must compute the episode likelihood for a specified user and time range.	FR-36–55
Epic 6: Experimentation	The system must enable systematic evaluation of different parameter configurations.	FR-56–59
Epic 7: Dashboard & Visualisation	The system must provide a web-based dashboard to manage key interactions with the system.	FR-60–84
Epic 8: Transparency & Reproducibility	The system must capture full pipeline computation traces.	FR-85–88

3.7.2 Non-Functional Requirements

The complete list of non-functional requirements is provided in Section B.2. Four non-functional requirements are highlighted as they directly align with the objectives of this thesis (see Table 1.1). First, the system must offer the *architectural flexibility* to integrate new data modalities (NFR-1), which is central property for the novel multimodal and context-aware extension pursued in Objectives 1 and 2. Second, the system must be *inherently explainable* (NFR-4), its results must be reproducible (NFR-5), and all pipeline decisions must be auditable (NFR-8). Those qualities collectively realise the white-box design goal of Objective 3.

Chapter 4

Implementation

This chapter describes how the design from Chapter 3 was realized as a working proof-of-concept prototype, addressing *Objective 4* in Table 1.1. The focus is on technically non-trivial decisions and implementation challenges that go beyond the design-level specification; readers are referred to Chapter 3 for the underlying algorithms and architectural rationale.

4.1 Technology Stack

The prototype implements the full analytical pipeline defined in Chapter 3. The source code is publicly available on GitHub [12]. Appendix A provides the full repository overview. When referencing implementation artifacts throughout this chapter, only the filename or entity name is mentioned; the repository README provides navigation details. Table 4.1 summarizes the principal technologies. The project is deployed using Docker Compose, with a FastAPI backend, PostgreSQL database, and Streamlit dashboard frontend. Core logic is built in Python, leveraging one language for the entire stack to simplify development and maintenance.

Table 4.1: Technology stack of the POC.

Role	Technology	Version	Key Rationale
Language	Python	3.11	Dominant language in data science; covers API, analysis, and dashboard with one runtime
Web Framework	FastAPI	0.109	Modern web framework for building APIs with Python
Database	PostgreSQL	15	Mature relational engine with native JSON column support
ORM	SQLAlchemy	2.0	Maps logical schema to Python model classes
Dashboard	Streamlit	1.41	Deliver an interactive frontend using the same language (Python) as the main code base
Orchestration	Docker Compose	—	Reproducible multi-service deployment

4.2 Data Persistence

Figure 4.1 shows the physical realization of the logical schema from Section 3.4.2, implemented as SQLAlchemy models in `models.py`. Eight tables are organized into three groups: input data (biomarkers, context markers, user baselines), context evaluation (evaluation runs and history records), and the analysis pipeline (analysis runs, computed indicators, configuration experiments). Two foreign-key relationships link computed results to the runs that produced them: each indicator references its `analysis_run`, and each context history record references its `context_evaluation_run`.

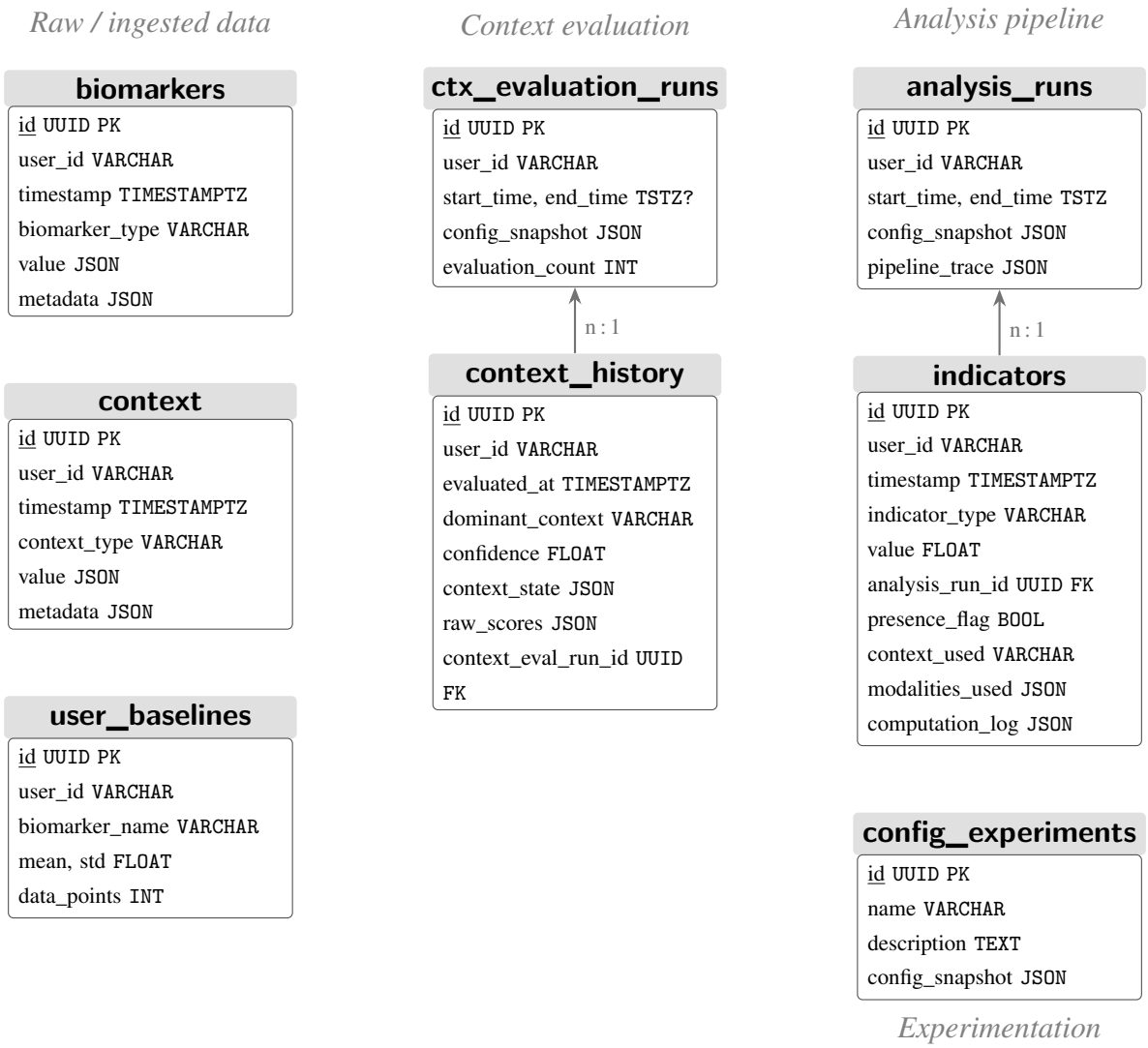


Figure 4.1: Overview of implemented database schema.

Relational-plus-JSON hybrid. Queryable fields such as `user_id`, `timestamp`, and `indicator_type` are stored as typed relational columns with composite indexes for temporal range queries. Variable-structure payloads, such as biomarker values that differ

per modality, full configuration snapshots, and per-step computation traces, are stored as PostgreSQL JSON columns. This avoids schema migrations whenever a payload structure changes, while still permitting relational joins and index-based filtering on the fixed columns. In practice, any new modality is supported out-of-the-box without database changes, as all modality-specific data is encapsulated in JSON blobs.

Run versioning for reproducibility. Both context and analysis runs store a full `config_snapshot` at creation time: a JSON serialization of the complete YAML configuration that governed the run. Reproducing a past result therefore requires no access to the current configuration files, satisfying the deterministic reproducibility principle from Section 3.2.1.

Three-level temporal granularity. The `indicators` table stores results at three temporal levels, distinguished by an `indicator_type` naming convention: per-window FASL scores (suffixed `_window`), daily aggregated summaries (plain indicator name), and episode-level gate decisions (the `presence_flag` column on the end-date row). Using a single table with a type discriminator avoids schema proliferation and enables querying at different temporal resolutions through simple filtering.

4.3 API Realization

The API contract from Section 3.4.1 was realized as FastAPI endpoints across three resource routers (`/biomarkers`, `/context`, `/indicators`) in `src/api/`. Pydantic schemas enforce request validation, and FastAPI dependency injection manages bearer-token authentication and database session lifecycle.

Modality-agnostic contract in Pydantic. The modality-agnostic data contract from Section 3.4.1 translates to a schema with a typed discriminator and an untyped payload. For biomarkers, `biomarker_type` is a `Literal["speech", "network"]` that identifies the modality, while `value` is a `dict[str, Any]` that accepts any JSON structure, because biomarker payloads differ fundamentally per modality and a fixed schema would require code changes for every new modality. Context schemas are even more open: `context_type` is an unconstrained string, reflecting the design principle that context sources are not predefined (Section 3.1.3).

As a result, adding a new modality or context source requires no changes to the API code. The submitting tool simply uses a new type value with its own payload structure.

The trade-off is that the API performs no validation of the `value` contents; structural correctness is the responsibility of the submitting tool.

Provenance in indicator responses. The read-only `GET /indicators` endpoint returns not just the computed numeric value but also provenance fields: `analysis_run_id` (linking to the run that produced it), `data_reliability_score` (a quality metric reflecting data completeness), and `config_snapshot` (the exact configuration used for the analysis). This means the API serves traceable results.

4.4 Configuration and Experimentation

The configuration externalises all pipeline parameters into declarative files. Each file targets one concern, as summarized in Table 4.2. The seven domain-specific YAML files in `config/` are composed into a single `AnalysisConfig` object.

Table 4.2: Configuration files used in the POC.

YAML File	Domain Concern
`indicators.yaml`	Biomarker-to-indicator weights and directions for all nine DSM-5 indicators
`context_weights.yaml`	Context-specific biomarker multipliers (e.g. dampened speech weights during solitary contexts)
`context_evaluation.yaml`	Fuzzy membership functions and context assumption rules
`ema.yaml`	Exponential moving average smoothing, hysteresis, and dwell-time parameters
`dsm_gate.yaml`	N-of-M rule: daily threshold θ, window length, required positive days
`episode.yaml`	Episode decision criteria: minimum indicator count and core symptom set
`reliability.yaml`	Data reliability scoring: coverage and quality weight balance

Validation at load time. Pydantic model validators enforce semantic constraints beyond type checking. For example, biomarker weights per indicator must sum to 1.0 within ± 0.001 (`IndicatorConfig`), context assumption condition weights are subject to the same constraint (`ContextAssumptionDef`), and the temporal window size is restricted to $\{5, 10, 15, 30, 60\}$ minutes. A `ConfigurationError` is raised on any violation, catching misconfiguration before pipeline execution rather than producing silent errors mid-computation.

Experiment management. When an analysis is triggered with an experiment, the experiment's configuration is loaded in place of the default, and the resulting run uses

the corresponding parameters. This decoupling allows multiple runs against the same data under different parameter sets. Each experiment record holds a name, an optional description, and a complete `AnalysisConfig` serialized as JSON.

4.5 Context Evaluation Engine

The six-step context evaluation pipeline from Section 3.5.1.2 is realized in the package `core/context/` and `membership.py` implements the fuzzy membership functions (Steps 1–2), `evaluator.py` orchestrates assumption scoring and context selection (Steps 3–6), and `smoother.py` encapsulates EMA smoothing with hysteresis and dwell-time gating (Steps 4–5). A fourth module, `history.py`, manages the run lifecycle. Algorithm 4.1 shows the orchestration flow as pseudocode.

Algorithm 4.1 Context evaluation pseudocode.

```
4.0.1  evaluate_context(user, start, end, config):
4.0.2    evaluator := ContextEvaluator(config)
4.0.3    IF prior history exists within staleness threshold
4.0.4      THEN initialize smoother from prior smoothed scores
4.0.5      ELSE reset smoother (cold start)
4.0.6
4.0.7    FOR EACH interval t IN [start, end] STEP eval_interval:
4.0.8      IF gap(t, last_eval) > staleness THEN reset smoother
4.0.9      snapshot := latest value per marker before t
4.0.10     IF snapshot = ∅ THEN CONTINUE
4.0.11     memberships  := evaluate linguistic sets per marker    - Steps 1-2
4.0.12     raw_scores   := weighted mean per assumption           - Step 3
4.0.13     smoothed     := EMA(raw, previous_smoothed)            - Step 4
4.0.14     candidate    := argmax(smoothed)
4.0.15     active       := apply hysteresis + dwell gates         - Step 5
4.0.16     persist(active, smoothed, raw, stabilization fields)   - Step 6
```

EMA state continuity across runs. The `ContextHistoryService` must seed the `EMASmoother` with prior state before processing a new time range; without this initialization, the first evaluations would produce unsmoothed scores that break temporal continuity. Before backfilling, `_initialize_smoother_state` queries the most recent `ContextHistoryRecord` and loads its smoothed scores and active context into the smoother. If no prior record exists or the gap exceeds the configurable staleness threshold, the smoother is reset to prevent outdated state from contaminating new evaluations. During iteration, the same staleness guard triggers a mid-range reset whenever a gap between consecutive intervals exceeds the threshold.

Stabilisation edge cases. Two decisions in `smoother.py` impact stabilisation behaviour. First, when a candidate context fails the hysteresis gate, its dwell counter is reset to zero rather than paused—the candidate must re-earn its full dwell count after any cycle where the margin drops below the threshold. Second, when the candidate equals the currently active context, the dwell counter is set to the required threshold immediately, avoiding a penalty for stability. Both decisions are captured in a frozen `SwitchDecision` dataclass whose transparency fields (`blocked_reason`, `dwell_progress`, `score_difference`) are persisted on each history record for dashboard drilldown (Section 4.8).

4.6 Analysis Pipeline

The analysis pipeline from Section 3.5.2 is implemented in `core/analysis.py`. Algorithm 4.2 shows the execution flow as pseudocode; including the major code-level steps. Notably, Steps 1—3 and 9—10 are infrastructure code. The core logic is realized in Steps 4—7. The DSM-5 gate is implemented in Step 8.

Algorithm 4.2 Analysis pipeline pseudocode.

```
run_analysis(user, start, end, baseline, config):
  Step 1   run_id := new UUID; tracer := new PipelineTracer(run_id)
  Step 2   IF context_run_id THEN load run; neutral for uncovered dates
           ELSE auto-generate context evaluations for [start, end]
  Step 3   biomarkers[] := query records for user in [start, end]

  Step 4   FOR EACH biomarker_name:
             FOR EACH window IN tumbling_windows(size_minutes):
               windows[biomarker_name] := aggregate(readings)

  FOR EACH indicator IN config.indicators:
    Step 5   FOR EACH biomarker IN indicator.biomarkers:
               (μ_b, σ_b) := baseline[biomarker]
               FOR EACH w IN windows[biomarker]:
                 z := (value - μ_b) / max(σ_b, min_std)
                 μ := sigmoid(z); ctx := lookup_context(w)

    Step 6   FOR EACH unique window:
               apply missing_strategy(present, missing biomarkers)
               ŵ_b := weight_b × (1 + (ctx_weight_b - 1) × conf_b)
               score := Σ(ŵ_b · μ_dir,b) / Σ(ŵ_b)

    Step 7   FOR EACH date d: likelihood := max consecutive mean of k windows

  Step 8   FOR EACH indicator: present := Σ(days ≥ θ) ≥ N of M
           episode := count(present) ≥ 5 AND core indicator present
  Step 9   persist indicators, daily summaries, run metadata
  Step 10  serialize tracer → JSON → pipeline_trace column
```

Processor decomposition. The implementation is realised as four independent processor modules from `core/processors/`: `window_aggregator` creates temporal windows (Step 1), `window_membership` computes per-biomarker membership scores (Step 2), `window_fasl` aggregates context-weighted indicator scores (Step 3), and daily summaries are produced by the `daily_aggregator` (Step 4). The orchestrator in `analysis.py` calls them as needed and each processor exchanges frozen dataclasses to prevent accidental mutation across step boundaries and support traceability.

Context temporal binding and deferred weighting. Each window aggregate retains the original reading timestamps so that the membership processor can look up the dominant context at evaluation time; windows without context coverage fall back to a neutral weight of 1.0. Context weights are deliberately *not* applied during membership computation (Step 5) but deferred to the FASL aggregation (Step 6). Applying them earlier would interact incorrectly with the directionality correction for `lower_is_worse` biomarkers, where the inversion $\mu_{\text{dir}} = 1 - \mu$ would operate on an already-weighted value.

Missing biomarkers and numerical guards. Step 6 applies a configurable missing-biomarker strategy when a window lacks readings for one or more expected biomarkers: `neutral_fill` substitutes a membership of 0.5 with a context weight of 1.0; `partial_fasl` renormalises the denominator over only the present biomarkers; and `skip_window` discards incomplete windows entirely. Two numerical guards complement this handling: the z-score computation in Step 5 applies a configurable floor (`min_std_deviation`, default 0.1) to the baseline standard deviation to prevent division by zero, and the DSM-5 gate in Step 8 maps NaN or infinity values to 0.0.

4.7 Explainability Realization

This section describes how the explainability principles from Section 3.2 are realized in code.

Trace capture. A `PipelineTracer` instance is created at the start of every analysis run. Each of the ten logical pipeline steps is bracketed by `start_step` and `end_step` calls that record the step's inputs, outputs, execution duration, and optional metadata into a frozen `PipelineStep` dataclass. Table 4.3 lists the ten logical steps and summarizes what each one captures. Upon completion, the tracer assembles a `PipelineTrace` object, which is itself a frozen dataclass, which is serialized to JSON and stored in the

`pipeline_trace` column of the `AnalysisRun` record. Because all trace dataclasses are frozen, recorded evidence cannot be mutated after capture.

Table 4.3: Captured data traces per pipeline step.

#	Step Name	Traced Inputs	Traced Outputs
1	Run ID Generation	—	Analysis run UUID
2	Context History	User, time range	Context source, coverage ratio, gaps
3	Read Data	User, time range	Biomarker count, context count
4	Window Aggregation	Window size, aggregation method	Window count, per-biomarker statistics
5	Membership Computation	Indicator name, context strategy	Membership count, z-score and weight statistics
6	Window FASL	Indicator name, missing strategy	Window indicator count, score distribution, peak window
7	Daily Aggregation	Indicator name, window indicator count	Daily summaries with likelihood and coverage
8	Episode Decision	Daily summaries, indicator list	Episode flag, rationale, per-indicator gate results
9	Persist Results	Run ID, summary count	Records saved confirmation
10	Save Trace	—	Trace stored in JSONB

Trace fragment. Listing 4.1 shows a trace fragment from Step 8, which produces a human-readable rationale for the episode decision to provide a self-contained summary. The rationale is stored as an immutable field on the `EpisodeDecision` dataclass and persisted inside the pipeline trace; the dashboard renders it directly as part of the transparency drilldown described in Section 4.8.

Listing 4.1: Trace fragment for episode decision.

```
{
  "step_name": "Episode Decision",
  "step_number": 30,
  "duration_ms": 2,
  "inputs": {
    "daily_summaries_count": 126,
    "indicators": ["depressed_mood", "sleep_disturbance", ...]
  },
  "outputs": {
```

```
    "episode_likely": true,
    "indicators_present": 6,
    "min_indicators_required": 5,
    "decision_rationale": "6 of 9 indicators present
        (>=5 required: MET); Core indicator(s) present:
        depressed_mood; => Episode criteria MET",
    "gate_results": {
      "depressed_mood": {"presence_flag": true,
          "days_above_threshold": 12, ...},
      "sleep_disturbance": {"presence_flag": true, ...}
    }
  }
}
```

4.8 Dashboard

The five dashboard capabilities defined in Section 3.6.1 are realized as a Streamlit multi-page application in `src/dashboard/`. Six pages have been implemented as listed in Table 4.4. Appendix C provides screenshots of the entire dashboard; this section focuses on the technical realization and non-trivial implementation aspects.

Table 4.4: Dashboard pages and their responsibilities.

Page File	Description
`1_Analysis.py`	Triggers analysis runs, displays episode decisions, and exposes the full indicator computation chain
`2_Context.py`	Visualizes context evaluation history, membership functions, and stabilization state
`3_Experiment.py`	Creates and edits configuration experiments for comparative parameter tuning
`4_Generate_Mock_Data.py`	Generates reproducible synthetic biomarker and context time series from scenario presets
`5_Data.py`	Provides tabular access to raw biomarkers, context records, indicators, and timeline charts

Figure 4.2 illustrates the high-level, typical user flow logic. Three user workflows (generating mock data, evaluating context, and running the analysis) each allow parameter configuration before execution and persist their results to PostgreSQL. The results view reads from the database and branches into three modes: a results overview, a pipeline transparency drilldown, and a side-by-side comparison of two analysis runs.

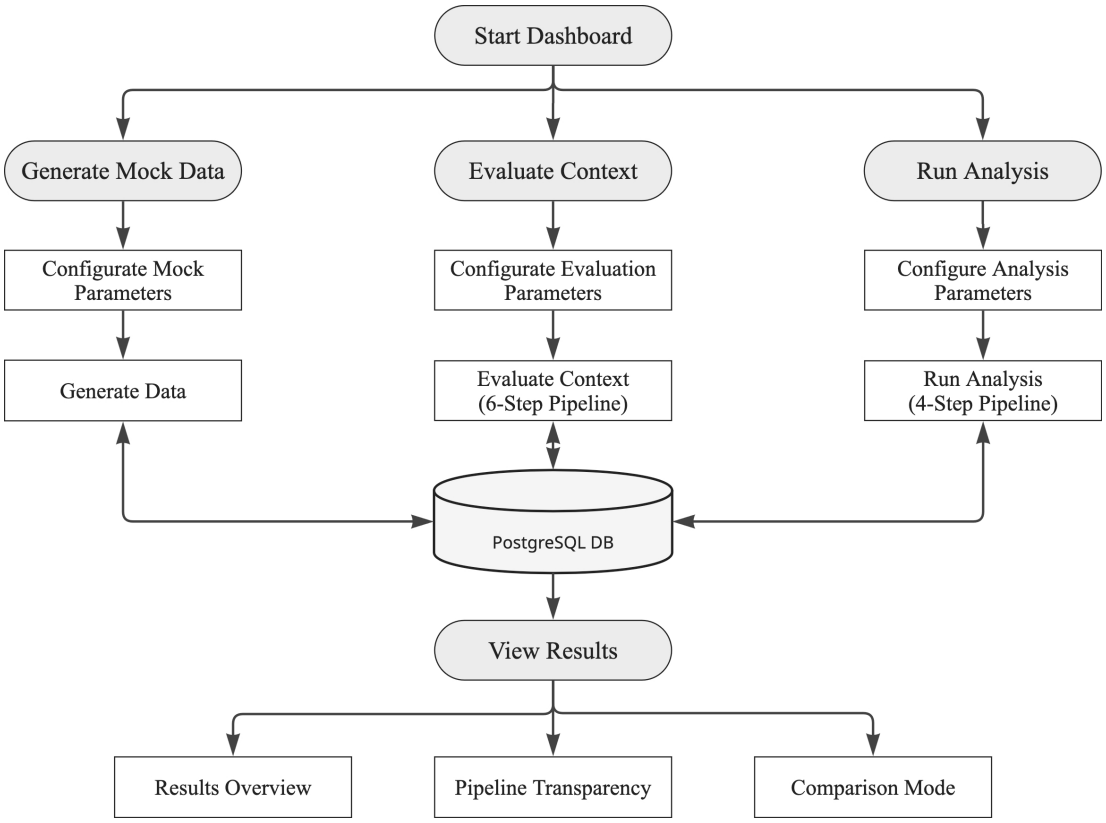


Figure 4.2: Process logic of the dashboard layer, showing the three primary user actions and their data flow through PostgreSQL to the results view.

Data access. Every data access runs through dedicated query modules, as implemented in `dashboard/data/`, which issue SQLAlchemy queries against the shared PostgreSQL database. The dashboard never calls the Linkage API; it relies exclusively on the Database Boundary defined in Section 3.3—the same decoupling mechanism that separates all internal components. Because the dashboard performs only read operations and runs within the same Docker network as the database, this direct query path avoids an unnecessary round-trip through the FastAPI layer while preserving the architectural separation.

Pipeline transparency. The dashboard makes the computational traces explicit to the user and provides dedicated transparency and accessible explainability. Both consume the JSON trace format persisted with each run. For each step of the pipeline, the user can drill into the steps relevant to the question at hand. Detailed screenshots are available in Appendix C.

Chapter 5

Evaluation

This chapter starts by evaluating the proof-of-concept prototype against the 88 functional requirements and nine non-functional requirements specified in Section 3.7. Within the DSR evaluation strategy framework of Venable et al. [40], the evaluation is positioned in the *ex post, artificial* quadrant: the artifact is assessed after construction, in a controlled setting with simulated data. This positioning reflects the absence of clinical access for naturalistic evaluation, ethical constraints that preclude real patient data, and the formative character of the evaluation, which targets functional correctness and system behaviour rather than clinical effectiveness. Following Hevner et al. [15], the evaluation addresses four artifact properties: *utility*, *efficacy*, *completeness*, and *consistency*.

Section 5.1 opens by establishing the lack of *determinism* for state-of-the-art LLM-based approaches, framing the proposed system's value. Section 5.2 presents the functional and non-functional assessment. Section 5.3 demonstrates system behaviour through predefined test scenarios. Section 5.5 discusses limitations, and Section 5.4 opens the discussion.

5.1 Determinism Comparison with LLM-based Approaches

This section empirically verifies the deterministic reproducibility principle established in Section 3.2.1 and contrasts it with the behaviour of state-of-the-art large language models (LLMs) on the same task.

5.1.1 Experimental Setup

The test scenario is generated with a fixed seed, producing 2,688 biomarker records and 336 context marker records (Section E.3). The prototype is executed five times via an automated verification script (Section E.1). Two LLMs, Claude Opus 4.6 (Extended Thinking) and ChatGPT 5.4 (Thinking), are each prompted five times with the same data and instructions (Section E.2). Both LLMs received the raw data and a finite list of

possible outputs (three context types, nine indicators) but no mapping between inputs and outputs, and were tasked with determining the active context and present indicators per 15-minute window.

5.1.2 Results

Across five runs, the prototype produced identical results at every pipeline level: context evaluation, window-level and daily indicator scores, and episode decision. However, neither LLM achieved full agreement: Table 5.1a and Table 5.1b show the unanimous agreement rates for context and indicator evaluation, respectively. The full LLM outputs are listed in Section E.2.

Table 5.1: Agreement rates across five runs for the prototype, Claude Opus 4.6, and ChatGPT 5.4: (a) context evaluation; (b) indicator evaluation.

(a) Context evaluation.

System	Unanimous	Agreement
Prototype	1,344	100.0 %
Claude Opus 4.6	989	73.6 %
ChatGPT 5.4	720	53.6 %

(b) Indicator evaluation.

System	Unanimous	Agreement
Prototype	12,096	100.0 %
Claude Opus 4.6	11,112	91.9 %
ChatGPT 5.4	6,286	52.0 %

5.2 System Architecture

This section assesses the prototype against the requirements specified in Section 3.7. Section 5.2.1 verifies each of the 88 functional requirements through requirement-level inspection, while Section 5.2.2 evaluates the nine non-functional requirements through design review, empirical measurement, and scenario demonstration.

5.2.1 Functional Assessment

Table 5.2 summarises the results for each of the 88 functional requirements defined in Section 3.7.1, grouped by epic. The complete requirement-level assessment with implementation notes is provided in Appendix B.

Table 5.2: Summarized assessment of all functional requirements, grouped by epic.

Epic	Total	✓	×
Epic 1: Infrastructure & Deployment	2	2	0
Epic 2: API	12	11	1
Epic 3: Database & Persistence	8	8	0
Epic 4: Context Evaluation	13	13	0
Epic 5: Analysis Pipeline	20	20	0
Epic 6: Experimentation	4	4	0
Epic 7: Dashboard & Visualisation	25	24	1
Epic 8: Transparency & Reproducibility	4	4	0
Total	**88**	**86**	**2**

Of the 88 requirements, 86 are fully implemented. The two requirements not implemented were explicitly excluded during scope refinement. FR-14 specified Model Context Protocol (MCP) exposure for LLM-based agent queries; this was determined to offer limited value for the current proof-of-concept objectives and can be added as a future API extension. FR-80 specified dynamic baseline computation from historical biomarker data; the database model for per-user baselines exists (FR-21), but the computation mechanism was deferred in favour of predefined baseline files and population defaults, which are sufficient for experimental evaluation.

5.2.2 Non-functional Assessment

Table 5.3 summarises the assessment for each of the nine non-functional requirements defined in Section 3.7.2. The full requirement descriptions are provided in Section B.2.

Table 5.3: Summarized assessment of all nine non-functional requirements.

NFR	Title	Assessment Method	Result
NFR-1	Architectural Flexibility	Design review	✓
NFR-2	Processing Efficiency	Design review	✓
NFR-3	Operational Simplicity	Design review	✓
NFR-4	Explainability	Scenario demonstration	✓
NFR-5	Reproducibility	Empirical verification	✓
NFR-6	Privacy	Architectural inspection	✓

NFR	Title	Assessment Method	Result
NFR-7	Maintainability	Design review	✓
NFR-8	Auditability	Scenario demonstration	✓
NFR-9	Robustness	Scenario demonstration	✓

NFR-1 Architectural Flexibility. The system follows a configuration-driven design in which biomarker definitions, indicator mappings, context types, and pipeline parameters are specified in YAML files. Adding a new biomarker requires only an entry in `indicators.yaml` (weight and direction), adding a new context type requires entries in `context_evaluation.yaml` and `context_weights.yaml`, and adjusting pipeline thresholds requires changes to the respective configuration files. None of these modifications require changes to the application code. Processing components such as membership functions, allowing additional methods to be added with minimal code changes. The codebase separates the API layer, the core analysis engine, and the persistence layer into distinct packages (`src/api`, `src/core`, `src/shared`), with data contracts defined through Pydantic models and dataclasses that decouple the layers from one another, making each layer independently extensible.

NFR-2 Processing Efficiency. All services run within Docker containers and the analysis pipeline operates without cloud-based computation dependencies. To verify the feasibility of running the system on a household device, the test scenario from Appendix E was executed; including 2,688 biomarker and 336 context records over 14 days. The full pipeline was executed five times on two devices. Table 5.4 reports the mean execution times.

Table 5.4: Mean pipeline execution time across five runs for a 14-day scenario (2,688 biomarker and 336 context records).

Device	Specification	Context	Analysis	Total
Device A	MacBook Pro M3, 18 GB RAM	1.43 s	121.71 s	123.14 s (∼2 min)
Device B	Raspberry Pi 4 Model B, 8 GB RAM	16.16 s	1375.32 s	1391.48 s (∼23 min)

Even on a Raspberry Pi 4 with 8 GB of RAM, the complete scenario executes in under 25 minutes, indicating that the pipeline is viable on low-cost household hardware for periodic assessments (e.g., daily summaries) without cloud dependencies. Notably,

the analysis pipeline dominates total execution time due to the computational cost of computing indicator scores for every window (Section 3.5.2.3). At 15-minute granularity and nine indicators, the system computes 864 indicator scores per day. For a production-ready system, improvements such as parallel computation and batch context resolution could be considered to optimise execution time.

NFR-3 Operational Simplicity. A single `docker-compose up` command deploys the complete platform, including the database, API, and dashboard. Configurations can be set via the user-facing interface and require no coding knowledge. New modalities are supported out-of-the-box without code changes thanks to the API's data-agnostic design.

NFR-4 Explainability. The system provides inherent white-box interpretability: every output is traceable to defined mappings between features and DSM-5 indicators. This is demonstrated through the scenario walkthroughs in Section 5.3, which trace each pipeline step from raw input to episode decision.

NFR-5 Reproducibility. Deterministic execution is empirically verified in Section 5.1, where five identical runs produced bit-identical results at every pipeline level. All pipeline runs are versioned with configuration snapshots, enabling exact re-execution.

NFR-6 Privacy. All data processing occurs locally within the Docker deployment. The system makes no external network calls and stores all data in the local PostgreSQL instance. The separation between model logic and data access layers supports auditable data governance.

NFR-7 Maintainability. The codebase follows a layered architecture with clear separation between API, service, and persistence layers. Rule definitions and evaluation procedures are externalised into configuration files, and the system is documented to support continued research beyond this thesis.

NFR-8 Auditability. The system captures complete computation traces with step-level inputs, outputs, execution duration, and metadata. Traces are serialised to JSON and persisted alongside each analysis run, enabling after-the-fact review as demonstrated in Section 5.3.

NFR-9 Robustness. The system implements configurable dropout strategies and minimum reading thresholds to handle incomplete or sparse input data. Graceful degradation under sensor failure conditions is demonstrated in Section 5.3.5.

5.3 System Behaviour

This section assesses system behaviour through four representative scenarios: end-to-end pipeline execution under a controlled context, behaviour under adversarial contexts, sensitivity to context changes, and robustness under degraded data conditions.

5.3.1 Scenario Setup

All scenarios operate on synthetic data generated by a seeded mock-data pipeline over a 14-day observation period (2,688 biomarker and 336 context marker records). Unless stated otherwise, each scenario inherits the configuration baseline of Scenario 1 (context assumptions, context weight profiles, indicator–biomarker mappings, user baseline, and DSM-5 gate parameters) and introduces only the scenario-specific deviation. The complete parameter set and the mock data generation procedure are documented with their reasoning in Appendix E.

5.3.2 Scenario 1: Solitary Digital Context

The first test scenario models a *young adult who begins extensively browsing social media alone in bed in a repeating pattern of two consecutive nights followed by one night off*, henceforth denoted as `solitary_digital`. Two configuration decisions shape the expected results: as the scenario is based on extensive social media browsing, the mock data primarily generates abnormal values for network biomarkers; context weights dampen speech and amplify network biomarkers accordingly.

Context Result. Figure 5.1 shows the context confidence scores over the 14-day period. The system correctly identifies `solitary_digital` as the dominant context during scenario nights, with confidence rising sharply at 20:00 and returning to `neutral` by midnight. On off-nights, confidence remains at baseline throughout. Visually, the scenario pattern shows the expected two nights on / one night off pattern. The EMA smoothing produces gradual transitions rather than hard switches at period boundaries, while the hysteresis threshold prevents brief fluctuations in marker values from triggering spurious context changes.

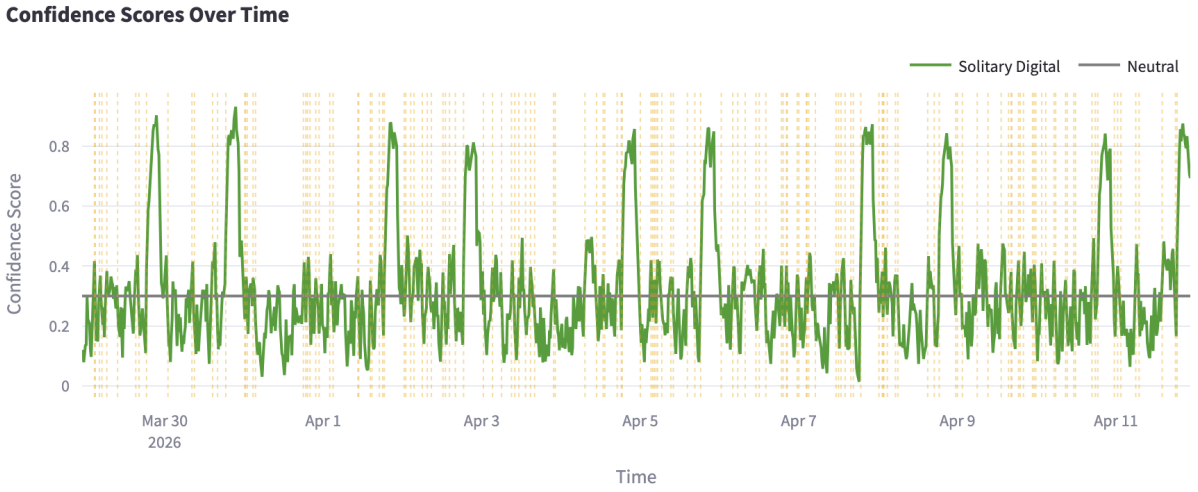
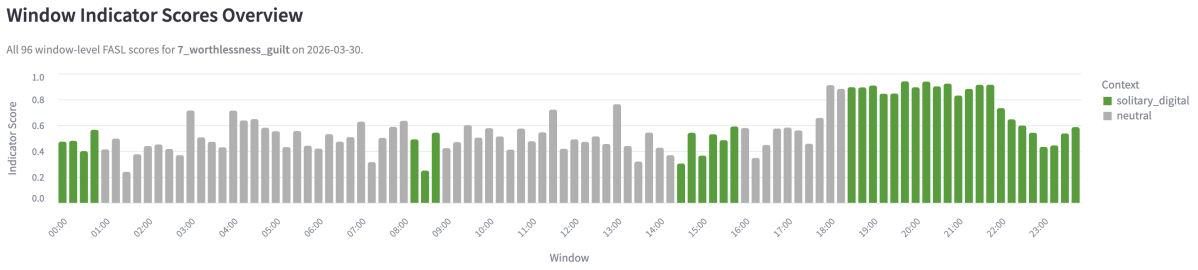
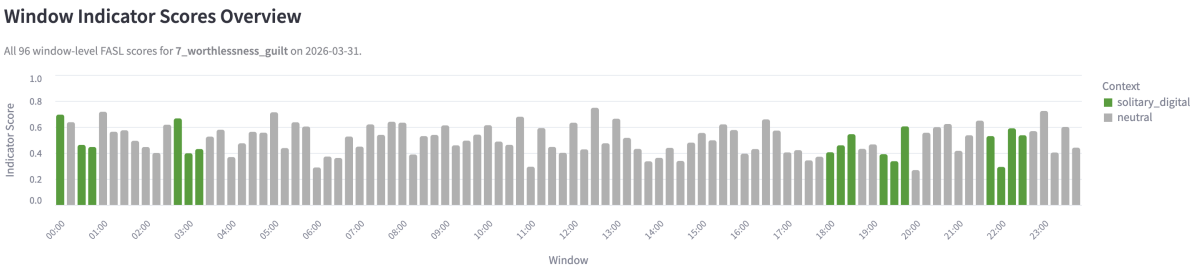


Figure 5.1: Context confidence scores over 14 days for the `solitary_digital` scenario. Peaks correspond to scenario-active evenings (two nights on / one night off).

Window-Level Indicator Likelihood. To illustrate the results, Figure 5.2 compares the window-level FASL scores for `7_worthlessness_guilt` on a scenario-active day and an off-day, respectively. On the active day (Figure 5.2a), daytime scores remain moderate, but evening windows evaluated under `solitary_digital` produce a distinct cluster of elevated values.



(a) Scenario-active day (2026-03-30).



(b) Off-day (2026-03-31); all windows evaluated under neutral weights.

Figure 5.2: Window-level FASL scores for `7_worthlessness_guilt`: (a) scenario-active day; (b) off-day.

On the off-day (Figure 5.2b), all windows remain under neutral weights and scores fluctuate around a moderate baseline throughout. The behaviour matches the expected pattern based on mock data and context evaluation from Figure 5.1.

Indicator Scores. Figure 5.3 presents the daily indicator likelihoods over the 14-day period. The system correctly classifies indicators as present, above the threshold at 0.6, in line with the scenario mock data. `7_worthlessness_guilt` scores highest at 73.9%, because it maps exclusively to network biomarkers, all of which receive the amplified context weight. `9_suicidality` remains below the presence threshold at 55.3% with zero days above threshold, confirming that the deliberately unamplified biomarkers produce the expected per-indicator contrast.

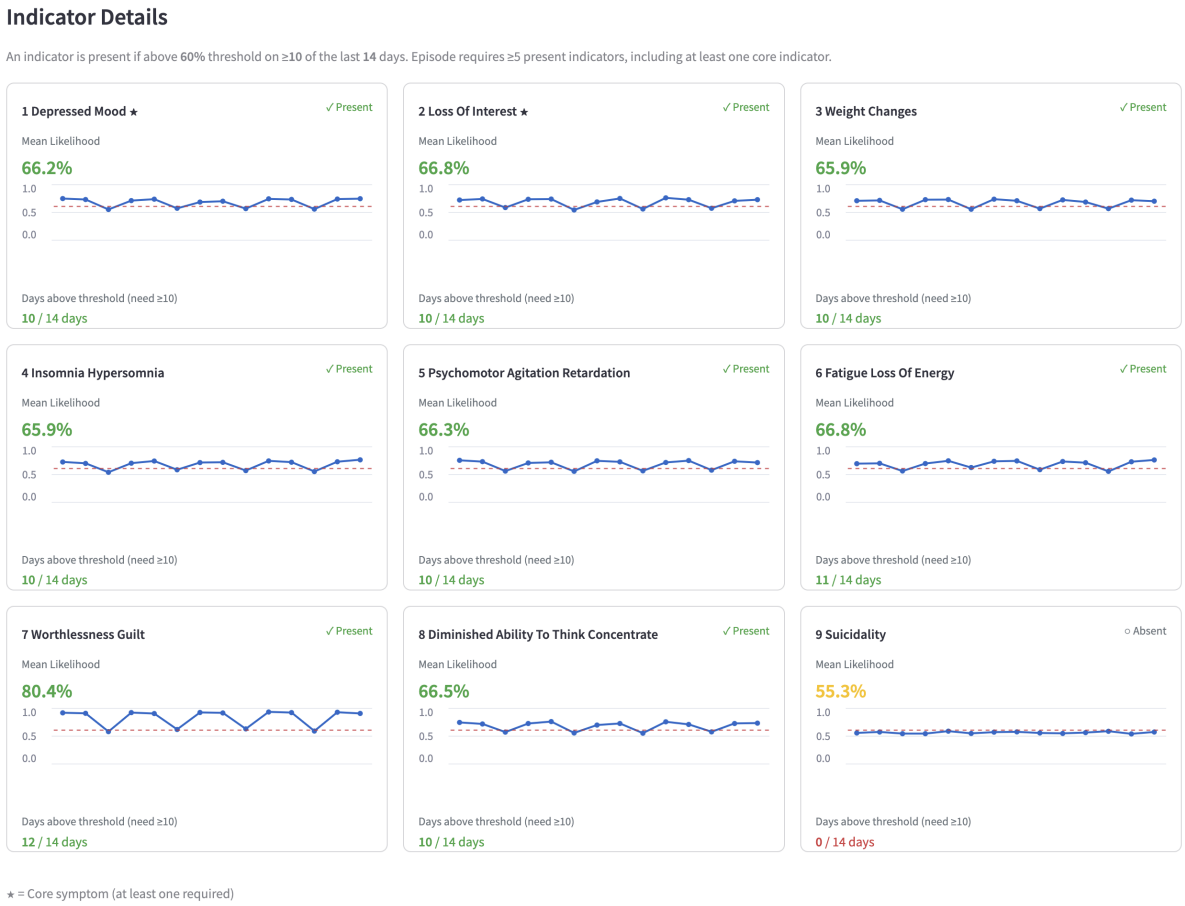


Figure 5.3: Dashboard shows daily indicator likelihoods for the `solitary_digital` scenario.

Episode Decision. The DSM-5 gate produces the final assessment shown in Figure 5.4. The system concludes *episode likely*: eight of nine indicators exceed the presence threshold, satisfying the minimum of five; both core symptoms (`1_depressed_mood` and

`2_loss_of_interest`) are present; and `9_suicidality` is absent, consistent with the deliberately omitted biomarkers. This outcome confirms that the pipeline correctly propagates the synthetic scenario (designed to exhibit a depressive episode pattern) through all processing stages to the expected conclusion.

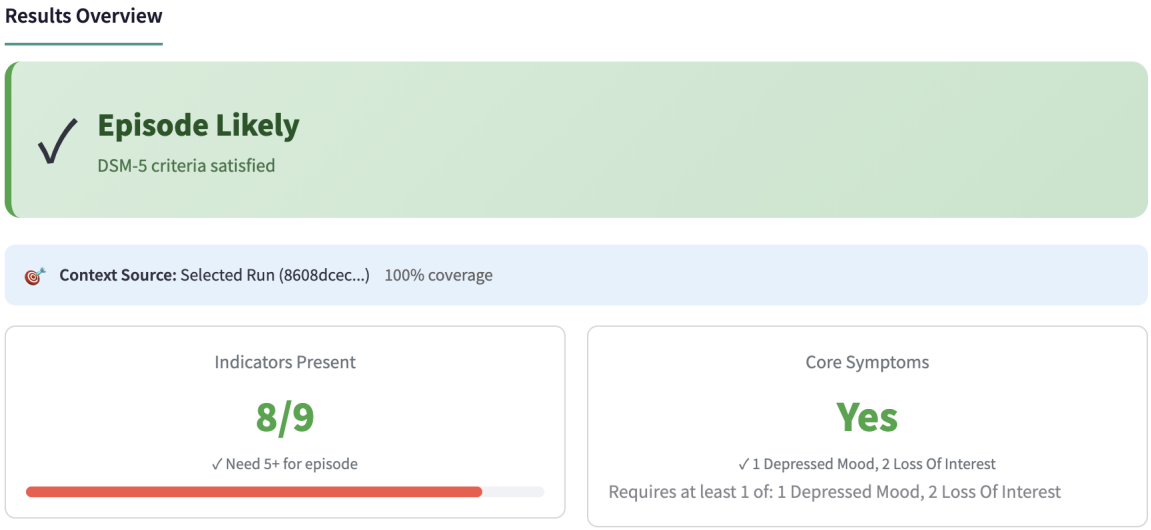


Figure 5.4: Dashboard shows the DSM-5 gate output for the `solitary_digital` scenario.

The scenario confirms that the system behaves correctly under the defined conditions and configurations, with all processing stages producing outputs consistent with the synthetic data and expected patterns. Thanks to the deterministic property of the system, the scenario can be reproduced using the configuration in Appendix E. The following scenarios will demonstrate how changes in context and data quality affect the system behaviour.

5.3.3 Scenario 2: Adversarial Context

The second scenario evaluates the system's behaviour for determining the dominant context under an adversarial context, which models a *young adult alone in room playing games online with friends*, denoted as `adversarial_social_digital_gaming`. Figure 5.5 contrasts the context assumptions defined for `solitary_digital` from scenario 1 and `adversarial_social_digital_gaming`. The difference being that `ambient_noise` is set to *high* for the adversarial context.

```
solitary_digital:
  conditions:
    people_in_room:
      set: low,      weight: 0.3
    network_activity_level:
      set: high,     weight: 0.3
    door_open_events:
      set: low,      weight: 0.1
    ambient_noise:
      set: low,      weight: 0.3
  operator: WEIGHTED_MEAN
```

(a) `solitary_digital`

```
adversarial_social_digital_gaming:
  conditions:
    people_in_room:
      set: low,      weight: 0.3
    network_activity_level:
      set: high,     weight: 0.3
    door_open_events:
      set: low,      weight: 0.1
    ambient_noise:
      set: high,     weight: 0.3
  operator: WEIGHTED_MEAN
```

(b) `adversarial_social_digital_gaming`

Figure 5.5: Configuration of contexts for main (a) and adversarial (b) scenarios; the sole difference is the expected linguistic set of `ambient_noise` (`low` vs. `high`).

Context evaluation is entirely dependent on the available sensors and a good configuration from domain experts. By changing the mocked sensor data for `ambient_noise` from *low* to *high*, the adversarial scenario takes over and classifies the context as `adversarial_social_digital_gaming` instead of `solitary_digital`, as seen in Figure 5.6.

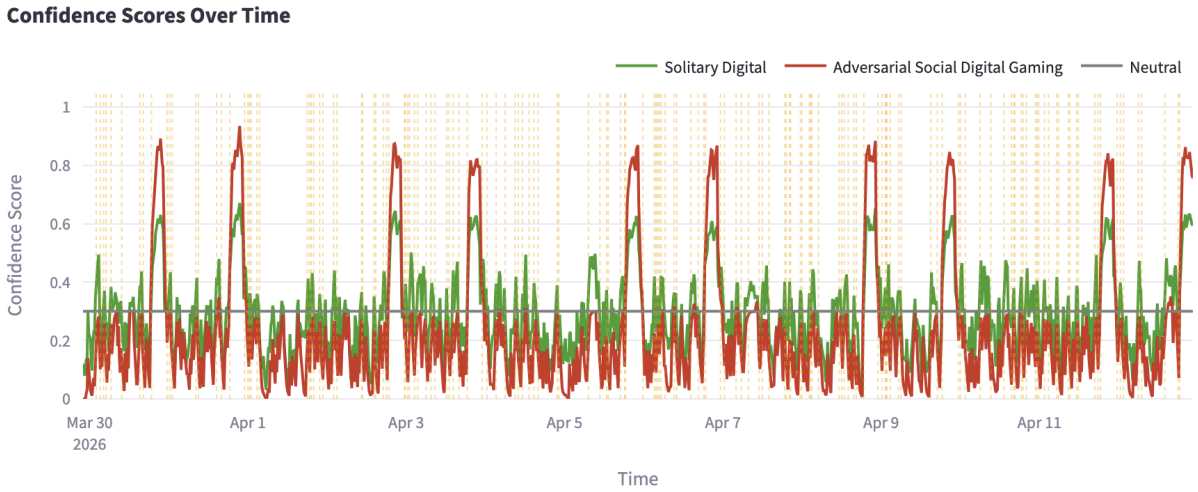


Figure 5.6: Adversarial scenario dominates on scenario evenings when `ambient_noise` is *high*, misclassifying the intended `solitary_digital` context.

The behaviour of the system is expected as context determination only depends on one sensor, which is not ideal. Consequently, the context evaluation is not yet robust

against adversarial scenarios as configuration is sparse. If we add further parameters to the configuration, the system becomes more robust in its context evaluation as it can rely on more datapoints to determine the context.

For instance, by additionally adding the context marker `audio_source_digital` to the configuration, the system can distinguish whether detected ambient noise originates from digital sources (speakers, headphones, gaming audio) or from physical human presence. The updated context assumptions and mock data parameters for this extended scenario are documented in Section E.4. With this additional marker, the system correctly classifies the context as `solitary_digital` even in the presence of high ambient noise, as shown in Figure 5.7.

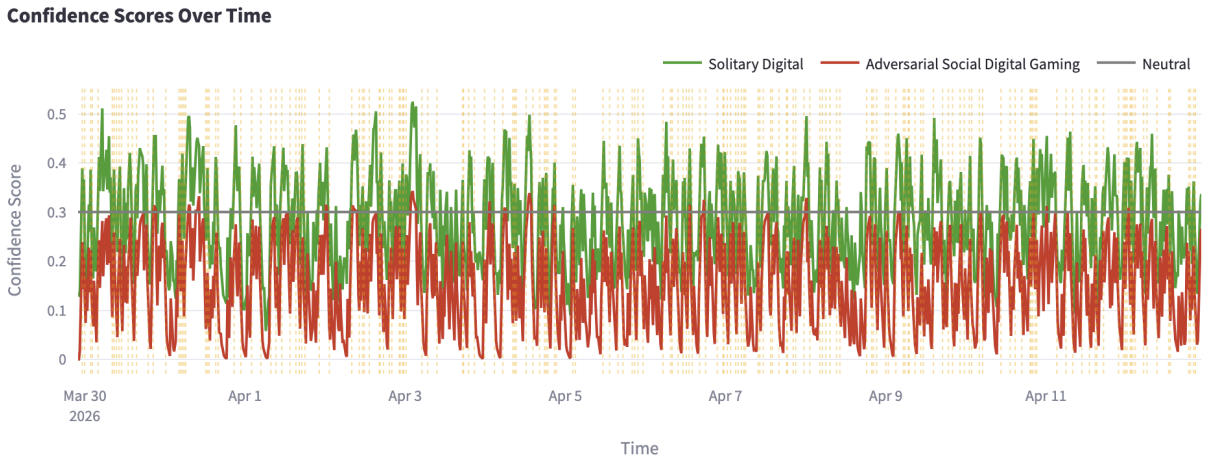


Figure 5.7: Main scenario is correctly classified despite high ambient noise; after adding `audio_source_digital` as a second discriminating marker.

The scenario confirms that context classification is sensitive to the available marker configuration. A single distinguishing marker between two context assumptions proved insufficient under adversarial conditions, whereas extending the configuration with `audio_source_digital` restored the expected classification without requiring changes to the evaluation logic. The system's ability to correctly determine the dominant context under adversarial conditions depends on the availability of sufficient and appropriately configured markers, highlighting the importance of good calibration from domain experts.

5.3.4 Scenario 3: Context shift

The third scenario evaluates the system's behaviour for adjusting the weights of biomarkers based on the active context. Figure 5.8 shows the context weight profiles for both contexts. Notably, as the adversarial context models a person gaming with friends via

voice chat, speech biomarkers are amplified (multiplier 1.5) and network biomarkers are dampened (multiplier 0.5).

```
5.0.1 solitary_digital:
5.0.2   # Speech biomarkers (dampened)
5.0.3   whispering:            0.5
5.0.4   prolonged_pauses:      0.5
5.0.5   ...                    0.5
5.0.6   # Network biomarkers (amplified)
5.0.7   reduced_social_interaction: 1.5
5.0.8   passive_media_binge:        1.5
5.0.9   ...                         1.5
```

(a) `solitary_digital`

```
5.0.1 adversarial_social_digital_gaming:
5.0.2   # Speech biomarkers (amplified)
5.0.3   whispering:            1.5
5.0.4   prolonged_pauses:      1.5
5.0.5   ...                    1.5
5.0.6   # Network biomarkers (dampened)
5.0.7   reduced_social_interaction: 0.5
5.0.8   passive_media_binge:        0.5
5.0.9   ...                         0.5
```

(b) `adversarial_social_digital_gaming`

Figure 5.8: Configuration of biomarker weights for main (a) and adversarial (b) scenarios.

While all data inputs are identical to scenario 1, the new context weights profile for `adversarial_social_digital_gaming` influences the computation such that different indicator scores are produced compared to Figure 5.2a, as seen in Figure 5.9.

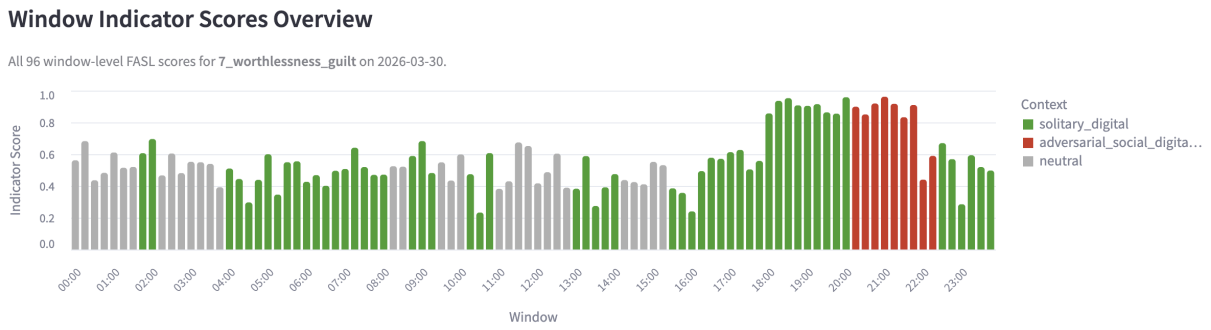


Figure 5.9: Daily indicator likelihoods under adversarial context weights, using the same biomarker data as Scenario 1.

The scenario confirms that context is not merely a label but an active modifier of the analysis output: identical data produces different indicator profiles depending on the active context weights. If two contexts shared identical weight profiles, no difference would occur. The system's sensitivity to context therefore depends entirely on the quality of weight calibration from domain experts.

5.3.5 Scenario 4: Degraded Data

The fourth scenario evaluates the system's resilience when sensor data is partially unavailable or faulty. To isolate the effect of data degradation, this scenario focuses on a single indicator, `1_depressed_mood`, which maps to four biomarkers evenly split across two modalities, as shown in Figure 5.10. The full degradation conditions and configurations are documented in Section E.5.

```
1_depressed_mood:
  biomarkers:
    # Speech modality
    whispering:                weight: 0.25
    prolonged_pauses:          weight: 0.25
    # Network modality
    reduced_social_interaction: weight: 0.25
    passive_media_binge:        weight: 0.25
```

Figure 5.10: Biomarker-to-indicator mapping for `1_depressed_mood`.

Six degradation conditions are applied systematically, progressing from single-biomarker dropout to full-modality dropout, and concluding with a faulty sensor that reports a constant maximum value. Figure 5.11 shows the absolute deviation from the full-data baseline score for each condition and each day. Cells that remain white indicate no deviation from the baseline, while darker cells indicate greater distortion.

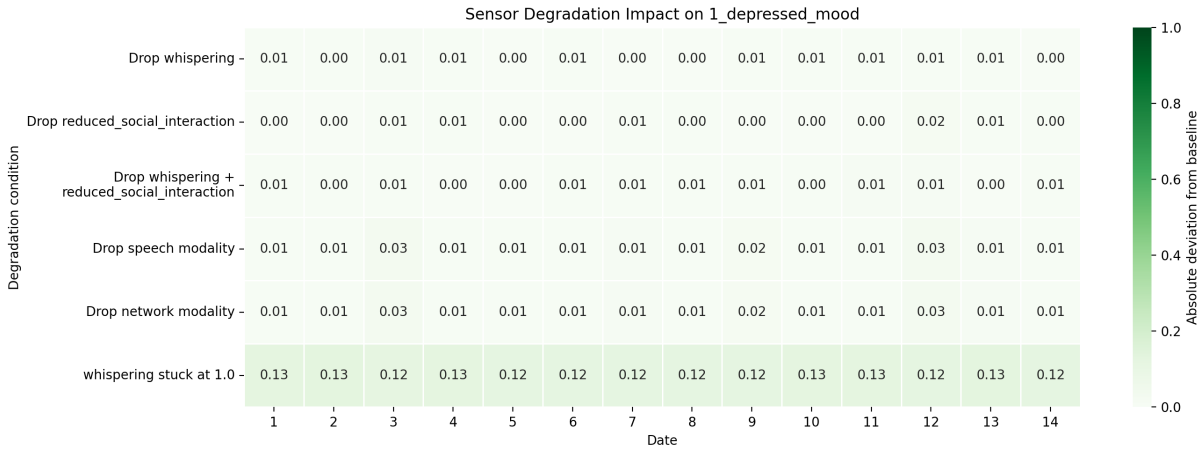


Figure 5.11: Absolute deviation from baseline indicator scores for `1_depressed_mood` under six degradation conditions over 14 days.

For the dropout conditions (rows 1–5), the system continues to produce indicator scores with little deviations (0.01-0.03) from baseline due to the implemented re-normalisation mechanism of weights over the the remaining biomarkers. Consequently, the system's behaviour degrades gracefully in the presence of missing data, given that remaining data provides robust signals.

The faulty sensor condition (row 6) differs qualitatively from the dropout conditions. All four biomarkers are present. However, the constant maximum value for `whispering` introduces a systematic deviation that is visible across all 14 days. Unlike the dropout conditions, where the system correctly adjusts the weights of the remaining biomarkers to maintain stable scores, the faulty sensor condition produces distorted scores. However, the faulty sensor is outweighed by the other three biomarkers, resulting in a moderate deviation of around 0.1 from baseline. In practice, an indicator can be constructed from more than four biomarkers, which makes computation more robust and dilutes influence of a faulty sensors.

The scenario confirms that the system degrades gracefully when sensors are absent: indicator scores remain computable with partial input and and faulty sensor data while the system degrades gracefully, showing robust results with moderate deviations.

5.4 Discussion

This thesis set out to investigate how the Linkage Framework can be extended to design an explainable, white-box system that supports multimodal and contextual data for monitoring of major depressive disorder in IoT-enabled household environments. Each evaluation will be discussed separately, followed by an overall interpretation.

Determinism. The results show that deterministic reproducibility is an inherent property of the white-box design of the proposed system. In contrast, LLMs, even if state-of-the-art, struggle with this property. In a clinical monitoring context where trust is important, this highlights the benefit of our system: a clinician who questions a particular score must be able to re-run the analysis and obtain the identical result. Even clinically insignificant variation undermines this traceability, because it introduces uncertainty about whether the output reflects the data or the model's internal state. The prototype's deterministic property thus directly supports Obj-3 (explainable automated health assessments).

System Architecture. The results show that 86 out of 88 functional requirements have been implemented, with two requirements excluded as deliberate scope decisions rather

than technical limitations; suggesting that the proposed architecture translates into a viable proof-of-concept implementation.

Among the non-functional requirements, two findings are particularly noteworthy. First, the pipeline is able to execute on a Raspberry Pi 4 within 25 minutes for a 14-day scenario *(NFR-2)*, suggesting that edge deployment for periodic background assessment (e.g., once a day) is feasible without cloud dependencies—an important property for privacy-preserving household monitoring. Second, the system's explainability property *(NFR-4)* is satisfied not through a post-hoc explanation layer but through the architecture itself: every computation uses parameterised, closed-form operations with no opaque components, providing inherent white-box interpretability as required by *Obj-3*.

Together, these results indicate that the architecture constitutes a robust prototypical foundation *(Obj-4)*. However, the system is not production-grade: execution time on constrained hardware can be optimized, and aspects such as parallel computation, incremental processing, and formal security hardening would need to be addressed before clinical deployment.

System Behavior. Taken together, the four scenario results suggest that the system's robustness is largely governed by the quality of its configuration by domain experts. Scenario 1 confirmed correct end-to-end behaviour under well-calibrated conditions *(Obj-1)*, while Scenarios 2 and 3 *(Obj-2)* revealed that both context classification and indicator scoring are sensitive to configuration quality: a sparse marker set left the system vulnerable to adversarial contexts, and poorly chosen weight profiles can steer indicator scores in unintended directions. Scenario 4 showed that the system degrades gracefully under data dropout and faulty sensors thanks to weight re-normalisation and compensation from other data sources, becoming less susceptible with better calibration from domain experts.

These findings suggest that the system's reliability scales with configuration effort: sufficient markers from diverse modalities improve context robustness, well-calibrated weights ensure meaningful context influence, and distributing an indicator across multiple biomarkers buffers against individual sensor failures.

The implication is clear: While the system's behaviour is theoretically robust, it depends largely on good calibration from domain experts. This positions calibration as a central concern for future work (Section 6.3).

Summary. Across the three evaluation dimensions, the results indicate that the Extended Linkage Framework and its proof-of-concept implementation are technically viable: the

system produces deterministic, traceable, and explainable outputs from multimodal and context-enriched input, satisfying the core capabilities targeted by the research question and fulfilling all five research objectives defined in Table 1.1. At the same time, the evaluation also highlights that domain-expert calibration plays a pivotal role in result quality, across all parameter ranges. The system's white-box design makes this dependency transparent and auditable, but does not eliminate it. Establishing systematic calibration processes therefore constitutes a key direction for future work.

5.5 Limitations

The following limitations constrain the generalisability of the findings. Where applicable, corresponding directions for future work are discussed in Section 6.3.

Absence of real patient data. No real patient data, clinical recordings, or sensor measurements were available for this work. The evaluation only assessed *system behaviour* rather than *clinical accuracy* or diagnostic validity. The system's ability to detect depressive episodes in real populations remains unverified, and clinical validation constitutes an essential prerequisite before any deployment consideration.

Parameter calibration. The evaluation used predefined parameter settings based on available information for all configurations and did not involve domain experts for calibration. Baseline values were set pragmatically rather than derived from real population distributions, and all parameter weights are provisional rather than empirically derived from domain expertise or clinical data. While acceptable for a proof-of-concept that did not evaluate clinical accuracy, this limits the generalisability of the findings and the system's readiness for real-world use.

Limited modalities. Only two modalities (speech and network) were integrated for the proof-of-concept. Although the architecture supports additional modalities without code changes (NFR-1), this extensibility is not demonstrated in the evaluation.

Chapter 6

Considerations & Future Work

6.1 Summary

This thesis investigated the design of an explainable white-box system that supports multimodal and contextual data for monitoring of major depressive disorder based on DSM-5 factors in IOT-enabled household environments. Based on prior work on the Linkage Framework, this thesis proposed the *CML Framework*, which follows a late-fusion multimodal architecture that preserves per-modality explainability, and a context model that treats environmental information as an active modifier during interpretation of biomarker data.

Following its formulated research objectives in Table 1.1, this thesis continued its contribution by implementing a proof-of-concept prototype that realized: multimodal fusion within an explainable framework, a context model that enriched automated assessments, and an end-to-end implementation thereof. The evaluation showed that the proposed system is technically viable, producing deterministic, traceable and explainable outputs with correct context-sensitive behaviour and graceful degradation under adversarial conditions or data dropout. Further, the proof of concept was measured on an edge deployment, showing feasibility of full pipeline execution on low-cost edge hardware without cloud dependencies, supporting privacy-preserving monitoring in domestic settings. The pipeline followed the explainability property through full pipeline inspection, following the system's white-box requirement. However, the evaluation also identified the need for strong calibration from domain experts to ensure robust system behaviour. Finally, the proof of concept was published as an open-source repository to support future research and development in this domain.

6.2 Final Considerations

This thesis achieved all five of its stated research objectives from Table 1.1 and demonstrated that explainable, multimodal, and context-aware depression monitoring is achievable through a late-fusion architecture combined with a context-as-modifier model, producing deterministic, traceable, DSM-5-aligned assessments deployable on household edge hardware. Objectives 1 and 2 were met through the introduced *CML Framework*, which extended the Linkage Framework with a late-fusion multimodal architecture and a context-as-modifier model. Objective 3 was fulfilled by a system architecture that cohesively integrates data ingestion, context evaluation, multimodal analysis, and explainability into a single pipeline. Objectives 4 and 5 were addressed through a proof-of-concept prototype that realized this architecture and was evaluated for functional correctness, white-box properties, and deployment feasibility. However, while Objective 5 was met, clinical accuracy was not part of the evaluation, which limits the conclusions that can be drawn about the system's real-world diagnostic performance.

Beyond the stated objectives, the work produced several broader impressions. First, calibration of the system emerged as the most critical determinant of system quality, underscoring that configuration from domain experts is essential for robust system behaviour. While this is a natural consequence of a parameterized white-box design, it also highlights that creating a robust system is only the first step towards real-world applicability, and that cross-disciplinary collaboration with medical domain experts will be essential for future development and validation. Second, a current ambiguity in that regard is the extent to which the system's architectural design, notably the properties of white-box explainability, will translate to real-world diagnostic accuracy. This thesis did not evaluate diagnostic accuracy, which leaves the question open of how well such a system performs compared to state-of-the-art black-box AI models, which come with their own trade-offs like the lack of inherent explainability, but arguably have strong predictive power. After all, a white-box system that does not produce accurate results would have limited practical utility. Third, while this thesis focused on MDD monitoring, the design of the CLM framework and proposed architecture is not bound to a specific domain and can be adapted to other monitoring domains by reconfiguring the DSM-5-aligned rule base, suggesting wider applicability than the original scope.

The primary challenge throughout this work was the absence of real multimodal MDD datasets, which necessitated full reliance on simulated sensor data and consequently limits the ecological validity of the evaluation results. Closely related was the difficulty of

parameter calibration: without empirical patient data or structured expert elicitation, all fuzzy membership thresholds, context weights, and baseline values were set pragmatically, introducing a degree of arbitrariness into the system configuration. As a consequence, the evaluation focused instead on technical properties and system behaviour under different conditions and left diagnostic accuracy as an open question for future research.

Despite these challenges, this thesis established a technically sound and transparent foundation for automated MDD monitoring in household environments. The CML Framework and its proof-of-concept implementation demonstrated that white-box explainability, multimodal fusion, and context-awareness can be unified within a deployable pipeline; not merely as a theoretical construct, but as a realised system. The open questions identified, notably the clinical validation and empirically grounded calibration, do not diminish this contribution; rather, they define the natural trajectory for its continuation.

6.3 Future Work

Two directions for future research follow directly from the identified limitations.

Clinical Validation with Real Patient Data A necessary next step is conducting longitudinal studies in real household environments, comparing the system's automated DSM-5-aligned assessments against established diagnostic instruments administered by clinicians. Such validation is currently constrained by the lack of publicly available multimodal depression datasets that combine household IoT sensor streams with clinical ground-truth labels. A clinical validation effort could therefore also produce a purpose-built multimodal dataset that could serve as a benchmark for future research in this domain.

Systematic Domain Expert Calibration The evaluation identified configuration quality as a primary determinant of system robustness, yet all parameters, including fuzzy membership thresholds, context weights, and baseline values, were set pragmatically rather than derived from clinical expertise or empirical data. Structured expert elicitation protocols combined with sensitivity analysis could establish empirically grounded parameter values and quantify how parameter variations propagate through the pipeline. As clinical validation produces real observational data, it simultaneously enables an iterative calibration–validation loop in which domain experts refine system parameters against measured diagnostic outcomes.

Bibliography

[1] Iftikhar Ahmed, Anushree Brahmacharimayum, Raja Hashim Ali, Talha Ali Khan, and Muhammad Ovais Ahmad. "Explainable AI for Depression Detection and Severity Classification from Activity Data: Development and Evaluation Study of an Interpretable Framework". In: *JMIR Mental Health* 12 (2025), e72038. DOI: 10.2196/72038.

[2] American Psychiatric Association. *Diagnostic and Statistical Manual of Mental Disorders*. 5th ed. 2013. DOI: 10.1176/appi.books.9780890425596.

[3] Navneet Bains and Sara Abdijadid. *Major Depressive Disorder*. 2025.

[4] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. "A Survey on Context-Aware Systems". In: *International Journal of Ad Hoc and Ubiquitous Computing* 2.4 (2007), pp. 263–277. DOI: 10.1504/IJAHUC.2007.014070.

[5] Moira N. Burns, Michael Begale, Jennifer Duffecy, Darren Gerhard, Christopher J. Karr, Elise Giangrande, Steven A. Safren, Clarissa Szobot, Linda A. Dimeff, and Adrian Aguilera. "Harnessing Context Sensing to Develop a Mobile Intervention for Depression". In: *Journal of Medical Internet Research* 13.3 (2011), e55. DOI: 10.2196/jmir.1838.

[6] Luca Canzian and Mirco Musolesi. "Trajectories of Depression: Unobtrusive Monitoring of Depressive States by Means of Smartphone Mobility Traces Analysis". In: *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*. 2015, pp. 1293–1304. DOI: 10.1145/2750858.2805845.

[7] Ahnryul Choi et al. "Digital Phenotyping for Stress, Anxiety, and Mild Depression". In: *JMIR mHealth and uHealth* (2024). DOI: 10.2196/52319.

[8] Nicholas Cummins, Stefan Scherer, Jarek Krajewski, Sebastian Schnieder, Julien Epps, and Thomas F. Quatieri. "A Review of Depression and Suicide Risk Assessment Using Speech Analysis". In: *Speech Communication* 71 (2015), pp. 10–49. DOI: 10.1016/j.specom.2015.03.004.

[9] Anind K. Dey. "Understanding and Using Context". In: *Personal and Ubiquitous Computing* 5.1 (2001), pp. 4–7. DOI: 10.1007/s007790170019.

[10] Everette S. Gardner Jr. "Exponential Smoothing: The State of the Art—Part II". In: *International Journal of Forecasting* 22.4 (2006), pp. 637–666. DOI: 10.1016/j.ijforecast.2006.03.005.

[11] Marzyeh Ghassemi, Luke Oakden-Rayner, and Andrew L. Beam. "The False Hope of Current Approaches to Explainable Artificial Intelligence in Health Care". In: *The Lancet Digital Health* 3.11 (2021), e745–e750. DOI: 10.1016/S2589-7500(21)00208-9.

[12] Tibor Haller. *Proof-of-Concept Repository for Explainable Multimodal-based Depression Awareness at the Edge.* Accessed 31 March 2026. 2026. URL: https://github.com/972C8/cml-depression-poc.

[13] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. "The Anatomy of a Context-Aware Application". In: *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '99)*. 1999, pp. 59–68. DOI: 10.1145/313451.313476.

[14] Institute for Health Metrics and Evaluation. *2021 Global Burden of Disease (GBD) [online database]*. Accessed 13 March 2026. Seattle, 2024. URL: https://vizhub.healthdata.org/gbd-results/.

[15] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. "Design Science in Information Systems Research". In: *MIS Quarterly* 28.1 (2004), pp. 75–106. DOI: 10.2307/25148625.

[16] ISO/IEC/IEEE. *Systems and Software Engineering—Life Cycle Processes—Requirements Engineering*. International Standard ISO/IEC/IEEE 29148:2018. International Organization for Standardization, 2018. DOI: 10.1109/IEEESTD.2018.8559686.

[17] Jyh-Shing Roger Jang, Chuen-Tsai Sun, and Eiji Mizutani. *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. 1997. ISBN: 978-0-13-261066-7.

[18] Lal Khan, Mohammad Zubair Khan, and Ibrahim Aljubayri. "A Comprehensive Framework for Multi-Modal Depression Detection: Integrating Adaptive Fusion, Fairness Regularization, and Explainable AI". In: *Mathematics* 14.4 (2026), p. 711. DOI: 10.3390/math14040711.

[19] George J. Klir and Bo Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. 1995. ISBN: 978-0-13-101171-7.

[20] Kurt Kroenke, Robert L. Spitzer, and Janet B. W. Williams. "The PHQ-9: Validity of a Brief Depression Severity Measure". In: *Journal of General Internal Medicine* 16.9 (2001), pp. 606–613. DOI: 10.1046/j.1525-1497.2001.016009606.x.

[21] Puneet Kumar, Shreshtha Misra, Zhuhong Shao, Bin Zhu, Balasubramanian Raman, and Xiaobai Li. "Multimodal Interpretable Depression Analysis Using Visual, Physiological, Audio and Textual Data". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2025. DOI: 10.1109/WACV61041.2025.00518.

[22] Guanjin Lam, Michael M. Tadesse, Yen-Wei Lin, Tu N. Ho, and Selvaraj Thangaraj. "Systematic Review and Meta-Analysis of Explainable Machine Learning Models for Clinical Depression Detection". In: *Behavioral Sciences* 15.11 (2025), p. 1476. DOI: 10.3390/bs15111476.

[23] Jonas Länzlinger. "Audio-centered Approach for Building a Multimodal Predictive AI Agent to Detect Depressive Behaviors". MA thesis. University of St. Gallen, 2025.

[24] Chenhao Lin, Pengwei Hu, Hui Su, Shaochun Li, Jie Mei, Jieyu Zhou, and Henry Leung. "SenseMood: Depression Detection on Social Media". In: *Proceedings of the 2020 International Conference on Multimedia Retrieval*. 2020, pp. 407–411. DOI: 10.1145/3372278.3391932.

[25] Haoliang Liu, Ruolan Li, Xinyu Wang, Sai Bi, Haoyang Lin, Chuanfu Wang, Jie Yu, Lan Ye, and Xingwei Zheng. "Multimodal Depression Recognition and Analysis: Facial Expression and Body Posture Changes via Emotional Stimuli". In: *Journal of Affective Disorders* 381 (2025), pp. 44–54. DOI: 10.1016/j.jad.2025.03.155.

[26] Faith Matcham et al. “Remote Assessment of Disease and Relapse in Major Depressive Disorder (RADAR-MDD): Recruitment, Retention, and Data Availability in a Longitudinal Remote Measurement Study”. In: *BMC Psychiatry* 22 (2022), p. 136. DOI: 10.1186/s12888-022-03753-1.

[27] Stephan Nef. “At the Edge of Empathy: Mental-Health Awareness Through Home-Router Network Traffic Insights”. MA thesis. University of St. Gallen, 2025.

[28] Aurélie Pahud de Mortanges, Haozhe Luo, Shelley Zixin Shu, Amith Kamath, Yannick Suter, Mohamed Shelan, Alexander Poellinger, and Mauricio Reyes. “Orchestrating Explainable Artificial Intelligence for Multimodal and Longitudinal Data in Medical Imaging”. In: *npj Digital Medicine* 7.1 (2024), p. 195. DOI: 10.1038/s41746-024-01190-w.

[29] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. “A Design Science Research Methodology for Information Systems Research”. In: *Journal of Management Information Systems* 24.3 (2007), pp. 45–77. DOI: 10.2753/MIS0742-1222240302.

[30] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. “Context Aware Computing for the Internet of Things: A Survey”. In: *IEEE Communications Surveys & Tutorials* 16.1 (2014), pp. 414–454. DOI: 10.1109/SURV.2013.042313.00197.

[31] Postman. *2025 State of the API Report*. Tech. rep. Postman, 2025.

[32] Zulqarnain Rashid et al. “Remote Monitoring of Patients Through RADAR-Base: An Open-Source Platform for Continuous Health Data Collection”. In: *JMIR Mental Health* (2024). DOI: 10.2196/50058.

[33] Cynthia Rudin. “Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead”. In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215. DOI: 10.1038/s42256-019-0048-x.

[34] Mahmood Sadeghi et al. “Harnessing Multimodal Approaches for Depression Detection Using Large Language Models”. In: *npj Digital Medicine* (2024). DOI: 10.1038/s44184-024-00112-8.

[35] Sohrab Saeb, Mi Zhang, Christopher J. Karr, Stephen M. Schueller, Marya E. Corden, Konrad P. Kording, and David C. Mohr. “Mobile Phone Sensor Correlates of Depressive Symptom Severity in Daily-Life Behavior: An Exploratory Study”. In: *Journal of Medical Internet Research* 17.7 (2015), e175. DOI: 10.2196/jmir.4273.

[36] Luis R. Soenksen, Yu Ma, Cynthia Zeng, Leonard Boussioux, Kimberly Villalobos Carballo, Liangyuan Na, Holly M. Wiberg, Michael L. Li, Ignacio Fuentes, and Dimitris Bertsimas. “Integrated Multimodal Artificial Intelligence Framework for Healthcare Applications”. In: *npj Digital Medicine* 5.1 (2022), p. 149. DOI: 10.1038/s41746-022-00689-4.

[37] Ian Sommerville. *Modernes Software-Engineering: Entwurf und Entwicklung von Softwareprodukten*. 2020. ISBN: 9783868943962.

[38] John Torous and Jukka-Pekka Onnela. “Characterizing the Clinical Relevance of Digital Phenotyping Data Quality with Applications to a Cohort with Schizophrenia”. In: *npj Digital Medicine* 1 (2018), p. 15. DOI: 10.1038/s41746-018-0022-8.

[39] U.S. Food and Drug Administration, Health Canada, and UK Medicines and Healthcare Products Regulatory Agency. *Good Machine Learning Practice for Medical Device Development: Guiding Principles*. Accessed 17 February 2026. 2021. URL: https://www.fda.gov/medical-devices/software-medical-device-samd/good-machine-learning-practice-medical-device-development-guiding-principles.

[40] John Venable, Jan Pries-Heje, and Richard Baskerville. "A Comprehensive Framework for Evaluation in Design Science Research". In: *Design Science Research in Information Systems. Advances in Theory and Practice (DESRIST 2012)*. Vol. 7286. 2012, pp. 423–438. DOI: 10.1007/978-3-642-29863-9_31.

[41] Rui Wang, Fanglin Chen, Zhenyu Chen, Tianxing Li, Gabriella Harari, Stefanie Tignor, Xia Zhou, Dror Ben-Zeev, and Andrew T. Campbell. "StudentLife: Assessing Mental Health, Academic Performance and Behavioral Trends of College Students Using Smartphones". In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*. 2014, pp. 3–14. DOI: 10.1145/2632048.2632054.

[42] Rui Wang, Weichen Wang, Alex daSilva, Jeremy F. Huckins, William M. Kelley, Todd F. Heatherton, and Andrew T. Campbell. "Tracking Depression Dynamics in College Students Using Mobile Phone and Wearable Sensing". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2.1 (2018), p. 43. DOI: 10.1145/3191775.

[43] Lotfi A. Zadeh. "Fuzzy Sets". In: *Information and Control* 8.3 (1965), pp. 338–353. DOI: 10.1016/S0019-9958(65)90241-X.

[44] Lotfi A. Zadeh. "The Concept of a Linguistic Variable and Its Application to Approximate Reasoning—I". In: *Information Sciences* 8.3 (1975), pp. 199–249. DOI: 10.1016/0020-0255(75)90036-5.

[45] Fei Zhao, Chengcui Zhang, and Baocheng Geng. "Deep Multimodal Data Fusion". In: *ACM Computing Surveys* 56.9 (2024), pp. 1–36. DOI: 10.1145/3649447.

[46] Hans-Jürgen Zimmermann and Peter Zysno. "Latent Connectives in Human Decision Making". In: *Fuzzy Sets and Systems* 4.1 (1980), pp. 37–51. DOI: 10.1016/0165-0114(80)90062-7.

[47] Hamad Zogan, Imran Razzak, Xianzhi Wang, Shoaib Jameel, and Guandong Xu. "Explainable Depression Detection with Multi-Aspect Features Using a Hybrid Deep Learning Model on Social Media". In: *World Wide Web* 25.1 (2022), pp. 281–304. DOI: 10.1007/s11280-021-00992-2.

Appendix A
GitHub Repository

The source code is publicly available on GitHub [12]: https://github.com/972C8/cml-depression-poc.

A.1 Repository Overview

The repository follows a modular package structure under `src/` that maps to the architectural components from Section 3.3:

`src/api/` : Linkage API – FastAPI-based REST endpoints and Pydantic schemas for biomarker ingestion, indicator retrieval, and context queries.

`src/core/` : Linkage Core – the central analysis engine comprising Context Evaluation (fuzzy membership functions, weighting strategies, history smoothing), the Analysis Pipeline (baseline computation, indicator aggregation, window processing), and the DSM-5 Gate (threshold evaluation and diagnostic gating logic). Domain models, data readers, and processor modules reside here.

`src/dashboard/` : Streamlit-based Dashboard – multi-page application with interactive components for analysis pipeline transparency, baseline selection, context evaluation inspection, experiment editing, and mock data generation.

`src/shared/` : Shared utilities for cross-cutting concerns including domain model definitions, structured logging, database access, and application-wide settings.

`config/` : YAML configuration files governing all tunable parameters such as indicator definitions, context evaluation weights, DSM-5 gate thresholds, EMA smoothing coefficients, episode detection settings, and reliability constraints.

Appendix B
Requirements Catalogue

This appendix provides the complete list of functional and non-functional requirements. Requirements marked with an asterisk were defined for completeness but are not within the implementation scope of this thesis. The *Result* and *Note* columns summarize the evaluation status. Each requirement is marked as fully implemented (✓), partially implemented ((✓)), or not implemented (×).

B.1 Functional Requirements

Epic 1: Infrastructure & Deployment. This epic establishes the foundational deployment model. The system must be deployable as a containerised platform using Docker Compose, with all services orchestrated from a single entry point. Configuration must be externalisable through environment variables and YAML files.

Table B.1: Functional requirements — Epic 1: Infrastructure & Deployment

Nr.	Description	Result	Note
FR-1	The system must be deployable as a containerised platform using Docker Compose, orchestrating all services (Database, API, Dashboard).	✓	Services are configured and deployable via Docker compose. See docker-compose.yaml.
FR-2	The system must load configuration from environment variables and YAML files at startup, initialise database connections, and verify service health before accepting requests.	✓	Environment variables loaded via Pydantic Settings, YAML configs loaded from config/ directory. Database initialized on startup, /health endpoint verifies service availability.

Epic 2: API. The API epic defines the external interface for data ingestion and retrieval. A REST API secured by API key authentication provides endpoints for submitting and querying biomarker and context data. All responses follow a standardised format to facilitate integration. FR-14 (MCP exposure) was explicitly excluded from the implementation scope, as Model Context Protocol support was determined to offer limited value for the current proof-of-concept objectives.

Table B.2: Functional requirements — Epic 2: API

Nr.	Description	Result	Note
FR-3	The system must provide a REST API for external data ingestion and retrieval.	✓	FastAPI with /biomarkers, /context, /indicators endpoints.
FR-4	The system must require API key authentication via Bearer token for all REST API data ingestion and retrieval endpoints.	✓	HTTPBearer with verify_api_key dependency on all data routes.
FR-5	The system must accept biomarker data submissions via POST /biomarkers endpoint.	✓	Accepts user_id, timestamp, biomarker_type (speech or network), value (JSON), and optional metadata.
FR-6	The system must accept context data submissions via POST /context endpoint.	✓	Accepts user_id, timestamp, context_type (open string for extensibility), value (JSON), and optional metadata.
FR-7	The system must support batch ingestion of multiple biomarker records in a single transaction via POST /biomarkers/batch.	✓	Atomic transaction with 1000 item limit.
FR-8	The system must support batch ingestion of multiple context records in a single transaction via POST /context/batch.	✓	Atomic transaction with 1000 item limit.
FR-9	The system must validate all records in a batch operation before persisting any, failing the entire batch if a single record is invalid.	✓	All records are validated before any database write. If validation or persistence fails, the entire batch is rolled back.
FR-10	The system must retrieve biomarker data via GET /biomarkers with optional filters.	✓	Includes optional filters: user_id (required), time range, biomarker_type.
FR-11	The system must retrieve context data via GET /context with optional filters.	✓	Includes optional filters: user_id (required), time range, context_type.

Nr.	Description	Result	Note
FR-12	The system must retrieve computed indicators via GET /indicators with optional filters.	✓	Includes optional filters: user_id (required), time range, indicator_type.
FR-13	The system must follow a standardised API response format with data wrapper and metadata, and include error code, message, details, and request_id in all errors.	✓	Success responses wrapped in {data, meta}. Errors include code, message, details, and a unique request_id for tracing.
FR-14*	The system must expose data via MCP so that LLM-based agents can perform queries for benchmarking.	×	It was decided not to implement MCP in the current scope due its limited use for achieving the project goal. In the future, extension of the API to support MCP is possible.

* *Defined for completeness; not within the implementation scope of this thesis.*

Epic 3: Database & Persistence. This epic governs the data model and persistence strategy. The database serves as the primary data store for all persistent data, including biomarkers, context markers, indicators, and data computed during the analysis.

Table B.3: Functional requirements — Epic 3: Database & Persistence

Nr.	Description	Result	Note
FR-15	The system must use PostgreSQL as the primary data store for all persistent data.	✓	PostgreSQL 15 with psycopg2, SQLAlchemy 2.0 ORM.
FR-16	The system must persist all ingested data (biomarkers and context markers) with user attribution and temporal ordering.	✓	Each record stores user_id and timestamp; retrieval is ordered chronologically.
FR-17	The system must persist computed indicators with traceability to their originating analysis run.	✓	Each indicator references its analysis run and stores a detailed computation log.
FR-18	The system must persist analysis runs with configuration snapshots enabling full reproducibility.	✓	AnalysisRun stores config_snapshot and pipeline_trace as JSON.

Nr.	Description	Result	Note
FR-19	The system must persist daily summary aggregations for historical trend analysis.	✓	Computed and persisted as Indicator records with a computation log that holds relevant results.
FR-20	The system must persist context evaluation history enabling temporal analysis of context transitions.	✓	Each evaluation is stored with dominant context, confidence scores, raw/smoothed values, and stabilization status, linked to its versioned evaluation run.
FR-21	The system must persist user baselines for personalised normalisation.	✓	Stores per-biomarker statistics (mean, std, percentiles) per user; one baseline entry per user/biomarker combination.
FR-22	The system must persist configuration experiments for comparative analysis.	✓	Each experiment stores a full configuration snapshot. Create, edit, and delete operations available via the dashboard.

Epic 4: Context Evaluation. Context evaluation requirements specify the six-step pipeline that determines the user's situational context from submitted context markers. The pipeline proceeds from reading context markers, through fuzzy membership computation using configurable membership functions, to rule-based aggregation of raw context assumption scores. Temporal stabilisation is achieved through exponential moving average smoothing, hysteresis thresholds, and dwell time delays, ensuring that the dominant context determination is robust against transient fluctuations. All intermediate and final results must be included in the output for transparency.

Table B.4: Functional requirements — Epic 4: Context Evaluation

Nr.	Description	Result	Note
FR-23	The system must allow modification of the context evaluation configuration via YAML files located in the configuration directory.	✓	Configured via config/context_evaluation.yaml, defining marker memberships and context assumptions.

Nr.	Description	Result	Note
FR-24	The system must support different context types.	✓	Three context types defined (solitary_digital, adversarial_social_digital_gaming, neutral). New types can be added via configuration without code changes.
FR-25	The system must execute context evaluation for a specified user and time range, producing a versioned context evaluation run.	✓	Each evaluation run receives a unique ID and is persisted with its configuration for reproducibility.
FR-26	The system must detect context evaluation coverage gaps for a given date range before analysis.	✓	Gaps are detected based on the configured evaluation interval and reported with coverage ratio and missing dates.
FR-27	The system must read context markers from the database for the specified user and time range (step 1).	✓	Reads context records filtered by user, time range, and optionally by evaluation run.
FR-28	The system must compute fuzzy membership degrees for each context marker using configurable membership functions (step 2).	✓	Supports triangular and trapezoidal membership functions, both configurable per marker.
FR-29	The system must support linguistic variables (low, medium, high) with different types of membership functions for fuzzy membership (step 2).	✓	Each marker defines named fuzzy sets (e.g. low, medium, high) with independently configurable function types and parameters.
FR-30	The system must apply fuzzy rules to compute raw context assumption scores for each context type (step 3).	✓	Fuzzy rules support AND/OR operators with weighted aggregation across marker conditions.
FR-31	The system must apply EMA smoothing to raw context evaluation scores to stabilise context detection and prevent rapid oscillations (step 4).	✓	Exponential moving average with configurable alpha (default 0.3), applied independently per context type.
FR-32	The system must implement hysteresis thresholds for context stabilisation, requiring confidence delta before allowing context switches (step 5).	✓	A minimum score difference (default 0.1) is required before a context switch is permitted. Blocked switches report the reason.

Nr.	Description	Result	Note
FR-33	The system must implement dwell time delays for context stabilisation, requiring consecutive readings before confirming a context switch (step 5).	✓	A candidate context must lead for a configurable number of consecutive readings (default 2) before the switch is confirmed. Progress is reported in results.
FR-34	The system must determine the dominant context as the context type with the highest smoothed confidence score (step 6).	✓	The context with the highest smoothed score becomes dominant, subject to hysteresis and dwell time checks.
FR-35	The system must include dominant context, confidence scores, raw scores, smoothed scores, and stabilisation status in context evaluation results (step 6).	✓	Results include all transparency fields: dominant context, per-context raw and smoothed scores, and stabilization details (blocked reason, candidate, dwell progress).

Epic 5: Analysis Pipeline. The analysis pipeline epic encompasses the complete multi-step computation from raw biomarker data to episode decision. Key stages include time-window aggregation with configurable window sizes and minimum reading thresholds, z-score normalisation against user baselines, sigmoid membership functions for mapping deviations to concern degrees, and the FASL formula for computing indicator scores with context-weighted biomarker contributions. The pipeline culminates in daily aggregation and DSM-gate logic, which applies an N-of-M criterion to determine indicator presence and requires both a minimum number of indicators and at least one core indicator for an episode decision. Each stage produces confidence metadata reflecting data completeness and quality.

Table B.5: Functional requirements — Epic 5: Analysis Pipeline

Nr.	Description	Result	Note
FR-36	The system must orchestrate the complete multi-step analysis pipeline from data reading to episode decision.	✓	Full pipeline orchestration across all steps with progress reporting and error handling at each stage.

Nr.	Description	Result	Note
FR-37	The system must allow modification of the analysis configuration via YAML files located in the configuration directory.	✓	Separate YAML files for indicators, context weights, EMA settings, and DSM-gate parameters in the config/ directory.
FR-38	The system must aggregate biomarker readings into configurable time windows for temporal pattern preservation (step 1).	✓	Tumbling windows with configurable size (5 to 60 minutes), aligned to clock boundaries.
FR-39	The system must compute per-biomarker statistics for each aggregated time window (step 1).	✓	Configurable aggregation method per window (mean, median, min, max).
FR-40	The system must handle sparse data gracefully, applying minimum readings thresholds per window (step 1).	✓	Windows with fewer readings than the configurable minimum are excluded from analysis.
FR-41	The system must normalise biomarker values using z-score computation to express deviation from user baseline (step 2).	✓	Each biomarker value is expressed as a standard deviation (z-score) from the user's baseline.
FR-42	The system must support multiple baseline sources: file-based upload or population defaults (step 2).	✓	Baseline files stored in config/baselines/. A population default is included out of the box.
FR-43	The system must support configurable directionality (positively or negatively correlated) for each biomarker (step 2).	✓	Each biomarker is marked as higher_is_worse or lower_is_worse in the indicator configuration.
FR-44	The system must handle missing biomarkers gracefully using configurable dropout strategies (step 2).	✓	Three strategies available: fill with neutral value, compute with available biomarkers only, or skip the window entirely.
FR-45	The system must apply membership functions to convert z-score deviations into degrees of symptom concerns (step 2).	✓	Sigmoid function maps z-scores to a 0-1 concern degree (0.5 = neutral, higher = more concern).

Nr.	Description	Result	Note
FR-46	The system must compute indicator scores using the FASL (Fuzzy Aggregated Symptom Likelihood) formula (step 3).	✓	Weighted sum of directionally adjusted membership values, normalized by the sum of effective weights.
FR-47	The system must adjust biomarker weights based on active context using configurable context-specific multipliers (step 3).	✓	Each context type defines multipliers per biomarker (e.g. speech_activity weighted differently depending on context). Weights are normalized.
FR-48	The system must scale context weight adjustments with the degree of confidence that the context indeed applies, to avoid overweighting uncertain contexts (step 3).	✓	Weight adjustments are blended with context confidence: at confidence 0 weights revert to neutral, at confidence 1 full multipliers apply.
FR-49	The system must compute the result of each analysis window as an analytical summary (step 4).	✓	Each window produces a summary with indicator score, biomarker completeness, context weight, and per-biomarker contributions.
FR-50	The system must provide aggregated daily summary metrics for DSM-gate evaluation and trend analysis (step 4).	✓	Each day produces a summary per indicator covering a range of metrics, like severity, duration, persistence, and data quality.
FR-51	The system must include a confidence value reflecting data completeness and quality in indicator scores (step 4).	✓	Confidence is tracked via biomarker completeness (ratio of available biomarkers), data coverage (actual vs expected windows), and context availability.
FR-52	The system must aggregate the results of all analysis windows into a daily result (step 5).	✓	All window-level indicator scores for a given date are aggregated into a single daily summary.
FR-53	The system must apply N-of-M criterion for DSM-gate logic: an indicator is present if N or more days are above threshold within an M-day window (step 5).	✓	Default based on DSM-5: An indicator is present if at least 5 out of 14 days exceed the threshold. Per-indicator overrides are supported.

Nr.	Description	Result	Note
FR-54	The system must require both minimum indicators present AND at least one core indicator present for an episode decision (step 5).	✓	Default based on DSM-5: Episode requires at least 5 indicators present and at least one core indicator (e.g. social_withdrawal or diminished_interest).
FR-55	The system must generate human-readable decision rationale for each episode evaluation (step 5).	✓	Each episode decision includes a text rationale stating how many indicators were present, which core indicators matched, and whether criteria were met.

Epic 6: Experimentation. The experimentation epic enables systematic evaluation of different parameter configurations. The system must support creating and managing experiments with adjustable parameters, generating mock data with configurable settings and reproducible seeds, and comparing versioned runs to assess how configuration changes affect analysis outcomes.

Table B.6: Functional requirements — Epic 6: Experimentation

Nr.	Description	Result	Note
FR-56	The system must support experiments to run evaluations and compare results across different parameter settings.	✓	Experiments with adjustable parameters can be created, run, and compared side-by-side in the dashboard.
FR-57	The system must generate mock data (biomarker and context) with configurable generation settings and seed for reproducibility.	✓	Scenario-based generation with seed, intervals, and modality ablation via dashboard.
FR-58	The system must enable comparison of versioned runs to assess how configuration changes affect analysis outcomes.	✓	All runs are versioned with configuration snapshots. Two runs can be compared side-by-side in the dashboard, showing differences in configuration and results.

Nr.	Description	Result	Note
FR-59	The system must version all pipeline runs (context evaluation and analysis) and make them selectable for reproducibility.	✓	Both analysis runs and context evaluation runs are versioned with unique IDs and persisted with their configuration. Past runs are selectable via the dashboard for review and reuse.

Epic 7: Dashboard & Visualisation. The dashboard epic specifies the web-based interface that serves as the primary interaction surface for the system. It comprises six pages: Home, Analysis, Context, Experiment, Mock Data, and Data. FR-80 (dynamic baseline computation from historical data) was excluded from the current scope; baselines are limited to predefined files and population defaults.

Table B.7: Functional requirements — Epic 7: Dashboard & Visualisation

Nr.	Description	Result	Note
FR-60	The dashboard must provide a web-based Streamlit interface for accessing and visualising data, running experimental analyses, and interpreting results.	✓	Streamlit application with six pages: Home, Analysis, Context, Experiment, Mock Data, and Data.
FR-61	The dashboard data page must provide access to raw data (e.g. biomarkers, context, indicators, results) in an interpretable format.	✓	Separate tabs for Biomarkers, Indicators, and Context, each showing records in paginated tables.
FR-62	The dashboard data page must provide data tables with sorting, pagination, filtering, and CSV export functionality for biomarkers, context, and indicators.	✓	Tables support type and name filters, configurable page size (25-250 rows), pagination controls, and CSV download.
FR-63	The dashboard mock page must allow creating new users with mocked data.	✓	User enters a user ID and selects a scenario; mock biomarker and context data is generated and persisted.

Nr.	Description	Result	Note
FR-64	The dashboard mock page must provide an interface to mock and test different scenarios via configuration.	✓	Predefined scenarios (solitary_digital, adversarial_social_digital_gaming) are selectable, each with a preview of expected behavior.
FR-65	The dashboard mock page must provide different mock generation settings (data interval, seed for reproducibility, selected modalities).	✓	Advanced options for biomarker and context intervals, reproducibility seed toggle, and modality selection checkboxes for ablation testing.
FR-66	The dashboard mock page must allow resetting user data to re-start mocking data from a clean slate.	✓	Reset button with confirmation dialog. Deletes all data for the selected user across all tables.
FR-67	The dashboard experiment page must allow creating different experiments with adjustable parameters for context evaluation and analysis.	✓	Creation form with name and description, followed by an interactive editor for all configuration parameters.
FR-68	The dashboard experiment page must allow editing and deleting experiments.	✓	Existing experiments can be modified and saved, reset to defaults, or deleted.
FR-69	The dashboard experiment page must allow exporting the configuration as YAML export.	✓	Active configuration can be exported as a YAML file via a download button.
FR-70	The dashboard context page must allow selecting an experiment configuration to run the context evaluation.	✓	Dropdown to choose between the default configuration or any saved experiment.
FR-71	The dashboard context page must display the results for each context evaluation.	✓	Run selector showing each evaluation with its dominant context, confidence score, and timestamp.
FR-72	The dashboard context page must display all computational steps how the result is determined by showing the entire transparency pipeline.	✓	Six-step pipeline breakdown: marker values, fuzzy memberships (with visual charts), raw scores, EMA smoothing, hysteresis/dwell checks, and final result.

Nr.	Description	Result	Note
FR-73	The dashboard context page must display the configuration used for the context evaluation.	✓	Expandable configuration view organized in tabs: EMA settings, Marker Memberships, and Context Assumptions.
FR-74	The dashboard context page must display the historical trends of the context evaluation.	✓	Interactive timeline chart showing confidence scores per context type over time.
FR-75	The dashboard analysis page must allow running analysis on demand for a selected user.	✓	Guided workflow: select date range, experiment config, context evaluation run, and baseline, then trigger analysis.
FR-76	The dashboard analysis page must allow selecting an analysis period to be used in the analysis.	✓	Start and end date/time pickers for defining the analysis period.
FR-77	The dashboard analysis page must allow selecting an experiment configuration to be used in the analysis.	✓	Dropdown to choose the default or a saved experiment configuration, with an expandable detail view.
FR-78	The dashboard analysis page must allow selecting a context evaluation run to be used in the analysis.	✓	Dropdown listing available context evaluation runs with their ID, creation time, and evaluation count.
FR-79	The dashboard analysis page must allow selecting a predefined user baseline or uploading a custom baseline to be used in the analysis.	✓	Selector offering predefined baseline files or a custom file upload option.
FR-80*	The dashboard analysis page must support computing dynamic user baselines from historical biomarker data for personalised normalisation.	×	The database model for per-user baselines exists (FR-21), but no mechanism for computing baselines from historical user data is implemented. Baselines are currently limited to predefined files and population defaults.
FR-81	The dashboard analysis page must display results for each analysis run individually.	✓	Run selector allowing navigation between past runs, each showing its indicators, likelihoods, and durations.

Nr.	Description	Result	Note
FR-82	The dashboard analysis page must display a clear overview of results (episode likelihood and indicator presence based on DSM-5).	✓	Overview of result shows the episode decision, what indicators are present, what core symptoms are present, and additional details for each indicator.
FR-83	The dashboard analysis page must display all computational steps how the result is determined by showing the entire transparency pipeline.	✓	Step-by-step pipeline breakdown for each run. Also supports a comparison mode showing two runs side-by-side.
FR-84	The dashboard analysis page must display the configuration used for the analysis.	✓	The configuration snapshot stored with the analysis run is displayed in a readable format.

* *Defined for completeness; not within the implementation scope of this thesis.*

Epic 8: Transparency & Reproducibility. This epic addresses cross-cutting concerns that underpin scientific rigour. The system must ensure deterministic execution so that identical inputs and configurations produce identical results. Full pipeline computation traces with step-level inputs, outputs, duration, and metadata must be captured and serialised to JSON for auditing and debugging. Human-readable explanations must be generated from the captured data to support white-box interpretability.

Table B.8: Functional requirements — Epic 8: Transparency & Reproducibility

Nr.	Description	Result	Note
FR-85	The system must ensure deterministic analysis execution so that identical inputs and configuration produce identical results.	✓	All core logic is deterministic by design (no random sampling). Mock data generation uses a fixed seed for reproducibility.
FR-86	The system must capture full pipeline computation traces with step-level inputs, outputs, duration, and metadata for white-box explainability.	✓	Each pipeline step is traced with its inputs, outputs, execution duration, and metadata. The full trace is available for inspection after each run.
FR-87	The system must serialise pipeline traces to JSON and store them with each analysis run for auditing and debugging.	✓	Traces are serialized to JSON and persisted alongside the analysis run, enabling retrieval and review at any time.

Nr.	Description	Result	Note
FR-88	The system must provide human-readable explanations based on captured data.	✓	Both context evaluation and the analysis pipeline provide detailed step-by-step explanations with formulas and visual charts.

B.2 Non-Functional Requirements

Table B.9: Non-functional requirements

Nr.	Title	Description
NFR-1	Architectural Flexibility	The system must be modular and extensible, allowing new data modalities, biomarkers, and processing components to be integrated without modifying the core pipeline. Interfaces between layers must be clearly separated and standardised to support future evolution.
NFR-2	Processing Efficiency	The system must be capable of executing on resource-constrained edge hardware (e.g. household devices) with acceptable latency, without requiring cloud-based computation for core analysis functionality.
NFR-3	Operational Simplicity	The system must be deployable and operable in a household environment with minimal configuration effort, using containerised orchestration (Docker Compose) to manage all services from a single entry point.
NFR-4	Explainability	The system must provide inherent white-box interpretability, enabling every output to be transparently traced back to defined mappings between features and DSM-5 indicators. Post-hoc explanation of opaque models is not sufficient; the reasoning path itself must be interpretable.
NFR-5	Reproducibility	The system must support scientific reproducibility by producing deterministic results for identical inputs and configurations. All pipeline runs must be versioned with configuration snapshots, enabling exact re-execution and comparative analysis across experiments.

Nr.	Title	Description
NFR-6	Privacy	The system must process all data locally without exporting raw data to external servers. Clear boundaries must exist between model logic and data access to keep data governance simple and auditable, supporting operation in privacy-sensitive household environments.
NFR-7	Maintainability	The system must be structured and documented so that future developers can understand, modify, and extend it. Interfaces, rule definitions, and evaluation procedures must be clearly documented to support continued research beyond this thesis.
NFR-8	Auditability	The system must capture and persist complete computation traces with step-level inputs, outputs, and metadata, enabling after-the-fact review and verification of any decision.
NFR-9	Robustness	The system must behave predictably under degraded conditions, including incomplete, sparse, or noisy input data. Graceful degradation strategies (e.g. dropout handling, minimum reading thresholds) must prevent silent failures or misleading outputs.

Appendix C
Dashboard Screenshots

This appendix provides annotated screenshots of the running prototype dashboard, supplementing the selective discussion in Section 4.8.

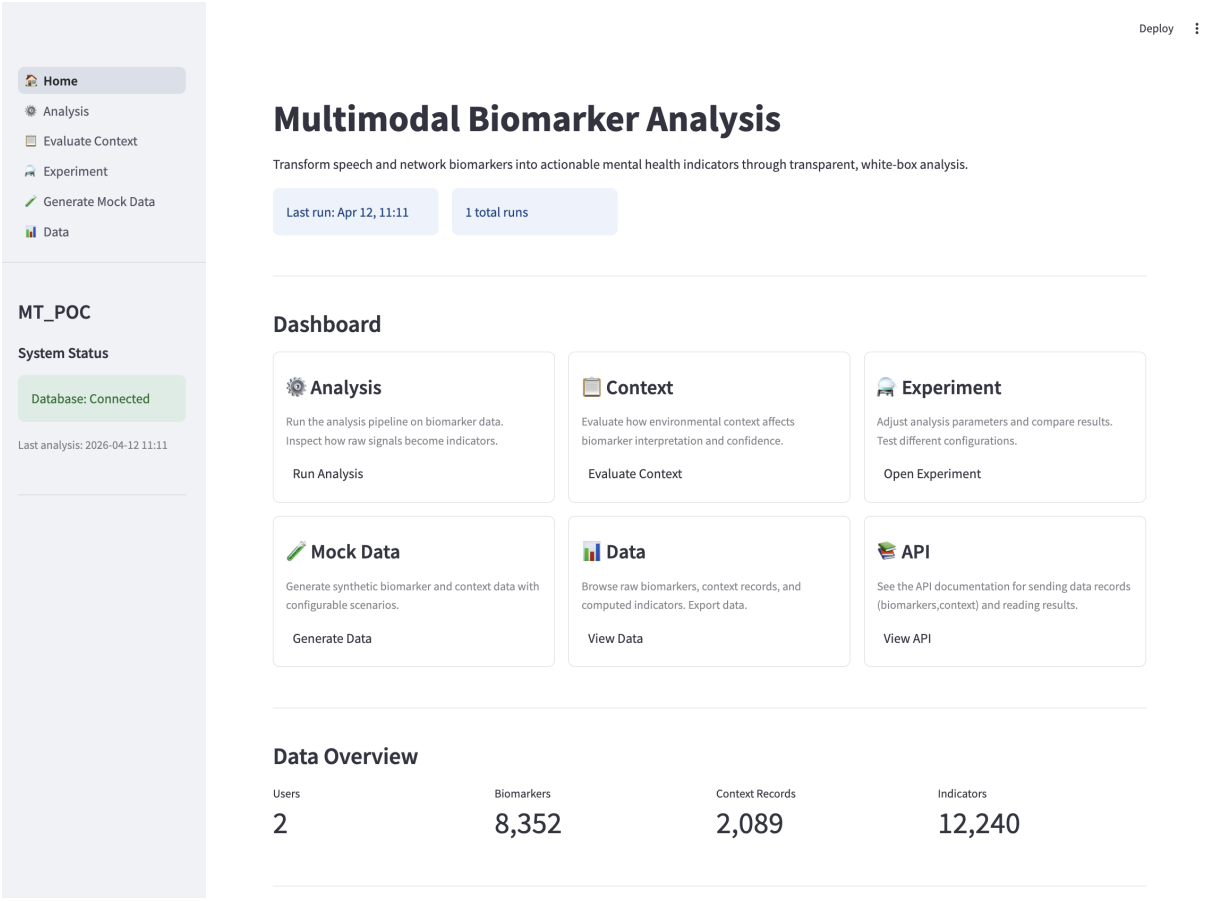


Figure C.1: Home dashboard — Entry point providing navigation to all major dashboard sections.

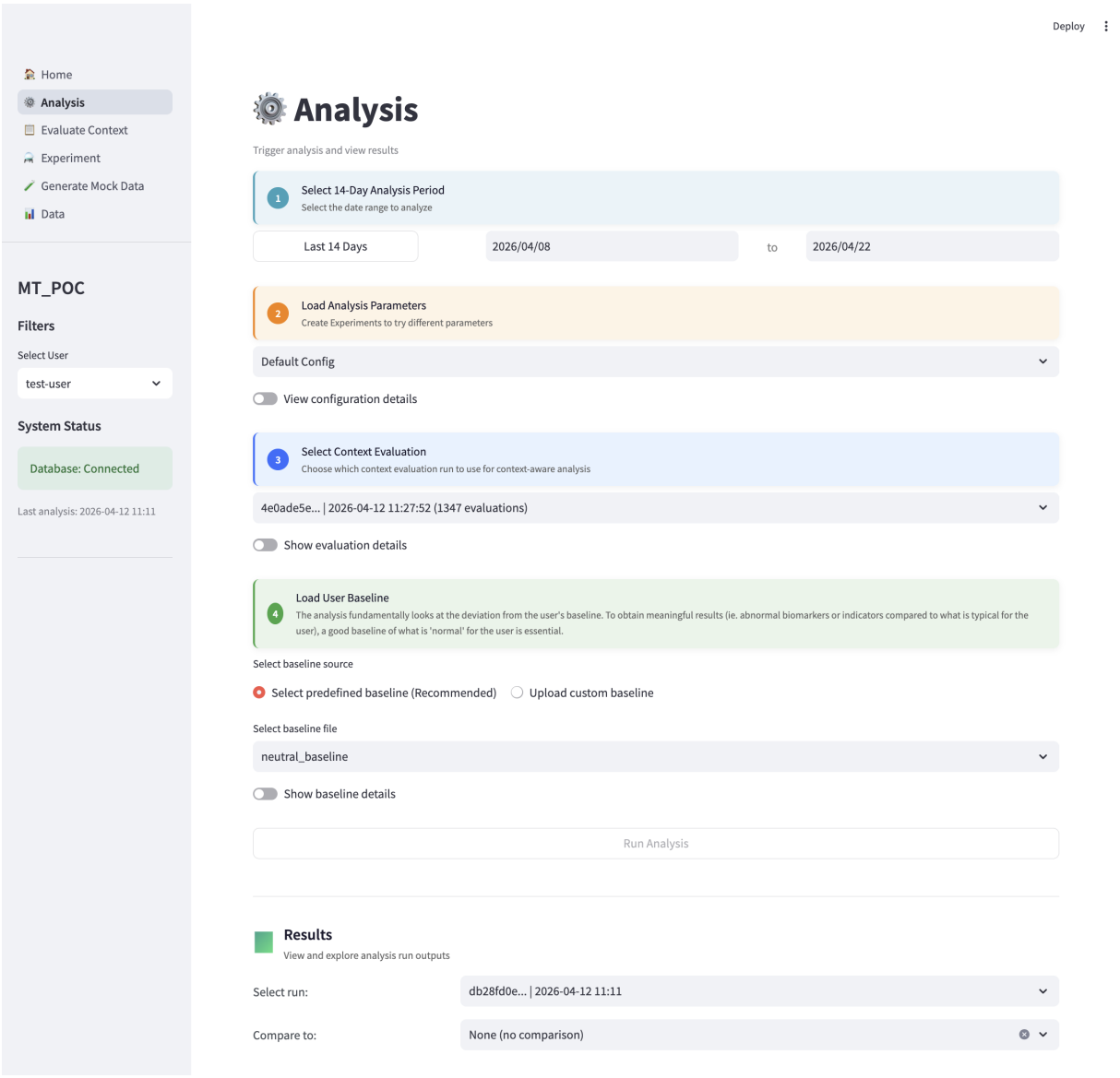


Figure C.2: Analysis dashboard (1 of 4) — Running the analysis pipeline.

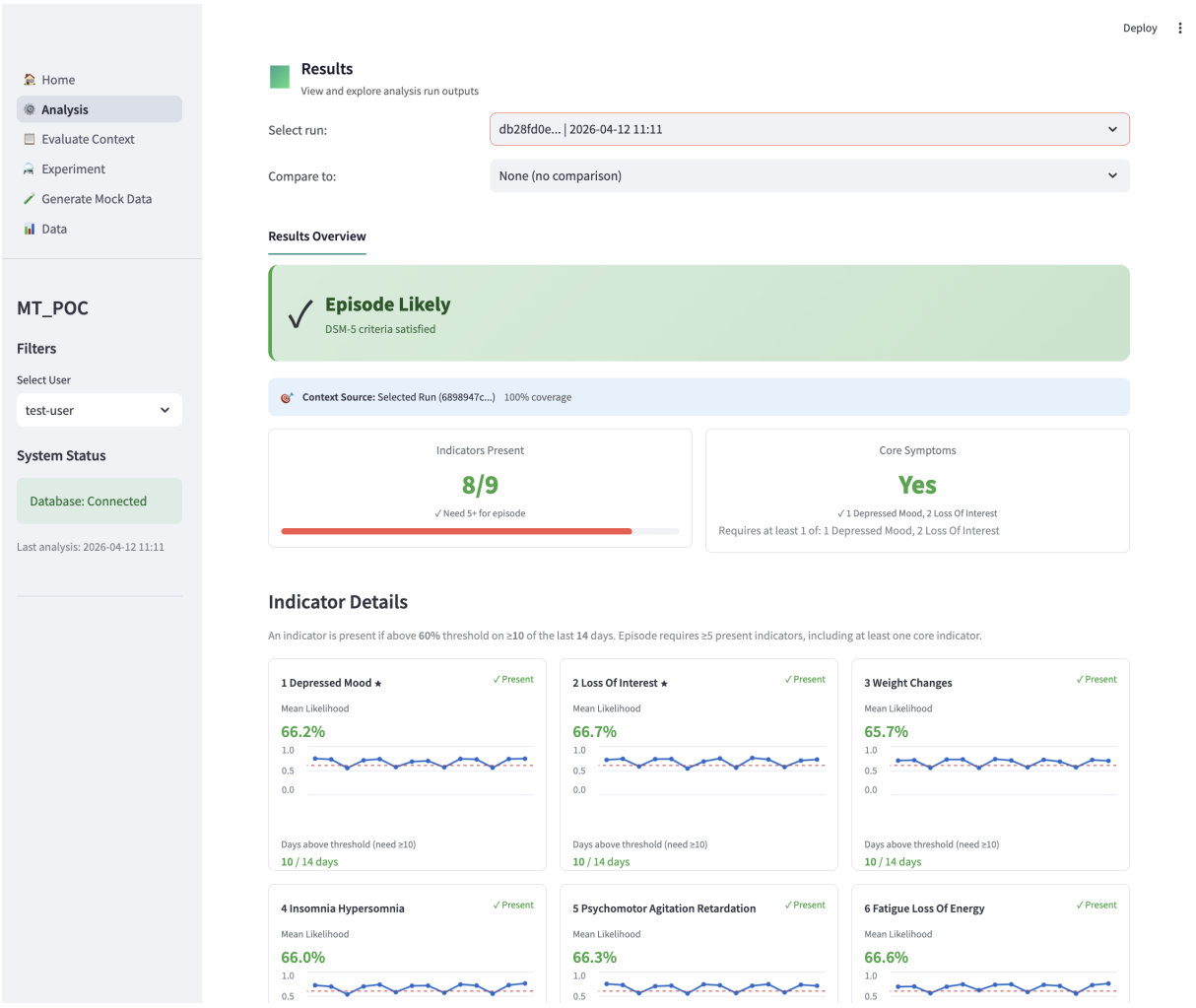


Figure C.3: Analysis dashboard (2 of 4) — Viewing the analysis results.

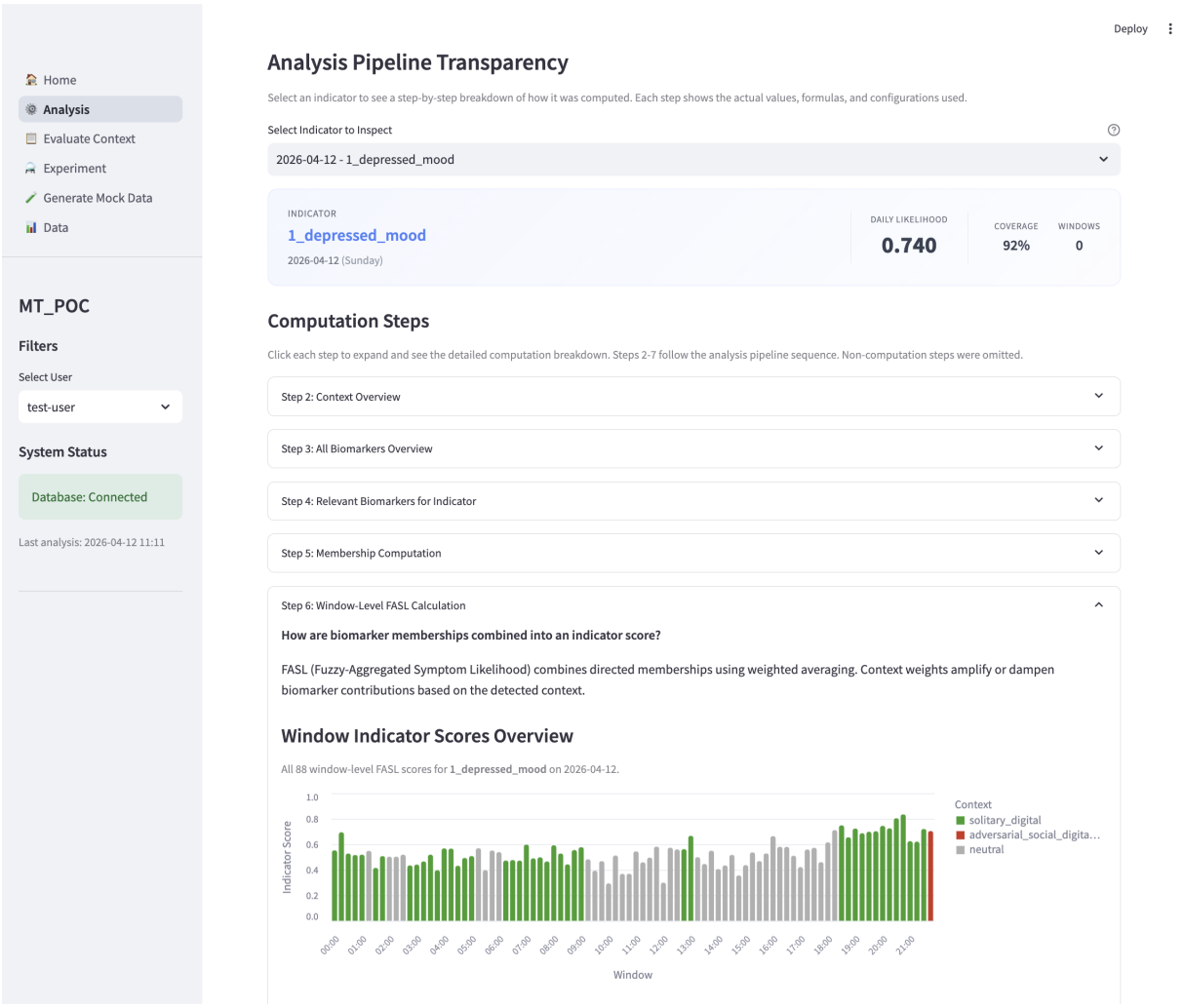


Figure C.4: Analysis dashboard (3 of 4) — Viewing extended details of the analysis pipeline trace.

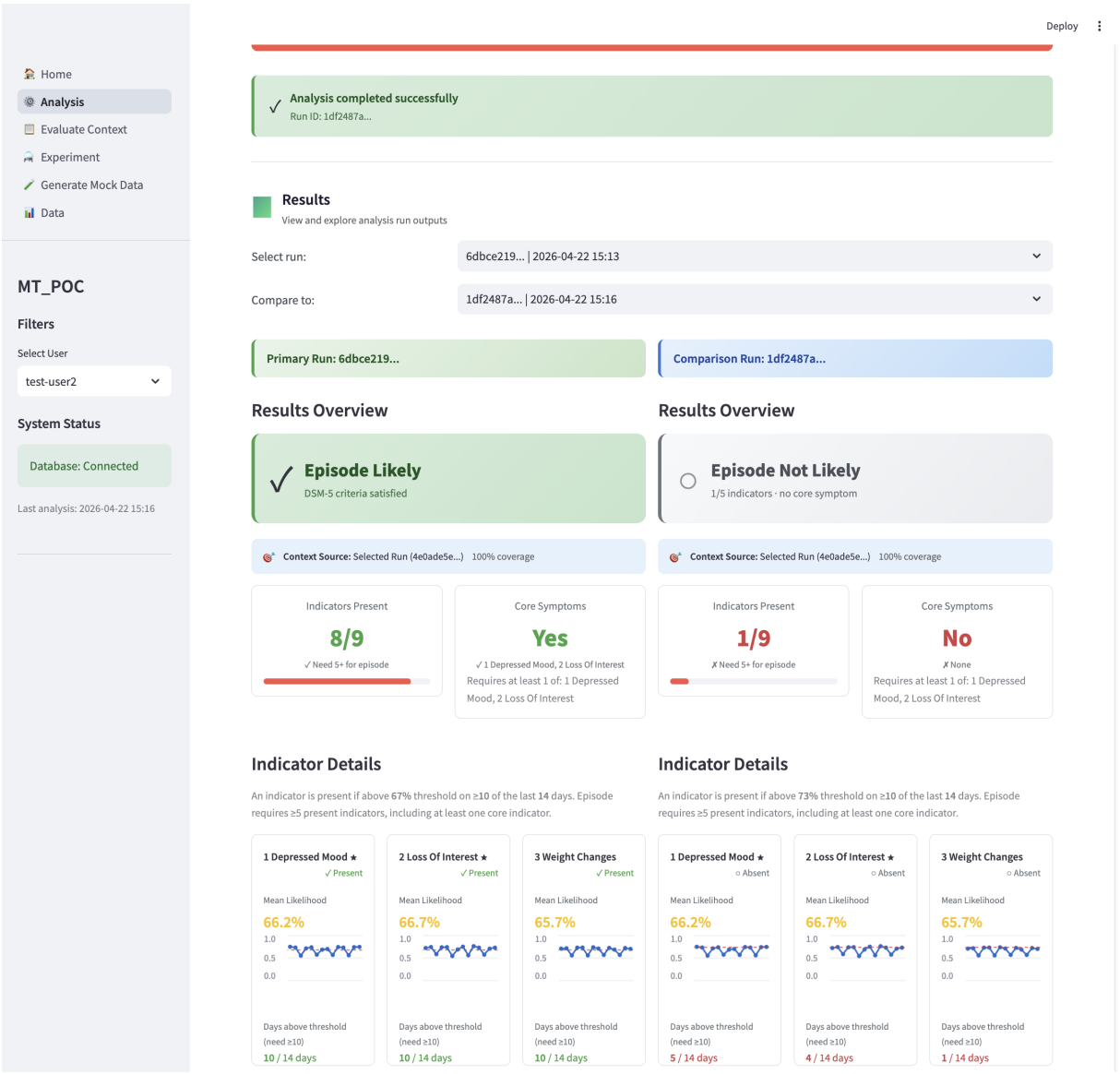


Figure C.5: Analysis dashboard (4 of 4) — Comparing two analysis results.

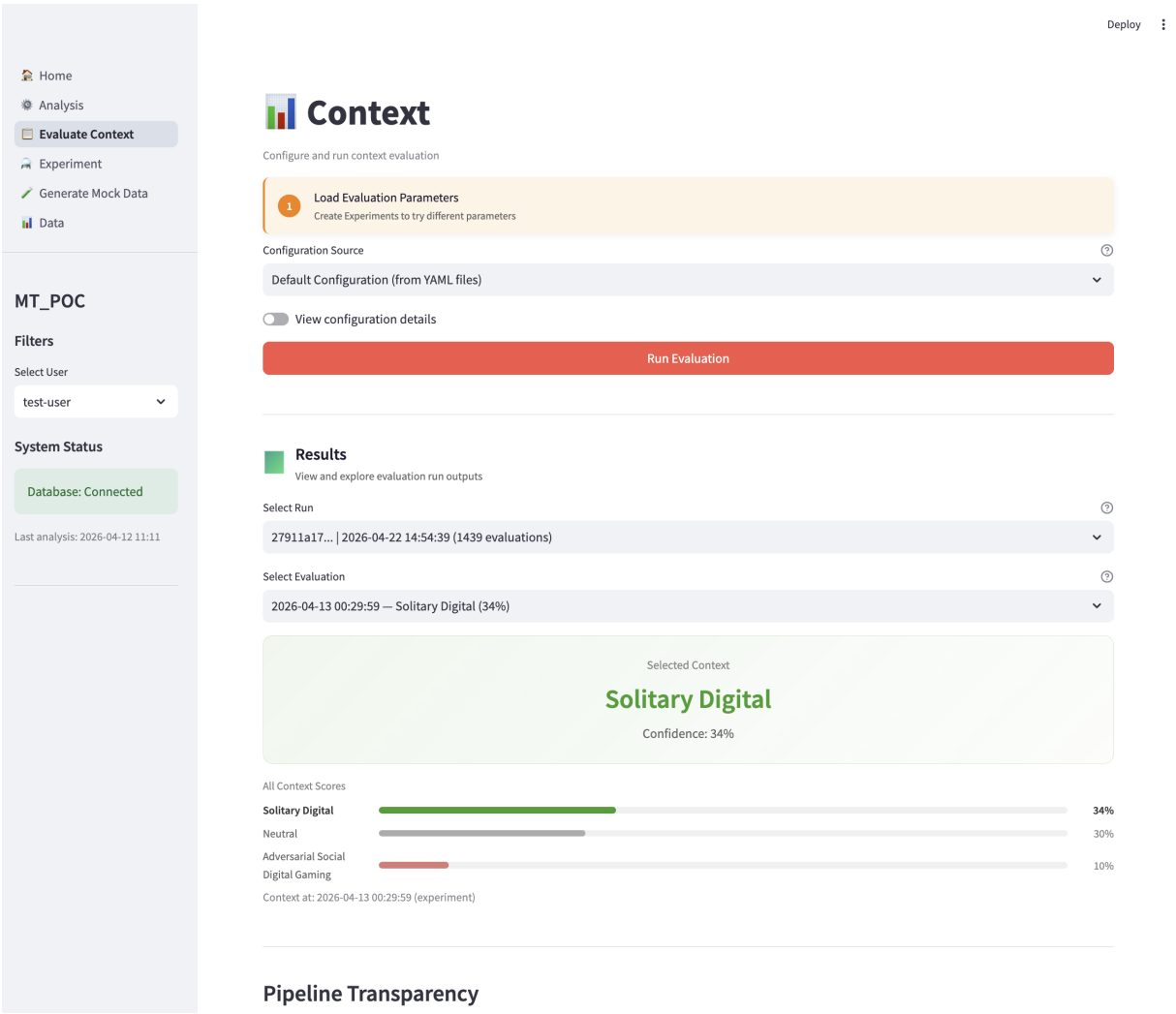


Figure C.6: Context dashboard (1 of 2) — Running a context evaluation and viewing the result.

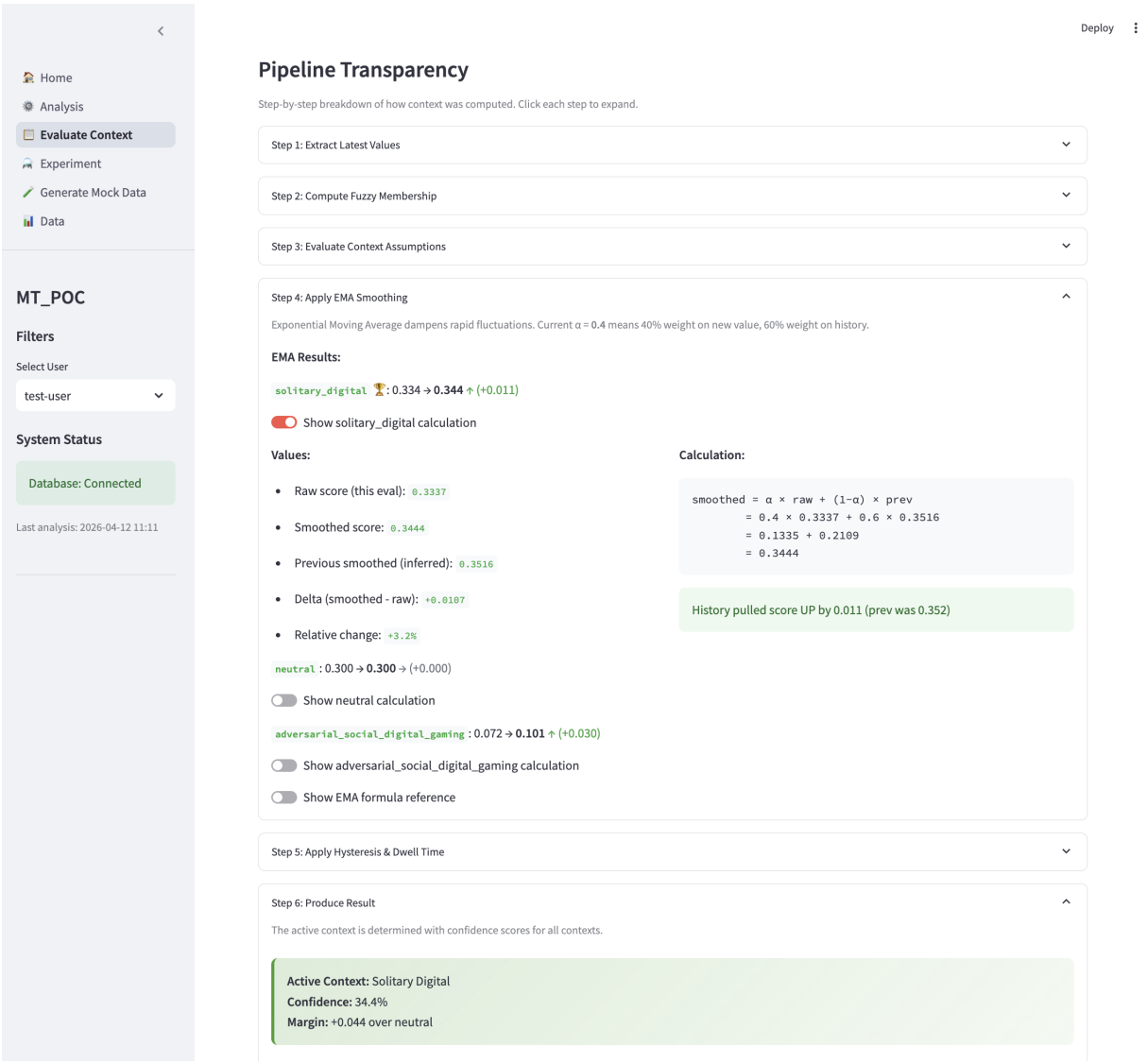


Figure C.7: Context dashboard (2 of 2) — Viewing extended details of the context pipeline trace.

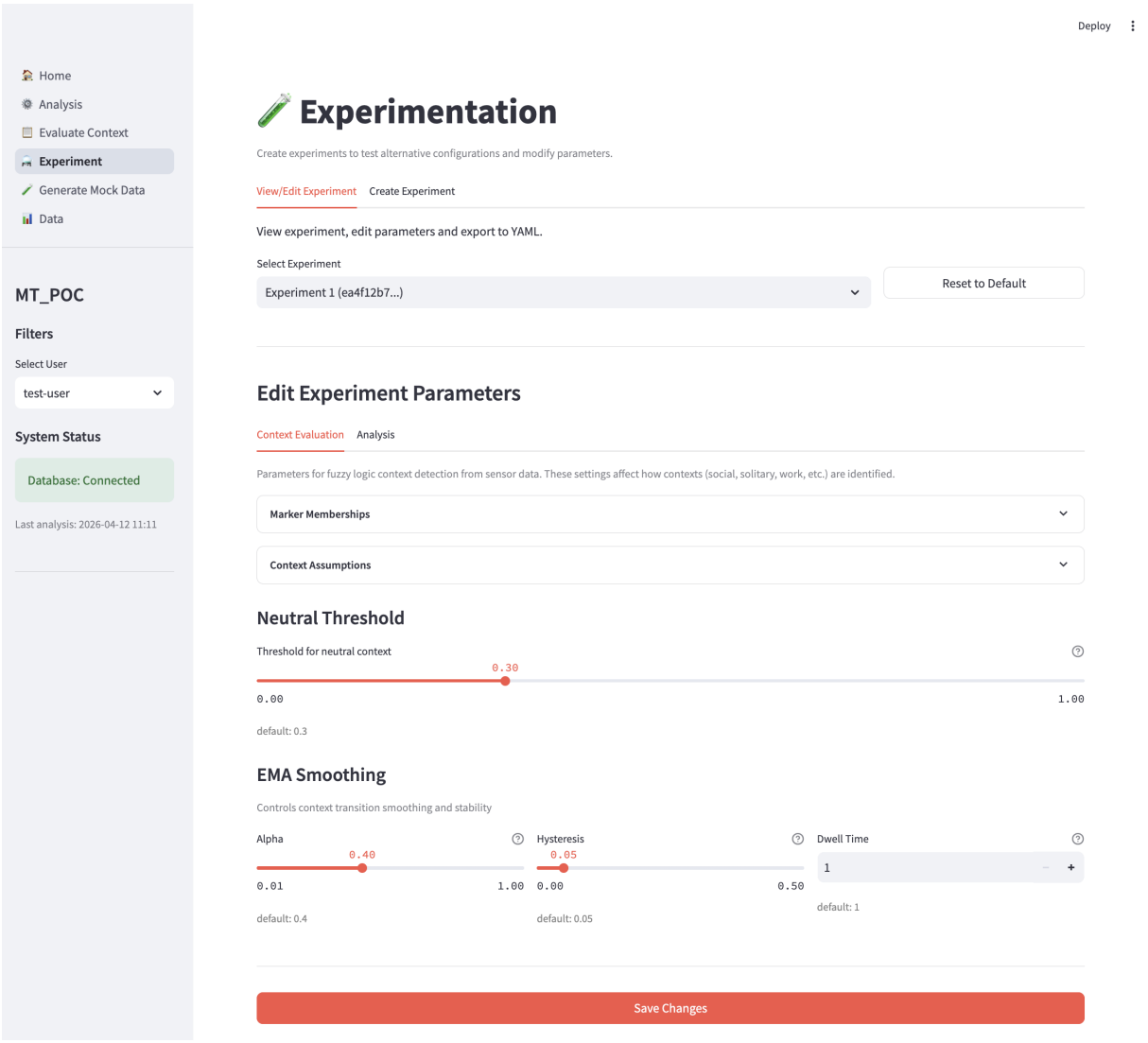


Figure C.8: Experiment dashboard — Creating and running experiments to adjust the pipeline.

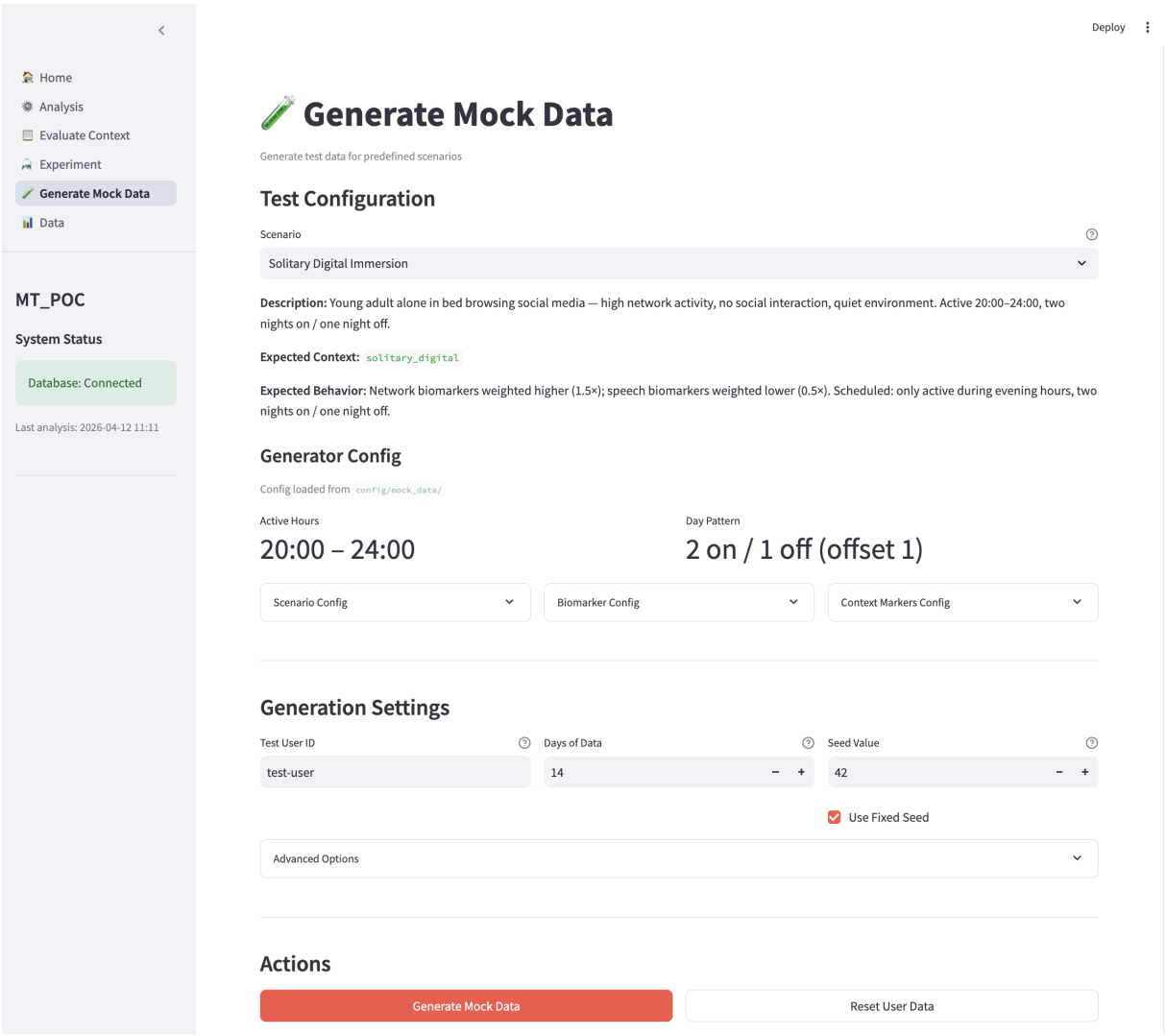


Figure C.9: Mock data dashboard — Creating mock data.

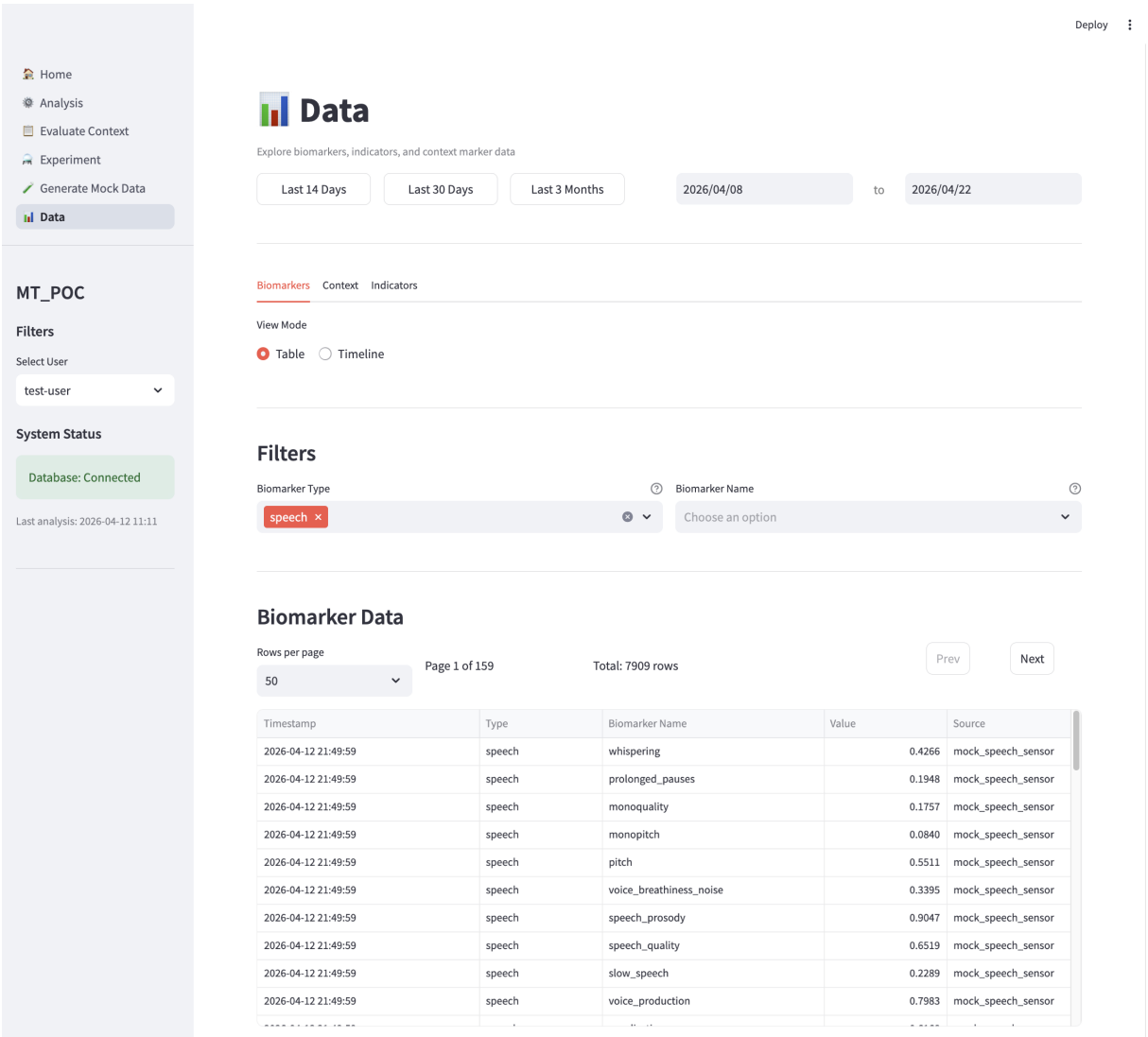


Figure C.10: Data dashboard — Inspecting the raw data ingested into the system.

Appendix D
Configuration Parameters

This appendix catalogues every parameter of the analysis pipeline that is externalised through configuration files, grouped by the component that consumes it. For each parameter, the design rationale states which decision the parameter encodes and who is expected to tune it. Concrete values adopted for the evaluation in Chapter 5 are listed in Appendix E.

Table D.1: Configurable parameters grouped by pipeline component.

Component	Configurable Parameter	Design Rationale
Analysis Pipeline	Biomarker-to-indicator mappings	Domain expert must tune what biomarkers constitute an indicator
	Per-biomarker weights	Domain expert must tune relative influence of each biomarker
	Directionality	Domain expert must define whether a higher or lower value is better/worse for each biomarker
	Biomarker type	Domain expert must define the type of biomarker
	EMA alpha Hysteresis threshold Dwell time	Domain expert must tune the smoothing parameters
Context Evaluator	Membership function type and parameters	Domain expert must define how context markers are mapped to context types
	Context assumption conditions	Domain expert must define what context markers constitute a context type
	Aggregation operator	Domain expert must define how context scores are aggregated
	Neutral fallback threshold	Domain expert must define threshold to apply neutral context

Component	Configurable Parameter	Design Rationale
	Per-context per-biomarker weight multipliers	Domain expert must tune relative influence of context on biomarkers
DSM-5 Gate	Gate presence threshold θ Consecutive-day window M Minimum indicator count Core indicator list Peak window k	Domain expert must define gate parameters based on disorder type

Appendix E
Evaluation Artefacts

This appendix documents the configurations, scripts, and prompts used in Chapter 5.

E.1 Determinism Verification

The verification script executes the full pipeline repeatedly on identical input and compares outputs at every level. For full script, see [12].

```
python scripts/verify_determinism.py \
    --runs 5 --seed 42 --days 14 \
    --scenario solitary_digital
```

```
================================================
DETERMINISM VERIFICATION REPORT
================================================
Seed: 42 | Days: 14 | Runs: 5 | Scenario: solitary_digital

PHASE 1: Context Evaluation
  Results per run:     336
  Total comparisons:   1,344
  Status:              PASS (all identical)

PHASE 2: Analysis Pipeline
  Window FASL Scores:
    Scores per run:    12,096
    Total comparisons: 48,384
    Status:            PASS (all identical)

  Daily Summaries:
    Summaries per run: 126
    Total comparisons: 504
    Status:            PASS (all identical)
```

```
  Episode Decision:
    Episode likely:    True
    Indicators present: 8
    Total comparisons: 4
    Status:            PASS (all identical)

==================================================
OVERALL: PASS - 50,236 total comparisons (587.4s)
==================================================
```

E.2 LLM Prompt Templates

Both prompts were executed five times each on March 13, 2026, using Claude Opus 4.6 (Extended Thinking) and ChatGPT 5.4 (Thinking).

E.2.1 Context Evaluation Prompt

```
5.0.1 I want you to determine the active context of 15-minute windows using the
        provided data and
5.0.2 instructions below.
5.0.3 Your analysis is performed using the following mock data: "2.1
        _mock_context.csv".
5.0.4 The mock data includes context markers (e.g. people_in_room,
        network_activity_levels), which
5.0.5 may be used to determine the active context for a given time window.
5.0.6 You must come up with your own contextual understanding between active
        contexts and context markers.
5.0.7 The active context must be one of: [neutral, solitary_digital,
5.0.8 adversarial_social_digital_gaming].
5.0.9 - neutral = a neutral context; no specific context is active
5.0.10 - solitary_digital = Young adult alone in bed browsing social media at
        night
5.0.11 - adversarial_social_digital_gaming = Young adult alone in room playing
        games online with friends
5.0.12 The confidence value must be provided as a value between 0% to 100%.
5.0.13 You must come up with your own contextual understanding how you determine
        the confidence and
5.0.14 consequently determine the active context.
```

```
5.0.15 The time windows must be 15-minutes. For every time window, evaluate the
         provided data and determine
5.0.16 the active context.
5.0.17 You must follow the instructions exactly as written down and you are not
         allowed to deviate at any
5.0.18 point.
```

E.2.2 Indicator Evaluation Prompt

```
 5.0.1 I want you to determine the present indicators of 15-minute windows using
         the provided data and instructions below.
 5.0.2 Your analysis is performed using the following mock data: "2.1
         _mock_biomarkers.csv".
 5.0.3 The mock data includes biomarkers (e.g. pitch, passive_media_binge), which
          may be used to determine the present indicators for a given time
         window.
 5.0.4 You must come up with your own contextual understanding between indicators
          and biomarkers.
 5.0.5 The indicators must be one of: [1_depressed_mood, 2_loss_of_interest, 3
         _weight_changes, 4_insomnia_hypersomnia, 5
         _psychomotor_agitation_retardation, 6_fatigue_loss_of_energy, 7
         _worthlessness_guilt, 8_diminished_ability_to_think_concentrate, 9
         _suicidality].
 5.0.6 The confidence value must be provided as a value between 0% to 100%.
 5.0.7 You must come up with your own contextual understanding how you determine
         the indicator_likelihood and consequently determine the present
         indicators.
 5.0.8 The time windows must be 15-minutes. For every time window, evaluate the
         provided data and determine the present indicators.
 5.0.9 You must evaluate all 9 indicators for each time window.
5.0.10 You must follow the instructions exactly as written down and you are not
         allowed to deviate at any point.
```

E.2.3 Run Index

Table E.1 indexes the raw output files produced by each of the ten LLM runs (two models, two tasks, five runs each).

Table E.1: LLM output files per model and task (5 runs each).

Model	Task	Files
Claude Opus 4.6	Context	`2.1_claude_context_output-run{1..5}.md`
	Indicators	`2.1_claude_biomarkers_output-run{1..5}.md`
ChatGPT 5.4	Context	`2.1_chatgpt_context_output-run{1..5}.md`
	Indicators	`2.1_chatgpt_biomarkers_output-run{1..5}.md`

All output and comparison files are available in the project repository (Appendix A).

E.3 Scenario 1: Solitary Digital Context

The test scenario is *young adult starts extensively browsing social media alone in bed in a repeating pattern of two consecutive nights followed by one night off*; henceforth denoted as `solitary_digital`.

E.3.1 Context Evaluation Parameters

The EMA parameters in Table E.2 are provisional, selected from their conventional operating ranges by empirical testing on mock data. $\alpha = 0.4$ sits at the responsive end of the typical 0.1–0.5 range: lower values failed to capture evening transitions within the first scenario windows, while higher values amplified jitter in the neutral period. The hysteresis threshold of 0.05 follows common practice. The dwell time is set to a single period because at 15-minute granularity a value of 2 already requires 30 minutes of sustained change before a transition is recorded, which proved too conservative for short evening scenario windows.

Table E.2: EMA smoothing parameters.

Parameter	Value
Alpha (smoothing factor)	0.4
Hysteresis threshold	0.05
Dwell time (periods)	1

The parameters weights of the `solitary_digital` assumption (Table E.3) were set arbitrarily based on best efforts to mock the scenario but have not been tuned against

real data and should be considered provisional. Intuitively, the context expects "low" number of people_in_room, "high" network_activity_level, "low" door_open_events. However, the focus is on testing the system behavior rather than setting correct values. A weighted-mean aggregation (Section 3.5.1.5) is used in preference to a strict conjunction so that the evaluator continues to return a graded confidence score under partial evidence, rather than collapsing to zero whenever a single marker deviates.

Table E.3: Context assumption: `solitary_digital`.

Marker	Fuzzy Set	Weight
people_in_room	low	0.4
network_activity_level	high	0.4
door_open_events	low	0.2

The membership functions in Table E.4 use trapezoidal shapes for end-of-scale sets where any value past a saturation point should map to full membership (e.g. `people_in_room` is high) and triangular shapes elsewhere. Parameter ranges encode intuitive thresholds for each marker but are not calibrated against real sensor distributions and would need to be revisited before deployment. Again, values were set based on best efforts to mock the scenario.

Table E.4: Fuzzy membership definitions per context marker.

Marker	Set	Function	Parameters
people_in_room	low	triangular	[0, 1, 2]
	medium	triangular	[2, 3, 5]
	high	trapezoidal	[4, 7, 10, 10]
network_activity_level	low	triangular	[0, 0.2, 0.5]
	medium	triangular	[0.4, 0.5, 0.7]
	high	triangular	[0.6, 1.0, 1.0]
door_open_events	low	triangular	[0, 0, 2]
	medium	triangular	[1, 3, 5]
	high	trapezoidal	[4, 6, 10, 10]

E.3.2 Indicator–Biomarker Mapping

The catalogue in Table E.5 contains 29 biomarkers across nine indicators, with at most two biomarkers per modality per indicator. The cap balances per-indicator signal diversity against a tractable sensor footprint for the POC. Candidates are drawn from the prior work on speech [23] and network [27] biomarkers and, where multiple candidates are available, those appearing in more than one indicator are preferred to shrink the overall catalogue. Indicator 7 (`worthlessness_guilt`) is the only incomplete row: the referenced literature does not identify suitable speech biomarkers, and rather than improvise candidates, the indicator retains only its two network biomarkers.

Table E.5: Indicator–biomarker mapping (29 biomarkers across 9 indicators). Biomarkers marked ↓ are `lower_is_worse`; all others are `higher_is_worse`.

Indicator	Speech	Network
1_depressed_mood	whispering, prolonged_pauses	reduced_social_interaction, passive_media_binge
2_loss_of_interest	monoquality, monopitch	shrinking_domain_diversity, reduced_interactive_engagement
3_weight_changes	pitch ↓, voice_breathiness_noise	food_ordering_pattern_shift, calorie_nutrition_information_seeking
4_insomnia_hypersomnia	speech_prosody ↓, speech_quality ↓	shifted_sleep_timing, changed_sleep_duration
5_psychomotor_agitation	slow_speech, prolonged_pauses	restless_device_switching, slowed_variable_typing_dynamics
6_fatigue_loss_of_energy	speech_prosody ↓, speech_quality ↓	extended_daytime_inactivity, decline_effortful_interaction
7_worthlessness_guilt	—	engagement_with_mental_health_content, digital_self_withdrawal
8_diminished_concentration	speech_quality ↓, monopitch	fragmented_focus, indecisive_information_seeking
9_suicidality	voice_production ↓, vocalization ↓	crisis_oriented_help_seeking, self_harm_community_engagement

Biomarkers within an indicator receive equal weights (i.e., 0.25 for 4 biomarkers). Unequal weighting would require evidence of differential predictive value per biomarker, which the available data cannot supply; equal weighting is the only neutral choice and is explicitly provisional pending empirical calibration.

Directionality per biomarker is adopted directly from prior work on speech and network analysis [23, 27]. In total, 22 biomarkers are `higher_is_worse` and 7 are `lower_is_worse`.

E.3.3 Context Weights and DSM-5 Gate

Context multipliers for `solitary_digital` use a binary scheme: all speech biomarkers are dampened to 0.5 and all network biomarkers are amplified to 1.5. As per-biomarker tuning is not feasible at this stage, values were set based on best efforts to mock the scenario. To simplify the evaluation, the same multipliers are applied to all indicators within a modality.

The gate and reliability parameters are listed in Table E.6. $M = 14$, the minimum-indicator count of 5, and the core-indicator set {`1_depressed_mood`, `2_loss_of_interest`} are taken directly from the DSM-5 criteria for a major depressive episode [2]. $\theta = 0.6$ is inherited from the network-modality Linkage prototype in which the FASL formulation was introduced [27]. $k = 8$ aggregates the daily indicator likelihood over the worst two hours rather than a 24-hour mean, so that symptom peaks are not diluted by long uninformative periods such as sleep.

Table E.6: DSM-5 gate and reliability parameters.

Parameter	Value
Presence threshold θ	0.6
Consecutive-day window M	14
Minimum indicators	5
Core indicators	1_depressed_mood, 2_loss_of_interest
Peak window k	8

E.3.4 User Baseline

The baseline convention in Table E.7 is pragmatic: without access to real population distributions, the evaluation adopts mean 0.25 for `higher_is_worse` biomarkers, mean 0.75 for `lower_is_worse` biomarkers, and a uniform standard deviation of 0.15.

Table E.7: User baseline convention (std = 0.15 for all biomarkers).

Direction	Mean
higher_is_worse (22 biomarkers)	0.25
lower_is_worse (7 biomarkers)	0.75

The asymmetric means encode a healthy-state starting point: each biomarker sits far from its concerning end of the scale, so deviations in the concerning direction translate to increased symptom likelihood after direction handling. The uniform standard deviation is a simplification of convenience; in deployment it would be replaced by per-biomarker, user- or cohort-specific values so that z-scores reflect genuine deviation from a user's typical behaviour.

E.3.5 Mock Data Generation

Mock data is generated at 15-minute intervals over 14 days with seed = 42, yielding 2,688 biomarker and 336 context marker records. The fixed seed is a prerequisite for the determinism verification in Section 5.1.

Context marker data follows a two-regime schedule (Table E.8). The daytime regime samples around values consistent with typical domestic activity, while the 20:00–24:00 regime on scenario-active evenings—repeating two-nights-on, one-night-off over the 14-day window—is tightened around the values expected under `solitary_digital`. The variance split is intentional: a wide spread during the neutral regime reproduces noisy baseline behaviour, whereas a tight spread during the scenario regime guarantees an inspectable signal that the context evaluator can react to without ambiguity.

Table E.8: Mock data generation parameters by period.

Period	Type	Baseline	Variance
Neutral (00:00–20:00)	people_in_room	1.5	0.5
	network_activity_level	0.4	0.3
	door_open_events	3	2
Scenario (20:00–24:00, every 2nd night)	people_in_room	1	0
	network_activity_level	0.9	0.1
	door_open_events	1	1

Biomarker values during the neutral period are sampled around the user baseline (Table E.7) with variance 0.2, set slightly wider than the baseline standard deviation of 0.15 so that the neutral period does not itself produce systematic deviations that would muddy the scenario contrast. During the scenario period, network biomarkers are overridden to baseline 0.8 with variance 0.15, producing a clear deviation of 0.55 from the healthy baseline. The two indicator-9 biomarkers (`crisis_oriented_help_seeking`,

self_harm_community_engagement) are intentionally *not* overridden: keeping them at baseline establishes a per-indicator ground truth for the episode decision—eight of nine indicators should become elevated while indicator 9 remains below threshold—and thereby demonstrates that the pipeline analyses each indicator independently rather than propagating a blanket amplification across the catalogue.

E.4 Scenario 2: Adversarial Context

This section documents the configuration changes applied to extend the base scenario (Section E.3) for adversarial robustness evaluation.

E.4.1 Extended Marker: audio_source_digital

A fifth context marker, audio_source_digital, is introduced to indicate whether detected audio originates from digital sources (e.g. speakers, headphones, gaming audio) rather than physical human presence. A low value indicates digital audio sources; a high value indicates physical presence. The marker is introduced specifically because ambient_noise alone cannot discriminate between the two scenarios: high noise can arise from either gaming audio (solitary) or actual co-located people (social), so a second marker is required to break the tie.

The fuzzy membership definition (Table E.9) mirrors that of network_activity_level (Table E.4): both are normalised to the [0, 1] range, and reusing the same partition keeps the evaluator's sensitivity to each marker comparable without introducing a new calibration surface.

Table E.9: Fuzzy membership definition for audio_source_digital.

Set	Function	Parameters
low	triangular	[0, 0.2, 0.5]
medium	triangular	[0.4, 0.5, 0.7]
high	triangular	[0.6, 1.0, 1.0]

E.4.2 Updated Context Assumptions

Both context assumptions are extended with the audio_source_digital marker, as listed in Tables E.10a and E.10b. Weights are redistributed from the original 0.4/0.4/0.2 layout to 0.30/0.30/0.10/0.15/0.15 to accommodate the two additional markers while maintaining a sum of 1.0. Weights are kept identical between the two assumptions

so that only the `ambient_noise` and `audio_source_digital` fuzzy sets differ across contexts—the remaining markers do not bias the outcome in either direction, isolating the audio pair as the sole discriminator between solitary and adversarial classification.

Table E.10: Updated context assumptions accommodating the `audio_source_digital` marker.

(a) `solitary_digital`.

Marker	Fuzzy Set	Weight
people_in_room	low	0.30
network_activity_level	high	0.30
door_open_events	low	0.10
ambient_noise	**low**	0.15
audio_source_digital	**low**	0.15

(b) `adversarial_social_digital_gaming`.

Marker	Fuzzy Set	Weight
people_in_room	low	0.30
network_activity_level	high	0.30
door_open_events	low	0.10
ambient_noise	**high**	0.15
audio_source_digital	**high**	0.15

E.4.3 Mock Data: Extended Adversarial Scenario

The extended scenario reuses the adversarial mock data unchanged, adding only the `audio_source_digital` series, as listed in Table E.11. The marker is held at baseline 0.1 with variance 0.1 across both the neutral and the scenario-active periods, reflecting that the audio source remains digital throughout (headphones, speakers, or game audio) rather than switching to physical presence. This choice is the crux of the extended scenario: the ambient-noise signal alone falsely suggests co-located human activity, but the constant low reading from `audio_source_digital` supplies the counter-evidence that lets the context evaluator correctly classify the state as `solitary_digital` despite the high ambient noise.

Table E.11: Mock data parameters for the extended adversarial scenario (delta from base scenario).

Period	Marker	Baseline	Variance
Neutral (00:00–20:00)	audio_source_digital	0.1	0.1
Scenario (20:00–24:00)	ambient_noise	0.9	0.1
	audio_source_digital	0.1	0.1

E.5 Scenario 4: Degraded Data

This section documents the degradation conditions applied to indicator `1_depressed_mood` in Scenario 4. The indicator maps to four equally weighted biomarkers across two modalities, as listed in Table E.12.

Table E.12: Biomarker configuration for `1_depressed_mood`.

Biomarker	Modality	Weight	Direction
whispering	speech	0.25	higher_is_worse
prolonged_pauses	speech	0.25	higher_is_worse
reduced_social_interaction	network	0.25	higher_is_worse
passive_media_binge	network	0.25	higher_is_worse

Table E.13 lists the six degradation conditions. Conditions 1–5 simulate sensor dropout by removing biomarkers from the input, causing the FASL formula to re-normalise over the remaining biomarkers. Condition 6 simulates a faulty sensor by overriding the computed membership value of `whispering` with a constant value of 1.0 (maximum symptom signal) for all days and time windows.

Table E.13: Degradation conditions for Scenario 4.

#	Condition	Biomarkers affected	Available
1	Drop `whispering`	1 speech removed	3/4
2	Drop `reduced_social_interaction`	1 network removed	3/4
3	Drop `whispering` + `reduced_social_interaction`	1 speech + 1 network removed	2/4
4	Drop speech modality	both speech removed	2/4
5	Drop network modality	both network removed	2/4
6	`whispering` stuck at 1.0	faulty sensor (all present)	4/4

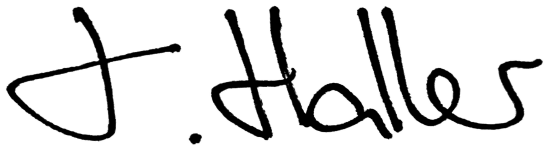
The evaluation script (`evaluation-heatmap.py`) that generates the degradation heatmap is available in the project repository (Appendix A).

Declaration of Authorship

‘I hereby declare,

- that I have written this thesis independently;
- that I have written the articles and contributions using only the aids listed in the index;
- that all parts of the thesis produced with the help of aids (incl. AI-Tools) have been precisely declared;
- that I have mentioned all sources used and cited them correctly according to established academic citation rules; this also includes the comprehensible disclosure of all personal publications;
- that I have acquired all immaterial rights to any materials I may have used, such as images or graphics, or that these materials were originally created by myself;
- that I am aware of the legal provisions regarding the publication and dissemination of parts or the entire thesis and that I comply with them accordingly;
- that I am aware that the University will prosecute a violation of this Declaration of Authorship and that disciplinary as well as criminal consequences may result, which may lead to expulsion from the University or to the withdrawal of my title.’

By submitting this thesis, I confirm through my conclusive action that I am submitting the statutory declaration, that I have read and understood it, and that it is true.



St. Gallen, April 24, 2026 Tibor Haller

Declaration of Auxiliary Aids

The creation of this thesis involved third-party software, including AI-based tools. I declare all uses of third-party software in the thesis text that contributed non-trivially to the thesis' content and are non-standard in such research. Further, by the University of St. Gallen's decree II.B.1.20 section 5.3, I explicitly declare the following uses:

- Private proofreading, DeepL, Anthropic Claude, Google Translate, Grammarly (Premium), and OpenAI ChatGPT (Plus), in their current versions available between 2025 and April 2026, have been used for spellchecking, grammar checking, and enhancing the conciseness and clarity of the thesis text.
- Anthropic Claude and OpenAI ChatGPT (Plus) in their current versions available between 2025 and April 2026, have been used for brainstorming, code assistance and refactoring and LaTex illustrations, algorithms and formulas.