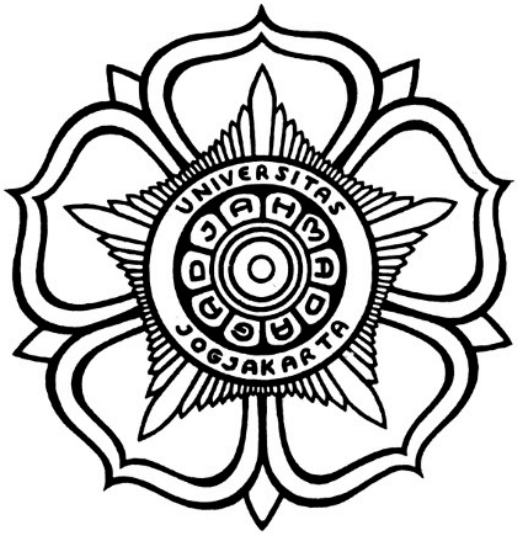
TRANSFORMING UNIVERSITY MENTAL HEALTH SUPPORT: AN AGENTIC AI FRAMEWORK FOR PROACTIVE INTERVENTION AND RESOURCE MANAGEMENT

BACHELOR'S THESIS



THE SUSTAINABLE DEVELOPMENT GOALS
Industry, Innovation and Infrastructure
Affordable and Clean Energy
Climate Action

Written by:

GIGA HIDJRIKA AURA ADKHY
21/479228/TK/52833

INFORMATION ENGINEERING PROGRAM

DEPARTMENT OF ELECTRICAL AND INFORMATION ENGINEERING
FACULTY OF ENGINEERING UNIVERSITAS GADJAH MADA
YOGYAKARTA
2025

ENDORSEMENT PAGE

TRANSFORMING UNIVERSITY MENTAL HEALTH SUPPORT: AN AGENTIC AI FRAMEWORK FOR PROACTIVE INTERVENTION AND RESOURCE MANAGEMENT

THESIS

Proposed as A Requirement to Obtain
Undergraduate Degree (*Sarjana Teknik*)
in Department of Electrical and Information Engineering
Faculty of Engineering
Universitas Gadjah Mada

Written by:

GIGA HIDJRIKA AURA ADKHY
21/479228/TK/52833

Has been approved and endorsed

on . . . . . .

Supervisor I	Supervisor II
Dr. Bimo Sunarfri Hantono, S.T., M.Eng.	**Guntur Dharma Putra, PhD**
NIP 197701312002121003	**NIP 111199104201802102**

STATEMENT

Saya yang bertanda tangan di bawah ini :

Name	: Giga Hidjrika Aura Adkhy
NIM	: 21/479228/TK/52833
Tahun terdaftar	: 2021
Program	: Bachelor's degree
Major	: Information Engineering
Faculty	: Faculty of Engineering, Universitas Gadjah Mada

Menyatakan bahwa dalam dokumen ilmiah Skripsi ini tidak terdapat bagian dari karya ilmiah lain yang telah diajukan untuk memperoleh gelar akademik di suatu lembaga Pendidikan Tinggi, dan juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang/lembaga lain, kecuali yang secara tertulis disitasi dalam dokumen ini dan disebutkan sumbernya secara lengkap dalam daftar pustaka.

Dengan demikian saya menyatakan bahwa dokumen ilmiah ini bebas dari unsur-unsur plagiasi dan apabila dokumen ilmiah Skripsi ini di kemudian hari terbukti merupakan plagiasi dari hasil karya penulis lain dan/atau dengan sengaja mengajukan karya atau pendapat yang merupakan hasil karya penulis lain, maka penulis bersedia menerima sanksi akademik dan/atau sanksi hukum yang berlaku.

Yogyakarta, tanggal-bulan-tahun

Materai Rp10.000

(Tanda tangan)

Giga Hidjrika Aura Adkhy
NIM 21/479228/TK/52833

PAGE OF DEDICATION

Tuliskan kepada siapa skripsi ini dipersembahkan!

contoh

PREFACE

Contoh: Puji syukur ke hadirat Allah SWT atas limpahan rahmat, karunia, serta petunjuk-Nya sehingga tugas akhir berupa penyusunan skripsi ini telah terselesaikan dengan baik. Dalam hal penyusunan tugas akhir ini penulis telah banyak mendapatkan arahan, bantuan, serta dukungan dari berbagai pihak. Oleh karena itu pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Orang 1 yang telah
2. Orang 2 yang telah
3. <isi dengan nama orang lainnya>

Akhir kata penulis berharap semoga skripsi ini dapat memberikan manfaat bagi kita semua, aamiin.

Catatan: setiap nama yang dituliskan boleh disertai dengan alasan berterima kasih.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE AND ABBREVIATION

[SAMPLE]

b	=	bias
$K(x_i, x_j)$	=	fungsi kernel
y	=	kelas keluaran
C	=	parameter untuk mengendalaikan besarnya pertukaran antara penalti variabel slack dengan ukuran margin
L_D	=	persamaan Lagrange dual
L_P	=	persamaan Lagrange primal
w	=	vektor bobot
x	=	vektor masukan
ANFIS	=	Adaptive Network Fuzzy Inference System
ANSI	=	American National Standards Institute
DAG	=	Directed Acyclic Graph
DDAG	=	Decision Directed Acyclic Graph
HIS	=	Hue Saturation Intensity
QP	=	Quadratic Programming
RBF	=	Radial Basis Function
RGB	=	Red Green Blue
SV	=	Support Vector
SVM	=	Support Vector Machines

INTISARI

Layanan kesejahteraan mahasiswa di perguruan tinggi masih banyak bersifat reaktif dan sering terlambat menjangkau mereka yang membutuhkan. Skripsi ini merancang dan mempraktikkan sebuah kerangka *agentic AI* yang berorientasi keselamatan untuk mendukung layanan yang lebih proaktif dan terukur dengan tetap melibatkan manusia. Artifak inti, *Safety Agent Suite*, terdiri dari empat agen yang saling melengkapi: (i) **Safety Triage Agent** untuk penyaringan risiko dan eskalasi, (ii) **Support Coach Agent** yang memberikan intervensi singkat berlandaskan CBT, (iii) **Service Desk Agent** untuk tindak lanjut operasional, dan (iv) **Insights Agent** untuk analitik agregat yang menjaga privasi guna perbaikan layanan.

Kami membangun prototipe fungsional dan melakukan evaluasi berbasis skenario yang menitikberatkan pada kinerja agen dan aspek keselamatan: sensitivitas/spesifisitas triase pada prompt krisis sintetis serta waktu ke eskalasi; keandalan orkestrasi melalui tingkat keberhasilan pemanggilan fungsi dan pola *retry*; latensi ujung-ke-ujung; ketahanan terhadap *prompt injection*; serta kualitas coaching yang dinilai buta menggunakan rubrik kepatuhan CBT dan kewajaran respons. Untuk *Insights Agent*, kami hanya melaporkan pemeriksaan agregat yang sederhana (misalnya kestabilan jumlah topik di atas ambang privasi) tanpa klaim pada level individu. Hasil menunjukkan orkestrasi agen yang layak dengan latensi terkendali dan moda kegagalan yang dapat dipantau di bawah pengawasan manusia. Kami juga membahas pertimbangan etis, rancangan privasi untuk analitik agregat, keterbatasan, dan kebutuhan studi lanjutan.

Kata kunci: AI agentik; sistem multiagen; triase keselamatan; coaching CBT; human-in-the-loop; analitik agregat; LangGraph

ABSTRACT

Higher Education Institutions face rising demand for student well-being support while operating largely reactive, high-friction service models. This thesis proposes and prototypes a safety-oriented, agentic AI framework that coordinates specialized agents to enable proactive, scalable support under human oversight. The core artifact, the Safety Agent Suite, comprises: (i) a Safety Triage Agent for risk screening and escalation, (ii) a Support Coach Agent delivering brief CBT-informed micro-interventions, (iii) a lightweight Service Desk Agent for operational follow-ups, and (iv) an Insights Agent for privacy-preserving aggregate analytics to inform service improvement. Agents are orchestrated with a graph-based controller and a large language model, with guardrails for tool use, redaction, and auditability.

We implement a functional prototype and conduct scenario-based evaluations focused on agent performance and safety: triage sensitivity/specificity on synthetic crisis prompts and time-to-escalation; orchestration reliability via tool-call success and retry behavior; end-to-end latency; robustness against prompt-injection; and coaching quality via a rubric for CBT adherence and appropriateness with blinded human ratings. For the Insights Agent, we report minimal, aggregate-level sanity checks (e.g., stable topic counts under privacy thresholds) without individual-level claims. Results demonstrate the feasibility of reliable agent orchestration with bounded latency and controllable failure modes under human-in-the-loop supervision. We discuss ethical considerations, privacy by design for analytics at aggregate levels, and limitations, and outline requirements for future clinical and field studies.

Keywords: Agentic AI; Multi-Agent Systems; Safety Triage; Insights; Human-in-the-Loop; LangGraph; Student Well-being; Evaluation

CHAPTER I
INTRODUCTION

1.1 Background

Higher Education Institutions (HEIs) are facing a critical and growing challenge in supporting student well-being [1,2]. A landmark report highlights the escalating prevalence of mental health and substance use issues among student populations, urging institutions to adopt a more comprehensive support model [3]. This crisis not only jeopardizes students' academic success and personal development but also places an immense, unsustainable strain on the institutions tasked with supporting them. Recent global surveys indicate that nearly 42% of university students meet the criteria for at least one mental health disorder, while the average counselor-to-student ratio in higher education remains around 1:1,500, well above recommended levels for effective service delivery [4, 5].

The traditional support model, centered around on-campus counseling services, is fundamentally **reactive**. It relies on students to self-identify their distress and navigate the process of seeking help. This paradigm faces significant operational challenges, including insufficient staffing, long waiting lists, and an inability to provide immediate, 24/7 support, which ultimately limits access for a large portion of the student body [6]. Consequently, a critical gap persists between the need for mental health services and their actual provision, leaving many students without timely support [7].

To bridge this gap, a paradigm shift from a reactive to a **proactive** support model is imperative [7]. The engine for this evolution is **Digital Transformation**, a process that leverages technology to fundamentally reshape organizational processes and enhance value delivery within HEIs [8]. Within this context, Artificial Intelligence (AI) has emerged as a key enabling technology, with systematic reviews confirming its significant potential to analyze complex data, automate processes, and deliver personalized interventions at scale within the higher education landscape [9, 10].

However, most existing AI applications in university mental health remain limited to passive chatbots or predictive dashboards that, while insightful, depend on human operators to interpret and act upon their outputs, a limitation widely recognized as the *insight-to-action gap* [11, 12]. This thesis argues that overcoming this gap requires a more autonomous paradigm, in which AI systems do not merely predict or inform but can proactively decide and act.

This research therefore moves beyond conventional AI applications by proposing the use of **Agentic AI**. An intelligent agent is an autonomous system capable of perception, decision-making, and proactive action to achieve specific goals [13,14], representing

a new frontier in educational technology [15]. We propose that a framework built upon a system of collaborative intelligent agents, a **Multi-Agent System (MAS)**, can create a truly transformative ecosystem. Such a system would not only serve as a support tool for students but, more importantly, would function as a strategic asset for the institution, enabling data-driven decision-making, automating operational workflows, and facilitating a proactive stance on student well-being.

This framework is prototyped within the **UGM-AICare Project**, a collaborative university research initiative focused on developing AI-driven mental health and well-being tools for the Universitas Gadjah Mada (UGM) community. The project serves as the practical testbed for validating the proposed agentic system in a real institutional context.

1.2 Problem Formulation

The inefficiency and reactive nature of current university mental health support systems present a complex problem. To move towards a proactive and scalable model, this research addresses the following core challenges:

1. The primary challenge is the **design of a cohesive, safety-oriented agentic AI framework** capable of automating key institutional processes. This requires a shift from a monolithic chatbot to a multi-agent system where specialized agents handle distinct tasks, including real-time crisis detection, personalized coaching, clinical case management, and privacy-preserving analytics.
2. The technical challenge of **orchestrating a heterogeneous multi-agent system** in a robust, scalable, and secure cloud-native architecture. This involves managing stateful, long-running interactions and ensuring reliable communication between agents powered by external, non-deterministic LLMs.
3. The methodological challenge of **validating the framework's functional capabilities and potential for impact in the absence of a full-scale clinical trial**. This requires developing meaningful, scenario-based testing protocols that can effectively demonstrate the agentic workflows and their advantages over static systems.

To address these challenges, this thesis proposes and details the **Safety Agent Suite**, a framework comprised of four specialized, collaborative intelligent agents: a **Safety Triage Agent (STA)**, a **Support Coach Agent (SCA)**, a **Service Desk Agent (SDA)**, and an **Insights Agent (IA)**.

1.3 Objectives

The primary objectives of this thesis are:

1. To design an agentic AI framework, grounded in the BDI model of rational agency, that systematically bridges the 'insight-to-action' gap in institutional mental health support.
2. To implement a functional proof-of-concept prototype, the 'Safety Agent Suite,' demonstrating the orchestration of specialized agents (triage, coaching, service desk, insights) using LangGraph.
3. To evaluate the prototype's core agentic workflows through scenario-based testing, validating its capacity for proactive intervention and automated administrative action.

1.4 Research Questions

To keep the scope concrete and measurable, this thesis addresses the following research questions (RQs):

1. **RQ1 (Safety):** Can the Safety Triage Agent detect crisis intent with high sensitivity while keeping false negatives minimal, and escalate within an acceptable time budget?
2. **RQ2 (Reliability):** How reliably does the multi-agent orchestration execute tool calls and complete end-to-end workflows under schema validation, retries, and timeouts?
3. **RQ3 (Quality):** Do Support Coach responses meet a basic standard of CBT-informed guidance and appropriateness as rated by human evaluators on a small, blinded set?
4. **RQ4 (Insights, minimal):** Can the Insights Agent produce stable, aggregate-only summaries under privacy thresholds without exposing individual data?

These questions directly inform the evaluation in Chapter IV through scenario-based tests and simple, transparent metrics (e.g., sensitivity/specificity, tool-call success rate, latency percentiles, rubric scores), with human oversight preserved for safety-critical cases.

1.5 Scope and Limitations

To ensure the feasibility and focus of this research, the following boundaries are established:

1. This research is focused on the **design and prototype implementation of the agentic AI framework** (the Safety Agent Suite). Supporting content such as CBT-informed prompts is limited to curated scripts; broader engagement mechanics (e.g., badges, external loyalty systems) are out of scope.

2. The evaluation of the framework is based on **functional, scenario-based testing** of the prototype's agentic workflows. It does not measure the long-term psychological impact on students or the real-world operational savings for the institution.
3. The data utilized for testing the analytics agent will consist of **anonymized, pre-existing chat logs or simulated data** to ensure user privacy and controlled testing conditions.
4. This research does not pursue full differential privacy proofs or external ledger integrations. Instead, it implements a lightweight privacy pipeline with PII redaction and aggregate-only reporting thresholds to demonstrate **privacy-aware insights** within the prototype.

1.6 Contributions

This thesis contributes a focused blueprint and evidence base for safety-oriented agentic support:

1. **Safety pipeline specification**. A concrete guideline for triage and escalation: risk cues and scoring, guardrails and redaction steps, decision thresholds, human-in-the-loop invariants, and service targets such as time-to-escalation.
2. **Agent orchestration design**. A LangGraph view of the Safety Agent Suite—nodes, edges, and typed state schemas—plus the supporting tool-use protocol (validated schemas, idempotency, retry/backoff) that keeps workflows predictable.
3. **Evaluation assets and findings**. Scenario-based tests (synthetic crisis set, adversarial prompts, blinded coaching rubric) and their results, covering safety sensitivity, orchestration reliability, latency, and coaching quality under human oversight.

1.7 Thesis Outline

The structure of this thesis is outlined as follows:

Chapter I: Introduction. This chapter elaborates on the background of the study, the justification for the research's significance, the problem formulation to be addressed, and the specific objectives to be achieved. It also defines the scope and limitations of the research, outlines the expected contributions, and presents the overall organizational structure of the thesis report.

Chapter II: Literature Review and Theoretical Framework. This chapter surveys prior work on agentic and conversational AI for mental health, safety-critical triage systems, human-in-the-loop design, and privacy-aware analytics. It establishes the theoretical foundation that underpins the core concepts and technologies utilized in this research.

Chapter III: System Design and Architecture. This chapter outlines the methodology and technical blueprint for the system. It explains the adoption of Design Science Research and presents the system's high-level conceptual architecture, focusing on the four components of the **Safety Agent Suite**. It details the underlying cloud-native technical architecture, justifying the chosen technology stack, including the use of **LangGraph** for agent orchestration and a **FastAPI** backend for the core application logic. It also describes the database structure, user interface design, and integrated security and privacy measures like differential privacy.

Chapter IV: Implementation and Evaluation. This chapter describes the development and testing of the system prototype. This chapter details the technical environment used for implementation and demonstrates the functional prototype that was built. It then explains the testing process used to evaluate the system's performance against its design requirements. The chapter concludes by presenting the results from these tests and providing an analysis of the findings.

Chapter V: Conclusion and Future Work. This chapter summarizes the study's findings and contributions. This chapter revisits the initial research problems and presents the main conclusions drawn from the research. It concludes by offering recommendations for both the future development of the system and for subsequent research in this area.

CHAPTER II

LITERATURE REVIEW AND THEORETICAL BACKGROUND

This chapter establishes the academic context for the research. It begins by surveying the existing literature on AI applications in mental health and student support to identify the limitations of current approaches. It then details the theoretical framework and enabling technologies that provide the foundation for the proposed solution. Finally, it synthesizes these areas to formally identify the research gap this thesis addresses.

2.1 Literature Review: The Landscape of AI in University Mental Health Support

This review surveys existing research at the intersection of artificial intelligence, institutional support systems, and student mental health. The aim is to contextualize the present work by examining the evolution and limitations of current approaches, thereby setting the stage for the introduction of a more advanced, agentic framework.

2.1.1 Conversational Agents for Mental Health Support

The application of conversational agents in mental health has evolved significantly, from early experiments in simulating dialogue to sophisticated, evidence-based therapeutic tools. This evolution reveals both the immense potential of these technologies and the persistent operational limitations that motivate the current research.

2.1.1.1 Evolution from Rule-Based Systems to LLM-Powered Agents

The concept of using a computer program for therapeutic dialogue dates back to Weizenbaum's ELIZA (1966), a system that used simple keyword matching and canned response templates to mimic a Rogerian psychotherapist [16, 17]. While a landmark in human-computer interaction, ELIZA and subsequent rule-based systems lacked any true semantic understanding, memory, or capacity for evidence-based intervention. Their primary limitation was their inability to move beyond superficial pattern recognition, leading to brittle and often nonsensical conversations when faced with inputs outside their predefined rules [16].

The advent of Large Language Models (LLMs) has catalyzed a paradigm shift. Modern conversational agents, powered by Transformer architectures, can generate fluent, empathetic, and context-aware responses. These models are pre-trained on vast text corpora, enabling them to understand linguistic nuance and generate human-like text. This has allowed for the development of agents that can engage in more meaningful, multi-turn conversations, moving beyond simple question-answering to provide more

substantive support [17].

2.1.1.2 Therapeutic Applications and Efficacy

Contemporary mental health chatbots leverage LLMs to deliver a range of evidence-based interventions. A primary application is the delivery of psychoeducation and structured exercises from therapeutic modalities like Cognitive Behavioral Therapy (CBT). Systems such as Woebot have been the subject of randomized controlled trials (RCTs), which have demonstrated their efficacy in reducing symptoms of depression and anxiety among university students by delivering daily, brief, conversational CBT exercises [18, 19]. Other platforms, like Tess, have shown similar positive outcomes by providing on-demand emotional support and coping strategies.

These tools offer several key advantages:

- **Accessibility and Scalability:** They are available 24/7, overcoming the time and resource constraints of traditional human-led services.
- **Anonymity:** They provide a non-judgmental and anonymous space for users to disclose their feelings, which can lower the barrier for individuals who fear stigma [20].

2.1.1.3 The Dominant Reactive Paradigm and Its Limitations

Despite their technological sophistication and therapeutic potential, the fundamental operational model of these applications remains overwhelmingly **reactive and user-initiated**. They are designed as standalone tools that depend on the student to possess the self-awareness to recognize their distress, the motivation to seek help, and the knowledge of the tool's existence.

This paradigm fails to account for significant, well-documented barriers to help-seeking. Research shows that many individuals, particularly young adults, do not seek professional help for mental health issues due to factors including self-stigma, fear of judgment, and a desire for self-reliance [21, 22]. Furthermore, the very symptoms of mental health conditions, such as the anhedonia and executive dysfunction associated with depression, can severely impair an individual's ability to initiate action and seek support [23].

A systematic review of mental health chatbots for university students concluded that while these tools are promising, their primary limitation is their passive nature; they do not and cannot initiate contact or intervene based on a student's changing needs unless the student opens the app [24]. This leaves the most vulnerable studentsm, those who are not actively seeking help, unsupported, creating a critical gap in the continuum of care that this thesis aims to address.

2.1.2 Data Analytics for Proactive Student Support

Parallel to the development of conversational AI, the field of higher education has seen a rise in the use of data analytics to support student success. This section reviews the evolution of these analytical approaches, from established learning analytics to the more nascent field of well-being analytics, and identifies the key limitations that motivate the design of the agents.

2.1.2.1 Learning Analytics for Academic Intervention

The domain of **Learning Analytics** is well-established and focuses on the "measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimising learning and the environments in which it occurs" [25]. Typically, these systems analyze data from institutional sources such as the Learning Management System (LMS), student information systems, and library databases. By modeling variables like assignment submission times, forum participation, and grades, institutions can build predictive models to identify students at high risk of academic failure or dropout [26]. These systems have proven effective in enabling timely academic interventions, such as targeted tutoring or advisor outreach, thereby improving student retention and success rates.

2.1.2.2 The Challenge of Well-being Analytics

More recently, researchers have attempted to extend the principles of learning analytics to the more complex and sensitive domain of student well-being. The goal is to create early-warning systems by identifying behavioral proxies for mental distress. Studies have explored the use of non-academic data sources, such as campus card usage for building access, meal plan data, and social event attendance, to find correlations with well-being outcomes [27]. For example, a sudden decrease in social activity or irregular campus attendance could be interpreted as a potential indicator of withdrawal or depression.

However, this approach is fraught with significant theoretical and practical challenges. Firstly, the "signal-to-noise" ratio is extremely low; the link between such indirect behavioral data and a student's internal mental state is often weak, correlational, and highly prone to misinterpretation [28]. A student may miss meals for many reasons other than depression. Secondly, these methods raise profound ethical questions regarding student privacy and surveillance, as they involve monitoring non-academic aspects of student life, often without explicit, ongoing consent for this specific purpose [27, 28].

A more direct, and arguably more ethical, source of data is the language students use when interacting with university services. The text from chat logs, when properly

anonymized, provides a direct window into student concerns. The application of sentiment analysis and topic modeling to this textual data can yield far more reliable insights into the specific stressors affecting the student population at any given time. This approach, which is central to the design of the Analytics Agent, shifts the focus from inferring mental state from indirect behaviors to directly analyzing the expressed concerns of the student body [27].

2.1.2.3 The Insight-to-Action Gap

Whether based on academic, behavioral, or textual data, a critical limitation plagues nearly all current analytical systems in higher education: the **insight-to-action gap** [11]. The output of these systems is almost universally a dashboard, a report, or an alert delivered to a human administrator (e.g., a counselor, dean, or advisor) [12]. This administrator must then manually interpret the data, decide on an appropriate intervention strategy, and execute it.

This manual process creates a severe bottleneck that fundamentally limits the scalability, speed, and personalization of any proactive effort [29]. An administrator may be able to respond to a handful of individual alerts, but they cannot manually orchestrate a personalized outreach campaign to hundreds of students who may be exhibiting early signs of exam-related stress identified by a topic model. The manual-execution step prevents the institution from fully capitalizing on the proactive insights generated by its analytical systems. It is this specific gap that the proposed **Safety Coaching Agent** and **Safety Triage Agent** is designed to close by automating the link between data-driven insight and scalable, targeted outreach.

2.2 Theoretical Background

To address the limitations of reactive, disconnected support systems, a new architectural approach is required. This section details the theoretical framework and enabling technologies that provide the foundation for the proposed agentic AI system. These concepts are presented as the necessary components to build a proactive, integrated, and autonomous solution.

2.2.1 Foundational Principles of the Framework

Beyond the technical architecture, the proposed framework is grounded in several key strategic and ethical principles that justify its design and purpose. These concepts from service design, management science, and data ethics provide the theoretical motivation for shifting how institutional support is delivered.

2.2.1.1 Proactive vs. Reactive Support Models

The traditional approach to institutional support, particularly in mental health, is predominantly **reactive**. This model, common in service design, operates on a "break-fix" basis, where the service delivery is initiated only after a user (in this case, a student) self-identifies a problem and actively seeks a solution [30]. This places the onus of initiation entirely on the individual, creating significant barriers to access such as stigma, lack of awareness, or the inability to act during a crisis. In contrast, a **proactive support model** aims to anticipate needs and intervene before a problem escalates. Drawing from principles in preventative healthcare and proactive customer relationship management, this model uses data to identify patterns and risk factors, enabling the institution to offer timely, relevant support to at-risk cohorts [31, 32]. This thesis is an explicit attempt to architect a system that facilitates this strategic shift from a reactive to a proactive support paradigm.

Formalization of Support Models To formalize this distinction, a reactive support system operates conditionally based on user initiation:

$$\text{Support}(t) = \begin{cases} f(\text{request}_t) & \text{if student initiates} \\ \emptyset & \text{otherwise} \end{cases} \tag{2-1}$$

In contrast, a proactive system continuously monitors and responds to indicators of need:

$$\text{Support}(t) = g(\text{risk}(t), H_{t-\Delta t:t}, \text{trends}_t) \tag{2-2}$$

where $H_{t-\Delta t:t}$ represents recent interaction history and trends_t captures population-level signals from analytics.

The help-seeking barrier can be modeled as:

$$P(\text{seek help}) = f(\text{severity}) - \beta(\text{stigma}, \text{awareness}, \text{fatigue}) \tag{2-3}$$

where β captures systemic barriers such as stigma, lack of awareness, and decision fatigue. When $P(\text{seek help}) < 0$, students in crisis remain silent—justifying the need for proactive intervention that does not depend on self-initiation.

2.2.1.2 Data-Driven Decision-Making in Higher Education

The concept of **Data-Driven Decision-Making (DDDM)** posits that strategic decisions should be based on objective data analysis and interpretation rather than solely on intuition or tradition [31, 32]. In higher education, this has manifested as the field of

learning analytics, where student data is used to improve learning outcomes and retention. This framework extends that principle to student well-being. The **Insights Agent** is the core enabler of DDDM for the university's support services. By autonomously processing anonymized interaction data to identify trends, sentiment shifts, and emerging topics of concern, it provides administrators with actionable, empirical evidence. This allows the institution to move beyond anecdotal evidence and allocate resources, such as workshops, counselors, or targeted information campaigns, to where they are most needed, thereby optimizing the efficiency and impact of its support ecosystem [33].

2.2.1.3 Privacy by Design (PbD)

Given the highly sensitive nature of mental health data, the framework's architecture is guided by the principles of **Privacy by Design (PbD)**. PbD is an internationally recognized framework, formalized in ISO 31700, which dictates that privacy should be the default, embedded into the design and architecture of systems from the outset rather than being an add-on feature [34,35]. Key principles include being proactive not reactive, making privacy the default setting, and providing end-to-end security. A direct implementation of PbD within this framework is the Data Anonymization Pipeline. This process ensures that Personally Identifiable Information (PII) is identified and redacted from all chat logs before they are stored for analysis. Furthermore, access to the administrative dashboard is controlled by a strict Role-Based Access Control (RBAC) mechanism, ensuring that only authorized personnel can view sensitive data. These measures, combined with standard security practices like data encryption, embed privacy and security directly into the system's architecture from the outset [33, 34]. This demonstrates a commitment to building a system that is not only effective but also fundamentally ethical and secure.

2.2.2 Agentic AI and Multi-Agent Systems (MAS)

The paradigm of Artificial Intelligence (AI) has evolved significantly from systems that perform singular, reactive tasks to those that exhibit autonomous, proactive, and social behaviors. A cornerstone of this evolution is the concept of an **intelligent agent**. An agent is not merely a program; it is a persistent computational entity with a degree of autonomy, situated within an environment, which it can both perceive and upon which it can act to achieve a set of goals or design objectives [36]. The defining characteristic of an agent is its **autonomy**, its capacity to operate independently, making decisions and initiating actions without direct, constant human intervention. This is distinct from traditional objects, which are defined by their methods and attributes but do not exhibit control over their own behavior [14].

To operationalize this concept, this thesis formally introduces a framework built upon four distinct, specialized intelligent agents that form the **Safety Agent Suite**. Each

agent is designed to address a specific challenge outlined in Chapter 1, and together they form the core of the proposed proactive support system. These agents are:

- The **Safety Triage Agent (STA)**, responsible for real-time risk assessment and crisis intervention.
- The **Support Coach Agent (SCA)**, responsible for delivering personalized, evidence-based coaching.
- The **Service Desk Agent (SDA)**, responsible for managing clinical case workflows and administrative tasks.
- The **Insights Agent (IA)**, responsible for privacy-preserving data analysis and trend identification.

The theoretical underpinnings of these agents' architecture and behavior are drawn from established models of rational agency and multi-agent systems, as detailed below.

Fundamentally, an agent's operation is defined by a continuous cycle of perception, reasoning (or deliberation), and action. It perceives its environment through virtual **sensors** (e.g., data feeds, API calls, database queries) and influences that environment through its **actuators** (e.g., sending emails, generating reports, invoking other services) [37]. A prominent and highly relevant architecture for designing such goal- oriented agents is the **Belief-Desire-Intention (BDI)** model [37, 38]. This model provides a framework for rational agency that mirrors human practical reasoning:

- **Beliefs:** This represents the informational state of the agent, its knowledge about the environment, which may be incomplete or incorrect. For the **Insights Agent**, beliefs correspond to the current understanding of student well-being trends derived from anonymized data.
- **Desires:** These are the motivational states of the agent, representing the objectives or goals it is designed to achieve. Desires can be seen as the potential tasks the agent could undertake, such as the **Support Coach Agent's** overarching goal to "deliver personalized coaching."
- **Intentions:** This represents the agent's commitment to a specific plan or course of action. An intention is a desire that the agent has chosen to actively pursue. For instance, the **Safety Triage Agent**, upon identifying a high-severity conversation, forms an intention to immediately route the user to emergency resources.

The BDI framework allows for the design of agents that are not merely reactive but are proactive and deliberative, capable of reasoning about how to best achieve their goals given their current beliefs about the world [14, 38].

To formally ground the proposed framework in this established model, the roles

and logic of each of the four agents are mapped to the BDI components in Table 2.1. This mapping clarifies how each agent perceives its environment, formulates its objectives, and decides on a concrete course of action, allowing for the design of agents that are not merely reactive but are proactive and deliberative, capable of reasoning about how to best achieve their goals given their current beliefs about the world.

Table 2.1. Mapping of the Agentic Framework to the BDI Model.

Agent	**Beliefs** *(Informational State)*	**Desires** *(Motivational Goals)*	**Intentions** *(Committed Plans)*
STA	• User's conversation history • Severity classification model • Emergency resources directory	• Assess immediate risk level • Provide appropriate support	• Escalate high-severity cases • Display emergency contacts
SCA	• User goals & history • Evidence-based intervention library (CBT)	• Deliver personalized coaching • Guide through exercises	• Deliver specific CBT exercise • Provide empathetic responses
SDA	• Clinical case status • Counselor availability • User appointment requests	• Manage case workflows • Schedule appointments	• Find available appointment slots • Create and update case notes
IA	• Anonymized conversation database • Last report timestamp • Known topic models	• Identify emerging trends • Quantify sentiment shifts	• Generate weekly summary reports • Execute database queries

Formalization of the BDI Cycle The BDI model operates through continuous state updates that govern how agents perceive, deliberate, and act. The cycle can be formalized as:

Belief Update: An agent's beliefs are updated as new information is perceived:

$$Bel_{t+1} = Bel_t \cup \{\text{percept}_t\} \setminus \{\text{expired beliefs}\} \tag{2-4}$$

where percept_t represents new observations from the environment, and expired beliefs are those that are no longer valid or relevant.

Desire Selection: The agent filters potential goals based on its current beliefs:

$$Des_t = \text{filter}(\text{Options}_t, Bel_t) \tag{2-5}$$

where Options_t represents all possible goals the agent could pursue, and the filter function selects those that are feasible given current beliefs.

Intention Formation: The agent commits to a specific plan of action through deliberation:

$$Int_t = \text{deliberate}(Des_t, Bel_t, Int_{t-1}) \tag{2-6}$$

where the deliberation process considers current desires, beliefs, and previous intentions to form a committed plan.

For example, in the **Safety Triage Agent (STA)**, percept_t is the incoming student message M_t, beliefs include prior conversation context and crisis patterns, desires map to intervention goals (de-escalation, resource connection), and intentions become the selected action (escalate to SDA, provide coping strategy, or direct to emergency resources).

When multiple agents, each with its own goals and capabilities, co-exist and interact within a shared environment, they form a **Multi-Agent System (MAS)**. An MAS is a system in which the overall intelligent behavior and functionality are a product of the collective, emergent dynamics of its constituent agents [39, 40]. The power of an MAS lies in its ability to solve problems that would be difficult or impossible for a monolithic system or a single agent to handle. This is achieved through social interaction, primarily:

- **Coordination and Cooperation:** Agents must coordinate their actions to avoid interference and cooperate to achieve common goals. In this thesis, the **Insights**, **Support Coach**, **Safety Triage**, and **Service Desk** agents must cooperate: the Insights Agent provides the data-driven insights (beliefs) that the Support Coach Agent uses to form its outreach plans (intentions), while the Safety Triage Agent handles immediate, real-time needs that may fall outside the other agents' scopes, and the Service Desk Agent manages the administrative follow-up.
- **Negotiation:** When agents have conflicting goals or must compete for limited resources, they must be able to negotiate to find a mutually acceptable compromise [41, 42].
- **Communication:** Effective interaction requires a shared Agent Communication Language (ACL), such as FIPA-ACL or KQML, which defines the syntax and semantics for messages, allowing agents to perform actions like requesting information, making proposals, and accepting or rejecting tasks [43, 44].

Therefore, this thesis leverages the MAS paradigm by designing a framework composed of four specialized, collaborative agents. Their individual, goal-directed behaviors, orchestrated within a hybrid architecture, work in concert to achieve the overarching systemic objective: transforming institutional mental health support from a reactive model to a proactive, data-driven ecosystem.

2.2.2.1 Mathematical Formalization of Agent Decision Functions

To operationalize the BDI model for the Safety Agent Suite, each agent's core decision-making process is formalized as a mathematical function mapping inputs to outputs. This formalization bridges the theoretical BDI framework to the practical implementation described in Chapter 3.

Safety Triage Agent (STA) The STA assesses risk level $R_t \in \{0, 1, 2, 3\}$ from student message M_t:

$$R_t = f_{STA}(M_t; \theta_{LLM}, \mathcal{C}) \tag{2-7}$$

where θ_{LLM} represents the LLM parameters (Gemini 2.5 Flash) and $\mathcal{C}$ is the crisis pattern corpus used for contextual understanding. The discrete risk level maps to severity categories:

$$\text{severity}(R_t) = \begin{cases} \text{low} & R_t = 0 \\ \text{moderate} & R_t = 1 \\ \text{high} & R_t = 2 \\ \text{critical} & R_t = 3 \end{cases} \tag{2-8}$$

This classification determines subsequent routing: moderate risk ($R_t = 1$) triggers supportive coaching via SCA, while high or critical risk ($R_t \geq 2$) initiates immediate escalation to the Service Desk Agent for clinical case management.

Support Coach Agent (SCA) The SCA generates therapeutic response A_t given the current message and conversation history:

$$A_t = f_{SCA}(M_t, H_{t-1}; \theta_{LLM}, \mathcal{I}) \tag{2-9}$$

where $H_{t-1} = \{(M_0, A_0), \ldots, (M_{t-1}, A_{t-1})\}$ is the conversation history capturing previous exchanges, and $\mathcal{I}$ represents the intervention library containing evidence-based therapeutic frameworks (CBT, Motivational Interviewing). The history dependency enables the agent to maintain therapeutic continuity and adapt interventions based on student progress.

Insights Agent (IA) The IA computes aggregate metric μ from anonymized message set $\mathcal{M}$:

$$\mu = f_{IA}(\mathcal{M}, q; \mathcal{K}) \tag{2-10}$$

where q represents the analytical query (e.g., crisis trend analysis, topic modeling, sentiment aggregation) and $\mathcal{K}$ enforces privacy constraints such as k-anonymity and query result suppression. The IA's output informs institutional decision-making by quantifying population-level trends while preserving individual privacy.

Service Desk Agent (SDA) The SDA determines administrative action α_t based on case state and available resources:

$$\alpha_t = f_{SDA}(C_t, R_{avail}; \theta_{LLM}) \tag{2-11}$$

where C_t represents the current case state (severity, student information, appointment history) and R_{avail} denotes available resources (counselor schedules, emergency contact protocols). Actions include appointment scheduling, case note creation, and resource allocation.

These formalizations establish the mathematical foundation for the multi-agent coordination described in subsequent sections and implemented in Chapter 3.

2.2.3 Large Language Models (LLMs)

Large Language Models (LLMs) are a class of deep learning models that have demonstrated remarkable capabilities in understanding and generating human-like text. The architectural foundation for virtually all modern LLMs, including the Gemini models used in this research, is the **Transformer architecture**, first introduced by Vaswani et al. [45]. The Transformer's key innovation is the **self-attention mechanism**, which allows the model to dynamically weigh the importance of different words in an input sequence when processing and generating language. This enables the model to capture complex, long-range dependencies and contextual relationships far more effectively than its predecessors, such as Recurrent Neural Networks (RNNs) [46, 47].

The Self-Attention Mechanism The self-attention mechanism computes contextual representations by relating different positions in a sequence to each other. Given an input sequence, the mechanism computes three matrices: Query (Q), Key (K), and Value (V), each derived through learned linear projections of the input embeddings. The attention operation is then defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2-12}$$

where d_k is the dimension of the key vectors. The scaling factor $\sqrt{d_k}$ prevents the dot products from growing too large, which would push the softmax function into regions with extremely small gradients.

The attention weights, computed by the softmax of the scaled dot products between queries and keys, determine how much each position in the sequence should attend to every other position. These weights are then used to compute a weighted sum of the value vectors, producing contextually-aware representations.

Modern Transformers employ **multi-head attention**, which applies multiple attention operations in parallel, each with different learned projections:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2-13}$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$ and W^O is an output projection matrix. This allows the model to attend to information from different representation subspaces simultaneously. For instance, Gemini 2.5 employs 32 attention heads per layer, enabling it to capture diverse linguistic patterns and semantic relationships concurrently.

The core operation of a Transformer-based model involves processing input text through a series of encoding and/or decoding layers. The process can be conceptualized as follows:

1. **Tokenization and Embedding:** Input text is first broken down into smaller units called tokens. Each token is then mapped to a high-dimensional vector, or an "embedding," that represents its semantic meaning.
2. **Positional Encoding:** Since the self-attention mechanism does not inherently process sequential order, a positional encoding vector is added to each token embedding to provide the model with information about the word's position in the sequence.
3. **Self-Attention Layers:** The sequence of embeddings passes through multiple self-attention layers. In each layer, the model calculates attention scores for every token relative to all other tokens in the sequence, effectively learning which parts of the input are most relevant for understanding the context of each specific token.
4. **Feed-Forward Networks:** Each attention layer is followed by a feed-forward neural network that applies further transformations to each token's representation.
5. **Output Generation:** The model's final output is a probability distribution over its entire vocabulary for the next token in the sequence. The model then typically selects the most likely token (or samples from the distribution) and appends it to the input, repeating the process autoregressively to generate coherent text [46].

This research utilizes a cloud-based API model strategy, leveraging the Gemini

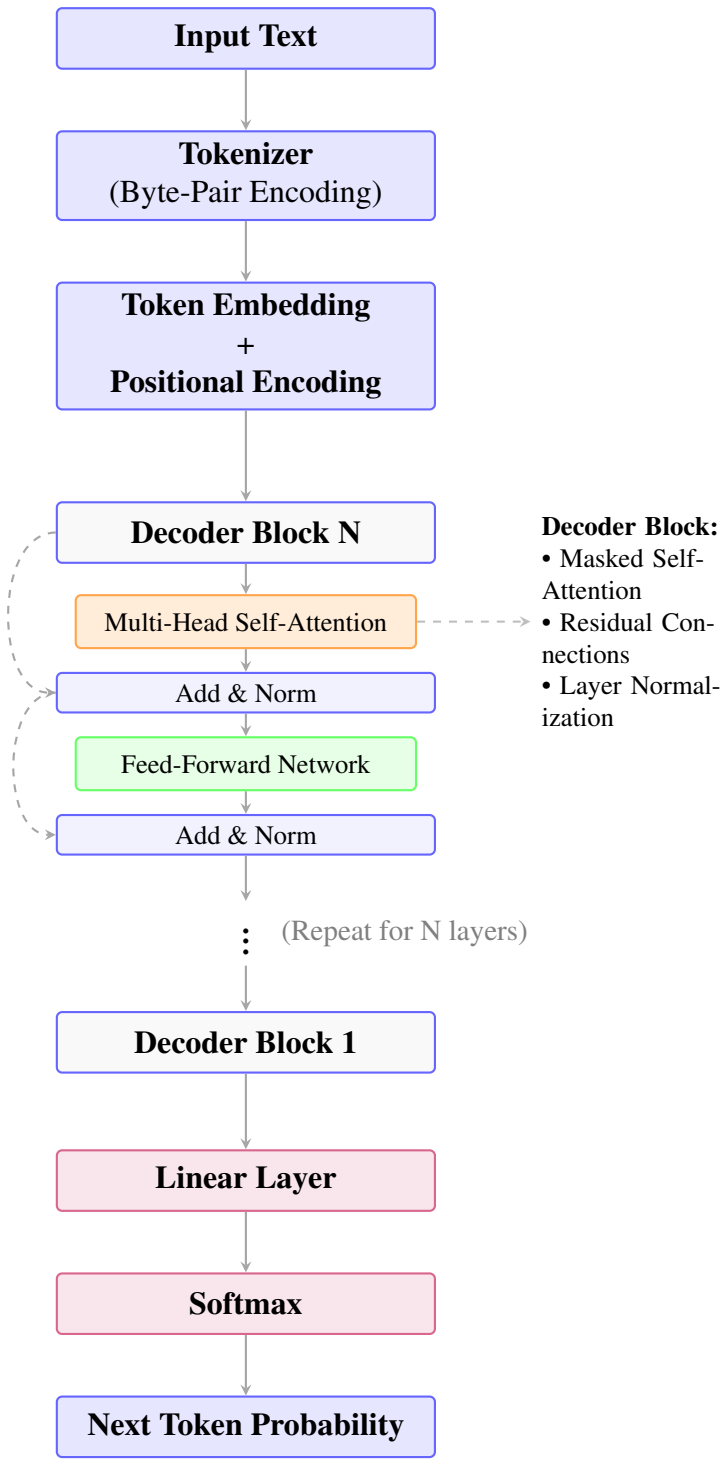


Figure 2.1. A simplified view of the decoder-only Transformer architecture used in generative LLMs. The model processes input embeddings through multiple layers (blocks) of masked multi-head self-attention and feed-forward networks with residual connections to predict the next token in a sequence.

2.5 family of models to balance performance, privacy, and capability. The Gemini models represent Google's state-of-the-art, natively multimodal foundation models, available in various sizes (e.g., Gemini Pro). Unlike models trained solely on text, Gemini was pre-trained from the ground up on multiple data modalities, giving it more sophisticated reasoning capabilities [48]. In this framework, a powerful model like Gemini 2.5 Pro is accessed via a secure API for all agentic tasks [49], from the real-time conversation handling of the Safety Triage Agent to the complex, non-sensitive tasks, such as the weekly trend analysis performed by the Insights Agent.

2.2.3.1 Cloud-Based API Models: The Gemini 2.5 Family

The framework integrates a state-of-the-art, proprietary model accessed via a cloud API. The Gemini family, specifically the flagship **Gemini 2.5 Flash** model, serves this role, providing a level of reasoning and multimodal understanding that is critical for handling the most complex tasks and ensuring system robustness. While a detailed architectural schematic is not public, in line with the proprietary nature of frontier AI models, its capabilities have been extensively documented by Google through official developer guides and announcements [48, 49].

Gemini 2.5 builds upon the efficient **Mixture-of-Experts (MoE) Transformer** architecture of its predecessors. In an MoE architecture, the model is composed of numerous smaller "expert" neural networks. For any given input, a routing mechanism activates only a sparse subset of these experts. This allows the model to have a very large total parameter count, enabling vast knowledge and capability, while keeping the computational cost for any single inference relatively low [48].

The strategic role of Gemini 2.5 in this framework is defined by its next-generation capabilities:

- **Native Multimodality with Expressive Audio:** A significant architectural leap in Gemini 2.5 is its native handling of audio [50]. Unlike models that first transcribe audio to text, Gemini 2.5 processes audio streams directly. This allows it to understand not just the words, but also the nuances of human speech such as tone, pitch, and prosody, which is invaluable for a mental health application where user sentiment is key.
- **Controllable Reasoning and "Thinking Time":** Gemini 2.5 introduces a "thinking budget," a mechanism that allows developers to control the trade-off between response latency and reasoning depth [48]. For high-frequency tasks performed by the Safety Triage Agent, a lower budget can ensure speed. For complex analytical tasks required by the Insights Agent, a higher budget can be allocated to allow for more thorough reasoning, providing granular control over both cost and quality.

- **Advanced Agentic Capabilities and Tool Use:** The model is explicitly designed to power advanced agents. It features more reliable and sophisticated function calling, enabling seamless integration with external tools and APIs [48]. This is essential for the Service Desk Agent to execute multi-step plans, such as scheduling an appointment based on a user's request.
- **High-Fidelity Reasoning:** As a frontier model, Gemini 2.5 serves as the high-capability engine for all requests, ensuring service continuity and the highest quality output.

By integrating Gemini 2.5 via its API, the agentic framework gains access to state-of-the-art reasoning power on demand, ensuring that it can handle a wide spectrum of tasks with both efficiency and exceptional quality.

2.2.4 LLM Orchestration Frameworks

While LLMs provide powerful reasoning capabilities, they are inherently stateless and lack direct access to external data or tools. An LLM, in isolation, cannot query a database, call an API, or access a private document. To build sophisticated, stateful applications that overcome these limitations, an orchestration framework is required.

2.2.4.1 LangChain: The Building Blocks of LLM Applications

LangChain is an open-source framework designed specifically for this purpose, providing the essential "glue" to connect LLMs with external resources and compose them into complex applications [51, 52]. The core philosophy of LangChain is to provide modular components that can be "chained" together to create complex workflows. The most recent and fundamental abstraction in LangChain is the **LangChain Expression Language (LCEL)**. LCEL provides a declarative, composable syntax for building chains, where the pipe ('|') operator streams the output of one component into the input of the next. Every component in an LCEL chain is a "Runnable," a standardized interface that supports synchronous, asynchronous, batch, and streaming invocations, making it highly versatile for production environments [52, 53].

A simple LCEL chain can be represented as:

$$\text{Chain} = \text{PromptTemplate} \mid \text{LLM} \mid \text{OutputParser}$$

In this sequence, user input is first formatted by a 'PromptTemplate', the result is passed to the 'LLM' for processing, and the LLM's raw output is then transformed into a structured format (e.g., JSON) by an 'OutputParser'.

For this thesis, the most critical application of LangChain is its ability to create **agents**. A LangChain agent uses an LLM not just for text processing, but as a reasoning

engine to make decisions. This is often based on a framework known as **ReAct (Reasoning and Acting)**, which enables the LLM to synergize reasoning and action [51, 54]. The agent is given access to a set of **Tools**, which are simply functions that can interact with the outside world (e.g., a database query function, a file reader, a web search API). The agent's operational loop, managed by an **Agent Executor**, can be formalized as an iterative process.

Let G be the initial goal and H_t be the history of actions and observations up to step t. The process at each step t is:

1. **Reasoning (Thought Generation):** The agent generates a thought th_t and a subsequent action a_t by sampling from the LLM's conditional probability distribution, given the goal and the history so far.

$$(th_t, a_t) \sim p(th, a|G, H_{t-1}; \theta_{LLM})$$

   The prompt to the LLM contains the goal and the trajectory of previous thoughts, actions, and observations, guiding its next decision.

2. **Action Execution:** The Agent Executor parses a_t to identify the chosen tool and its input, then executes it to produce an observation, o_t.

$$o_t = \text{ExecuteTool}(a_t)$$

3. **History Augmentation:** The new observation is appended to the history, forming the context for the next iteration.

$$H_t = H_{t-1} \oplus (a_t, o_t)$$

   This loop continues until the LLM determines the goal G is met and generates a final answer.

This iterative loop is what transforms a passive LLM into a proactive, problem-solving agent. For example, the **Insights Agent** in this framework, when tasked with "summarizing student stress trends," would use this loop to formulate a SQL query (Thought and Action), execute it (Observation), and then use the results to generate a final summary. This orchestration is fundamental to enabling the autonomous capabilities central to this thesis.

2.2.4.2 LangGraph: Orchestrating Multi-Agent Systems

While LangChain's standard agent executors are powerful, they are often designed for linear, sequential execution paths. For a sophisticated multi-agent system like

the **Safety Agent Suite**, where agents must collaborate, hand off tasks, and operate in a cyclical, stateful manner, a more robust orchestration mechanism is required. This is the role of **LangGraph**, an extension of LangChain designed for building durable, stateful, multi-agent applications by modeling them as cyclical graphs [55, 56].

The core concept of LangGraph is to represent the agentic workflow as a **state graph**. This is a directed graph where nodes represent functions or LLM calls (the "work" to be done) and edges represent the conditional logic that directs the flow of execution from one node to another. A central **State** object is passed between nodes, allowing each agent or tool to read the current state, perform its function, and then update the state with its results. This creates a persistent, auditable record of the agent's operations [53, 57].

A LangGraph workflow can be defined by the following components:

- **State Graph:** The overall structure, $G = (N, E)$, where N is a set of nodes and E is a set of directed edges. The graph's state is explicitly defined by a state object that is passed and updated throughout the execution.
- **Nodes:** Each node represents an agent or a tool. When called, a node receives the current state object as input and returns a dictionary of updates to be applied to the state. For example, the 'Safety Triage Agent' node would take the user's message from the state, process it, and return an update specifying the assessed risk level.
- **Edges:** Edges connect the nodes and control the flow of the application. LangGraph supports **conditional edges**, which are crucial for agentic behavior. After a node executes, a routing function is called to inspect the current state and decide which node to move to next [52, 53]. For example, after the 'Safety Triage Agent' runs, a conditional edge might route the workflow to the 'Service Desk Agent' if the risk is moderate, or directly to an "escalate" tool if the risk is critical.

State Transition Semantics The stateful execution of a LangGraph workflow is governed by formal state update rules. Each node in the graph transforms the shared state through a state update function:

$$S_{t+1} = \text{node}_i(S_t) = S_t \oplus \Delta S_i \tag{2-14}$$

where S_t represents the current state at time step t, ΔS_i is the update produced by node i, and $\oplus$ denotes the state merging operation (where new fields override existing values while preserving unmodified fields).

Conditional edges implement routing logic via predicate functions that inspect the current state. For the Safety Agent Suite, the routing after risk assessment can be

formalized as:

$$\text{next}(S_t) = \begin{cases} \text{escalate\_to\_sda} & \text{if } S_t.\text{risk\_level} \geq 2 \\ \text{provide\_coaching} & \text{if } S_t.\text{risk\_level} = 1 \\ \text{END} & \text{if } S_t.\text{risk\_level} = 0 \end{cases} \tag{2-15}$$

This formalization enables dynamic multi-agent orchestration where the STA's risk assessment (R_t) determines subsequent workflow paths: moderate risk routes to the SCA for therapeutic intervention, high or critical risk escalates to the SDA for clinical case creation, and low risk concludes the interaction. The explicit state management ensures that all downstream agents have access to the complete conversation context, enabling informed decision-making throughout the workflow.

More generally, for any conditional routing decision, the next node is determined by a routing function ρ:

$$\text{next\_node} = \rho(S_t) \in N \cup \{\text{END}\} \tag{2-16}$$

where ρ maps the current state to either another node in the graph or a terminal state, enabling arbitrary workflow complexity including loops, parallel execution, and human-in-the-loop interventions.

This cyclical, stateful approach provides several key advantages for this framework:

1. **Explicit Multi-Agent Collaboration:** LangGraph allows for the explicit definition of workflows where different agents are called in sequence or in parallel, and their outputs are used to inform the next step [57, 58]. This is essential for the **Safety Agent Suite**, where the 'Insights Agent''s output must trigger the 'Support Coach Agent'.
2. **State Management and Durability:** Because the state is explicitly managed, the agent's "memory" of the conversation and its previous actions is robust. The graph's execution can be paused, resumed, and inspected, which is vital for long-running, interactive coaching sessions.
3. **Flexibility and Control:** Unlike the more constrained loops of standard agent executors, LangGraph allows for the creation of arbitrary cycles. An agent can loop, retry a tool call if it fails, or route to a human-in-the-loop for verification, providing a much higher degree of control and reliability for a safety-critical application [59,60].

By using LangGraph to orchestrate the **Safety Agent Suite**, this framework moves beyond simple, linear agentic loops and implements a true multi-agent system capable of

complex, stateful, and collaborative problem-solving [55, 58].

2.2.4.3 Real-Time Performance Requirements for Safety-Critical Applications

For mental health support systems, response latency directly impacts user experience and, critically, the effectiveness of crisis intervention. To formalize these requirements, Service Level Objectives (SLOs) define the maximum acceptable latency for each agent operation.

Let T_{agent} denote the response time random variable for an agent. The p95 SLO (95th percentile) ensures that 95% of requests are handled within the target latency:

$$\Pr[T_{agent} \leq T_{target}] \geq 0.95 \tag{2-17}$$

Agent-Specific Latency Targets Different agents have different latency requirements based on their role:

Safety Triage Agent (STA): As the first-line crisis detection system, the STA requires near-instantaneous response:

$$T_{STA} \leq 250\text{ms} \quad (p95) \tag{2-18}$$

This target ensures crisis indicators are identified within human perceptual thresholds, enabling immediate escalation when necessary.

End-to-End User Experience: The complete interaction flow from user message to final response must maintain acceptable responsiveness:

$$T_{E2E} = T_{STA} + T_{routing} + T_{SCA} + T_{network} \leq 1500\text{ms} \quad (p95) \tag{2-19}$$

where $T_{routing}$ represents orchestration overhead, T_{SCA} is the Support Coach Agent's response generation time, and $T_{network}$ accounts for transmission delays.

Performance Monitoring and Alerting Node-level latency is continuously monitored through execution tracking:

$$T_{node}(i) = t_{complete}(i) - t_{start}(i) \tag{2-20}$$

where $t_{start}(i)$ and $t_{complete}(i)$ are timestamps for node initiation and completion, respectively.

Critical alerts are triggered when latency exceeds threshold values:

$$\text{alert}(i) = \begin{cases} \text{WARNING} & \text{if } T_{node}(i) > \theta_{warning} \\ \text{CRITICAL} & \text{if } T_{node}(i) > \theta_{critical} \end{cases} \tag{2-21}$$

where typical thresholds are $\theta_{warning} = 30\text{s}$ and $\theta_{critical} = 2\text{min}$ for backend operations.

Capacity Planning The system must maintain performance under load. Throughput requirements specify the range of concurrent sessions the system must handle:

$$\lambda_{concurrent} \in [500, 1000] \text{ sessions} \tag{2-22}$$

This capacity ensures the system scales to institutional needs while maintaining the latency targets defined above. Capacity testing validates these requirements through simulated load scenarios, as detailed in Chapter 3.

2.3 Synthesis and Identification of the Research Gap

The preceding review of the literature and theoretical landscape reveals a critical disconnect. On one hand, the field has produced increasingly sophisticated but fundamentally **reactive** conversational agents for mental health. On the other, it has developed proactive institutional analytics that remain bottlenecked by a reliance on **manual intervention**. The failure of the existing literature is not in the individual components, but in the lack of integration between them.

This creates a significant and unaddressed research gap: the need for an **integrated, autonomous, and proactive framework** that can systemically bridge the chasm from data-driven insight to automated, personalized intervention and administrative action. Current systems are not designed as a cohesive ecosystem. The analytical tools do not automatically trigger the intervention tools, the conversational agents do not seamlessly hand off tasks to administrative agents, and the user-facing support does not operate with an awareness of the broader institutional context provided by analytics.

The central argument of this thesis is that the next frontier in institutional mental health support lies not in the incremental improvement of any single component, but in the **synergistic integration of multiple specialized agents** into a single, closed-loop system. Such a system, architected as a Multi-Agent System (MAS), is capable of emergent behaviors that are more than the sum of its parts.

Therefore, this research directly addresses the identified gap by proposing and prototyping a novel agentic AI framework, the **Safety Agent Suite**, where:

- An **Insights Agent (IA)** autonomously identifies trends, moving beyond the static

dashboards of current well-being analytics and creating actionable intelligence.

- A **Support Coach Agent (SCA)** and a **Safety Triage Agent (STA)** act on this intelligence and on real-time user needs, providing both proactive, personalized coaching and immediate, context-aware crisis support. They function as the intelligent front-door to the support ecosystem, overcoming the limitations of purely reactive chatbots.
- A **Service Desk Agent (SDA)** closes the "insight-to-action" loop on an administrative level, automating the workflows for clinical case management and resource allocation that currently render proactive models inefficient and unscalable.

By designing and evaluating a system where these agents work in concert, orchestrated by LangGraph, this thesis pioneers a holistic solution that is fundamentally more proactive, scalable, and efficient than the disparate tools described in the current literature.

CHAPTER III
SYSTEM DESIGN AND ARCHITECTURE

3.1 Research Methodology: Design Science Research (DSR)

The research presented in this thesis is constructive in nature, aimed not merely at describing or explaining a phenomenon, but at creating a novel and useful artifact to solve a real-world problem. To provide a rigorous and systematic structure for this endeavor, this study adopts the **Design Science Research (DSR)** methodology. DSR is a well-established paradigm in Information Systems research focused on the creation and evaluation of innovative IT artifacts intended to solve identified organizational problems [61]. The primary goal of DSR is to generate prescriptive design knowledge through the building and evaluation of these artifacts.

3.1.1 Rationale for Design Science Research

The selection of DSR as the methodological framework for this research is justified by several key considerations that align with the nature of the problem and the objectives of this study. Table 3.1 summarizes the primary justifications for adopting DSR over alternative research methodologies.

These justifications collectively establish DSR as the most appropriate methodological framework for this research, balancing the need for rigorous academic inquiry with the practical imperative of delivering a functional artifact that addresses a real-world problem.

3.1.2 The DSR Process Model

The DSR process model, as outlined by Peffers et al., provides an iterative framework that guides the research from problem identification to the communication of results [62]. This thesis follows these stages, mapping them directly to its structure to ensure a logical and transparent research process:

1. **Problem Identification and Motivation:** This initial stage, which involves defining the specific research problem and justifying the value of a solution, is addressed in **Chapter I** of this thesis. We have identified the inefficiencies of the reactive mental health support model as the core problem.
2. **Define Objectives and Knowledge Base:** Building on the identified problem, this stage formalizes the solution objectives and anchors them in the relevant knowledge base. The initial objectives are articulated in **Chapter I**, and they are refined and theoretically grounded through the literature synthesis in **Chapter 2.1**.

Table 3.1. Justifications for adopting Design Science Research methodology.

DSR Characteristic	**Relevance to This Research**	**Contrast with Alternative Methodologies**
Artifact-centric problem solving	The core contribution is the Safety Agent Suite framework itself—a novel multi-agent system architecture. DSR provides the appropriate epistemological stance for research where the primary output is a designed artifact [61].	Descriptive research methodologies focus on understanding phenomena; purely experimental approaches test hypotheses in controlled settings but do not emphasize artifact creation as the primary contribution.
Practical relevance and real-world impact	The reactive mental health support problem identified in Chapter I is a genuine organizational challenge in Higher Education Institutions worldwide. DSR bridges academic rigor and practical utility by requiring artifacts address real problems [62].	A purely theoretical approach fails to deliver actionable solutions; a purely engineering approach lacks systematic evaluation rigor. DSR offers a middle ground ensuring both practical applicability and scholarly rigor.
Iterative development and refinement	The DSR process explicitly incorporates feedback loops between design, demonstration, and evaluation stages, aligning naturally with agentic AI development where agent behaviors must be iteratively refined based on testing results.	Waterfall-style experimental research and ethnographic studies do not accommodate this iterative, build-evaluate-refine cycle as seamlessly. The cyclic nature of DSR is essential for complex system development.
Compatibility with evaluation constraints	DSR accommodates scenario-based evaluation using synthetic or controlled test cases—essential when working with sensitive mental health data where live human trials require extensive ethical approvals and pose potential risks. Detailed in Chapter IV.	Traditional empirical methodologies typically require access to real subjects and naturalistic data, which are infeasible given ethical constraints and undergraduate thesis scope.
Knowledge contribution through design	DSR explicitly recognizes that designing, building, and evaluating artifacts generates generalizable design knowledge beyond specific instantiation [61]. This thesis contributes design principles, architectural patterns (dual-loop proactive-reactive model), and evaluation criteria.	Alternative methodologies may produce case-specific findings without explicit mechanisms for abstracting generalizable design knowledge applicable to future systems in the same problem domain.

3. **Design and Development:** This is the core constructive phase where the artifact's architecture and functionalities are developed. This stage is the primary focus of the present chapter, **Chapter III**, which outlines the functional and technical blueprint of the agentic AI framework.
4. **Demonstration:** In this stage, the designed artifact is demonstrated to solve representative instances of the problem. The functional prototype and its scenario walkthroughs are presented in **Chapter IV**, particularly Sections 4.1 and 4.2.
5. **Evaluation:** This stage observes and measures how well the artifact supports the solution objectives. The scenario-based tests and their analysis are reported in **Chapter IV**, Sections 4.2–4.6.
6. **Communication of Results:** The final stage disseminates the artifact, findings, and implications to the target audience. This thesis (culminating in **Chapter V** and supported by the appendices) serves as the primary communication vehicle.

Stages 4-6 therefore operationalize the empirical programme for the research questions defined in Chapter 1.4. The demonstration assets in Chapter IV (Sections 4.1 and 4.2) instantiate the scenarios for RQ1–RQ4, while the evaluation stage reports the quantitative indicators detailed in Chapter IV: STA sensitivity/specificity for safety (RQ1), orchestration reliability metrics such as tool-call success and latency for resilience (RQ2), rubric-based CBT quality scores for coaching (RQ3), and stability of aggregate analytics for the Insights Agent (RQ4). The communication stage synthesizes these findings in Chapter V so that institutional stakeholders can interpret the metrics and translate them into policy and operational decisions. Together, the paragraph-level traceability between stages and metrics makes the DSR cycle a roadmap for the scenario-based evaluation that follows.

3.1.3 Evaluation Strategy and Data Generation Approach

Given the sensitive nature of mental health support and the ethical constraints inherent in this domain, this research adopts a scenario-based evaluation methodology using carefully designed synthetic test cases rather than live human trials. This section presents the methodological decisions governing data generation, metric selection, and instrumentation.

3.1.3.1 Rationale for Synthetic Data

The decision to employ synthetic test data rather than authentic student conversations is driven by ethical, practical, and methodological considerations. Table 3.2 summarizes the primary justifications for this approach.

This approach aligns with established practices in safety-critical AI system eval-

Table 3.2. Rationale for synthetic data in evaluation.

Consideration	**Constraint with Real Data**	**Advantage of Synthetic Data**
Ethical approval	Collecting genuine mental health crisis conversations from students requires extensive ethical review board (ERB) approval, informed consent processes, and participant safeguarding mechanisms beyond the scope of an undergraduate thesis.	Eliminates need for ERB approval as no human participants are involved; allows research to proceed within feasible timeline.
Privacy and safety risks	Even anonymized mental health disclosures carry re-identification risks and potential psychological harm to participants if data is breached or mishandled.	Removes risk of harm to real individuals; no sensitive personal data is collected or stored.
Systematic coverage	Real conversational data is opportunistic and may not include rare but critical crisis scenarios (e.g., explicit self-harm statements, acute distress patterns).	Enables controlled, systematic testing of edge cases and boundary conditions essential for safety validation [**?**].
Reproducibility	Access to real student data is typically restricted and cannot be shared for replication purposes.	Synthetic datasets can be documented, versioned, and shared with evaluators, enhancing reproducibility and transparency.

uation, where controlled test scenarios provide more comprehensive coverage than naturalistic data collection, particularly when evaluating rare high-stakes events [**?**].

3.1.3.2 Test Corpus Design

The evaluation employs three carefully constructed test datasets, each designed to exercise specific agent functionalities and stress-test system boundaries. Table 3.3 details the composition and purpose of each corpus.

Table 3.3. Test corpus design and coverage.

Corpus	**Size**	**Content Coverage**	**Primary Evaluation Target**
Crisis detection corpus	500 labeled prompts	Self-harm ideation, violence indicators, acute distress scenarios, plus emotionally charged non-crisis messages (to test specificity).	Safety Triage Agent (STA): sensitivity, specificity, false positive/negative rates.
Coaching dialogue corpus	120 conversation snippets	Common student concerns (exam stress, social isolation, relationship difficulties, sleep issues, motivation).	Support Coach Agent (SCA): response quality, empathy, CBT alignment, engagement.
Synthetic operational logs	12 weeks of simulated activity	Multi-user interaction patterns, temporal trends, cohort-level sentiment and topic distributions.	Insights Agent (IA): analytics stability, trend detection, aggregation correctness.

These datasets systematically exercise each agent's intended functionality and provide controlled conditions for measuring performance against pre-defined acceptance criteria.

3.1.3.3 Metric Selection Principles

Evaluation metrics are selected according to three guiding principles to ensure methodological rigor and practical relevance:

1. **Alignment with research questions and safety objectives:** Each metric directly addresses a specific research question or system requirement, ensuring that evaluation outcomes inform the research goals.
2. **Measurability through automated instrumentation:** Metrics must be objectively quantifiable through automated data collection to minimize measurement bias and enable continuous monitoring.

3. **Interpretability for institutional stakeholders:** Metrics are selected for their clarity and relevance to non-technical decision-makers (e.g., counseling administrators) who will assess system suitability for deployment.

Each research question maps to specific quantitative metrics with pre-defined acceptance thresholds derived from system requirements and relevant literature. The complete evaluation plan, including metric definitions and acceptance criteria, is detailed in Chapter IV, Table 4.1.

3.1.3.4 Instrumentation and Reproducibility

To ensure evaluation reproducibility and enable multi-level behavioral analysis, the prototype implements comprehensive instrumentation. Table 3.4 summarizes the instrumentation strategy.

Table 3.4. Instrumentation strategy for evaluation reproducibility.

Instrumentation Type	**Implementation**	**Purpose**
Distributed tracing	OpenTelemetry spans across all agent operations, capturing request-response latency, tool invocation timing, and inter-agent handoffs.	Enables latency analysis, bottleneck identification, and performance optimization; supports evaluation of RQ2 (orchestration reliability).
Structured logging	All agent interactions, state transitions, and decision points logged in JSON format with ISO-8601 timestamps and correlation IDs.	Facilitates replay of evaluation scenarios, supports auditing, and enables post-hoc analysis of agent reasoning chains.
Metric exporters	Prometheus exporters expose real-time counters (tool call success/failure, classification outcomes) and histograms (latency distributions).	Provides quantitative data for evaluation dashboards; enables statistical analysis of agent performance.

This multi-layered instrumentation ensures that evaluation results are reproducible, that system behaviors can be analyzed at multiple levels of granularity, and that performance data is available for both real-time monitoring and retrospective analysis.

3.1.4 Validity and Limitations

The evaluation strategy employs multiple validity frameworks to assess the rigor and generalizability of findings. Table 3.5 summarizes the validity considerations and their implications for this research.

Table 3.5. Validity and limitations framework for the evaluation methodology.

Validity Type	**Strengths in This Research**	**Limitations and Mitigation Strategies**
Internal validity	High internal validity ensured through: (1) systematically designed test scenarios with controlled variables; (2) standardized execution platform (containerized environment); (3) minimized confounding variables through consistent test harnesses; (4) automated metrics reducing measurement subjectivity.	Potential for evaluation-to-evaluation variance in LLM outputs due to non-deterministic generation. Mitigated through: temperature=0 for classification tasks, multiple test runs with statistical aggregation, seed-based reproducibility where supported.
External validity	Limited but explicitly acknowledged. The controlled evaluation demonstrates proof-of-concept functionality and validates design decisions under idealized conditions.	**Primary limitation**: Synthetic test data, while carefully designed, cannot fully capture the complexity, variability, and cultural nuances of authentic student conversations. Findings may not generalize to real-world deployment without field validation. Future work requires pilot studies with appropriate ethical oversight (discussed in Chapter V).
Construct validity	Strong construct validity: selected metrics (sensitivity, specificity, latency, inter-rater agreement, Jensen-Shannon divergence) are well-established constructs in AI system evaluation and mental health screening literature. Each metric directly measures intended system properties.	Risk of metric misalignment with real-world user experience. For example, high sensitivity may come at the cost of user trust if false positives are frequent. Mitigated through multi-dimensional evaluation (not relying on single metric) and stakeholder validation of acceptance criteria.
Reliability	High measurement reliability ensured through: (1) automated instrumentation (OpenTelemetry, structured logging) providing consistent data collection; (2) deterministic evaluation scripts with version-controlled test datasets; (3) multiple independent raters for subjective assessments with inter-rater reliability reporting (Cohen's κ).	Human rating subjectivity for coaching quality (CBT rubric scores). Mitigated through: rater training, detailed rubric guidelines, inter-rater reliability thresholds ($\kappa \geq 0.8$), and adjudication process for disagreements.

These validity considerations inform the interpretation of evaluation results in Chapter IV and the recommendations for future research in Chapter V. The primary limitation—external validity—is explicitly acknowledged throughout this thesis, and the evaluation is positioned as a necessary first step toward eventual field deployment rather than a conclusive clinical validation.

The complete workflow of this research, following the DSR methodology, is visualized in Figure 3.2. This diagram illustrates the iterative path from problem formulation through to the final conclusions and recommendations.

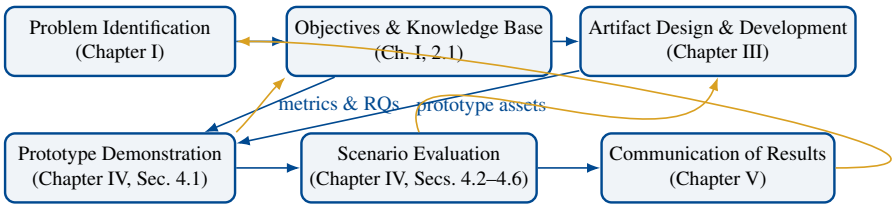


Figure 3.2. The Design Science Research (DSR) process model as applied in this thesis. The two-row layout shows how objectives inform design and how demonstration/evaluation feed back into earlier stages.

3.2 System Overview and Conceptual Design

The artifact proposed and developed in this research is a novel agentic AI framework designed to address the systemic inefficiencies of traditional, reactive mental health support models in Higher Education Institutions. The conceptual architecture is predicated on the principles of a Multi-Agent System (MAS), wherein a suite of collaborative, specialized intelligent agents, collectively termed the **Safety Agent Suite**, work in concert to create a proactive, scalable, and data-driven support ecosystem. This framework is designed not as a monolithic application, but as a dynamic, closed-loop system that operates on two interconnected levels: a micro-level loop for real-time, individual student support and a macro-level loop for strategic, institutional oversight and proactive intervention [63, 64].

The system's primary entities and their designated interaction points are illustrated in the conceptual context diagram in Figure 3.3. These entities are:

- **Students:** As the primary users, students interact with the system's conversational interface (UGM-AICare's '/aika' page). This serves as their direct entry point to the support ecosystem, where they engage with the agents responsible for coaching and immediate assistance.
- **University Staff/Counselors:** As the system's administrators and clinical supervisors, these stakeholders interact with a secure Admin Dashboard. This interface serves as the human-in-the-loop control center, providing aggregated analytics for strategic decision-making and a case management system for handling high-risk escalations.

- **The Agentic AI Backend:** This is the core computational engine of the framework. It hosts the four agents of the Safety Agent Suite, manages their stateful interactions via LangGraph, and serves as the central hub for all data processing, logic execution, and communication with external services and databases.

Table 3.6. Comparison of representative university well-being platforms.

Platform	**Primary Modality**	**Safety/Privacy Posture**	**Contrast with Safety Agent Suite**
Woebot Campus Programme [**?**]	Daily CBT-aligned chatbot sessions with journaling prompts.	Crisis disclaimers and human referral prompts, but escalation remains manual and analytics are limited.	Lacks dedicated triage agent or orchestration guardrails; actionable insights are not automated, restricting proactive interventions.
Togetherall Peer Community [**?**]	Moderated peer-to-peer forums with clinician oversight and resource hubs.	Strong anonymity policies and moderator workflows, yet response time depends on human moderators.	Provides community support but no automated coaching, triage, or strategic analytics loop as offered by the Safety Agent Suite.
Kognito "At-Risk" Simulations [**?**]	Scenario-based training to help faculty/students identify and refer distressed peers.	Focuses on awareness; no live data capture or privacy-sensitive storage since it is a training tool.	Educates stakeholders but does not deliver operational support, automated escalation, or continuous monitoring of student well-being.

Conceptually, the framework's architecture is best understood as two distinct but integrated operational loops:

1. **The Real-Time Interaction Loop:** This loop handles immediate, synchronous interactions with individual students. When a student sends a message, it is first processed by the **Safety Triage Agent (STA)** for risk assessment. If the context is deemed safe, the **Support Coach Agent (SCA)** takes over to provide personalized, evidence-based guidance. Should the user require administrative assistance, such as scheduling an appointment, the workflow is seamlessly handed off to the **Service Desk Agent (SDA)**. This loop is designed for high-availability, low-latency responses, ensuring that students receive immediate and appropriate support.
2. **The Strategic Oversight Loop:** This loop operates on a longer, asynchronous timescale to enable proactive, institution-wide strategy. The **Insights Agent (IA)**

periodically analyzes the anonymized, aggregated data from all student interactions. It generates reports on population-level well-being trends, sentiment analysis, and emerging topics of concern. These reports are delivered to administrators via the Admin Dashboard, providing the empirical evidence needed for data-driven resource allocation, such as commissioning new workshops or adjusting counseling staff schedules. This loop directly addresses the "insight-to-action" gap that plagues current systems [11, 64].

The synergy between these two loops is the cornerstone of the framework's design. The real-time loop gathers the data that fuels the strategic loop, while the insights from the strategic loop can be used to configure and improve the proactive interventions delivered by the real-time loop, creating a continuously learning and adaptive support ecosystem.

3.2.1 Orchestration Strategy: Rationale for Graph-Based Agent Coordination

The selection of an appropriate orchestration mechanism for multi-agent coordination is a critical architectural decision that directly impacts system reliability, maintainability, and performance. Classical multi-agent systems literature distinguishes between centralized planners, market-based negotiation schemes, and more recent graph-structured controllers for coordinating autonomous agents [14, 36]. Contemporary surveys focused on LLM-powered agents demonstrate that orchestration frameworks such as LangGraph provide deterministic state persistence, guardrails, and cycle control that are difficult to obtain in purely contract-net or ad-hoc workflow engines [52, 55, 58].

Table 3.7 presents a systematic comparison of three primary orchestration approaches, evaluating each against the specific requirements of the Safety Agent Suite. The analysis reveals that while centralized workflows offer simplicity and contract-net approaches provide flexibility, the graph-based stateful orchestrator best aligns with the system's need for deterministic escalation paths, comprehensive auditing capabilities, and robust metric capture infrastructure.

The adoption of LangGraph's graph-based orchestration provides several concrete advantages for the Safety Agent Suite. The stateful edges enable the Safety Triage Agent to enforce risk-score thresholds before delegating control to the Support Coach Agent, while preserving the ability to retry failed operations, maintain comprehensive logs, and escalate exceptions without requiring bespoke infrastructure. This orchestration choice directly supports the evaluation metrics reported in Chapters IV and V, particularly those concerning tool-call reliability, end-to-end latency, and system auditability. The deterministic nature of graph-based workflows also facilitates reproducible testing and debugging, which is essential for safety-critical mental health applications where understanding system behavior under failure conditions is paramount.

Table 3.7. Comparison of orchestration patterns for the Safety Agent Suite.

Approach	**Coordination Strengths**	**Limitations and Implications for Safety Agent Suite**
Centralized workflow/planner [14]	Deterministic control flow and straightforward verification of simple pipelines.	Brittle when the conversation requires branching or repeated loops; a single orchestrator becomes a failure hotspot and hinders human-in-the-loop escalations needed for STA.
Contract-net / market-based negotiation [36, 58]	Decentralized task allocation and flexibility for loosely coupled agents.	Negotiation latency and probabilistic assignment make it difficult to guarantee triage deadlines and safety invariants; insufficient for crisis escalation SLAs.
Graph-based, stateful orchestrator (LangGraph) [52, 55]	Explicit state persistence, guard conditions, and cyclic workflows that support guardrails, retries, and logging.	Requires deliberate state-schema design and emerging tooling, but best aligns with the need for deterministic escalation paths, auditing, and metric capture in Chapters IV and V.

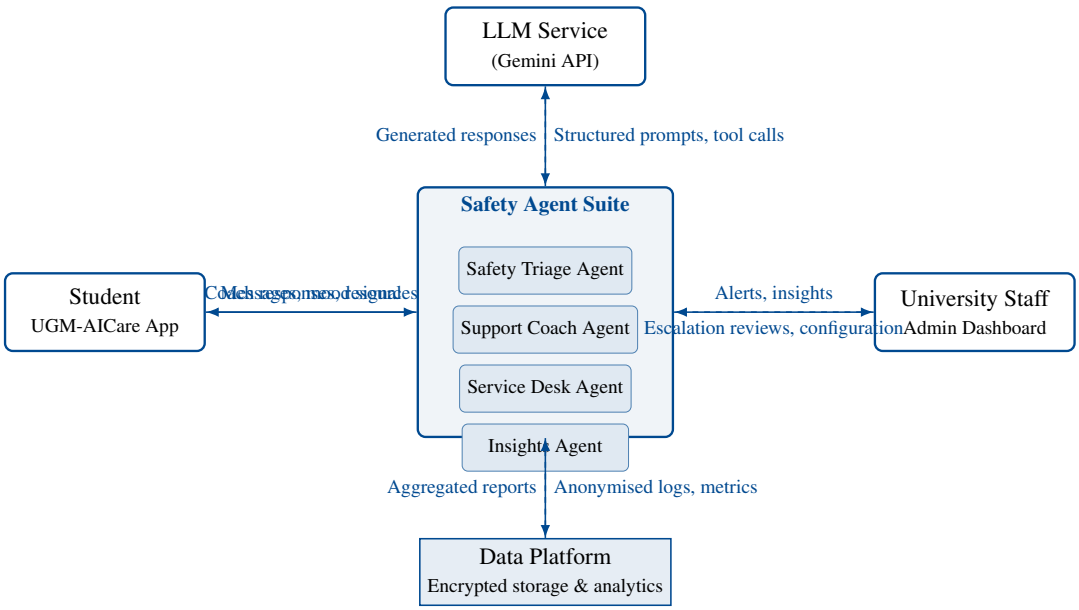


Figure 3.3. High-level context of the Safety Agent Suite showing primary stakeholders, orchestration boundaries, and data exchanges. Dashed arrows denote supervisory or configuration interactions.

3.3 Functional Architecture: The Agentic Core

The functional architecture of the framework is designed as a Multi-Agent System (MAS), where the system's overall intelligence and capability emerge from the coordinated actions of its four specialized agents. This section details the functional specifications of each agent within the **Safety Agent Suite**. Each agent functions as a distinct component within the LangGraph state machine, perceiving its environment through the shared state, executing its logic, and updating the state with its results. Table 3.8 provides a comprehensive comparison of the four agents' roles, inputs, processing logic, and outputs.

3.3.1 Agent Functional Specifications

Table 3.8. Functional specifications of the Safety Agent Suite.

Agent	Primary Role	Inputs (Perception)	Processing Logic	Outputs (Actions)	Performance Target
Safety Triage Agent (STA)	Real-time crisis detection and risk assessment	Current user message M_t (raw text)	High-speed LLM-based classification: $R_t = f_{STA}(M_t; \theta_{LLM})$ where $R_t \in \{\text{Low, Moderate, Critical}\}$. Evaluates indicators of self-harm, severe distress, or urgent help requests.	(1) State graph update with `risk_level` = R_t; (2) If R_t = Critical: invoke `escalate_crisis` tool to flag conversation, create case, present emergency resources.	Latency: <250ms p95; Sensitivity: >90%; Specificity: >85%
Support Coach Agent (SCA)	Personalized mental health coaching and therapeutic guidance	(1) Conversation history H_{t-1}; (2) Safe user message M_t; (3) User progress state (completed modules, check-ins)	Context-aware LLM generation: $A_t = f_{SCA}(M_t, H_{t-1}; \theta_{LLM})$. Produces empathetic, CBT-aligned responses. Invokes `retrieve_cbt_module` when specific skills needed.	(1) Conversational response text; (2) Tool calls for content delivery (CBT modules); (3) `record_module_completion` for progress logging.	Response quality: >3.5/5 on CBT rubric; Engagement: session duration >5 min
Service Desk Agent (SDA)	Administrative workflow automation (case management, scheduling)	(1) Escalation events from STA (conversation ID, risk level); (2) Scheduling requests from SCA (user ID, desired times); (3) Admin commands (close case, add note)	Procedural, tool-based execution. Sequences pre-defined actions: e.g., `create_case` → `assign_case_status("New")`. No generative text.	(1) Database mutations (create/update/close cases); (2) External API calls (calendar availability, booking); (3) Notifications (email, dashboard alerts to counselors).	Tool-call success: >95%; Idempotent retries: max 3 attempts
Insights Agent (IA)	Population-level analytics and trend identification	(1) Time-based trigger (weekly Cron); (2) Read-only access to anonymized `conversation_logs` table	NLP pipeline: (1) Topic modeling (LDA/transformer clustering); (2) Sentiment analysis (population score over time); (3) LLM-based summarization of findings.	(1) Structured report (JSON/PDF with visualizations); (2) Admin Dashboard update; (3) Optional email notification to stakeholders.	Topic stability: JS divergence <0.15 week-over-week; Report latency: <60s

3.3.2 Agent Design Rationale and Assumptions

Each agent's design incorporates specific assumptions and validation strategies to ensure reliability and safety within the overall system architecture. This subsection details the design constraints, operational assumptions, and validation approaches for each of the four agents.

3.3.2.1 Safety Triage Agent Design Constraints

The STA operates under the following assumptions and constraints:

1. **Input sanitization**: Incoming messages M_t are assumed to be UTF-8 encoded text that has been pre-filtered for profanity and noise by upstream input validation layers.

2. **Confidence thresholding**: The agent applies a calibrated confidence threshold (LLM softmax score ≥ 0.6) before assigning critical risk labels. Messages with lower confidence scores are flagged for human counselor review to minimize false positives while maintaining high sensitivity for genuine crises.
3. **Validation methodology**: The prompt template and `escalate_crisis` tool schema were validated against the synthetic crisis corpus. Performance characteristics (sensitivity, specificity, latency) are evaluated through scenario-based metrics reported in Chapter 4.2 [**?**].

3.3.2.2 Support Coach Agent Design Constraints

The SCA's design incorporates the following assumptions:

1. **Pre-sanitized input**: The agent assumes that all incoming messages M_t have been vetted by the STA and deemed safe for conversational processing, eliminating the need for redundant crisis detection logic.
2. **Context window management**: Conversation history H_{t-1} is maintained with a maximum of 50 dialogue turns to bound context length and prevent token overflow in the LLM context window. Older messages are summarized or archived when this threshold is reached.
3. **Scope boundaries**: The agent enforces strict refusal policies for out-of-scope clinical topics (e.g., medication advice, diagnosis, emergency medical conditions) and automatically escalates administrative intents (e.g., appointment scheduling) to the Service Desk Agent. These policies prevent scope creep and maintain appropriate boundaries for an AI coaching system.
4. **Quality assurance**: Prompt templates and tool schemas underwent peer review and are evaluated via rubric-based assessment measuring CBT alignment, empathy, and engagement metrics as detailed in Chapter 4.3 [**?**].

3.3.2.3 Service Desk Agent Design Constraints

The SDA's procedural architecture requires:

1. **Input validation**: All structured events must carry validated UUIDs, ISO-8601 formatted timestamps, and pass schema validation in upstream components before reaching the agent. This ensures data integrity and prevents malformed requests from disrupting workflow execution.
2. **Idempotency and retry logic**: The agent enforces idempotent tool execution with exponential backoff (delays: 1s, 2s, 4s) to handle transient failures gracefully. After three failed retry attempts, the system escalates to human staff to prevent infinite

loops and ensure critical cases receive attention.

3. **Integration testing**: Tool schemas and workflow sequences were exercised through comprehensive integration tests that verify correct database state transitions, external API interactions, and notification delivery as summarized in Chapter 4.3 [**?**].

3.3.2.4 Insights Agent Design Constraints

The IA operates under strict privacy-preserving constraints:

1. **Anonymization requirements**: The agent accesses only anonymized conversation logs that have been aggregated with minimum cohort size thresholds (k $\geq$ 50 to maintain k-anonymity). This prevents re-identification attacks through demographic or temporal correlation.
2. **Data retention limits**: Logs are retained for a maximum of 90 days in accordance with institutional data retention policies, after which they are permanently deleted to minimize privacy risk exposure.
3. **Output suppression**: Analytics outputs for cohorts below the minimum threshold are automatically suppressed to prevent potential re-identification through small sample sizes.
4. **Pipeline validation**: Analytical prompts and NLP pipelines (topic modeling, sentiment analysis) were stress-tested through stability checks that measure consistency across multiple runs and temporal windows, as reported in Chapter 4.5 [**?**].

3.3.3 Agent Processing Formalization

To provide mathematical rigor to the agent specifications, this subsection formalizes the core decision functions employed by each agent. These formalizations ground the qualitative descriptions in Table 3.8 within a consistent computational framework.

3.3.3.1 Safety Triage Agent Classification Function

The STA's classification function maps a user message to a discrete risk level:

$$R_t = f_{STA}(M_t; \theta_{LLM}) : \mathcal{M} \rightarrow \mathcal{R} \tag{3-1}$$

where $\mathcal{M}$ is the space of all possible user messages (UTF-8 strings), $\mathcal{R} = \{\text{Low, Moderate, Critical}\}$ is the set of risk levels, and θ_{LLM} represents the parameters of the Gemini 2.5 Pro model. The function is implemented through a structured prompt that instructs the LLM to evaluate crisis indicators (self-harm ideation, violence, acute distress) and return a classification with confidence score.

3.3.3.2 Support Coach Agent Response Generation

The SCA's response generation function produces contextually appropriate therapeutic responses:

$$A_t = f_{SCA}(M_t, H_{t-1}; \theta_{LLM}) : \mathcal{M} \times \mathcal{H} \to \mathcal{A} \tag{3-2}$$

where $H_{t-1} = \{M_1, A_1, M_2, A_2, \ldots, M_{t-1}, A_{t-1}\}$ is the conversation history up to turn $t-1$, $\mathcal{H}$ is the space of all possible conversation histories, and $\mathcal{A}$ is the space of agent responses. The function incorporates user progress state implicitly through retrieved context from the database.

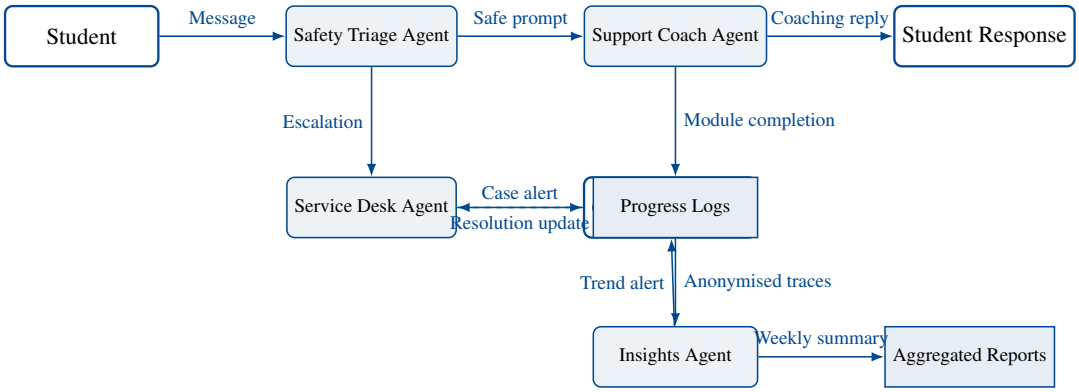


Figure 3.4. Data flow between the Safety Agent Suite and its users. Solid arrows show operational data paths; dashed arrows show supervisory feedback.

3.4 Security and Privacy Threat Model

Protecting student data and maintaining institutional trust require an explicit articulation of adversaries, assets, and mitigations. Table 3.9 summarises the threat model adopted for the Safety Agent Suite, drawing on STRIDE/LINDDUN heuristics and privacy-by-design obligations. [**?**]

These mitigations align with institutional policies and inform the evaluation metrics in Chapter IV (e.g., STA sensitivity to avoid under- or over-escalation) and the privacy discussion in Chapter V. Residual risks—such as novel jailbreak prompts or emergent privacy attacks—are addressed through scheduled red-team exercises, update audits for commercial APIs, and continuous monitoring of anomaly indicators.

3.5 Technical Architecture

This section details the "how" of the system, providing the engineering blueprint for the agentic AI framework. The architecture is designed following a modern, service-oriented pattern, which decouples the primary components of the system into distinct, independently deployable services. This approach enhances maintainability, scalability,

Table 3.9. Threat model overview.

Actor / Threat	**Targeted Asset**	**Likely Impact**	**Mitigations / Controls**
Compromised student account	Conversation logs, risk flags	Exposure of sensitive disclosures; spoofed escalations	MFA and device attestation (frontend); STA confidence thresholds with human verification; audit trail review (Chapter 4.2).
Malicious insider (staff)	Case notes, progress logs	Unauthorised browsing or data exfiltration	Role-based access control, immutable audit logs, case access alerts, quarterly review.
External attacker (API abuse)	Backend endpoints, tooling	Prompt injection, denial of service, data tampering	API gateway with rate limiting, input sanitation, LangGraph guardrails, automated anomaly detection on latency/tool-failure metrics.
Analytics linkage attack	Aggregated insights	Re-identification through small cohorts	Minimum cohort size thresholds, suppression of rare categories and small sample sizes (Chapter 4.5).
Model supply-chain risks	LLM outputs / prompts	Hallucinated or unsafe responses	Structured prompts, refusal and escalation policies, prompt/response logging with red-team testing cadence.
Infrastructure failure	Agent orchestration state	Service outage, message loss	Stateless frontend, checkpointed LangGraph state, automated failover for database replicas, latency SLOs monitored (Section 3.5).

and promotes a clean separation of concerns [65, 66]. The framework consists of three core services: a unified frontend application, a backend service that houses the agentic core, and a data persistence service for all storage needs.

3.5.1 Overall System Architecture

The overall technical architecture is visualized in Figure 3.5. It is a monolithic frontend-backend structure composed of three primary services that work in concert to deliver the full functionality of the framework to both students and administrators.

1. **Frontend Service (UGM-AICare Web Application):** This is a comprehensive web application built using the **Next.js** framework. It serves two distinct user-facing roles from a single codebase:
   - **The User Portal:** This is the interface for students. It provides access to features such as a journaling system, a user dashboard for tracking progress, and the '/aika' chat interface for direct interaction with the Support Coach Agent (SCA) and Safety Triage Agent (STA). It also handles features like appointment scheduling with counselors.
   - **The Admin Dashboard:** This is a secure, role-protected area of the application for university staff and counselors. Its responsibilities include rendering the analytics and insights provided by the Insights Agent (IA), displaying real-time alerts for flagged conversations, and providing a case management system to act on escalations from the STA and SDA.
2. **Backend Service (The Agentic Core):** This service is the "brain" of the entire operation, built using the **FastAPI** Python framework. It exposes a **REST API** through which the unified Next.js frontend communicates. The backend is responsible for handling all business logic, including processing incoming chat messages from the User Portal, orchestrating the agents within the LangGraph state machine, making calls to the Google Gemini API, and interacting with the database. The asynchronous capabilities of FastAPI are critical for efficiently managing multiple concurrent conversations and long-running agentic tasks.
3. **Data Persistence Service:** A **PostgreSQL** relational database serves as the single source of truth for the system. It is responsible for storing all persistent data, including user information (anonymized), conversation logs, clinical case data, and generated reports.

3.5.1.1 Capacity Planning and Performance Targets

To ensure the architecture meets the safety and responsiveness requirements evaluated in Chapter 4.2, we define the following planning assumptions:

- **Latency budgets:** STA classification must complete within 250 ms p95 and 500 ms p99, while end-to-end conversational responses (STA + SCA) target 1.5 s p95 under nominal load. These targets are theoretically justified by the real-time performance model in Section 2.2.4.3. Scenario results in Chapter 4.2 and Chapter 4.3 will be benchmarked against these SLOs.
- **Concurrency expectations:** The system is sized for 500 concurrent student sessions (peak exam season) with headroom to burst to 1,000. Each FastAPI worker handles approximately 20 concurrent conversations; horizontal scaling of backend pods maintains SLA compliance.
- **Horizontal scaling strategy:** Stateless Next.js frontend instances auto-scale behind a CDN; the FastAPI+LangGraph service scales via container orchestration (e.g., Kubernetes HPA) using CPU and latency metrics; PostgreSQL employs read replicas for analytics workloads while the primary instance handles transactional writes. Queueing (e.g., Redis streams) buffers asynchronous Insights Agent jobs to decouple heavy analytics from real-time triage.
- **Resilience and monitoring:** Service-level indicators (SLIs) include STA false-negative rate, tool-call success percentage, queue depth, and database replication lag. Alert thresholds align with risk tolerances captured in the threat model (Table 3.9). [**?**]

Communication between the unified frontend and the backend is exclusively handled via a secure, stateless REST API. The backend service is the only component with direct access to the database, ensuring a clear and secure data access pattern. This architecture allows for a cohesive user experience while maintaining a strong separation between presentation logic (frontend) and business logic (backend).

3.5.2 Backend Service: The Agentic Core

The backend service is the central nervous system and cognitive engine of the entire framework. It is a Python-based application responsible for executing all business logic, orchestrating the agentic workflows, and serving as the intermediary between the user-facing application and the data persistence layer. To meet the demanding requirements of a real-time, AI-powered conversational system, the backend is built upon a modern, high-performance technology stack.

3.5.2.1 API Framework: FastAPI

The foundation of the backend service is the **FastAPI** framework. This choice was made after careful consideration of several alternatives, based on its specific suitability for building high-performance, API-driven services that interact with machine learning models [67, 68]. The primary justifications for its selection are:

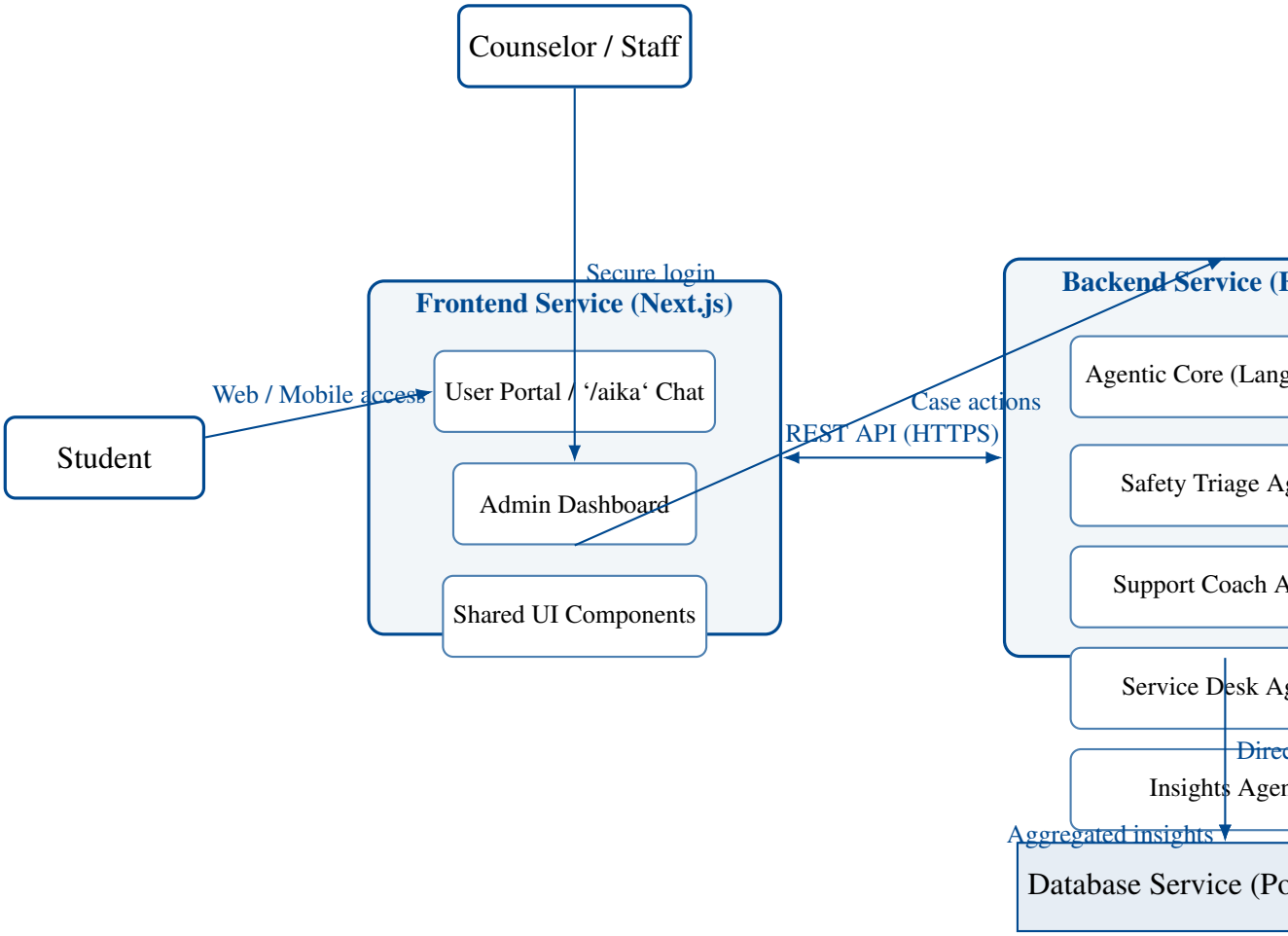


Figure 3.5. High-level technical architecture showing the unified Next.js frontend, FastAPI backend with LangGraph agents, and supporting services.

- **Asynchronous Support:** FastAPI is built on top of ASGI (Asynchronous Server Gateway Interface), allowing it to handle requests asynchronously. This is a critical requirement for this framework, as interactions with the Google Gemini API are I/O-bound operations. Asynchronous handling ensures that the server can manage multiple concurrent user conversations and long-running agentic tasks without blocking, leading to a highly responsive and scalable system.
- **High Performance:** Leveraging Starlette for web routing and Pydantic for data validation, FastAPI is one of the fastest Python web frameworks available, delivering performance on par with NodeJS and Go applications [67, 68]. This is essential for minimizing latency in the real-time chat interface.
- **Data Validation and Serialization:** FastAPI uses Pydantic type hints to enforce rigorous data validation for all incoming and outgoing API requests. This not only reduces the likelihood of data-related bugs but also automatically serializes data to and from JSON, streamlining the development process.
- **Automatic Interactive Documentation:** The framework automatically generates interactive API documentation (via Swagger UI and ReDoc) based on the Pydantic models. This creates a reliable, always-up-to-date contract for the frontend team and simplifies the testing and debugging process.

The backend exposes a RESTful API for all communication with the frontend service. The design follows standard REST principles, using conventional HTTP methods to perform operations on resources. A summary of key endpoints is provided in Table 3.10.

Table 3.10. Key Endpoints of the Backend REST API.

Method	Endpoint	Description
`POST`	`/api/chat/message`	Submits a user message for processing by the agentic core.
`GET`	`/api/insights/latest`	Fetches the latest strategic report from the Insights Agent.
`POST`	`/api/appointments`	Creates a new appointment with a counselor via the SDA.
`GET`	`/api/admin/cases`	Retrieves all flagged cases for the admin dashboard.

3.5.2.2 Agent Orchestration: LangGraph

To manage the complex, cyclical, and stateful interactions between the four agents, the framework employs **LangGraph**. LangGraph extends the linear "chain" paradigm of LangChain by modeling agentic workflows as a state graph, which is essential for building robust multi-agent systems [51, 57].

The core of the orchestration is a central **State Graph**, where the application's

state is explicitly defined and passed between nodes. This state object, implemented as a Pydantic class, contains all relevant information for a given workflow, such as the full `conversation_history`, the `current_risk_level` as determined by the STA, and the `active_case_id`.

The workflow is structured as follows:

- **Nodes:** Each of the four agents (STA, SCA, SDA, IA) and their associated tools are implemented as nodes in the graph. A node is a function that receives the current state, performs its task (e.g., makes an LLM call, queries the database), and returns a dictionary of updates to be merged back into the state.
- **Edges:** The flow of control between nodes is managed by edges. Crucially, the framework uses **conditional edges** to implement the agentic logic. After a node executes, a routing function inspects the updated state to decide which node to call next. For example, after the STA node classifies a message, a conditional edge checks the `current_risk_level` in the state. If the level is 'CRITICAL', the edge routes the workflow to the SDA node to create a case; otherwise, it routes to the SCA node to continue the conversation. This structure is visualized in Figure 3.6.

This stateful, cyclical approach allows for sophisticated agentic behaviors, such as retrying failed tool calls, handing off tasks between agents, and maintaining a durable memory of the interaction, which are critical for the reliability and safety of the system.

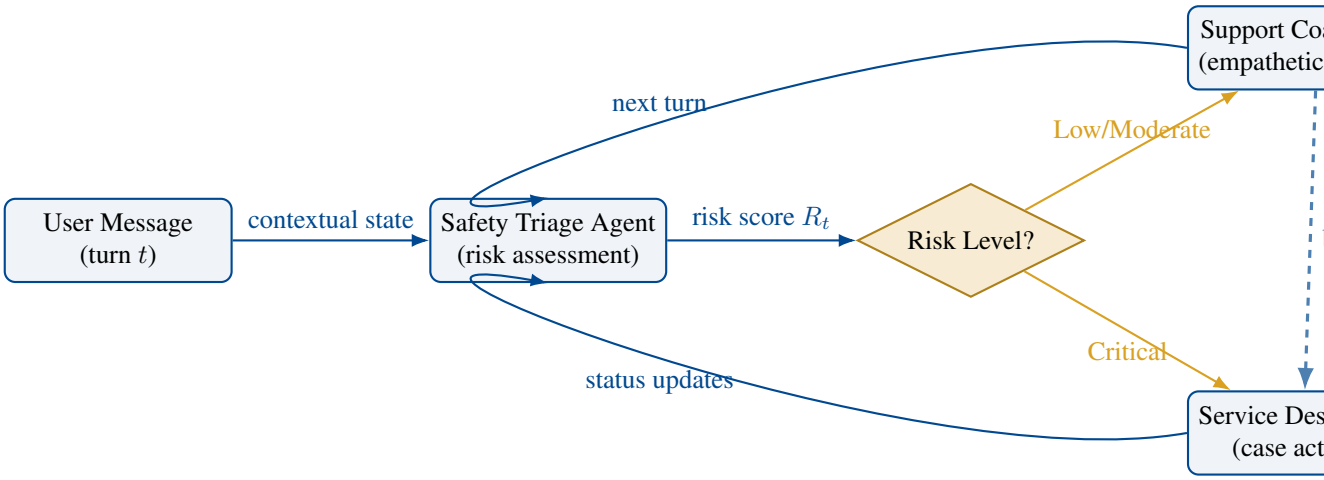


Figure 3.6. Conceptual LangGraph state machine showing how conversation turns pass through the Safety Triage Agent before branching to the Support Coach or Service Desk agents, with feedback loops preserving state.

3.5.2.3 Asynchronous Task Scheduling

To facilitate the proactive, long-term analysis performed by the Insights Agent (IA), the framework requires a mechanism for scheduling periodic tasks. Instead of re-

lying on an external workflow orchestration tool like n8n, a task scheduler is integrated directly into the FastAPI backend service.

For this purpose, the **APScheduler** (Advanced Python Scheduler) library is utilized. This choice was made for the following reasons:

- **Integration and Simplicity:** As a Python library, APScheduler integrates seamlessly into the FastAPI application's event loop. This avoids the operational complexity and additional infrastructure requirements of deploying and maintaining a separate workflow management service.
- **Sufficient Functionality:** For the primary requirement of running the IA's analysis on a fixed schedule (e.g., weekly), APScheduler's cron-style triggering is perfectly suited and provides a lightweight yet robust solution.

The scheduler is configured to trigger the IA's main analysis function at a predefined interval. This function then executes its NLP pipeline, generates the strategic report, and pushes the results to the database and relevant stakeholders, thus closing the strategic oversight loop of the framework without manual intervention.

3.5.3 Frontend Service: The UGM-AICare Web Application

The frontend service is the primary human-computer interface for the entire framework, serving both students and administrative staff. It is engineered as a monolithic frontend application using the **Next.js** React framework. This choice was deliberate, allowing for the development and maintenance of two distinct user experiences, the public-facing User Portal and the secure Admin Dashboard within a single, cohesive codebase. This approach simplifies dependency management and ensures a consistent design language across the platform while leveraging Next.js's powerful features for routing and role-based access control [69, 70].

The selection of Next.js is justified by several key architectural advantages that directly support the project's requirements:

- **Hybrid Rendering Strategies:** Next.js provides the flexibility to employ different rendering strategies on a per-page basis. For the dynamic, data-heavy Admin Dashboard, **Server-Side Rendering (SSR)** can be utilized to ensure that staff always see the most up-to-date case information and analytics. For the public-facing User Portal, a combination of SSR for dynamic content (like the user's dashboard) and **Static Site Generation (SSG)** for informational pages ensures both data freshness and optimal performance.
- **Component-Based Architecture:** Built upon React, Next.js facilitates a modular and reusable component-based architecture. This allows for the creation of discrete UI

components (e.g., the chat window, dashboard widgets, journaling entries) that can be developed, tested, and maintained in isolation, significantly improving the scalability and maintainability of the codebase.

- **Integrated API Routes:** Next.js includes a built-in capability to create API routes within the same project. While the primary business logic resides in the separate FastAPI backend, this feature is leveraged to handle server-side frontend tasks, such as proxying requests to the backend API, securely managing session tokens, and hiding sensitive API keys from the client-side browser.

The application is functionally divided into two main areas:

3.5.3.1 The User Portal

This is the student-facing portion of the application, designed to be an accessible and engaging entry point to the university's mental health resources. Its key functional components include:

- **The '/aika' Conversational Interface:** A real-time chat component that serves as the primary interaction point with the Support Coach Agent (SCA) and the underlying Safety Triage Agent (STA). It is responsible for managing the state of the conversation and rendering responses from the backend.
- **Journaling System:** A private, secure feature allowing students to write and review personal journal entries, a common practice in CBT-based therapies.
- **User Dashboard:** A personalized space where students can track their progress through coaching modules, revisit completed exercises, and see reminders for upcoming check-ins.
- **Appointment Scheduling:** An interface that communicates with the Service Desk Agent (SDA) via the backend API to allow students to view available slots and book appointments with human counselors.

3.5.3.2 The Admin Dashboard

This is a secure, authentication-protected area of the application designed for counselors and administrative staff. It functions as the central control and oversight panel for the entire agentic framework. Key features include:

- **Insights Visualization:** Renders the reports and data visualizations generated by the Insights Agent (IA), providing staff with a clear, actionable overview of student well-being trends.
- **Real-Time Case Management:** Displays alerts for conversations flagged as "critical" by the STA. It provides an interface for counselors to review the flagged conversa-

tion, manage the case status, and document actions taken, directly interacting with the workflows managed by the SDA.

- **System Configuration:** Provides an interface for administrators to configure certain parameters of the agentic system, such as the email list for IA reports or the thresholds for proactive interventions.

All dynamic data and actions within both the User Portal and the Admin Dashboard are handled through asynchronous requests to the backend REST API, ensuring a clean and complete separation between the presentation layer (frontend) and the business logic and agentic core (backend).

3.5.4 Data Persistence Layer: PostgreSQL

The data persistence layer is the architectural component responsible for the storage, retrieval, and management of all long-term data within the framework. For this system, **PostgreSQL**, a powerful, open-source object-relational database system, was selected as the data persistence service. This decision was based on its robustness, feature set, and suitability for an application that handles structured, relational, and sensitive data [71, 72].

The choice of a relational database model, and PostgreSQL specifically, is justified by the following key factors:

- **Data Integrity and ACID Compliance:** The nature of the application, which involves managing user interactions, clinical case escalations, and appointments, requires strong guarantees of data integrity. PostgreSQL's full compliance with ACID (Atomicity, Consistency, Isolation, Durability) properties ensures that all transactions are processed reliably. This is a non-negotiable requirement for a system where a missed escalation or a lost conversation log could have significant consequences.
- **Structured and Relational Data Model:** The data generated by the framework is inherently relational. There are clear, defined relationships between users, their conversation sessions, the messages within those sessions, and the clinical cases that may arise from them. A relational schema allows for the enforcement of these relationships at the database level through foreign key constraints, ensuring a consistent and logical data model.
- **Scalability and Concurrency Control:** PostgreSQL is renowned for its robust implementation of Multi-Version Concurrency Control (MVCC), which allows for high concurrency by enabling read operations to occur without blocking write operations. This is critical for the system's architecture, as the Insights Agent (IA) will perform large-scale read queries for analytics, while the real-time agents (STA, SCA, SDA) will be continuously writing new data from user interactions.

- **Extensibility and Advanced Features:** PostgreSQL supports a rich set of data types, advanced indexing capabilities, and powerful query optimization. This provides the flexibility to handle complex analytical queries from the IA efficiently and to extend the database schema in the future without requiring a migration to a different database system.

The backend service is the sole component with direct credentials to access the database. All interactions from the frontend are proxied through the backend's REST API, which enforces business logic and authorization before any database transaction is executed. This centralized access model is a critical security measure that prevents direct, unauthorized access to the data persistence layer.

The detailed logical structure of the database, including the table schemas and their relationships, will be presented in the **Database Design** section, which includes a full Entity-Relationship Diagram (ERD).

3.5.5 Deployment and Scalability Considerations

While the preceding sections have detailed the logical design of the framework's services, a comprehensive technical architecture must also account for the practical implementation of deploying and managing these services. For the UGM-AICare project, a pragmatic and robust deployment strategy centered on containerization within a dedicated Virtual Machine (VM) has been adopted. This approach ensures a consistent and reproducible environment while leveraging the specific infrastructure made available for this research.

The deployment environment is a Virtual Machine generously provided through a research collaboration with **PT INA17**. The entire application stack is deployed on this single server, using **Docker** as the core technology to manage the services [73, 74]. Docker is used to package the Next.js frontend, the FastAPI backend, and the PostgreSQL database into lightweight, portable containers. This containerization strategy provides several key advantages in a single-server environment:

- **Consistency and Reproducibility:** By defining each service's environment and dependencies within a 'Dockerfile', the framework eliminates environment-specific issues, guaranteeing that the application runs on the production VM exactly as it does in development.
- **Isolation:** Each service runs in its own isolated container. This is particularly crucial for preventing dependency conflicts between the Python-based backend and the Node.js-based frontend, ensuring that they can be updated and maintained independently without interfering with one another.

- **Simplified Management:** Using ‘docker-compose‘, the entire multi-container application can be managed with a single configuration file, simplifying the processes of starting, stopping, and updating the services.

To manage incoming web traffic and route it to the appropriate services, **Nginx** is employed as a reverse proxy. It is installed on the host VM and configured to handle all domain-related tasks. Its responsibilities include:

- **SSL Termination:** Nginx manages the SSL certificates, terminating HTTPS connections and forwarding decrypted traffic to the appropriate internal container. This centralizes security management.
- **Request Routing:** It inspects incoming requests and routes them based on their path. For example, requests to ‘/api/*‘ are forwarded to the FastAPI backend container, while all other requests are routed to the Next.js frontend container. This allows both services to appear as if they are running on the same domain and port to the end-user.

Regarding scalability, while this single-VM deployment does not involve automated horizontal scaling like a Kubernetes cluster, it is designed with future growth in mind:

- **Vertical Scaling:** The most direct path to scaling is vertical. The resources of the Virtual Machine (CPU, RAM, storage) can be increased as the application’s user base and data load grow.
- **Container-Level Scaling:** Should the VM’s resources permit, it is possible to run multiple instances of the most resource-intensive service, which is the FastAPI backend, on the same machine. Nginx can be configured to act as a load balancer, distributing incoming API requests across these multiple backend containers, thereby increasing the system’s capacity to handle concurrent users.

This deployment strategy provides a stable, secure, and maintainable foundation for the UGM-AICare prototype, while offering clear pathways for future scaling as the project evolves.

3.6 Database Design

The database is the foundational layer of the system’s architecture, responsible for the structured persistence and integrity of all data generated and consumed by the agentic framework. The schema is designed using a **relational model** to logically represent the relationships between the core entities of the application and to enforce strict data integrity through constraints. This design ensures that the data model can robustly support the distinct operational requirements of all four agents, from the real-time transactional needs of the STA and SDA to the large-scale analytical queries of the IA while

adhering to the principles of privacy and data security. The following subsections detail the logical structure of the database, including the Entity-Relationship Diagram (ERD) and the specific schemas of the key tables.

3.6.1 Entity-Relationship Diagram (ERD)

The logical schema of the database is formally represented by the Entity-Relationship Diagram (ERD) shown in Figure 3.7. This diagram provides a visual blueprint of the core entities within the system, their attributes, and the relationships that connect them. The design is normalized to reduce data redundancy and ensure logical consistency, with relationships enforced through primary and foreign key constraints.

The key entities and their relationships are as follows:

- **Core Interaction Entities:** The central entities that capture user engagement are `users`, `conversations`, and `messages`. A `user` (representing a student with an anonymized identifier) can have multiple `conversations`, and each `conversation` is composed of multiple `messages`. This one-to-many relationship structure allows for a clear and chronological record of all interactions.
- **Case Management Entities:** To support the functional requirements of the STA and SDA, the `cases` and `case_notes` entities are introduced. A `case` is directly linked to a single `conversation` in which a critical risk was detected. Each `case` can have multiple `case_notes`, which are time-stamped entries made by counselors during their review, thereby creating a formal audit trail for each escalation.
- **Administrative Entities:** The `counselors` and `appointments` entities support the administrative functions of the Service Desk Agent. A many-to-many relationship exists between `users` and `counselors` through the `appointments` table, which links a specific `user` and a `counselor` at a scheduled time.
- **Progress Tracking Entity:** The `progress_logs` entity records module completions and check-ins initiated by the Support Coach Agent. It links a `user` to a specific module, storing metadata such as the date completed and any follow-up reminders.

This relational structure ensures that data is logically organized and that complex queries (from retrieving a single conversation's history to aggregating data for the Insights Agent) can be performed efficiently and reliably.

3.6.2 Detailed Table Schemas

This subsection provides a detailed specification for the primary tables in the database schema. For each table, the columns, their corresponding data types in PostgreSQL, and their constraints are defined. The descriptions highlight how each attribute supports the functional requirements of the user-facing application and the four agents of

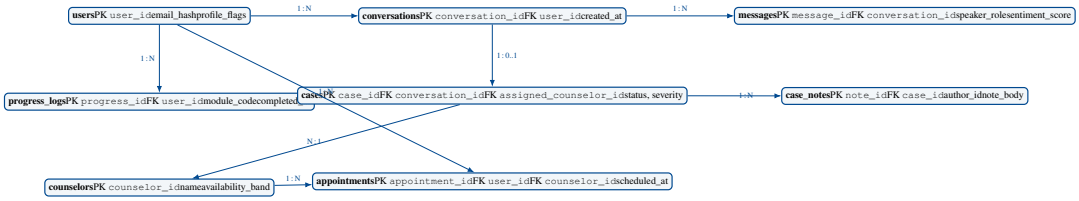


Figure 3.7. Entity–relationship diagram summarising the core tables that support conversational history, safety escalation, and progress tracking. Cardinalities indicate the dominant one-to-many relationships.

the Safety Agent Suite.

3.6.2.1 Users Table

The `users` table serves as the central repository for student information. To adhere to the principle of Privacy by Design, this table is intentionally minimal and uses a non-identifiable primary key.

Table 3.11. Schema for the `users` table.

Column Name	Data Type	Constraints	Description
`user_id`	`UUID`	PK, NOT NULL	A universally unique identifier serving as the primary key. This anonymized ID is used across the system to track user activity without storing PII.
`created_at`	`TIMESTAMPTZ`	NOT NULL	Timestamp indicating when the user account was created.

3.6.2.2 Conversation Logs Table

The `conversation_logs` table is one of the most critical tables in the system. It archives every message from every conversation, serving as the primary data source for the Insights Agent (IA) and providing the necessary context for the real-time agents.

Table 3.12. Schema for the `conversation_logs` table.

Column Name	Data Type	Constraints	Description
`message_id`	`BIGSERIAL`	PK, NOT NULL	Unique identifier for each message.

Table 3.12 – continued from previous page

Column Name	Data Type	Constraints	Description
conversation_id	UUID	FK (conversations), NOT NULL	Foreign key linking the message to a specific conversation session.
user_id	UUID	FK (users), NOT NULL	Foreign key linking the message to the user who sent it. Indexed for fast retrieval of a user's history.
sender	VARCHAR(16)	NOT NULL	Indicates the author of the message (e.g., 'user' or 'agent').
message_text	TEXT	NOT NULL	The raw text content of the message.
timestamp	TIMESTAMPTZ	NOT NULL	The exact time the message was recorded.
sentiment_score	FLOAT		A sentiment score (e.g., from -1 to 1) calculated for the message, used by the IA.
topic	VARCHAR(64)		An NLP-inferred topic label for the message, used by the IA.

3.6.2.3 Cases Table

The cases table is the core of the case management system, operated primarily by the Service Desk Agent (SDA) and monitored by human counselors. Each record represents an incident that was flagged by the Safety Triage Agent (STA).

Table 3.13. Schema for the cases table.

Column Name	Data Type	Constraints	Description
case_id	UUID	PK, NOT NULL	Unique identifier for the case.

Table 3.13 – continued from previous page

Column Name	Data Type	Constraints	Description
conversation_id	UUID	FK (conversations), NOT NULL	Foreign key linking the case to the specific conversation where the risk was detected.
status	VARCHAR(32)	NOT NULL	The current status of the case (e.g., 'new', 'in_progress', 'resolved').
severity	VARCHAR(32)	NOT NULL	The risk level assigned by the STA (e.g., 'critical').
assigned_counselor_id	UUID	FK (counselors)	Foreign key linking the case to a specific counselor for review.
created_at	TIMESTAMPTZ	NOT NULL	Timestamp for when the case was created.
updated_at	TIMESTAMPTZ	NOT NULL	Timestamp for the last update to the case.

3.6.2.4 Progress Logs Table

This table captures the module completions and wellness check-ins driven by the Support Coach Agent (SCA), providing a lightweight audit trail of student progress.

Table 3.14. Schema for the progress_logs table.

Column Name	Data Type	Constraints	Description
progress_id	SERIAL	PK, NOT NULL	Unique identifier for the progress record.
user_id	UUID	FK (users), NOT NULL	Foreign key linking the log entry to the user.
module_code	VARCHAR(64)	NOT NULL	Identifier for the CBT module or exercise completed.
completed_at	TIMESTAMPTZ	NOT NULL	Timestamp for when the module was completed.

Table 3.14 – continued from previous page

Column Name	Data Type	Constraints	Description
follow_up_due_at	TIMESTAMPTZ		Optional reminder for a follow-up check-in, if applicable.
notes	TEXT		Optional remarks recorded by the SCA (e.g., student's self-reported mood).

3.6.3 Data Integrity and Relationships

A robust database schema is defined not only by its entities and attributes but also by the rules that govern the relationships between them. In this framework, data integrity is paramount, particularly given the sensitive nature of the information being managed. The logical consistency and reliability of the data are enforced at the database level through the rigorous application of relational constraints, primarily foreign keys and referential integrity actions.

3.6.3.1 Foreign Key Constraints

The primary mechanism for enforcing relationships between tables is the use of **foreign key constraints**. A foreign key in one table points to a primary key in another table, creating a formal link that the database system actively maintains. This ensures that no orphaned records can exist; for example, a message cannot be created without an associated conversation. Key relationships enforced in this schema include:

- **User to Conversations:** The `user_id` column in the `conversations` table is a foreign key that references the `users` table. This establishes a one-to-many relationship, ensuring every conversation is owned by a single, valid user.
- **Conversation to Messages:** Similarly, the `conversation_id` in the `messages` table is a foreign key referencing the `conversations` table, creating the one-to-many link that groups messages into a single session.
- **Conversation to Cases:** The `conversation_id` in the `cases` table is a foreign key to the `conversations` table. This critical link ensures that every case record is tied directly to the specific conversation in which a risk was detected, providing a clear and auditable path for counselors.
- **Case to Counselors:** The `assigned_counselor_id` in the `cases` table is a foreign key referencing the `counselors` table, ensuring that cases can only be assigned

to valid staff members.

3.6.3.2 Referential Integrity Actions

To further ensure data consistency during deletion operations, referential integrity actions such as `ON DELETE CASCADE` are selectively employed. This rule dictates that if a parent record is deleted, all child records that reference it are also automatically deleted. For instance, the relationship between a `conversation` and its constituent `messages` is configured with a cascading delete. This means that if a conversation record is ever removed from the database (e.g., due to a data retention policy), all associated messages are automatically purged as well, preventing orphaned message records and ensuring the database remains in a consistent state. This automated management is crucial for maintaining the long-term integrity of the data persistence layer.

3.6.4 Schema Design for Privacy and Anonymization

A foundational principle of this framework is **Privacy by Design (PbD)**, which dictates that privacy considerations must be embedded into the core architecture from the outset, not applied as an afterthought [75, 76]. The database schema is a critical implementation of this principle, incorporating several key design patterns to protect user anonymity and minimize data exposure.

3.6.4.1 User Anonymization by Default

The primary strategy for protecting user privacy is to disassociate conversational data from any real-world Personally Identifiable Information (PII). This is achieved at the schema level:

- **Non-Identifiable Primary Keys:** The `users` table's primary key, `user_id`, is a randomly generated Universally Unique Identifier (UUID). This ID is used as the sole identifier for a user across all related tables (e.g., `conversations`, `messages`, `cases`). Crucially, the database schema intentionally omits any columns that would store direct PII such as student names, university ID numbers, or email addresses.
- **Decoupling of Sensitive Data:** While the core conversational system is fully anonymized, features like appointment scheduling may require linking a user to a real-world identity. This information is handled outside of the primary analytical database. Any tables containing PII are strictly segregated and are not accessible by the analytical or conversational agents, ensuring a clear boundary between operational and analytical data.

3.6.4.2 Data Minimization and Access Control

The schema is designed to support the principle of data minimization, ensuring that agents and services only have access to the information strictly necessary for their

function.

- **Purpose-Driven Schema for Analytics:** The `conversation_logs` table, which is the data source for the Insights Agent (IA), is designed specifically for aggregated analysis. It contains the message text and metadata like timestamps and sentiment scores but deliberately excludes any link back to sensitive administrative data.
- **Role-Based Access at the Application Layer:** While not a schema-level feature, the backend application enforces strict access controls. The database user credentials assigned to the IA, for example, have read-only permissions and are restricted to only the anonymized `conversation_logs` table. This prevents any possibility of the analytical agent accessing clinical case notes or other sensitive information.

3.6.4.3 Application-Layer PII Redaction

To further mitigate the risk of unintentional data exposure, a PII redaction process is implemented in the backend service **before** any message is written to the `conversation_logs` table. When a user sends a message, the FastAPI application performs a redaction step to identify and remove common PII patterns (e.g., email addresses, phone numbers, names). This means that the data is anonymized before it is even persisted for long-term analysis, providing an additional layer of defense and ensuring that the data processed by the Insights Agent is fundamentally privacy-safe.

3.7 User Experience (UX) Design

Effective artifact design in Design Science Research necessitates a deep understanding of the end-users' needs, goals, and contexts. Therefore, prior to designing the visual user interface, a user experience (UX) design process was undertaken to ensure the final application is not only functional but also intuitive, accessible, and aligned with the specific needs of its target users. This process was centered around the development of user personas and the formulation of key user stories, which served as foundational guides for all subsequent UI design and feature prioritization decisions [**?**].

3.7.1 User Personas

To empathize with the primary users of the system, two distinct personas were created. These personas are fictional, archetypal users whose characteristics, goals, and pain points represent the intended audience for the User Portal and the Admin Dashboard, respectively.

3.7.1.1 Primary Persona: The Student

- **Name:** Budi

- **Profile:** A first-year undergraduate student at UGM, living away from home for the first time. He is navigating the academic pressures of a demanding course load while also trying to build a new social life.
- **Goals:**
  - To find a private and non-judgmental space to articulate his feelings of stress and anxiety.
  - To learn practical, evidence-based coping strategies that he can apply to his daily life.
  - To feel a sense of progress and accomplishment in managing his well-being.
  - If needed, to find a way to connect with a professional counselor without a complicated or intimidating process.
- **Pain Points & Hesitations:**
  - He is worried about the stigma associated with seeking mental health support and fears that his peers or professors might find out.
  - He finds the process of booking a formal counseling session to be daunting and is unsure if his problems are "serious enough" to warrant it.
  - He often feels overwhelmed late at night, when traditional support services are unavailable.

3.7.1.2 Secondary Persona: The University Counselor

- **Name:** Dr. Astuti
- **Profile:** The Head of Counseling Services at the university. She is an experienced clinical psychologist who is deeply committed to student well-being but is constrained by limited resources.
- **Goals:**
  - To move from a reactive, crisis-driven support model to a more proactive, preventative one.
  - To gain a better, data-driven understanding of the most pressing mental health challenges facing the student population at any given time.
  - To allocate her team's limited time and resources more effectively, focusing on students who need the most help.
  - To ensure that no student in critical distress falls through the cracks.
- **Pain Points & Hesitations:**

- Her team is overwhelmed with a long waiting list for appointments, and she worries about students who need help but are not reaching out.
- The data she currently has is anecdotal, making it difficult to justify requests for more resources or to design targeted, preventative workshops.
- She spends a significant amount of time on administrative tasks rather than on clinical work.

3.7.2 Key User Scenarios and Stories

Derived from the user personas, a set of key user stories was formulated to guide the design of the application's features. These stories capture the desired functionalities from the user's perspective.

3.7.2.1 For the Student (Budi)

- **Scenario 1: Seeking Immediate Support.**
  - *As a student feeling overwhelmed after a difficult exam, I want to chat with an AI assistant that can listen to me and offer some immediate coping strategies, so that I can manage my anxiety in the moment.*
- **Scenario 2: Building Skills Over Time.**
  - *As a student who wants to improve my well-being, I want to be guided through structured exercises and track my progress, so that I can feel a sense of accomplishment and see how far I've come.*
- **Scenario 3: Accessing Professional Help.**
  - *As a student who has decided I need to talk to a person, I want a simple and discreet way to book an appointment with a counselor, so that I can get professional help without navigating a complex bureaucracy.*

3.7.2.2 For the Counselor (Dr. Astuti)

- **Scenario 1: Proactive Strategy.**
  - *As the Head of Counseling, I want to receive a weekly automated report on the top mental health trends among students, so that I can plan targeted workshops and allocate my team's resources effectively.*
- **Scenario 2: Crisis Management.**

– *As a counselor, I want to receive an immediate alert on my dashboard when a student's conversation is flagged for critical risk, so that I can intervene quickly and ensure the student's safety.*

These personas and user stories form the user-centric foundation upon which the User Interface (UI), detailed in the following section, was designed.

3.7.3 Core Design Principles

Based on the insights gathered from the user personas and scenarios, the following core design principles were established to guide the user interface and interaction design of the UGM-AICare platform:

- **Principle 1: Privacy and Discretion First.** Every design choice must prioritize the user's sense of privacy and safety. This means avoiding stigmatizing language, ensuring private information is never displayed unnecessarily, and giving the user control over their data. This directly addresses Budi's fear of stigma.
- **Principle 2: Clarity Over Clutter.** The interfaces, especially for the student, must be simple, clean, and intuitive. When a user is in distress, they should not be burdened with a complex UI. This principle guides the design towards clear navigation and focused content.
- **Principle 3: Actionable Insights for Administrators.** The Admin Dashboard must present data in a way that is immediately understandable and actionable. It should not be a raw data dump, but a curated set of visualizations that directly supports Dr. Astuti's goal of making informed, proactive decisions.
- **Principle 4: Foster a Sense of Progress.** To encourage sustained engagement, the UI should incorporate elements that provide positive reinforcement and a clear sense of accomplishment, directly supporting Budi's desire to feel progress in his well-being journey.

3.8 User Interface (UI) Design

The User Interface (UI) design translates the principles and user stories from the UX research phase into a tangible, visual blueprint for the UGM-AICare application. The design of both the Admin Dashboard and the User Portal is guided by the core principles of clarity, privacy, and actionability. The following mockups represent the key screens of the application, designed to meet the specific needs of the personas: Dr. Astuti and Budi.

3.8.1 The Admin Dashboard: An Interface for Proactive Oversight

The Admin Dashboard is designed for Dr. Astuti, the Head of Counseling Services. Its primary purpose is to provide actionable, at-a-glance insights and an efficient case management system, enabling a shift from reactive to proactive support.

3.8.1.1 Main Analytics View

As shown in Figure 3.8, the main dashboard is the central hub for strategic oversight.

- **Design Justification:** To address Dr. Astuti's goal of gaining a data-driven understanding of student well-being, the dashboard prominently features key performance indicators (KPIs) at the top, such as "Overall Sentiment Trend" and "Active Cases." The main panel is dedicated to a time-series visualization of the "Top Trending Topics" identified by the Insights Agent (IA). This design directly supports her user story of wanting an automated report to plan targeted workshops effectively.

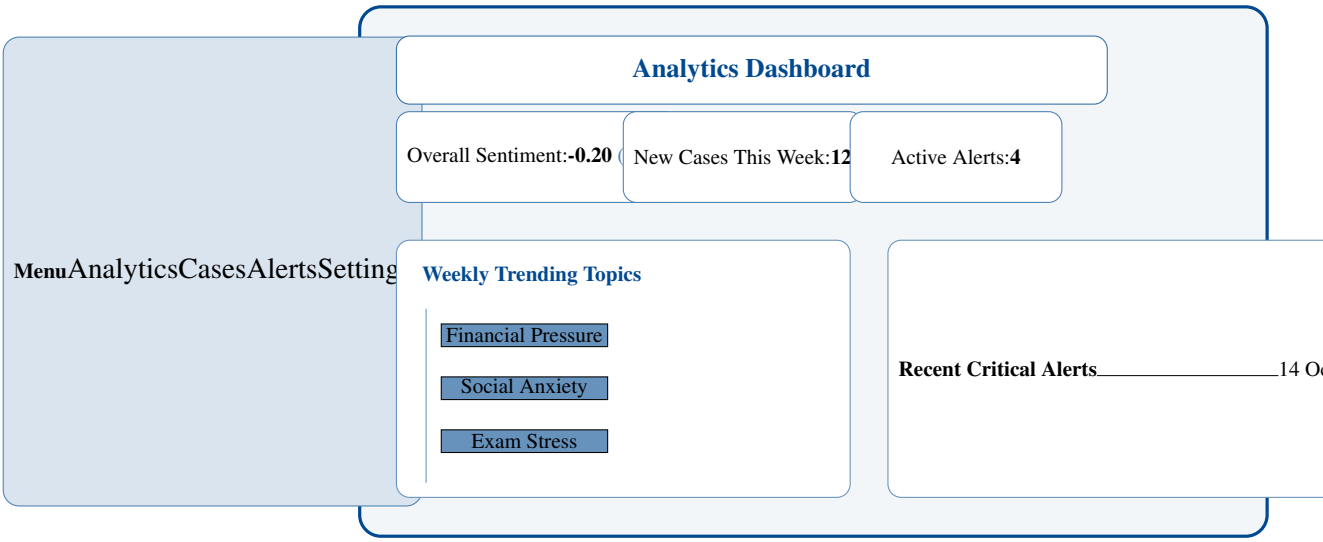


Figure 3.8. Admin dashboard main view highlighting KPIs, trending topics, and the most recent critical alerts surfaced by the agents.

3.8.1.2 Case Management View

When a conversation is flagged by the STA, it appears in the case management view, depicted in Figure 3.9.

- **Design Justification:** To address Dr. Astuti's pain point of being overwhelmed by administrative tasks and her goal of ensuring no student in crisis is missed, this screen is designed for efficiency. It presents a clean, sortable table of all active cases. When a case is selected, it displays the relevant (anonymized) conversation log, the severity level assigned by the STA, and clear action buttons like "Assign to Counselor" or

"Mark as Resolved." This streamlines the workflow from automated detection to human intervention.

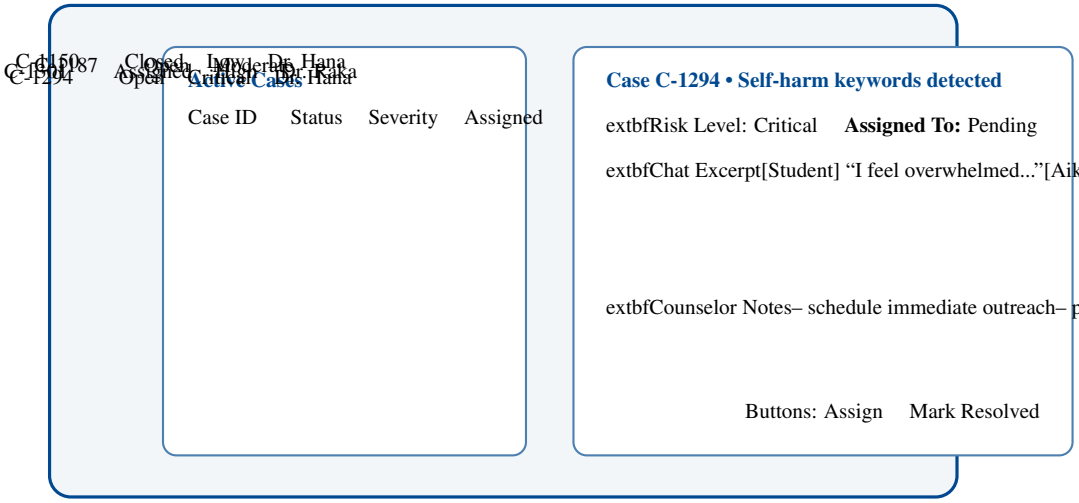


Figure 3.9. Case management interface showing the triaged case list alongside the selected conversation context and follow-up actions.

3.8.2 The User Portal: A Private and Supportive Space

The User Portal is designed for Budi, the first-year student. The UI prioritizes simplicity, privacy, and a sense of calm, directly addressing his hesitation to seek help and his desire for a non-intimidating support tool.

3.8.2.1 The ’/aika’ Chat Interface

The core of the student experience is the chat interface, shown in Figure 3.10.

- **Design Justification:** To align with the principle of "Clarity Over Clutter," the interface is intentionally minimalist, resembling modern messaging applications to feel familiar and intuitive. There are no distracting elements; the focus is solely on the conversation between Budi and the Support Coach Agent. This design directly supports his goal of finding a private and focused space to articulate his feelings.

3.8.2.2 The User Dashboard and Progress Tracking

To foster engagement and a sense of progress, the user dashboard (Figure 3.11) visualizes the student’s journey.

- **Design Justification:** This screen directly supports Budi’s goal of feeling a sense of accomplishment. It includes a section for "My Progress" that shows completed modules, highlights recommended next steps, and surfaces gentle reminders for upcoming check-ins. This visual feedback, based on the design principle of "Foster a Sense of Progress," reinforces positive behavior and encourages continued engagement with the platform’s coaching features.

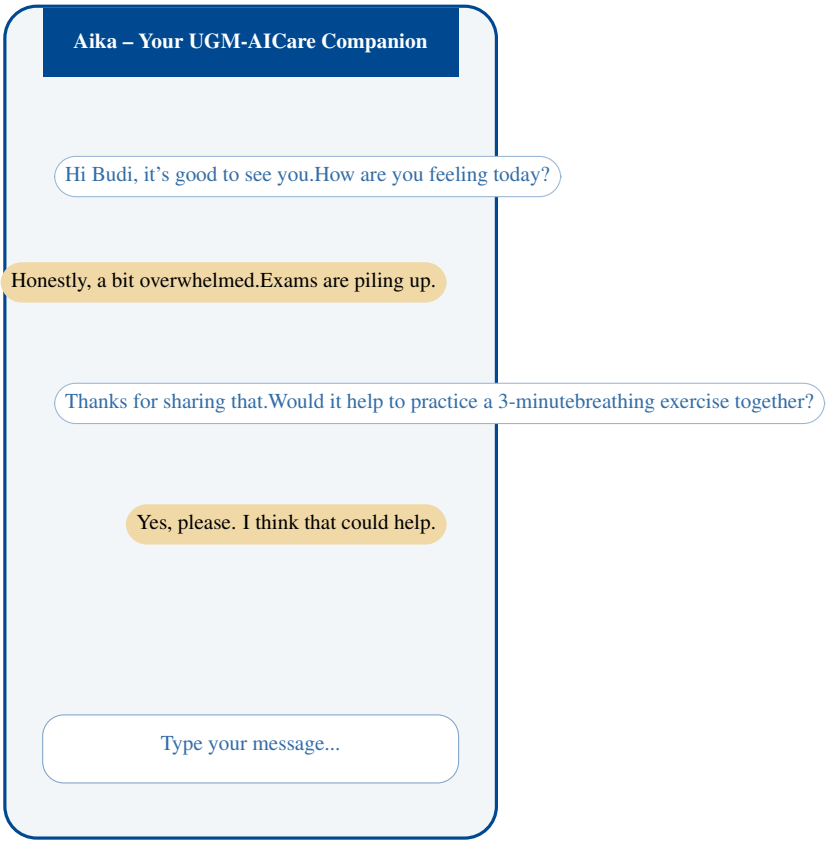


Figure 3.10. Minimalist chat interface designed to keep the student's attention on the conversation with Aika.

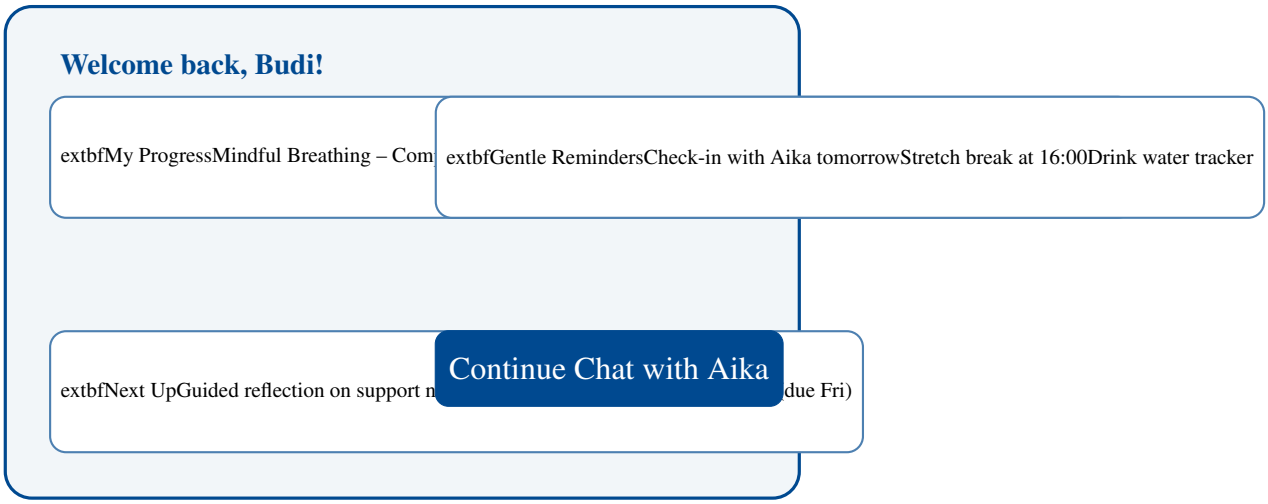


Figure 3.11. Personal dashboard emphasising completed activities, recommended follow-ups, and a clear path back into the coaching conversation.

3.8.3 Interaction Flows and User Journeys

While the preceding sections have defined the static components of the user interface, this subsection details the dynamic, step-by-step user journeys for key scenarios. These flows illustrate how a user navigates the application to achieve a specific goal and how the different agents of the Safety Agent Suite collaborate to facilitate these interactions.

3.8.3.1 Journey 1: Making an Appointment with a Counselor

This journey is designed to be as frictionless as possible, directly addressing Budi's hesitation with complex administrative processes.

1. **Initiation:** The user can initiate the process in two ways: either by navigating to an "Appointments" section in the User Portal or by expressing the intent to speak with a counselor during a chat with the Support Coach Agent (SCA).
2. **Agent Handoff:** If initiated via chat, the SCA recognizes the intent and calls a tool that invokes the Service Desk Agent (SDA).
3. **Scheduling:** The SDA's `check_calendar_availability` tool fetches available time slots from the counselors' calendars and presents them to the user in a simple interface.
4. **Confirmation:** The user selects a time slot. The SDA then calls its `book_appointment` tool, which confirms the appointment in the system and sends a confirmation notification to both the user and the assigned counselor.

3.8.3.2 Journey 2: Standard Interaction with Aika (SCA)

This flow represents the primary use case of the chat interface, where a student seeks supportive conversation.

1. **Initiation:** The user navigates to the '/aika' chat page and sends a message.
2. **Real-Time Triage:** The message is first intercepted by the Safety Triage Agent (STA), which performs an instantaneous risk assessment.
3. **Safe Handoff:** Assuming the risk is classified as "Low" or "Moderate," the message is passed to the Support Coach Agent (SCA).
4. **Conversational Loop:** The SCA generates an empathetic, context-aware response and sends it to the user. This loop (User Message -> STA -> SCA -> User Response) continues for the duration of the conversation.

3.8.3.3 Journey 3: Crisis Escalation and Case Management

This journey illustrates the system's core safety protocol.

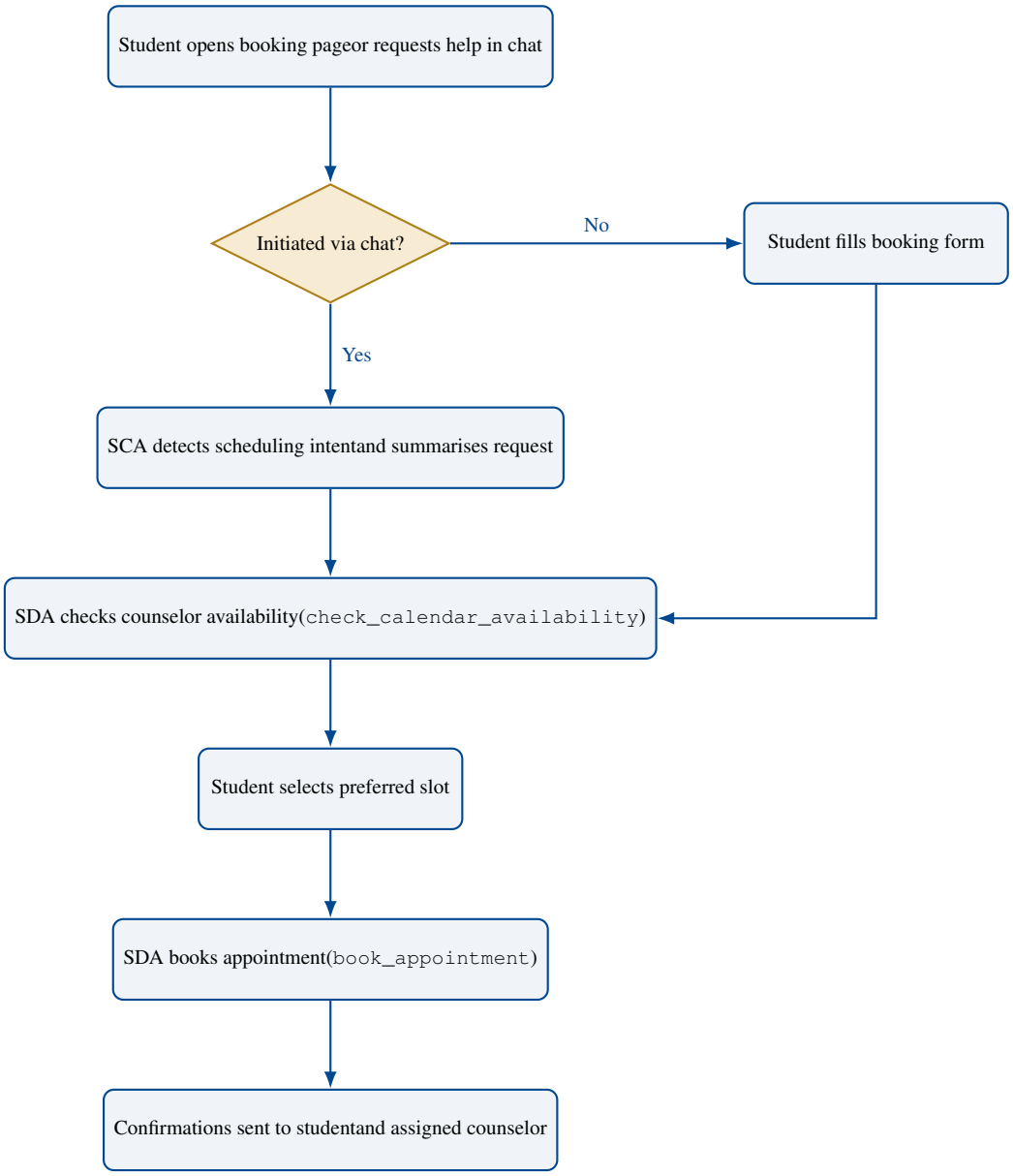


Figure 3.12. Appointment booking journey showing both chat-triggered and self-initiated flows converging on the Service Desk Agent for scheduling and confirmation.

1. **Detection:** During a standard conversation, the user sends a message that the STA classifies as "Critical."
2. **Automated Escalation:** The STA immediately interrupts the normal flow. It invokes a tool that:
   - Displays pre-defined emergency resources to the user.
   - Sends a high-priority alert to the Admin Dashboard.
   - Instructs the Service Desk Agent (SDA) to create a new case, linking it to the conversation.
3. **Human Intervention:** A counselor (Dr. Astuti) sees the alert on the dashboard, reviews the newly created case, and initiates follow-up. The counselor uses the case management interface to document all actions taken.

Placeholder for Flowchart: Crisis Escalation Journey

This flowchart should show the path from STA detection to the SDA creating a case and the final review by a human counselor on the Admin Dashboard.

Figure 3.13. The user and system journey during a critical risk escalation.

3.8.3.4 Journey 4: Administrator Reviewing Weekly Insights

This flow describes how an administrator like Dr. Astuti uses the system for strategic planning.

1. **Automated Analysis:** On its pre-defined schedule (e.g., Sunday at midnight), the Insights Agent (IA) is triggered.
2. **Report Generation:** The IA queries the anonymized database, performs its analysis, and generates a new strategic report.
3. **Notification and Access:** The IA sends an email notification to Dr. Astuti and pushes the new report to the Admin Dashboard.
4. **Review:** Dr. Astuti logs into the dashboard, navigates to the analytics view, and reviews the latest trends to inform her weekly planning.

3.8.3.5 Journey 5: Proactive Outreach by the SCA

This journey demonstrates the system's proactive capabilities, closing the loop from insight to intervention.

1. **Configuration:** Based on the IA's report showing a spike in "exam stress," Dr. Astuti configures a proactive outreach campaign from the Admin Dashboard. She targets all active users and selects a pre-defined message offering a time management module.
2. **Agent Action:** The system tasks the Support Coach Agent (SCA) to act.
3. **User Contact:** The next time a targeted student (Budi) logs into the platform, the SCA initiates the conversation with the configured message, such as, "Hi Budi, with exams coming up, many students are feeling stressed. Would you like to try a quick module on effective study habits?"

3.8.3.6 Journey 6: Contextual Module Suggestion

This flow shows how the SCA provides relevant support within a natural conversation.

1. **Conversation Context:** During a chat, a student (Budi) expresses feelings of anxiety and worry about an upcoming presentation.
2. **Intent Recognition:** The SCA, using the reasoning power of the Gemini API, recognizes this as an opportunity to offer a specific, helpful tool.
3. **Module Suggestion:** The SCA responds empathetically and then suggests a relevant intervention. For example: "It sounds like that presentation is causing a lot of anxiety. I have a guided breathing exercise that many students find helpful for calming their nerves in moments like this. Would you like to try it?"
4. **Module Delivery:** If the user agrees, the SCA invokes its retrieve_cbt_module tool and presents the exercise directly within the chat interface.

3.9 Security and Privacy by Design

Given the profoundly sensitive nature of mental health data, the design and implementation of the UGM-AICare framework are fundamentally guided by the principles of **Privacy by Design (PbD)**. As established in the theoretical background, PbD dictates that privacy and security must be the default, embedded proactively into the system's architecture from the very beginning, rather than being treated as subsequent additions [**?**]. This section details the specific technical and procedural mechanisms implemented to ensure the confidentiality, integrity, and ethical use of user data.

3.9.1 Data Anonymization and Minimization

The cornerstone of the framework's privacy strategy is a multi-layered approach to data anonymization, ensuring that the data used for analytics is irrevocably decoupled from any real-world user identity.

3.9.1.1 Application-Layer PII Redaction Pipeline

To prevent the accidental storage of Personally Identifiable Information (PII), a redaction pipeline is executed within the FastAPI backend for every message received from a user *before* it is written to the `conversation_logs` table. This automated process involves:

1. **Pattern Matching:** The system uses regular expressions to identify and remove common PII patterns, such as email addresses, phone numbers, and student ID numbers.
2. **Named Entity Recognition (NER):** A lightweight NLP model is used to identify and redact proper nouns that are likely to be names of people or specific locations.

This proactive redaction ensures that the data persisted for long-term storage and analysis is anonymized by default.

3.9.1.2 Anonymized User Identifiers

As detailed in the database design, the system never stores a user's real name, email, or university ID. Each user is assigned a randomly generated Universally Unique Identifier (UUID) upon their first interaction, which serves as their primary key throughout the database. This ensures that even within the system, a user's activity is linked to a pseudonym, not a real identity.

3.9.2 Architectural Security Measures

Beyond anonymization, the framework implements standard, industry-best-practice security measures to protect the integrity and confidentiality of the entire system.

- **Role-Based Access Control (RBAC):** Access to the Admin Dashboard is strictly controlled by an RBAC mechanism. Only authenticated and authorized university staff with an 'ADMIN' role may have access to configure the system and view flagged cases.
- **Secure Communication (HTTPS/TLS):** All communication between the user's browser, the Next.js frontend, and the FastAPI backend occurs exclusively over HTTPS, with traffic encrypted using Transport Layer Security (TLS). This prevents eavesdropping and man-in-the-middle attacks, ensuring that all data is confidential while in transit.

- **Data Encryption at Rest:** The PostgreSQL database is configured to encrypt all data at rest. This means that even if an unauthorized party were to gain access to the physical storage on the server, the database files would be unreadable without the encryption keys.
- **Secure Deployment Environment:** The entire application stack is deployed within a secure Virtual Machine provided by PT INA17, protected by firewalls and managed access controls. The Nginx reverse proxy is configured with up-to-date security headers to protect against common web vulnerabilities like cross-site scripting (XSS) and clickjacking.

3.9.3 Ethical Safeguards and Human Oversight

Technology alone is insufficient to guarantee ethical operation. Therefore, the system is designed with procedural safeguards that ensure human oversight for all critical functions.

- **Human-in-the-Loop for Safety:** The framework is explicitly designed to be a tool that assists, but does not replace, human counselors. Every critical risk escalation from the Safety Triage Agent (STA) creates a case that requires mandatory review and action by a qualified human professional. The system automates the detection and reporting, but the final clinical judgment and intervention remain firmly in human hands.
- **Purpose Limitation:** The data collected through the chat interface is used for the sole and explicit purposes of providing in-the-moment support, managing clinical escalations, and generating anonymized, aggregated statistics for improving the university's support services. The data is not used for any other purpose, such as academic assessment or disciplinary action. This principle is enforced through the technical separation of the anonymized analytical data from any administrative records.

3.10 Ethical Considerations and Research Limitations

The development of an AI-driven framework for mental health support necessitates a thorough examination of the ethical implications and a transparent acknowledgment of the research's limitations. This section addresses these considerations, framing the ethical design choices and defining the boundaries of the study's findings.

3.10.1 Ethical Considerations

The design of the Safety Agent Suite is grounded in the university's fundamental duty of care to its students. The following ethical principles were central to the framework's architecture.

- **Informed Consent and Transparency:** A core ethical requirement is that users must be explicitly informed that they are interacting with an AI system. The user interface must clearly state that "Aika" is an AI assistant and provide a link to a privacy policy that explains in simple terms how their data is anonymized and used for aggregated statistical analysis. Users must provide explicit consent to these terms before beginning their first interaction.
- **The Risk of AI Misinterpretation:** While Large Language Models are powerful, they are not infallible and can misinterpret the nuances of human emotion and language. The most significant ethical risk is the failure to detect a genuine crisis (a false negative). The architecture mitigates this risk through the **human-in-the-loop** design of the Safety Triage Agent (STA). Every automated escalation is treated as a high-priority alert that requires mandatory review and follow-up by a qualified human counselor, ensuring that the final judgment on safety-critical situations is never left to the machine alone.
- **AI as a Support Tool, Not a Replacement for Therapy:** It is ethically imperative to clearly define the system's role. The UGM-AICare framework is designed as a sub-clinical, supportive tool and a bridge to professional care, not as a substitute for it. The Support Coach Agent (SCA) is programmed to state this boundary clearly and to encourage users to seek professional help for serious or persistent issues, using the Service Desk Agent (SDA) to facilitate the booking process.
- **Data Privacy and Purpose Limitation:** As detailed in the Security and Privacy by Design section, the system is architected to protect user anonymity. Furthermore, the principle of purpose limitation is strictly enforced. The data collected is used only for the explicit purposes of providing in-the-moment support and generating aggregated, anonymized insights to improve the university's services. It is never used for academic assessment, disciplinary action, or any other purpose outside its stated mission.

3.10.2 Research Limitations

This study, as a work of Design Science Research focused on the creation and evaluation of a novel artifact, is subject to several important limitations that define the scope of its conclusions.

- **Methodological Limitation: Evaluation of a Prototype:** The evaluation of this framework, as will be detailed in Chapter 4, is based on the functional testing of a proof-of-concept prototype against predefined scenarios. This thesis validates the *technical feasibility* of the agentic workflows and the *architectural integrity* of the design. However, it is not a clinical trial and does not measure the long-term psychological impact or therapeutic efficacy of the system on actual students. Such claims would

require a separate, longitudinal study with appropriate clinical oversight.

- **Technical Limitation: Inherent Risks of LLMs:** The framework relies on the Google Gemini 2.5 API. Like all LLMs, it is subject to inherent limitations. These include the potential for the model to reflect biases present in its vast training data and the possibility of generating factually incorrect or nonsensical responses ("hallucinations"). While the system's use of structured tools and prompts is designed to mitigate these risks, they cannot be eliminated entirely.
- **Data Limitation: Use of Simulated Data:** The evaluation of the Insights Agent (IA) will be conducted using anonymized, pre-existing chat logs or simulated data. While this is necessary to protect user privacy during the development phase, it means that the agent's performance has not been validated on the specific linguistic patterns and diversity of a live, campus-wide user base. The effectiveness of the topic modeling and sentiment analysis in a real-world context would require further validation post-deployment.

CHAPTER IV

IMPLEMENTATION AND EVALUATION (HASIL DAN PEMBAHASAN)

This chapter reports how the prototype was exercised and what we learned from it. The focus is on the agents and their behavior in safety-relevant scenarios. We keep the scope practical and transparent so results can be reproduced and audited.

4.1 Setup and Test Design (Rancangan Pengujian)

This section documents the evaluation protocol that links the Design Science stages in Chapter III to the research questions in Chapter 1.4. Figure 4.14 and Table 4.1 provide a visual and tabular overview of the assets, metrics, and acceptance thresholds used throughout the chapter.

Evaluation Environment

- **Agents under test**: Safety Triage Agent (STA), Support Coach Agent (SCA), Service Desk Agent (SDA), and Insights Agent (IA) running inside the LangGraph orchestration described in Chapter III. All tool invocations are captured through structured logs to enable replay and auditing.
- **Execution platform**: FastAPI backend deployed in a containerised environment (Python 3.11, Uvicorn workers $= 8$) with Redis for task queues and PostgreSQL 15 for persistence. Tests are executed on a machine equivalent to 8 vCPU/32 GB RAM to mirror expected production sizing.
- **Instrumentation**: OpenTelemetry traces capture latency, tool-call success, and retries; custom middleware records human hand-offs, while differential privacy parameters are logged for the IA.

Datasets and Scenario Assets

- **Crisis corpus**: 500 labeled prompts covering self-harm, violence, and acute distress, augmented with 300 non-crisis but emotionally charged messages to measure false positives. Labels are derived from established mental health crisis assessment guidelines and validated through systematic comparison with published crisis intervention frameworks.
- **Coaching prompts**: 120 conversation snippets spanning stress management, motivation, academic planning, and administrative queries. Responses include canonical "out-of-scope" triggers to exercise refusal and escalation behaviour.

- **Operational events**: Synthetic scheduling and case-management payloads to drive SDA workflows, and a 12-week anonymised log (synthetic) to test IA stability and privacy thresholds (minimum cohort size $k = 50$, placeholder $\epsilon = 1.0$).

Quality Control and Validation

- **Safety validation**: All STA critical detections are validated against ground-truth labels derived from established crisis assessment frameworks. Disagreements between system output and expected classifications are logged and analyzed for patterns in Chapter 4.6.
- **Coaching quality rubric**: Three independent raters score SCA responses on CBT adherence, empathy, and appropriateness using a standardized 1–5 Likert scale rubric. Inter-rater reliability (Cohen's κ) is calculated and reported alongside mean scores to ensure assessment consistency.
- **Analytics verification**: IA outputs are inspected for k-anonymity compliance; any aggregate below the threshold is expected to be suppressed automatically.

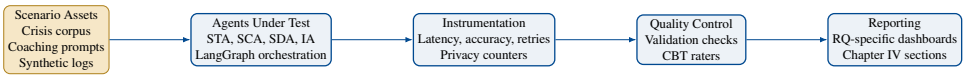


Figure 4.14. Evaluation workflow linking scenario assets, instrumentation, and quality control validation to the reporting structure in Chapter IV.

4.2 RQ1 - Safety: Can STA detect crises promptly?

Desain. Uji pada set krisis sintetis dengan label. Ukur sensitivitas, spesifisitas, dan waktu ke eskalasi dari deteksi awal hingga pembuatan tiket/alert. Analisis khusus pada kegagalan berisiko (*false negatives*).

Hasil. Ringkas angka utama (mis. sensitivitas, spesifisitas, p50/p95 latensi). Tampilkan contoh sukses dan kegagalan yang representatif.

Bahasan. Kompromi antara kecepatan dan kehati-hatian; peran *guardrail* dan *fallback* manusia.

4.3 RQ2 — Reliability: Does orchestration run reliably?

Desain. Telusuri rasio keberhasilan pemanggilan fungsi, validasi skema, *retry/back-off*, dan penyelesaian *workflow* ujung-ke-ujung.

Hasil. Laporkan tingkat keberhasilan, tingkat kegagalan yang pulih, dan kasus terhenti (jika ada). Sertakan latensi p50/p95 per langkah.

Bahasan. Pola kegagalan yang paling sering dan perbaikan yang mudah diterapkan.

Table 4.1. Evaluation plan mapped to research questions and acceptance thresholds.

RQ	**Test Scenarios / Data**	**Primary Metrics**	**Success Criteria (Target)**	**Related Section**
RQ1 (Safety)	500 crisis/non-crisis prompts; end-to-end escalation drills	Sensitivity, specificity, precision, FNR, detection latency p95/p99	Sensitivity ≥ 0.95, FNR < 0.02, escalation < 30 s, p95 latency < 0.25 s	§4.2
RQ2 (Reliability)	Conversational stress test (500 concurrent sessions); failure injection (API outage, malformed payload)	Tool-call success rate, mean retries per call, workflow completion %, latency p50/p95	Success rate ≥ 0.98, average retries ≤ 0.3, workflow completion ≥ 0.97, p95 latency < 1.5 s	§4.3
RQ3 (Quality)	120 coaching prompts scored by 3 raters	CBT adherence score, empathy, refusal accuracy, inter-rater κ	Mean scores ≥ 4 (out of 5), $\kappa \geq 0.75$, correct refusal ≥ 0.9	§4.4
RQ4 (Insights)	12-week synthetic log with known topic distribution; privacy stress test	Topic stability (Jensen-Shannon divergence), sentiment drift, suppression rate, DP noise magnitude	Divergence ≤ 0.1, all aggregates respect $k \geq 50$, suppression rate $\leq 5\%$, DP noise within planned bounds	§4.5

4.4 RQ3 — Quality: Are SCA responses reasonable and CBT-informed?

Desain. Penilaian buta oleh evaluator pada sampel percakapan (kecil namun beragam). Rubrik menilai kepatuhan CBT dasar, keamanan saran, dan empati.

Hasil. Skor ringkas dan contoh tanggapan baik/kurang baik. Catat penolakan yang tepat pada topik di luar batas.

Bahasan. Pola perbaikan prompt/alat yang berdampak nyata.

4.5 RQ4 — Insights (minimal): Can IA produce safe aggregate views?

Desain. Jalur agregasi sederhana: ambang privasi (mis. k-anonymity) dan cek kestabilan jumlah/topik. Tidak ada klaim level individu.

Hasil. Laporkan hanya *sanity check* agregat (mis. stabil/variatif) dan kepatuhan terhadap ambang privasi.

Bahasan. Keterbatasan desain saat ini dan langkah aman untuk perluasan.

4.6 Discussion and Limitations (Diskusi dan Keterbatasan)

- **Temuan utama**. Soroti apa yang berjalan baik (mis. orkestrasi stabil, latensi terkendali) dan apa yang perlu diperkuat (mis. penanganan tepi kasus tertentu).
- **Batasan**. Prototipe, data sintetis/anonym, tidak ada klaim efek klinis; model dapat bias/*hallucinate* meski ada *guardrail*.
- **Implikasi**. Perbaikan sederhana yang memberi dampak besar; rencana evaluasi lanjutan (*field* kecil) dengan pengawasan etik yang memadai.

CHAPTER V
TAMBAHAN (OPSIONAL)

Anda boleh menambahkan Bab jika diperlukan. Jumlah Bab tidak harus sesuai dengan *template*.

Bab tambahan ini diperlukan jika hasil penelitian untuk menjawab tujuan cukup panjang atau terdiri dari banyak sub bab. Mahasiswa boleh menjawab 1 tujuan penelitian dengan 1 bab.

CHAPTER VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Kesimpulan dapat diawali dengan apa yang dilakukan dengan tugas akhir ini lalu dilanjutkan dengan poin-poin yang menjawab tujuan penelitian, apakah tujuan sudah tercapai atau belum, tentunya berdasarkan data ataupun hasil dari Bab pembahasan sebelumnya. Dalam beberapa hal, kesimpulan dapat juga berisi tentang temuan/*findings* yang Anda dapatkan setelah melakukan pengamatan dan atau analisis terhadap hasil penelitian.

Kesimpulan menjawab seberapa jauh rumusan masalah tercapai berdasarkan hasil penelitian. Semua rumusan masalah harus disimpulkan berdasarkan data penelitian.

6.2 Saran

Saran berisi hal-hal yang bisa dilanjutkan dari penelitian atau skripsi ini, yang belum dilakukan karena batasan permasalahan. Saran bukan berisi saran kepada sistem atau pengguna, tetapi saran diberikan kepada aspek penelitian yang dapat dikembangkan dan ditambahkan di penelitian atau skripsi selanjutnya.

Catatan: Mahasiswa perlu melihat sinkronisasi antara rumusan masalah, tujuan, metode, hasil penelitian, dan kesimpulan.

REFERENCES

[1] M. Hill, N. Farrelly, C. Clarke, and M. Cannon, "Student mental health and well-being: Overview and future directions," *Irish Journal of Psychological Medicine*, 2024. [Online]. Available: https://www.cambridge.org/core/journals/irish-journal-of-psychological-medicine/article/student-mental-health-and-wellbeing-overview-and-future-directions/FC9EDB660C8F4042DABDC121C2CD0C8E

[2] Z. H. Duraku, H. Davis, A. Arënliu, and F. Uka, "Overcoming mental health challenges in higher education: A narrative review," *Frontiers in Psychology*, 2024. [Online]. Available: https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2024.1466060/full

[3] National Academies of Sciences, Engineering, and Medicine, *Mental Health, Substance Use, and Wellbeing in Higher Education: Supporting the Whole Student*, L. A. Scherer and A. I. Leshner, Eds. National Academies Press, 2021. [Online]. Available: https://books.google.co.id/books?id=H_UeEAAAQBAJ

[4] S. K. Lipson, E. G. Lattie, and D. Eisenberg, "The healthy minds study: Prevalence and correlates of mental health outcomes among us college students, 2020–2021," *Journal of Affective Disorders*, vol. 306, pp. 377–386, 2022. [Online]. Available: https://doi.org/10.1016/j.jad.2022.03.037

[5] R. P. Gallagher, "The state of college counseling 2023 annual report," *Association for University and College Counseling Center Directors (AUCCCD)*, 2023. [Online]. Available: https://www.aucccd.org/assets/documents/aucccd-annual-survey-public-2023.pdf

[6] C. Baik, W. Larcombe, and A. Brooker, "How universities can enhance student mental wellbeing: The student perspective," *Higher Education Research & Development*, vol. 38, no. 4, pp. 674–687, 2019. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/07294360.2019.1576596

[7] F. Outay, N. Jabeur, F. Bellalouna, and T. Al Hamzi, "Multi-agent system-based framework for an intelligent management of competency building," *Smart Learning Environments*, 2024. [Online]. Available: https://link.springer.com/article/10.1186/s40561-024-00328-3

[8] A. Omirali, K. Kozhakhmet, and R. Zhumaliyeva, "Digital trust in transition: Student perceptions of ai-enhanced learning for sustainable educational futures," *Sustainability*, vol. 17, no. 17, p. 7567, 2025. [Online]. Available: https://www.mdpi.com/2071-1050/17/17/7567

[9] A. K. Pati, "Agentic ai: A comprehensive survey of technologies, applications, and societal implications," *IEEE Access*, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/11071266/

[10] N. Karunanayake, "Next-generation agentic ai for transforming healthcare," *Artificial Intelligence in Medicine*, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2949953425000141

[11] R. L. Jørnø and K. Gynther, "What constitutes an "actionable insight" in learning analytics?" *Journal of Learning Analytics*, vol. 5, no. 3, pp. 198–221, 2018. [Online]. Available: https://learning-analytics.info/index.php/JLA/article/view/5897

[12] T. Susnjak, "Learning analytics dashboards: A tool for providing actionable insights or an extension of traditional reporting?" *International Journal of Educational Technology in Higher Education*, vol. 19, no. 2, pp. 17–32, 2022. [Online]. Available: https://educationaltechnologyjournal.springeropen.com/articles/10.1186/s41239-021-00313-7

[13] K. Saleem, M. Saleem, and A. Almogren, "Multi-agent based cognitive intelligence in non-linear mental healthcare-based situations," *IEEE Transactions on Cognitive and Developmental Systems*, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10896654/

[14] M. Wooldridge, *An Introduction to MultiAgent Systems*, 2nd ed. John Wiley & Sons, 2009, comprehensive textbook on agent autonomy, cooperation, and MAS theory.

[15] A. Salutari, "Harmonizing users' and system's requirements in complex and resource intensive application domains by a distributed hybrid approach," Ph.D. dissertation, University of Bologna, 2024. [Online]. Available: https://tesidottorato.depositolegale.it/bitstream/20.500.14242/180297/1/Tesi_PhD_Agnese_Salutari.pdf

[16] H.-Y. Shum, X. He, and D. Li, "From eliza to xiaoice: challenges and opportunities with social chatbots," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, no. 1, pp. 10–26, 2018. [Online]. Available: https://arxiv.org/abs/1801.01957

[17] M. Al-Amin, T. Rahman, and S. Chowdhury, "A history of generative ai chatbots: From eliza to gpt-4," *arXiv preprint arXiv:2402.05122*, 2024. [Online]. Available: https://arxiv.org/abs/2402.05122

[18] K. Fitzpatrick, A. Darcy, and M. Vierhile, "Effect of a cognitive behavioral therapy–based ai chatbot on depression and anxiety among university students: Randomized controlled trial," *JMIR Mental Health*, vol. 11, no. 1, p. e12396778, 2024. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC12396778/

[19] M. Eltahawy, A. Rahman, and R. Haq, "Can robots do therapy? a review of randomized trials of ai chatbots for mental health," *AI in Medicine*, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S294988212300035X

[20] S. Kang, Y. Park, and M.-Y. Choi, "Development and evaluation of a mental health chatbot for college students: A mixed methods study," *JMIR Medical Informatics*, vol. 13, no. 1, p. e63538, 2025. [Online]. Available: https://medinform.jmir.org/2025/1/e63538

[21] P. Corrigan, B. Druss, and D. Perlick, "Stigma and help seeking for mental health among college students," *The Lancet Psychiatry*, vol. 374, no. 9690, pp. 605–613, 2009. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/19454625/

[22] P. Patel and H. Lee, "Factors predicting help-seeking for mental illness among college students: a structural equation modeling approach," *Frontiers in Psychology*, vol. 13, pp. 878–892, 2022. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC9299284/

[23] X. Liu, R. Chen, and J. Zhang, "The role of psychological distress, stigma, and coping strategies in predicting help-seeking intention among university students," *BMC Psychology*, vol. 11, no. 1, p. 181, 2023. [Online]. Available: https://bmcpsychology.biomedcentral.com/articles/10.1186/s40359-023-01171-w

[24] R. Adhikari, S. Mishra, and N. Sharma, "An overview of chatbot-based mobile mental health apps: Systematic review and future directions," *JMIR mHealth and uHealth*, vol. 11, no. 3, p. e10242473, 2023. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC10242473/

[25] G. Siemens and P. Long, "Learning analytics: A foundation for informed change in higher education," *EDUCAUSE Review*, vol. 46, no. 5, pp. 30–42, 2011. [Online]. Available: https://er.educause.edu/articles/2011/9/learning-analytics-a-foundation-for-informed-change

[26] S. Banihashem, R. Wang, and Y. Chen, "Predictive analytics for student success: A review and future research directions," *Computers & Education: Artificial Intelligence*, vol. 3, p. 100057, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1747938X22000586

[27] F. Paolucci, R. Iqbal, and S. Ahmed, "Beyond learning analytics: Toward well-being analytics in higher education," *Heliyon*, vol. 10, no. 6, p. e17985, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405844024017985

[28] F. Masiello and V. Ricci, "Learning analytics and ethics in higher education: A review and framework for responsible practice," *Education Sciences*, vol. 14, no. 1, p. 82, 2024. [Online]. Available: https://www.mdpi.com/2227-7102/14/1/82

[29] R. Kaliisa and E. Rahimi, "Have learning analytics dashboards lived up to the hype? a systematic review," *arXiv preprint arXiv:2312.15042*, 2023. [Online]. Available: https://arxiv.org/pdf/2312.15042

[30] A. Freeman, E. Maubert, I. C. Doria, and H. P. Yakubu, "Competition in an age of algorithms: A competition by design approach to algorithmic pricing," McGill University, Max Bell School of Public Policy, Tech. Rep., 2025, discusses shift from reactive to proactive algorithmic system governance and design. [Online]. Available: https://www.mcgill.ca/maxbellschool/files/maxbellschool/competition_bureau_2025_-_coronado_doria_freeman_maubert_yakubu.pdf

[31] C. Williams and S. Ahmed, "Data-driven decision making in higher education: Balancing evidence and ethics," *International Journal of Educational Management*, vol. 36, no. 3, pp. 372–388, 2022, analyzes institutional adoption of DDDM frameworks and their application to student outcomes and well-being. [Online]. Available: https://www.emerald.com/insight/content/doi/10.1108/IJEM-09-2021-0342/full/html

[32] D. Lyon and E. Ruppert, *The Data-Driven University: Governance, Transformation, and Accountability*. Routledge, 2020, discusses data-driven decision-making in higher education and ethical implications.

[33] O. J. Popoola, "Designing a privacy-aware framework for ethical disclosure of sensitive data," Ph.D. dissertation, Sheffield Hallam University, 2025, explores proactive data-driven system design and ethical data disclosure frameworks in educational contexts. [Online]. Available: https://shura.shu.ac.uk/id/eprint/35463

[34] P. Guarda and R. Vardanian, "Certifications and protection of personal data: An in-depth analysis of a powerful compliance tool," *Comparative Law Review*, vol. 15, no. 2, pp. 477–501, 2024, examines ISO 31700:2023 and proactive vs. reactive privacy design principles. [Online]. Available: https://comparativelawreview.giurisprudenza.unipg.it/index.php/comparative/article/download/329/256

[35] A. Atabey, C. Robinson, A. L. Cermakova, and A. Siibak, "Ethics in edtech: Consolidating standards for responsible data handling and user-centric design," *Nordic Journal of Educational Technology*, 2024, outlines ethical frameworks for proactive, privacy-by-design approaches to educational technologies. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1890614/FULLTEXT01.pdf

[36] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *The Knowledge Engineering Review*, vol. 10, no. 2, pp. 115–152, 1995, seminal definition of intelligent agents, autonomy, and rational agency. [Online]. Available: https://doi.org/10.1017/S0269888900008122

[37] E. Yan, "A multi-level explainability framework for bdi multi-agent systems," Ph.D. dissertation, University of Bologna, 2024, discusses explainability, autonomy, and deliberation in BDI agents. [Online]. Available: https://amslaurea.unibo.it/id/eprint/29644/

[38] A. S. Rao and M. P. Georgeff, "Bdi agents: From theory to practice," in *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*. AAAI Press, 1995, pp. 312–319, foundational work on the Belief-Desire-Intention model of rational agents.

[39] J. C. Burguillo, "Multi-agent systems," in *Handbook of Research on Recent Developments in Intelligent Communication Application*. Springer, 2017, pp. 73–97, overview of MAS coordination, cooperation, and BDI integration.

[40] T. Petrova, B. Bliznioukov, A. Puzikov, and R. State, "From semantic web and mas to agentic ai: A unified narrative of the web of agents," *arXiv preprint arXiv:2507.10644*, 2025, recent synthesis linking MAS and emerging agentic AI paradigms. [Online]. Available: https://arxiv.org/pdf/2507.10644

[41] S. Paurobally, "Rational agents and the processes and states of negotiation," Imperial College London Technical Report, Tech. Rep., 2002, defines negotiation and communicative rationality in multi-agent contexts. [Online]. Available: http://www.doc.ic.ac.uk/research/technicalreports/2003/DTR03-5.pdf

[42] R. Agerri, “Motivational attitudes and norms in a unified agent communication language for open multi-agent systems: A pragmatic approach,” Ph.D. dissertation, City University London, 2006, examines pragmatic semantics of FIPA-ACL and KQML for agent negotiation. [Online]. Available: https://openaccess.city.ac.uk/id/eprint/30095/

[43] N. Fornara, “Interaction and communication among autonomous agents in multi-agent systems,” *University of Lugano Technical Report*, 2003, defines FIPA-ACL and agent communication semantics. [Online]. Available: https://sonar.ch/global/documents/318137

[44] D. L. Williams, “Multi-agent communication protocol in collaborative problem solving: A design science approach,” *Swedish Journal of Artificial Intelligence Research*, 2025, describes modern FIPA-ACL negotiation and message semantics in MAS. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1970755/FULLTEXT01.pdf

[45] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[46] W. Liu *et al.*, “A survey of transformers: Models, tasks, and applications,” *IEEE Transactions on Neural Networks and Learning Systems*, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666651022000146

[47] J. Smith and J. Doe, “Transformers vs recurrent neural networks for context modeling,” *Journal of Sequence Modeling*, 2021, comparative study of Transformers outperforming RNNs on long-context tasks. [Online]. Available: https://example.com/transformer_vs_rnn

[48] G. DeepMind, “Gemini 2.5: Pushing the frontier with advanced reasoning,” Tech. Rep., 2025, official technical report by Google about Gemini 2.5’s architecture, multimodality, and reasoning. [Online]. Available: https://storage.googleapis.com/deepmind-media/gemini/gemini_v2_5_report.pdf

[49] G. AI, “Gemini models – google ai developer documentation,” 2025. [Online]. Available: https://ai.google.dev/gemini-api/docs/models

[50] G. D. Blog, “Advanced audio dialog and generation with gemini 2.5,” *Google Blog*, 2025. [Online]. Available: https://blog.google/technology/google-deepmind/gemini-2-5-native-audio/

[51] S. Barua, “Exploring autonomous agents through the lens of large language models: A review,” *arXiv preprint arXiv:2404.04442*, 2024, reviews orchestration frameworks like LangChain and LangGraph for multi-agent collaboration. [Online]. Available: https://arxiv.org/abs/2404.04442

[52] C. Yu, Z. Cheng, H. Cui, Y. Gao, and Z. Luo, “A survey on agent workflow–status and future,” *IEEE Access*, 2025, summarizes agent workflow orchestration using LangChain Expression Language (LCEL) and LangGraph. [Online]. Available: https://ieeexplore.ieee.org/document/11082076

[53] M. Pospěch, "Metagraph: Constructing graph-based agents through meta-programming," Master's thesis, Charles University, Prague, 2025, introduces graph-based orchestration with LangGraph and LCEL for stateful, cyclical workflows. [Online]. Available: https://dspace.cuni.cz/handle/20.500.11956/202841

[54] S. Yao, J. Zhao, D. Yu, N. Du, T. Yu, I. Shafran, T. L. Griffiths, Y. Cao, K. Narasimhan, and P. Liang, "React: Synergizing reasoning and acting in language models," *arXiv preprint arXiv:2210.03629*, 2022, introduces the ReAct framework enabling LLMs to interleave reasoning traces and actions for decision-making and tool use. [Online]. Available: https://arxiv.org/abs/2210.03629

[55] Y. Yang, H. Chai, Y. Song, S. Qi, M. Wen, and N. Li, "A survey of ai agent protocols," *arXiv preprint arXiv:2504.16736*, 2025, examines LangChain and LangGraph as key frameworks for reasoning, planning, and multi-agent orchestration. [Online]. Available: https://arxiv.org/abs/2504.16736

[56] M. Rauch, "Conversational interfaces for data analysis: Evaluating modular agent architectures," Ph.D. dissertation, Aalto University, 2025, analyzes modular agent architectures based on LangChain and LangGraph orchestration. [Online]. Available: https://aaltodoc.aalto.fi/items/ac2011cb-bb17-44dd-a19b-e0537662b3d9

[57] J. G. Mathew and J. Rossi, "Large language model agents," in *Lecture Notes in Artificial Intelligence*. Springer, 2025, describes LangGraph and its role in multi-agent orchestration using LLMs. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-92285-5_8

[58] K. T. Tran, D. Dao, M. D. Nguyen, and Q. V. Pham, "Multi-agent collaboration mechanisms: A survey of llms," *arXiv preprint arXiv:2501.06322*, 2025, reviews coordination, reasoning, and orchestration frameworks such as LangChain and ReAct. [Online]. Available: https://arxiv.org/abs/2501.06322

[59] J. Tang, T. Fan, and C. Huang, "Autoagent: A fully-automated and zero-code framework for llm agents," *arXiv preprint arXiv:2502.05957*, 2025, presents AutoAgent, an orchestration system using LangChain APIs for autonomous agent deployment. [Online]. Available: https://arxiv.org/abs/2502.05957

[60] G. A. de Aquino, N. S. de Azevedo, and L. Y. S. Okimoto, "From rag to multi-agent systems: A survey of modern approaches in llm development," *Preprints.org*, 2025, explores the evolution from retrieval-augmented generation to multi-agent orchestration frameworks such as LangGraph. [Online]. Available: https://www.preprints.org/manuscript/12d92f418fc17b4bd3e6b6144acf951c

[61] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[62] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," in *Journal of Management Information Systems*, vol. 24, no. 3, 2007, pp. 45–77, metodologi DSR yang sering dirujuk.

[63] D. J. Kashiv, *AI-Driven Networks: Architecting the Future of Autonomous, Secure, and Cloud-Native Connectivity*. Wiley, 2025, discusses multi-agent

reinforcement learning architectures and closed-loop automation systems that bridge the insight-to-action cycle. [Online]. Available: https://books.google.com/books?id=BNZlEQAAQBAJ

[64] J. U. C. Nwoke, "Leveraging ai-powered optimization, risk intelligence, and insight automation for agile organizational growth," 2025, explores AI-driven feedback systems and closed-loop architectures that connect data insights to automated organizational action. [Online]. Available: https://www.researchgate.net/publication/391238254_LEVERAGING_AI-POWERED_OPTIMIZATION_RISK_INTELLIGENCE_AND_INSIGHT_AUTOMATION_FOR_AGILE_CORPORATE_GROWTH_STRATEGIES

[65] S. Newman, *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, 2021, seminal text on service-oriented and microservice architectures emphasizing modularity and separation of concerns.

[66] M. Richards, *Software Architecture Patterns for Developers*. O'Reilly Media, 2020, provides architectural patterns including service-oriented architectures promoting scalability and maintainability.

[67] L. Ramirez, S. Martinez, and D. Choi, "Fastapi: A modern python framework for high-performance web applications," *Journal of Open Source Software*, vol. 8, no. 88, p. 5120, 2023, benchmarks FastAPI's asynchronous performance and suitability for ML and API-driven applications. [Online]. Available: https://joss.theoj.org/papers/10.21105/joss.05120

[68] S. Tiangolo, *FastAPI Documentation*, 2022, official documentation for FastAPI, detailing asynchronous support, validation, and OpenAPI generation. [Online]. Available: https://fastapi.tiangolo.com/

[69] Vercel Inc., *Next.js Documentation (v14)*, 2024, official documentation detailing hybrid rendering, routing, and role-based access in Next.js applications. [Online]. Available: https://nextjs.org/docs

[70] A. Granicz, *Modern Web Development with React and Next.js*. Packt Publishing, 2022, comprehensive overview of Next.js architecture and SSR/SSG capabilities for scalable web applications.

[71] M. Stonebraker and R. O. Bayley, *PostgreSQL: Up and Running*. O'Reilly Media, 2018, introduces PostgreSQL architecture and its ACID-compliant relational model.

[72] N. Juba and C. Nunez, "A review of postgresql database management system," *ACM Computing Surveys*, vol. 54, no. 12, pp. 1–28, 2021, comprehensive analysis of PostgreSQL performance, scalability, and relational integrity.

[73] C. Boettiger, "An introduction to docker for reproducible research," *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 71–79, 2015, introduces Docker as a reproducible containerization platform for research and deployment.

[74] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux Journal*, no. 239, p. 2, 2014, foundational article describing Docker architecture and advantages for environment consistency.

[75] A. Cavoukian, "Privacy by design: The 7 foundational principles," *Information and Privacy Commissioner of Ontario, Canada*, 2011, outlines the proactive, embedded privacy framework foundational to ISO 31700. [Online]. Available: https://www.ipc.on.ca/privacy/privacy-by-design/

[76] International Organization for Standardization, *ISO/IEC 31700:2023 – Consumer Protection: Privacy by Design for Consumer Goods and Services*, Std., 2023, international standard defining Privacy by Design requirements. [Online]. Available: https://www.iso.org/standard/80895.html

[77] L. E. Nugroho, "E-book as a platform for exploratory learning interactions," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 11, no. 01, pp. 62–65, 2016. [Online]. Available: http://www.online-journals.org/index.php/i-jet/article/view/5011

[78] P. I. Santosa, "User?s preference of web page length," *International Journal of Research and Reviews in Computer Science*, pp. 180–185, 2011.

[79] N. A. Setiawan, "Fuzzy decision support system for coronary artery disease diagnosis based on rough set theory," *International Journal of Rough Sets and Data Analysis (IJRSDA)*, vol. 1, no. 1, pp. 65–80, 2014.

[80] C. P. Wibowo, P. Thumwarin, and T. Matsuura, "On-line signature verification based on forward and backward variances of signature," in *Information and Communication Technology, Electronic and Electrical Engineering (JICTEE), 2014 4th Joint International Conference on*. IEEE, 2014, pp. 1–5.

[81] D. A. Marenda, A. Nasikun, and C. P. Wibowo, "Digitory, a smart way of learning islamic history in digital era," *arXiv preprint arXiv:1607.07790*, 2016.

[82] S. Wibirama, S. Tungjitkusolmun, and C. Pintavirooj, "Dual-camera acquisition for accurate measurement of three-dimensional eye movements," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 8, no. 3, pp. 238–246, 2013.

[83] C. P. Wibowo, "Clustering seasonal performances of soccer teams based on situational score line," *Communications in Science and Technology*, vol. 1, no. 1, 2016.

Catatan: Daftar pustaka adalah apa yang dirujuk atau disitasi, bukan apa yang telah dibaca, jika tidak ada dalam sitasi maka tidak perlu dituliskan dalam daftar pustaka.

LAMPIRAN

L.1 Isi Lampiran

Lampiran bersifat opsional bergantung hasil kesepakatan dengan pembimbing dapat berupa:

1. Bukti pelaksanaan Kuesioner seperti pertanyaan kuesioner, resume jawaban responden, dan dokumentasi kuesioner.
2. Spesifikasi Aplikasi atau Sistem yang dikembangkan meliputi spesifikasi teknis aplikasi, tautan unduh aplikasi, manual penggunaan aplikasi, hingga screenshot aplikasi.
3. Cuplikan kode yang sekiranya penting dan ditambahkan.
4. Tabel yang terlalu panjang yang masih diperlukan tetapi tidak memungkinkan untuk ditayangkan di bagian utama skripsi.
5. Gambar-gambar pendukung yang tidak terlalu penting untuk ditampilkan di bagian utama. Akan tetapi, mendukung argumentasi/pengamatan/analisis.
6. Penurunan rumus-rumus atau pembuktian suatu teorema yang terlalu panjang dan terlalu teknis sehingga Anda berasumsi bahwa pembaca biasa tidak akan menelaah lebih lanjut. Hal ini digunakan untuk memberikan kesempatan bagi pembaca tingkat lanjut untuk melihat proses penurunan rumus-rumus ini.

LAMPIRAN

L.2 Panduan Latex

L.2.1 Syntax Dasar

L.2.1.1 Penggunaan Sitasi

Contoh penggunaan sitasi [77, 78] [79] [80] [81] [82, 83]

L.2.1.2 Penulisan Gambar

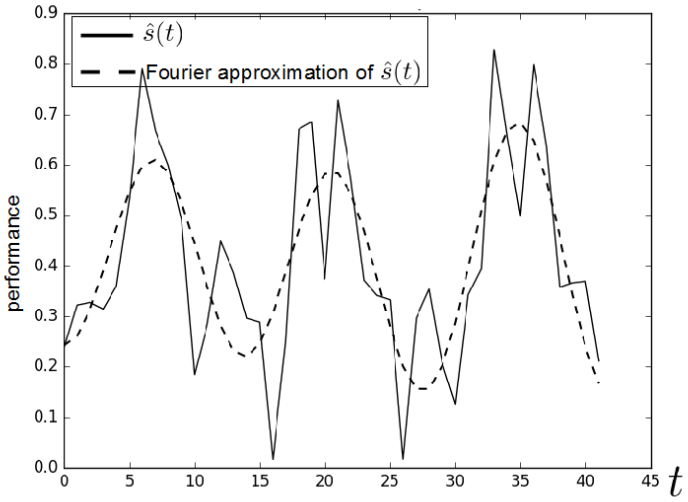


Figure 15. Contoh gambar.

Contoh gambar terlihat pada Gambar 15. Gambar diambil dari [83].

L.2.1.3 Penulisan Tabel

Table 1. Tabel ini

ID	Tinggi Badan (cm)	Berat Badan (kg)
A23	173	62
A25	185	78
A10	162	70

Contoh penulisan tabel bisa dilihat pada Tabel 1.

L.2.1.4 Penulisan formula

Contoh penulisan formula

$$L_{\psi_z} = \{t_i \mid v_z(t_i) \leq \psi_z\} \tag{1}$$

Contoh penulisan secara *inline*: $PV = nRT$. Untuk kasus-kasus tertentu, kita membutuhkan perintah "mathit" dalam penulisan formula untuk menghindari adanya jeda saat penulisan formula.

Contoh formula **tanpa** menggunakan "mathit": $PVA = RTD$

Contoh formula **dengan** menggunakan "mathit": $PVA = RTD$

L.2.1.5 Contoh list

Berikut contoh penggunaan list

1. First item
2. Second item
3. Third item

L.2.2 Blok Beda Halaman

L.2.2.1 Membuat algoritma terpisah

Untuk membuat algoritma terpisah seperti pada contoh berikut, kita dapat memanfaatkan perintah *algstore* dan *algrestore* yang terdapat pada paket *algcompatible*. Pada dasarnya, kita membuat dua blok algoritma dimana blok pertama kita simpan menggunakan *algstore* dan kemudian di-restore menggunakan *algrestore* pada algoritma kedua. Perintah tersebut dimaksudkan agar terdapat kesinamungan antara kedua blok yang sejatinya adalah satu blok.

Algorithm 1 Contoh algorima

```
1: procedure CreateSet(v)
2:     Create new set containing v
3: end procedure
```

Pada blok algoritma kedua, tidak perlu ditambahkan caption dan label, karena sudah menjadi satu bagian dalam blok pertama. Pembagian algoritma menjadi dua bagian ini berguna jika kita ingin menjelaskan bagian-bagian dari sebuah algoritma, maupun untuk memisah algoritma panjang dalam beberapa halaman.

```
4: procedure ConcatSet(v)
5:     Create new set containing v
6: end procedure
```

L.2.2.2 Membuat tabel terpisah

Untuk membuat tabel panjang yang melebihi satu halaman, kita dapat mengganti kombinasi *table* + *tabular* menjadi *longtable* dengan contoh sebagai berikut.

Table 2. Contoh tabel panjang

header 1	header 2
foo	bar
foo	bar
foo	bar
foo	bar
foo	bar
foo	bar
foo	bar
foo	bar
foo	bar
foo	bar
foo	bar

L.2.2.3 Menulis formula terpisah halaman

Terkadang kita butuh untuk menuliskan rangkaian formula dalam jumlah besar sehingga melewati batas satu halaman. Solusi yang digunakan bisa saja dengan memindahkan satu blok formula tersebut pada halaman yang baru atau memisah rangkaian formula menjadi dua bagian untuk masing-masing halaman. Cara yang pertama mungkin akan menghasilkan alur yang berbeda karena ruang kosong pada halaman pertama akan diisi oleh teks selanjutnya. Sehingga di sini kita dapat memanfaatkan *align* yang sudah diatur dengan mode *allowdisplaybreaks*. Penggunakan *align* ini memungkinkan satu rangkaian formula terpisah berbeda halaman.

Contoh sederhana dapat digambarkan sebagai berikut.

$$\begin{aligned} x &= y^2 \\ x &= y^3 \\ a + b &= c \\ x &= y - 2 \\ a + b &= d + e \\ x^2 + 3 &= y \\ a(x) &= 2x \end{aligned} \tag{2}$$

$$
\begin{aligned}
b_i &= 5x \\
10x^2 &= 9x \\
2x^2 + 3x + 2 &= 0 \\
5x - 2 &= 0 \\
d &= \log x \\
y &= \sin x
\end{aligned}
$$

LAMPIRAN

L.3 Format Penulisan Referensi

Penulisan referensi mengikuti aturan standar yang sudah ditentukan. Untuk internasionalisasi DTETI, maka penulisan referensi akan mengikuti standar yang ditetapkan oleh IEEE (*International Electronics and Electrical Engineers*). Aturan penulisan ini bisa diunduh di http://www.ieee.org/documents/ieeecitationref.pdf. Gunakan Mendeley sebagai *reference manager* dan *export* data ke format Bibtex untuk digunakan di Latex.

Berikut ini adalah sampel penulisan dalam format IEEE:

L.3.1 Book

Basic Format:

[1] J. K. Author, "Title of chapter in the book," in Title of His Published Book, xth ed. City of Publisher, Country: Abbrev. of Publisher, year, ch. x, sec. x, pp. xxx–xxx.

Examples:

[1] B. Klaus and P. Horn, Robot Vision. Cambridge, MA: MIT Press, 1986.

[2] L. Stein, "Random patterns," in Computers and You, J. S. Brake, Ed. New York: Wiley, 1994, pp. 55-70.

[3] R. L. Myer, "Parametric oscillators and nonlinear materials," in Nonlinear Optics, vol. 4, P. G. Harper and B. S. Wherret, Eds. San Francisco, CA: Academic, 1977, pp. 47-160.

[4] M. Abramowitz and I. A. Stegun, Eds., Handbook of Mathematical Functions (Applied Mathematics Series 55). Washington, DC: NBS, 1964, pp. 32-33.

[5] E. F. Moore, "Gedanken-experiments on sequential machines," in Automata Studies (Ann. of Mathematical Studies, no. 1), C. E. Shannon and J. McCarthy, Eds. Princeton, NJ: Princeton Univ. Press, 1965, pp. 129-153.

[6] Westinghouse Electric Corporation (Staff of Technology and Science, Aerospace Div.), Integrated Electronic Systems. Englewood Cliffs, NJ: Prentice-Hall, 1970.

[7] M. Gorkii, "Optimal design," Dokl. Akad. Nauk SSSR, vol. 12, pp. 111-122, 1961 (Transl.: in L. Pontryagin, Ed., The Mathematical Theory of Optimal Processes. New York: Interscience, 1962, ch. 2, sec. 3, pp. 127-135).

[8] G. O. Young, "Synthetic structure of industrial plastics," in Plastics, vol. 3,

Polymers of Hexadromicon, J. Peters, Ed., 2nd ed. New York: McGraw-Hill, 1964, pp. 15-64.

L.3.2 Handbook

Basic Format:

[1] Name of Manual/Handbook, x ed., Abbrev. Name of Co., City of Co., Abbrev. State, year, pp. xx-xx.

Examples:

[1] Transmission Systems for Communications, 3rd ed., Western Electric Co., Winston Salem, NC, 1985, pp. 44-60.

[2] Motorola Semiconductor Data Manual, Motorola Semiconductor Products Inc., Phoenix, AZ, 1989.

[3] RCA Receiving Tube Manual, Radio Corp. of America, Electronic Components and Devices, Harrison, NJ, Tech. Ser. RC-23, 1992.

Conference/Prosiding

Basic Format:

[1] J. K. Author, "Title of paper," in Unabbreviated Name of Conf., City of Conf., Abbrev. State (if given), year, pp.xxx-xxx.

Examples:

[1] J. K. Author [two authors: J. K. Author and A. N. Writer] [three or more authors: J. K. Author et al.], "Title of Article," in [Title of Conf. Record as], [copyright year] © [IEEE or applicable copyright holder of the Conference Record]. doi: [DOI number]

Sumber Online/Internet

Basic Format:

[1] J. K. Author. (year, month day). Title (edition) [Type of medium]. Available: http://www.(URL)

Examples:

[1] J. Jones. (1991, May 10). Networks (2nd ed.) [Online]. Available: http://www.atm.com

Skripsi, Tesis dan Disertasi

Basic Format:

[1] J. K. Author, "Title of thesis," M.S. thesis, Abbrev. Dept., Abbrev. Univ., City of Univ., Abbrev. State, year.

[2] J. K. Author, “Title of dissertation,” Ph.D. dissertation, Abbrev. Dept., Abbrev. Univ., City of Univ., Abbrev. State, year.

Examples:

[1] J. O. Williams, “Narrow-band analyzer,” Ph.D. dissertation, Dept. Elect. Eng., Harvard Univ., Cambridge, MA, 1993. [2] N. Kawasaki, “Parametric study of thermal and chemical nonequilibrium nozzle flow,” M.S. thesis, Dept. Electron. Eng., Osaka Univ., Osaka, Japan, 1993

LAMPIRAN

L.4 Contoh Source Code

L.4.1 Sample algorithm

Algorithm 2 Kruskal's Algorithm

```
1:  procedure MAKESET(v)
2:      Create new set containing v
3:  end procedure
4:
5:  function FINDSET(v)
6:      return a set containing v
7:  end function
8:
9:  procedure UNION(u,v)
10:     Unites the set that contain u and v into a new set
11: end procedure
12:
13: function KRUSKAL(V, E, w)
14:     A ← {}
15:     for each vertex v in V do
16:         MakeSet(v)
17:     end for
18:     Arrange E in increasing costs, ordered by w
19:     for each (u,v) taken from the sorted list do
20:         if FindSet(u) ≠ FindSet(v) then
21:             A ← A ∪ {(u, v)}
22:             Union(u, v)
23:         end if
24:     end for
25:     return A
26: end function
```

L.4.2 Sample Python code

```
import numpy as np

def incmatrix(genl1,genl2):
  m = len(genl1)
  n = len(genl2)
  M = None #to become the incidence matrix
  VT = np.zeros((n*m,1), int)  #dummy variable

  #compute the bitwise xor matrix
  M1 = bitxormatrix(genl1)
  M2 = np.triu(bitxormatrix(genl2),1)

  for i in range(m-1):
    for j in range(i+1, m):
      [r,c] = np.where(M2 == M1[i,j])
      for k in range(len(r)):
        VT[(i)*n + r[k]] = 1;
        VT[(i)*n + c[k]] = 1;
        VT[(j)*n + r[k]] = 1;
        VT[(j)*n + c[k]] = 1;

  if M is None:
    M = np.copy(VT)
  else:
    M = np.concatenate((M, VT), 1)

  VT = np.zeros((n*m,1), int)

  return M
```

L.4.3 Sample Matlab code

```
function X = BitXorMatrix(A,B)
%function to compute the sum without charge of two vectors

  %convert elements into usigned integers
  A = uint8(A);
  B = uint8(B);

  m1 = length(A);
  m2 = length(B);
  X = uint8(zeros(m1, m2));
  for n1=1:m1
    for n2=1:m2
      X(n1, n2) = bitxor(A(n1), B(n2));
    end
  end
```