# 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 |
| $\mathbf{w}$ | = | vektor bobot |
| $\mathbf{x}$ | = | 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

Intisari ditulis menggunakan bahasa Indonesia dengan jarak antar baris 1 spasi dan maksimal 1 halaman. Intisari sekurang-kurangnya berisi tentang latar belakang dan tujuan penelitian, metodologi yang digunakan, hasil penelitian, kesimpulan dan implikasi, dan Kata kunci yang berhubungan dengan penelitian.

Kata Kunci ditulis maksimal 5 kata yang paling berhubungan dengan isi skripsi. Silakan mengacu pada ACM / IEEE *Computing classification* jika Anda adalah mahasiswa Sarjana TI http://www.acm.org/about/class/ atau mengacu kepada IEEE keywords http://www.ieee.org/documents/taxonomy_v101.pdf jika Anda berasal dari Prodi Sarjana TE.

Kata kunci : Kata kunci 1, Kata kunci 2, Kata kunci 3, Kata kunci 4, Kata kunci 5

**Contoh Abstrak Teknik Elektro:**

"Penelitian ini bertujuan untuk mengembangkan sistem pengendalian suhu ruangan dengan menggunakan microcontroller. Metodologi yang digunakan adalah desain sirkuit, implementasi sistem pengendalian, dan pengujian performa. Hasil penelitian menunjukkan bahwa sistem pengendalian suhu ruangan yang dikembangkan mampu mengendalikan suhu ruangan dengan akurasi sebesar ±0,5°C. Kesimpulan dari penelitian ini adalah sistem pengendalian suhu ruangan yang dikembangkan efektif dan efisien.

Kata kunci: microcontroller, sistem pengendalian suhu, akurasi."

**Contoh Abstrak Teknik Biomedis:**

"Penelitian ini bertujuan untuk mengevaluasi keefektifan prototipe alat pemantau denyut jantung berbasis elektrokardiogram (ECG) untuk pasien jantung. Metodologi yang digunakan meliputi desain dan pembuatan prototipe, pengujian dengan pasien, dan analisis data. Hasil penelitian menunjukkan bahwa prototipe alat pemantau denyut jantung berbasis ECG memiliki akurasi yang baik dan mampu memantau denyut jantung pasien secara efektif. Kesimpulan dari penelitian ini adalah prototipe alat pemantau denyut jantung berbasis ECG merupakan solusi yang efektif dan efisien untuk memantau pasien jantung.

Kata kunci: elektrokardiogram, alat pemantau denyut jantung, akurasi."

**Contoh Abstrak Teknologi Informasi:**

"Penelitian ini bertujuan untuk mengevaluasi keamanan dan privasi pengguna aplikasi media sosial terpopuler. Metodologi yang digunakan meliputi analisis kebijakan privasi dan pengaturan keamanan, pengujian penetrasi, dan survei pengguna. Hasil penelitian menunjukkan bahwa beberapa aplikasi media sosial memiliki kebijakan privasi yang kurang jelas dan rendahnya tingkat keamanan. Kesimpulan dari penelitian ini adalah pentingnya meningkatkan kebijakan privasi dan tingkat keamanan pada aplikasi media sosial untuk melindungi privasi dan data pengguna.

Kata kunci: media sosial, keamanan, privasi, pengguna."

# ABSTRACT

The provision of mental health support in Higher Education Institutions (HEIs) is often hampered by a reactive model that struggles with scalability and timely intervention. This research addresses these challenges by proposing, designing, and implementing a novel **agentic AI framework** to enable a proactive, data-driven approach to student well-being. The framework's core contribution is a multi-agent system, the **Safety Agent Suite**, which is orchestrated by LangGraph and powered by Google's Gemini Large Language Model. This suite provides a robust architecture for real-time crisis detection, personalized coaching, and privacy-preserving analytics, all under human oversight.

To validate the framework, a functional prototype was developed. The system's capability to deliver evidence-based support was demonstrated by implementing a **Support Coach Agent** that provides interventions based on Cognitive Behavioral Therapy (CBT) principles. Furthermore, to address the critical challenge of user engagement, a novel gamification system was integrated, utilizing **blockchain technology** to issue NFT achievement badges. The prototype's feasibility was confirmed through testing scenarios that validated the agentic workflows. The results indicate that this agentic framework presents a viable and focused solution for transforming university mental health services, with the integrated features demonstrating its potential for delivering effective and engaging support.

**Keywords** : Agentic AI, Mental Health, Proactive Support, LangGraph, Student Well-being

# 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 [4].

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 [4]. Consequently, a critical gap persists between the need for mental health services and their actual provision, leaving many students without timely support [5].

To bridge this gap, a paradigm shift from a reactive to a **proactive** support model is imperative [5]. The engine for this evolution is **Digital Transformation**, a process that leverages technology to fundamentally reshape organizational processes and enhance value delivery within HEIs [6]. 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 [7, 8].

This research 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 [9], representing a new frontier in educational technology [10]. We propose that a framework built upon a system of collaborative intelligent agents, a Multi-Agent System (MAS), a concept already explored for smart campus management [10], 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 thesis details the design, development, and evaluation of such a framework, prototyped within the UGM-AICare project.

## 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. A significant technical challenge lies in the **implementation of a cloud-native architecture** to realize this framework. This involves orchestrating multiple AI agents using **LangGraph**, powered by a robust Large Language Model (Google Gemini), and integrating them within a secure FastAPI backend with an internal task scheduler.
3. Finally, there is a need to **evaluate the potential impact** of such a framework on institutional operational efficiency and its ability to support data-driven, safety-first decision-making. This will be validated through a proof-of-concept prototype and scenario-based testing focused on the agentic workflows.

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 **Isingts Agent (IA)**.

## 1.3 Objectives

The primary objectives of this thesis are:

1. To design the conceptual and technical framework for the agentic AI system.
2. To implement a functional proof-of-concept prototype.
3. To evaluate the prototype's capabilities against predefined functional scenarios.

## 1.4 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). The integration of content based on Cognitive Behavioral Therapy (CBT) and a blockchain-based gamification system

serve as proof-of-concept features to validate the framework's capabilities, not as primary research areas themselves.

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 will not provide an exhaustive analysis of differential privacy algorithms or blockchain scalability. It will, however, demonstrate their integration into the system to fulfill the architectural requirements of **privacy-first analytics** and **novel user engagement**.

## 1.5 Contributions

1. Academic: A novel framework for applying agentic AI in an institutional (higher education) context.
2. Practical: A blueprint for UGM to develop a more proactive, data-driven, and efficient mental health support system.

## 1.6 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 presents a comprehensive review of relevant prior research in the fields of conversational AI, the application of gamification in educational and well-being contexts, related blockchain applications, and studies on user engagement and student welfare. Furthermore, this chapter 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 [11, 12]. 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 [11].

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 [12].

#### 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 [13, 14]. 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 [15].

#### 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 [16, 17]. 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 [18].

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 [19]. This leaves the most vulnerable students—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" [20]. 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 [21]. 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 [22]. 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 [23]. 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 [22, 23].

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 [22].

#### 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** [24]. 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) [25]. 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 [26]. 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 [27]. 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 [28, 29]. This thesis is an explicit attempt to architect a system that facilitates this strategic shift from a reactive to a proactive support paradigm.

#### 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 [28, 29]. 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 [30].

#### 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 [31,32]. 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 [30, 31]. 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 [33]. 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 [34].

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) [35]. A prominent and highly relevant architecture for designing such goal- oriented agents is the **Belief-Desire-Intention (BDI)** model [35, 36]. 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 [34, 36].

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** *(Comm Plans)* |
|---|---|---|---|
| **Safety Triage Agent (STA)** | • The user's live conversation history. • A classification model for conversation severity. • A directory of emergency resources. | • To assess a user's immediate risk level accurately. • To provide the most appropriate support for the user's current state. | • Upon detecting high-severity terance, form intention to imm ately escalate. • To retrieve and play emergency tact information w out delay. |
| **Support Coach Agent (SCA)** | • User's stated goals and conversation history. • A library of evidence-based interventions (e.g., CBT exercises). | • To deliver personalized coaching. • To guide the user through therapeutic exercises. | • Based on user in form an intentio deliver a specific C exercise. • To provide em thetic and suppor responses. |
| **Service Desk Agent (SDA)** | • Status of clinical cases. • Availability of counselors. • User requests for appointments. | • To manage clinical case workflows. • To schedule appointments efficiently. | • Upon a user requ form an intentio find an available pointment slot. • To create and up case notes. |
| **Insights Agent (IA)** | • Access to the anonymized conversation log database. • The timestamp of the last generated report. • A model of known topics. | • To identify emerging trends in student concerns. • To quantify shifts in overall student sentiment. | • Upon its weekly ger, form an in tion to generate a summary report. • To formulate and cute a database qu for the past we data. |

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 [37, 38]. 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 [39, 40].
- **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 [41, 42].

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.

### 2.2.3 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. [43]. 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) [44, 45].

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 [44].

**Placeholder for Diagram: Simplified Transformer Architecture for Generative LLMs**

This diagram should illustrate the flow of information:
1. Input Text -> Tokenizer
2. Tokens -> Embedding Layer + Positional Encoding
3. Embedded Tokens -> A stack of 'N' Decoder Blocks
4. Inside a Decoder Block: Multi-Head Self-Attention -> Feed-Forward Network
5. Final Output -> Linear Layer -> Softmax -> Probability of Next Token

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

This research utilizes a cloud-based API model strategy, leveraging the Gemini 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 [46]. In this framework, a powerful model like Gemini 2.5 Pro is accessed via a secure API for all agentic tasks [47], 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 [46, 47].

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 [46].

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 [48]. 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 [46]. 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 [46]. 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 [49, 50]. 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 [50, 51].

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 [49, 52].

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 [53, 54].

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 [51, 55].

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 [50, 51]. 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.

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 [55, 56]. 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 [57,58].

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 [53, 56].

## 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 [59]. The primary goal of DSR is to generate prescriptive design knowledge through the building and evaluation of these artifacts.

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 [60]. 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. **Definition of Objectives for a Solution:** Based on the identified problem, this stage involves defining the objectives and desired capabilities of the artifact. These objectives, which center on creating a proactive, automated, and data-driven framework, are also detailed in **Chapter I**.
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 one or more instances of the problem. This will be accomplished through the implementation of a functional prototype, as will be detailed in **Chapter IV**.
5. **Evaluation:** This stage involves observing and measuring how well the artifact supports a solution to the problem. The prototype's capabilities will be evaluated against predefined functional scenarios in **Chapter IV**, with the findings and their implications discussed.

6. **Communication:** The final stage involves communicating the problem, the artifact, and its utility to a relevant audience. This entire thesis document serves as the primary communication artifact for this research.

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.

**Placeholder for Diagram: DSR Research Workflow**

This flowchart should illustrate the DSR stages as they apply to this thesis:
1. Box: "Problem Identification (Chapter 1)" ->
2. Box: "Literature Review & Theoretical Grounding (Chapter 2)" ->
3. Box: "Artifact Design & Architecture (Chapter 3)" ->
4. Box: "Prototype Implementation (Chapter 4)" ->
5. Box: "Scenario-Based Evaluation (Chapter 4)" ->
6. Box: "Conclusion & Future Work (Chapter V)"
*(Note: Follow guidelines from http://ugm.id/flowcharttutorial for styling.)*

Figure 3.2. The Design Science Research (DSR) process model as applied in this thesis.

## 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 [**?**].

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.

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 [**?**].

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.

**Placeholder for Diagram: System Context Diagram**

This diagram should be a high-level illustration of the system's architecture and stakeholders. It should visually represent:
1. An external "Student" entity interacting via a "UGM-AICare User App (/aika)" interface.
2. An external "University Staff/Counselor" entity interacting via an "Admin Dashboard" interface.
3. A central "Agentic AI Backend" system, which contains four internal components: the STA, SCA, SDA, and IA.
4. Arrows indicating the flow of information: Students send messages to the Backend; the Backend provides real-time responses. The Backend sends aggregated data and alerts to the Admin Dashboard; Staff use the dashboard to configure and oversee the Backend.

Figure 3.3. A high-level context diagram illustrating the primary entities, system interfaces, and the central role of the Agentic AI Framework.

## 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 "what" of the system by defining the specific role, operational logic, and capabilities 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.

### 3.3.1 The Safety Triage Agent (STA): The Real-Time Guardian

#### 3.3.1.1 Goal

The primary objective of the STA is to function as a real-time, automated safety monitor for every user interaction. Its goal is to assess the immediate risk level of a user's conversation to detect potential crises and trigger an appropriate escalation protocol without delay, ensuring that safety is the foremost priority of the system.

#### 3.3.1.2 Perception (Inputs)

The STA perceives the conversational environment by intercepting each user message before it is processed by other agents. Its primary input is the raw text of the user's current utterance. Let $M_t$ be the user's message at time $t$. The STA's perception is solely focused on this message:

- **Current User Message ($M_t$):** A string containing the user's latest input.

#### 3.3.1.3 Processing Logic

The core logic of the STA is a high-speed classification task. Upon receiving the message $M_t$, the agent invokes a specialized function, powered by the Gemini 2.5 Pro model, to classify the message into one of several predefined risk categories. The classification function, $f_{STA}$, can be represented as:

$$R_t = f_{STA}(M_t; \theta_{LLM})$$

where $\theta_{LLM}$ represents the parameters of the underlying Large Language Model, and the output, $R_t$, is an element of the set of possible risk levels, $R \in \{\text{Low, Moderate, Critical}\}$. The prompt for this classification is highly optimized for speed and accuracy, instructing the model to evaluate the text for indicators of self-harm, severe distress, or explicit requests for urgent help.

#### 3.3.1.4 Action (Outputs)

Based on the classification result $R_t$, the STA's action is to update the system's state, which in turn determines the next step in the LangGraph workflow.

- **State Update:** The agent's primary output is an update to the shared state graph, setting the `risk_level` variable to the value of $R_t$.
- **Trigger Escalation (if $R_t$ = Critical):** If a critical risk is detected, the agent's action triggers a conditional edge in the graph that invokes the `escalate_crisis` tool. This tool flags the conversation on the Admin Dashboard, logs the event, and instructs the Service Desk Agent (SDA) to create a high-priority case. It also immediately presents the user with pre-defined emergency resources.

### 3.3.2 The Support Coach Agent (SCA): The Empathetic Mentor

#### 3.3.2.1 Goal

The SCA is the primary user-facing conversational agent, designed to provide personalized, evidence-based mental health coaching. Its goal is to engage the student in a supportive, empathetic dialogue, guiding them through structured self-help modules based on established therapeutic principles like Cognitive Behavioral Therapy (CBT), and to foster engagement through a gamified reward system.

#### 3.3.2.2 Perception (Inputs)

The SCA operates on the history of the conversation and the user's profile. Its key inputs from the state graph are:

- **Conversation History ($H_{t-1}$):** The full transcript of the conversation up to the previous turn.
- **User's Current Message ($M_t$):** The message deemed safe by the STA.
- **User State:** Information about the user's progress, including completed modules and earned achievements.

#### 3.3.2.3 Processing Logic

The SCA's logic is generative and context-aware. It uses the Gemini 2.5 Pro model to generate a conversational response, $A_t$, that is empathetic and relevant to the user's message and history.

$$A_t = f_{SCA}(M_t, H_{t-1}; \theta_{LLM})$$

This agent has access to a toolset that allows it to retrieve and present structured content. When a user's query or the conversation flow indicates a need for a specific skill (e.g., managing anxiety), the SCA can decide to invoke its `retrieve_cbt_module` tool to fetch and present the relevant exercise.

#### 3.3.2.4 Action (Outputs)

- **Conversational Response:** A human-like text response to be displayed to the user.
- **Tool Call (Content Delivery):** Invocation of tools to present CBT exercises or other self-help modules.
- **Tool Call (Gamification):** Upon completion of a module, the SCA can call the `mint_achievement` tool, which interacts with the blockchain service to issue a non-fungible token (NFT) to the user's account as a verifiable record of their progress.

### 3.3.3 The Service Desk Agent (SDA): The Administrative Orchestrator

#### 3.3.3.1 Goal

The SDA functions as the administrative backbone of the support system. Its primary goal is to automate the operational workflows related to clinical case management and resource scheduling, thereby reducing the manual burden on university staff and ensuring that escalations and requests are handled efficiently and reliably.

#### 3.3.3.2 Perception (Inputs)

The SDA is primarily triggered by events from other agents or direct commands from the Admin Dashboard. Its inputs are structured data, not conversational text:

- **Escalation Event:** A signal from the STA containing the conversation ID and risk level of a flagged case.
- **Scheduling Request:** A structured request from the SCA (initiated by a user) containing the user's ID and desired appointment times.
- **Admin Commands:** Directives from a human administrator via the dashboard (e.g., "close case," "add note").

#### 3.3.3.3 Processing Logic

The SDA's logic is procedural and tool-based. It does not engage in open-ended conversation but rather executes a sequence of pre-defined actions based on its inputs. For example, upon receiving an escalation event, its logic is to execute the `create_case` tool, followed by the `assign_case_status` tool with the "New" parameter.

#### 3.3.3.4 Action (Outputs)

- **Database Operations:** The SDA's primary actions are database mutations, such as creating, updating, or closing case records in the clinical management database.
- **API Calls to External Services:** It can interact with external calendar systems to check for counselor availability and book appointments.
- **Notifications:** It sends automated email or dashboard notifications to counselors when a new case is assigned to them or when a student books an appointment.

### 3.3.4 The Insights Agent (IA): The Strategic Analyst

#### 3.3.4.1 Goal

The IA is designed to function as the institution's automated well-being analyst. Its goal is to autonomously process anonymized, aggregated conversation data to identify population-level mental health trends, sentiment shifts, and emerging topics of concern. This provides the institution with actionable, data-driven intelligence to inform resource allocation and proactive strategy.

#### 3.3.4.2 Perception (Inputs)

The IA is activated by a time-based trigger (e.g., a weekly Cron job) and its primary input is the entire corpus of anonymized conversation logs.

- **Time-Based Trigger:** A signal from the n8n orchestration layer to begin its analysis.
- **Anonymized Database Access:** Read-only access to the `conversation_logs` table, from which all personally identifiable information (PII) has been redacted.

#### 3.3.4.3 Processing Logic

The IA's logic involves a pipeline of Natural Language Processing (NLP) tasks performed on the collected data. This includes:

- **Topic Modeling:** Using algorithms like Latent Dirichlet Allocation (LDA) or modern transformer-based clustering to identify the most prevalent topics of discussion (e.g., "exam stress," "social isolation").
- **Sentiment Analysis:** Calculating the overall sentiment score for the student population over the given period and tracking its change over time.
- **Summarization:** Using the Gemini 2.5 Pro model to generate concise, human-readable summaries of the key findings from the topic and sentiment analysis.

#### 3.3.4.4 Action (Outputs)

- **Structured Report Generation:** The final output is a structured report (e.g., in JSON or PDF format) containing visualizations (e.g., charts of topic frequency over time) and the generated summaries.
- **Dashboard Update:** The agent pushes this report to the Admin Dashboard, updating the analytics view for university staff.
- **Email Notification:** It can be configured to automatically email the report to a list of stakeholders, such as the head of counseling services.

## 3.4 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, and promotes a clean separation of concerns [?]. 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.

### 3.4.1 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

**Placeholder for Diagram: Data Flow Diagram (DFD)**

This diagram should illustrate the flow of data between the four agents, the user, the admin, and the database. Key flows to show include:
1. User Message -> STA -> SCA -> User Response (Normal Flow).
2. User Message -> STA -> SDA -> Admin Dashboard (Crisis Flow).
3. SCA -> Blockchain Service (Gamification Flow).
4. IA -> Database -> Admin Dashboard (Analytics Flow).

Figure 3.4. A Data Flow Diagram illustrating the movement of information between the agents of the Safety Agent Suite and external entities.

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.

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).

**Placeholder for Diagram: Overall System Architecture**

This diagram should be a component diagram illustrating the core services and user interfaces. It should show:
1. A large box for the "Frontend Service (Next.js)" which contains two smaller boxes within it: "User Portal" and "Admin Dashboard".
2. An external "Student" actor pointing to the "User Portal".
3. An external "Administrator/Counselor" actor pointing to the "Admin Dashboard".
4. A box for the "Backend Service (FastAPI)" containing the "Agentic Core (LangGraph)" and the four agents (STA, SCA, SDA, IA).
5. A box for the "Database Service (PostgreSQL)".
6. A double-arrow labeled "REST API (HTTPS)" connecting the "Frontend Service" box to the "Backend Service" box.
7. An arrow labeled "Direct DB Connection" connecting the Backend to the Database.
8. An arrow from the Backend to an external "Google Gemini API" service.

Figure 3.5. The high-level technical architecture of the system, illustrating the unified frontend serving both users and admins, the backend agentic core, and the database service.

### 3.4.2 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.

#### 3.4.2.1 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 [**?**]. The primary justifications for its selection are:

- **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 [**?**]. 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 3.1.

Table 3.1. 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. |

#### 3.4.2.2 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 [**?**].

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.

#### 3.4.2.3 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-

**Placeholder for Diagram: LangGraph Conceptual State Graph**

This diagram should illustrate the agent workflow as a state graph. It should show:
1. An entry point "User Message".
2. A node for "Safety Triage Agent (STA)".
3. A conditional branch from STA: An edge labeled "Risk: Low/Moderate" leads to the "Support Coach Agent (SCA)" node. An edge labeled "Risk: Critical" leads to the "Service Desk Agent (SDA)" node.
4. The SCA node has a loop back to itself for conversation and can also branch to the SDA node for administrative tasks like booking.
5. The SDA node connects to the "Admin Dashboard" and database.

Figure 3.6. Conceptual diagram of the LangGraph state machine, showing how conditional edges route the workflow between the STA, SCA, and SDA based on the conversation's state.

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.

### 3.4.3 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 [**?**].

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:

#### 3.4.3.1 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 and view their earned gamification rewards, including the NFT achievement badges.
- **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.

#### 3.4.3.2 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 conversation, 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).

### 3.4.4 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 [**?**].

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).

### 3.4.5 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 [**?**]. 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—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.

## 3.5 Database Design

Purpose: To define the data persistence layer of the system.

Elaboration Points:

Present a clean Entity-Relationship Diagram (ERD).

Table 3.2. Key Columns and Data Types for `conversation_logs` Table

| Column Name | Data Type | Description |
|---|---|---|
| id | SERIAL PRIMARY KEY | Unique identifier |
| user_id | UUID | Reference to user (anonymized) |
| timestamp | TIMESTAMP WITH TIME ZONE | Time of message |
| message | TEXT | User or agent message content |
| sender | VARCHAR(16) | 'user' or 'agent' |
| sentiment_score | FLOAT | Sentiment analysis result |
| topic | VARCHAR(64) | NLP-inferred topic label |

## 3.6 User Interface (UI) Design

Purpose: To show the design of the human interface for the system's administrative users.

Elaboration Points:

Define the primary user persona for the dashboard (e.g., "Dr. Astuti, Head of Counseling Services").

Present wireframes or high-fidelity mockups for the key screens of the Admin Dashboard (e.g., the main analytics view, the report history page).

## 3.7 Security and Privacy by Design

Purpose: To demonstrate that critical security and privacy considerations are integral to the architecture.

Elaboration Points:

Detail the Data Anonymization Pipeline: How is Personally Identifiable Information (PII) identified and redacted from chat logs before they are stored for analysis?

Describe the Role-Based Access Control (RBAC) mechanism for the admin dashboard.

Mention standard security practices like data encryption in transit (TLS) and at rest.

## 3.8 Alur Tugas Akhir

Menguraikan prosedur yang akan digunakan dan jadwal atau alur penyelesaian setiap tahap. Alur penelian ini dapat disajikan dalam bentuk diagram. Diagram dapat disusun dengan aturan yang baik semisal menggunakan *flowchart*. Aturan dan tutorial pembuatan *flowchart* dapat dilihat di http://ugm.id/flowcharttutorial. Setelah menggambarkannya, penulis wajib menjelaskan langkah-langkah setiap alur tugas akhir dalam sub bab tersendiri sesuai dengan kebutuhan.

## 3.9 Etika, Masalah, dan Keterbatasan Penelitian (Opsional))

Bagian ini membahas pertimbangan etis penelitian dan [potensi] masalah serta keterbatasannya. Jika menyangkut penelitian dengan makhluk hidup, maka dibutuhkan adanya *ethical clearance*, di bagian ini hal itu akan dibahas. Demikian juga tentang keterbatasan ataupun masalah yang akan timbul.

# CHAPTER IV
# HASIL DAN PEMBAHASAN

Berikut ini adalah yang perlu diperhatikan untuk mengisi bab hasil dan pembahasan:

1. Setiap rumusan masalah boleh memiliki lebih dari 1 tujuan.
2. Setiap subbab harus spesifik menjawab setiap tujuan yang dituliskan.
3. Setiap rumusan masalah boleh dijawab dengan 1 subbab atau lebih.

Berikut ini adalah contoh sub bab untuk menjelaskan tujuan penelitian.

## 4.1 Pembahasan Tujuan 1 dengan Hasil Penelitian 1 (Ubah Judul Sesuai dengan Hal yang Hendak dibahas)

Sub bab pertama adalah membahas tujuan penelitian pertama dengan hasil penelitian ke-1. Dapat ditambahkan beberapa sub bab jika diperlukan.

## 4.2 Pembahasan Tujuan 1 dengan Hasil Penelitian 2 (Ubah Judul Sesuai dengan Hal yang Hendak dibahas)

Sub bab kedua adalah membahas tujuan penelitian pertama dengan hasil penelitian ke-2. Sub bab ini merupakan contoh tambahan sub bab pertama.

## 4.3 Pembahasan Tujuan 2 dengan Hasil Penelitian 3 (Ubah Judul Sesuai dengan Hal yang Hendak dibahas)

Sub bab ketiga adalah membahas tujuan penelitian kedua. Dapat ditambahkan beberapa sub bab jika diperlukan.

## 4.4 Perbandingan Hasil Penelitian dengan Hasil Terdahulu

Pembahasan penutup dapat menjelaskan mengenai kelebihan hasil pengembangan / penelitian dan kekurangan dibandingkan dengan skripsi atau penelitian terdahulu atau perbandingan terhadap produk lain yang ada di pasaran. Penulis dapat menggunakan tabel untuk membandingkan secara gamblang dan menjelaskannya.

# 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] 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

[5] 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

[6] 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

[7] 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/

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

[9] 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/

[10] 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]. Avail-

able: https://tesidottorato.depositolegale.it/bitstream/20.500.14242/180297/1/Tesi_PhD_Agnese_Salutari.pdf

[11] 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

[12] 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

[13] 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/

[14] 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

[15] 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

[16] 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/

[17] 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/

[18] 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

[19] 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/

[20] 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

[21] 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

[22] 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

[23] 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

[24] 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

[25] 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

[26] 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

[27] 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

[28] 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

[29] 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.

[30] 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

[31] 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

[32] 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

[33] 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

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

[35] 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/

[36] 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.

[37] 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.

[38] 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

[39] 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

[40] 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/

[41] 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

[42] 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

[43] 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

[44] 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

[45] 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

[46] 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

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

[48] 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/

[49] S. Barua, “Exploring autonomous agents through the lens of large language models: A review,” *arXiv preprint arXiv:2404.04442*, 2024, comprehensive review of LLM-based agents, including ReAct, LangChain, and multi-agent coordination. [Online]. Available: https://arxiv.org/abs/2404.04442

[50] 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

[51] 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

[52] 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

[53] 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

[54] 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

[55] 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

[56] 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

[57] 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

[58] 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

[59] 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.

[60] 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.

[61] 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

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

[63] 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.

[64] 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.

[65] 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.

[66] 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.

[67] 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 [61, 62] [63] [64] [65] [66, 67]

#### L.2.1.2 Penulisan Gambar

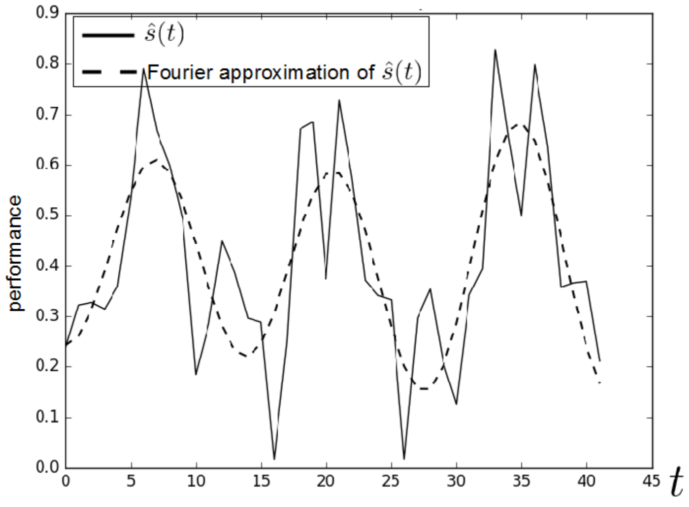

Figure 7. Contoh gambar.

Contoh gambar terlihat pada Gambar 7. Gambar diambil dari [67].

#### 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": $\mathit{PVA} = \mathit{RTD}$

#### 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.

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

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

## 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

```
procedure MakeSet($v$)
    Create new set containing $v$
end procedure

function FindSet($v$)
    return a set containing $v$
end function

procedure Union($u$,$v$)
    Unites the set that contain $u$ and $v$ into a new set
end procedure

function Kruskal($V, E, w$)
    $A \leftarrow \{\}$
    for each vertex $v$ in $V$ do
        MakeSet($v$)
    end for
    Arrange $E$ in increasing costs, ordered by $w$
    for each ($u$,$v$) taken from the sorted list do
        if FindSet($u$) $\neq$ FindSet($v$) then
            $A \leftarrow A \cup \{(u, v)\}$
            Union($u, v$)
        end if
    end for
    return A
end function
```

### L.4.2 Sample Python code

```python
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
```