Department of Computer Science,
North Carolina A&T State University

Detailed Design Specification

Comp 496: Senior Design II 2025

Stack Underflow Lumi

Jolisa Fields
Maya Swan
Laila Donaldson
Dana Brunson
Nicolas Harris

Revision History

Revision	Date	Author(s)	Description
0.1	4.18.2025	JF	Document Creation
0.2	4.19.2025	MS, LD	Section 2 and 5
0.3	4.21.2025	LD, MS, JF, NH, DB	Adding diagrams
1.0	4.23.2025	NH, DB	Section 3 and 4
1.1	5.08.2025	MS, JF, DB, LD	Final document

Contents

List of Figures

List of Tables

1 Introduction

Mental health challenges affect people around the world. Stigma, lack of time, and limited personalized support often make these challenges worse. Lumi was developed to address these issues by providing a web-based mental wellness companion. It combines artificial intelligence and augmented reality to help users take care of their emotions through accessible and personalized engagement.

This document explains the system architecture and design reasons for Lumi. It is based on earlier materials that describe core features like sentiment-driven journaling, AR wellness exercises, and daily mood tracking. For more background on user needs, readers can refer to the complete requirements document. For a closer look at the system's modular design, the architectural specification is also available.

2 System Overview

The architectural design of Lumi is composed of three primary layers: Layer X, Layer Y, and Layer Z. Each layer is responsible for a distinct set of functionalities that collectively drive the system’s core operations. These layers are designed to interact seamlessly, enabling a smooth user experience while maintaining modularity, scalability, and maintainability of the overall platform.

Layer X serves as the user interaction layer. It includes the Check-in/Journal UI, which facilitates user input through daily check-ins and journaling. This module allows users to express their thoughts and track their emotional state through a friendly, intuitive interface. It also houses the AR Module, built using Three.js, which delivers immersive relaxation experiences like guided breathing exercises or calming visuals. These AR features help ground users in moments of high stress or anxiety, making mindfulness practices more engaging and accessible. Communication from the UI and AR module is routed to the backend, where it is further processed for sentiment analysis or journaling storage.

Layer Y represents the intelligence layer, where a sentiment analysis Natural Language Processing Model is leveraged to analyze user input. The NLP/Sentiment Analysis model combines two pretrained emotion and text classification models to detect mood indicators, emotional tone, and signs of distress within journal entries. The analysis feeds into the Activity Recommendation engine, which determines the most suitable wellness activity or support resource for the user based on their emotional state. This dynamic and responsive functionality enables Lumi to adapt to the user’s changing needs, reinforcing consistent mental health engagement.

Layer Z is the backend logic and data management layer. At the heart of this layer is the Logic Coordinator, which receives inputs from both Layer X and Layer Y, directing them to the appropriate backend services. It connects to the Journaling Engine, responsible for handling and formatting user entries, and the Resource Hub, which aggregates and presents relevant support links, self-help materials, or emergency contacts. All user interactions, analysis results, and system outputs are captured in the Data Logger, powered by MongoDB, ensuring that a comprehensive and secure record of each user’s journey is maintained.

The seamless communication between these layers, illustrated in the data flow diagram, allows Lumi to support a continuous feedback loop between user input, AI interpretation, and actionable suggestions. This architecture not only ensures system responsiveness and personalization but also lays the groundwork for future enhancements, such as adaptive learning or expanded resource integration. Ultimately, Lumi's layered system supports its mission of making emotional care accessible, consistent, and personalized for all users.

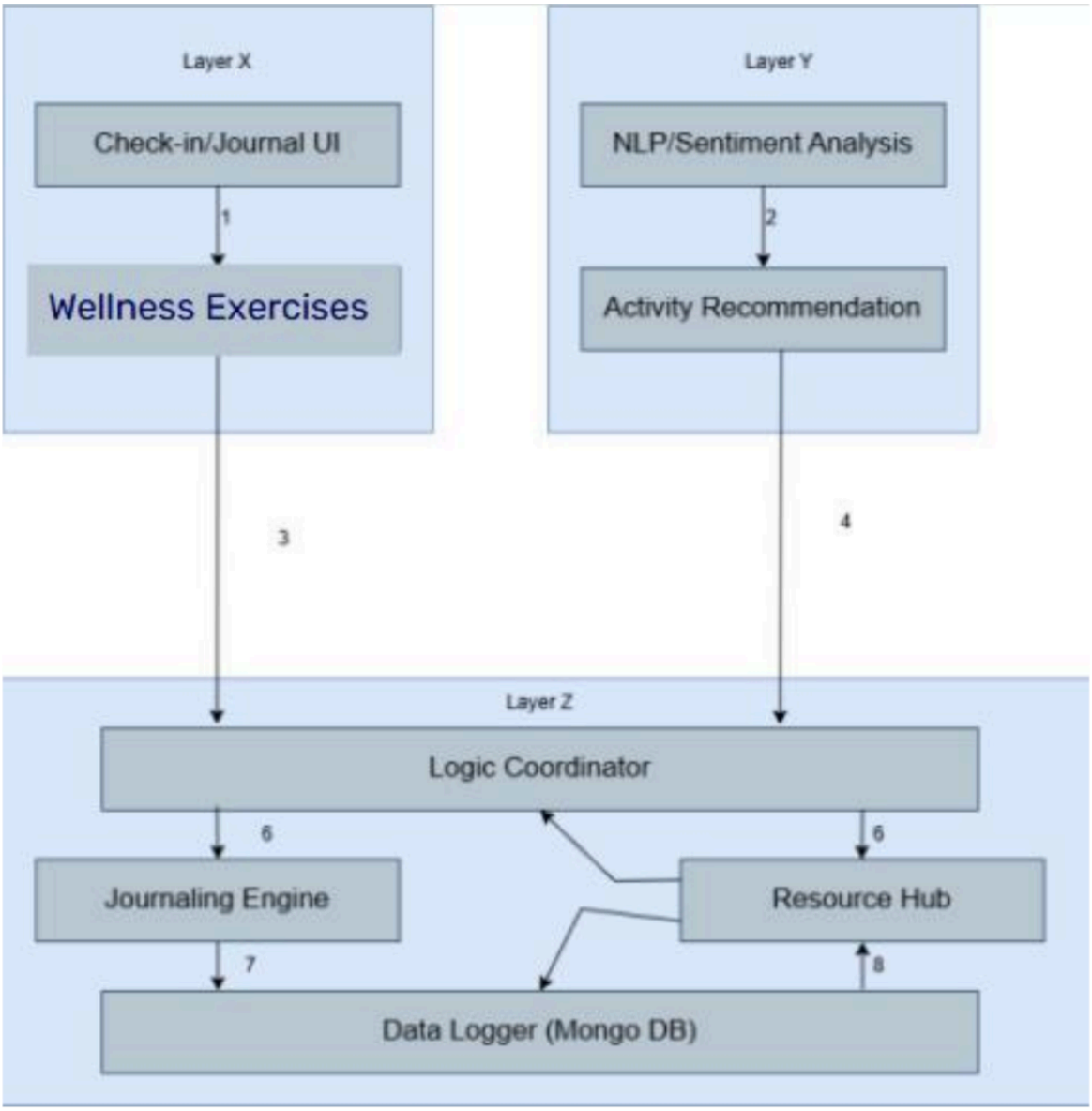

Figure 1: System Architecture

3 X Layer Subsystems

Layer X focuses on Augmented Reality (AR). The primary purpose of this layer is to provide a selection of immersive relaxation experiences , such as breathing exercises and calming visuals, for users to engage with. The activities are also accessible without requiring an AR headset.

3.1 Layer Hardware

This layer does not require hardware beyond standard computing devices capable of running modern web applications and supporting AR functionality. The AR subsystem runs on devices compatible with the needed APIs.

3.2 Layer Operating System

The layer is designed to function on any operating system that supports modern browsers with AR capabilities.

3.3 Layer Software Dependencies

The systems required by the layer are Three.js, React, browser APIs, and A-Frame.

3.4 Subsystem X1 - Visual Components

The purpose of the subsystem is to act as a relaxing activity for the user to participate in to help ground them in the case of overwhelming feelings or feelings of anxiety. It is software that works the user through exercises that distract them from their feelings at the moment, such as guided breathing animations and other environments.

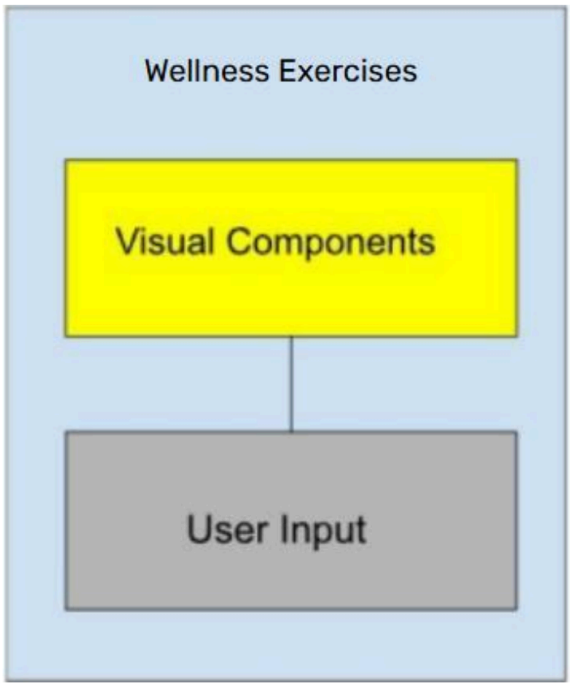

Figure 2: Architecture of the X Layer

3.4.1 Subsystem Hardware

No dedicated hardware is required beyond the user's device equipped with a camera and capable of running a compatible web browser. Devices may include smartphones, tablets, or laptops.

3.4.2 Subsystem Operating System

The subsystem functions on the web browser and only requires a device that supports modern web browsers.

3.4.3 Subsystem Software Dependencies

- JavaScript: Used to develop all interactive features.
- HTML/CSS: Used for structuring and styling the AR interface.

3.4.4 Subsystem Programming Languages

The subsystem uses JavaScript as its primary programming language. JavaScript is used to define configuration objects and control logic for AR exercises, animations, and duration settings within the application.

3.4.5 Subsystem Data Structures

The subsystem includes key data structures used to define and control the breathing animation logic. One core structure is an animation configuration object, which stores parameters for the guided breathing exercise.

```
const ARExercise = {

animationType: breathing,

Duration: 60,

};
```

3.4.6 Subsystem Data Processing

The subsystem handles user interactions and environmental simulation using a combination of real-time animation loops, waypoint detection, and responsive user feedback.

In the guided walking exercise, the system tracks user movement relative to a series of predefined waypoints in 3D space. The core processing strategies include:

- Waypoint Proximity Detection: The distance between the camera (user's position) and each waypoint is calculated every frame using Euclidean distance. When the user is within a threshold (< 1.5 units), the system triggers a prompt.

```
if (currentWaypoint < waypoints.length) {

    const target = waypoints[currentWaypoint];

    const distance = camera.position.distanceTo(target.position);
```

```
            if (distance < 1.5) {
              showPrompt(currentWaypoint);
              currentWaypoint++;
            }
        }
```

- Continuous Input Handling: Keyboard input is interpreted as directional movement. A combination of keydown/keyup events and a real-time animation loop (setAnimationLoop) updates the user's position based on time delta calculations. This allows fluid movement even on varied hardware.
- Prompt Timing: Prompt text is faded in using setInterval-based opacity interpolation, offering a calming UX experience without harsh transitions.

The breathing exercise simulates a breathing rhythm through procedural animation of a 3D sphere, guided visually and auditorily:

- Breathing Cycle Animation: The sphere scales up and down smoothly using a linear transformation updated on each animation frame. This simulates inhalation and exhalation.

```
        if (sphere) {
          const scale = 1 + 0.05 * Math.sin(Date.now() * 0.002 + index);
          sphere.scale.set(scale, scale, scale);
        }
```

- Textual Prompt Switching: The prompt updates to "Breathe in..." or "Breathe out..." depending on the current scale direction. This reinforces the visual cue with textual guidance.
- Ambient Audio Feedback: A looping background track is triggered upon session start, enhancing the user’s focus and relaxation through multisensory feedback.

3.5 Subsystem X1 - User Input

The User Input subsystem handles the interactions from the user within the application. All of the forms, buttons, and other elements.

3.5.1 Subsystem Hardware

The hardware components required for the user input are keyboards and a trackpad for scrolling through and hovering.

3.5.2 Subsystem Operating System

The system can operate on any modern web browser that supports the React application.

3.5.3 Subsystem Software Dependencies

The user input system requires the React library, the MongoDB database to store the information, and the backend APIs.

3.5.4 Subsystem Programming Languages

The subsystem is built using:

- JavaScript
- JSX
- CSS
- Java
- Python

3.5.5 Subsystem Data Structures

The Journal subsystem uses simple, structured data to record user reflections or notes during wellness sessions. The primary data structure is the JournalEntry object, which stores individual journal inputs along with their associated timestamps.

```
const JournalEntry = {

timeStamp: date,

Text: string,

};
```

3.5.6 Subsystem Data Processing

The input system is responsible for capturing, validating, and transmitting user-generated data like journal entries, check-ins, and exercise selections. The system captures user interactions and uses event listeners to track changes in real time. The input data is validated on the client side to ensure correctness and completeness before submission.

4 Y Layer Subsystems

Layer Y is a software-based layer which focuses on natural language processing (NLP). Its primary purpose is to analyze the user-input journal entries for emotional sentiment and potential signs of harmful intentions.

4.1 Layer Hardware

Not Applicable

4.2 Layer Operating System

Layer Y is designed to run on cloud environments that support Python and the required NLP libraries (Transformers, Pytorch etc.).

4.3 Layer Software Dependencies

Key software dependencies that will be used for this layer include:

- Transformers: for loading and running pretrained NLP models.
- Python Logging: for recording model actions and flagging hazardous outputs.
- Regular Expressions: For basic pattern matching of high-risk words (e.g., “hurt,” “hopeless,” “end it”).

4.4 Subsystem Y1 - Journal Entry

This subsystem is responsible for capturing user-submitted journal entries. Users input text describing their thoughts or experiences. These journal entries are then passed to downstream components, such as the NLP Analyzer, for sentiment and emotion analysis.

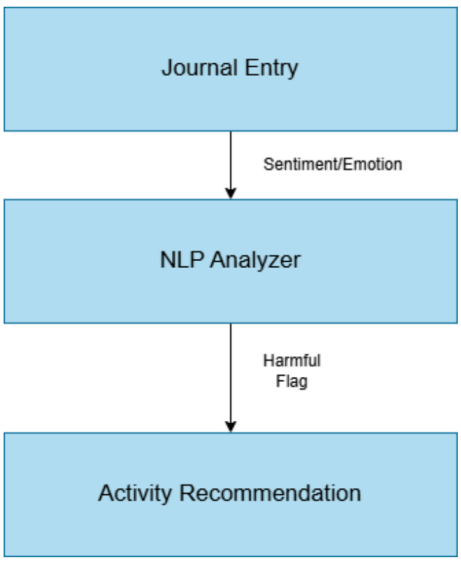

Figure 3: Architecture of the Y Layer

4.4.1 Subsystem Hardware

Not Applicable

4.4.2 Subsystem Operating System

The subsystem functions on the web browser and only requires a device that supports modern web browsers.

4.4.3 Subsystem Software Dependencies

This subsystem has minimal software dependencies. If part of a web or app interface, it relies on basic front-end technologies for UI elements like text boxes and submit buttons.

4.4.4 Subsystem Programming Languages

- JavaScript (for web-based input forms)
- HTML/CSS (for interface layout and style)

4.4.5 Subsystem Data Structures

The core data structure is a simple journal entry which captures the time and content of a user-submitted journal entry. It is passed to downstream subsystems like the NLP Analyzer for emotional interpretation and sentiment evaluation.

4.4.6 Subsystem Data Processing

The Journal Entry subsystem does not perform significant data processing. Its role is to accept user-written input, timestamp the entry upon submission, and package it into a simple object structure for downstream analysis. No machine learning or transformation is performed here. It acts simply as a data collection layer.

4.5 Subsystem Y2 - NLP Analyzer

This subsystem analyzes journal entries, extracting sentiment and emotional tone while checking for potentially harmful or self-harm-related language. It uses pretrained models and phrase-matching to identify risk indicators.

4.5.1 Subsystem Hardware

Not Applicable

4.5.2 Subsystem Operating System

Compatible with any OS that supports Python and the necessary machine learning libraries, with primary testing done in a Colab environment.

4.5.3 Subsystem Software Dependencies

Transformers, such as HuggingFace, will be used to load and run sentiment

analysis models.

- The DistilBERT model will be used to train our model
- The j-hartmann model will be used to classify emotions in English text data

Logging is used to track the processing of journal entries and flags for potentially harmful words

4.5.4 Subsystem Programming Languages

Python is the sole programming language used for the implementation.

4.5.5 Subsystem Data Structures

Input for this subsystem will be the raw journal entry strings. The output will be dictionaries containing sentiment and sentiment score; top emotion, emotion confidence, and full emotion score breakdown; and a Boolean flag to detect harmful language.

4.5.6 Subsystem Data Processing

A fine-tuned DistilBERT model will be used for sentiment analysis. Harmful language detection will be done through keyword-based matching of common self-harm expressions. Logging will be done with warning-level alerts when harmful content is identified.

4.6 Subsystem Y3 -Activity Recommendation

This subsystem generates personalized feedback based on the output from the NLP subsystem. It provides supportive responses or recommendations to the user, depending on the emotional tone, sentiment, and presence of harmful language in their journal entry.

4.6.1 Subsystem Hardware

Not Applicable

4.6.2 Subsystem Operating System

Compatible with any OS that supports Python and the necessary machine learning libraries.

4.6.3 Subsystem Software Dependencies

Built in Python, leveraging standard libraries

4.6.4 Subsystem Programming Languages

Python is the sole programming language used for the implementation.

4.6.5 Subsystem Data Structures

Input is taken from the NLP subsystem, and the output will be a feedback dictionary including messages for positive sentiment/emotion as well as empathetic responses or support resources for negative/harmful entries.

4.6.6 Subsystem Data Processing

Positive sentiment and uplifting emotions should lead the AI to reinforce positivity with motivational feedback. Negative sentiments or emotions such as anger or sadness should lead to the AI offering gentle guidance or journaling tips. Detected harmful language should trigger support messaging and suggest available helplines for mental health resources.

5 Z Layer Subsystems

Layer Z is the foundational logic and storage layer for Lumi. It supports journaling, daily check-ins, mood tracking, and resource delivery. This layer manages structured user input and enables data persistence, feeding upward to Layer Y for sentiment analysis. It operates fully within a software environment and interacts with a cloud-based MongoDB database.

5.1 Layer Hardware

Not Applicable

5.2 Layer Operating System

Layer Z operates on any OS that supports modern web development and database services.

5.3 Layer Software Dependencies

- MongoDB: For storing journal entries, and mood check-ins.
- Express.js: For handling API routing and requests.
- Mongoose: For schema validation and interaction with the MongoDB database.

5.4 Subsystem Z1

This subsystem manages the recording and retrieval of structured journaling and emotional data. It acts as the primary interface between the user inputs and the persistent data store.

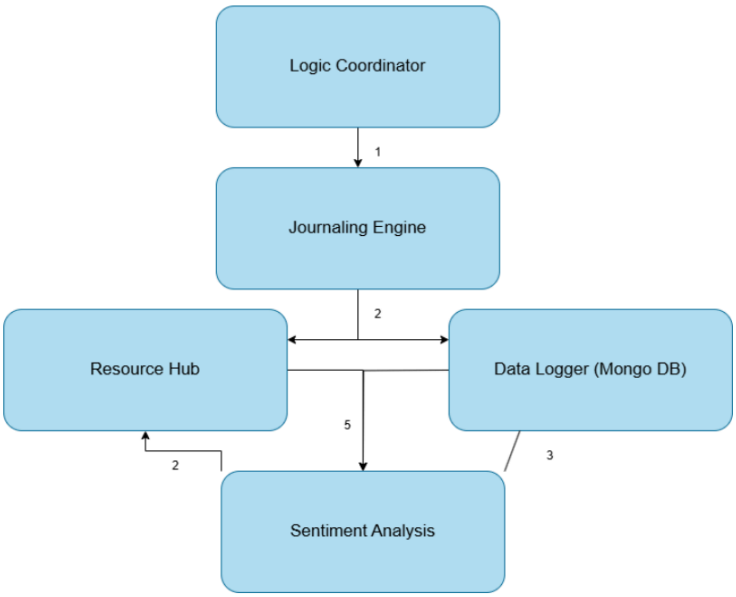

Figure 4: Architecture of the Z Layer

5.4.1 Subsystem Hardware

Not applicable. This subsystem runs as a Node.js service connected to MongoDB.

5.4.2 Subsystem Operating System

It is compatible with all OS environments that support Node.js and MongoDB.

5.4.3 Subsystem Software Dependencies

- Node.js: Core runtime for API functionality.
- MongoDB: Cloud-based NoSQL database.
- Mongoose: For structuring and validating journal entries.

5.4.4 Subsystem Programming Languages

- JavaScript: Primary language used for backend logic.
- JSON: Used for data storage formats and HTTP response bodies.

5.4.5 Subsystem Data Structures

Each journal entry is stored as a MongoDB document with the following schema:

```
{
 username: String,
 title: String,
content: String,
date: Date,
}
```

A separate collection stores mood check-in data:

```
{
 username: String,
 mood: String,
notes: String,
date: Date,
}
```

5.4.6 Subsystem Data Processing

- Journal entries and mood logs are time stamped and saved to collections.
- Data is retrieved for frontend visualization or passed to Layer Y for sentiment analysis.
- Journal history is filtered by date or mood to generate a mood trend graph.

6 APPENDIX A

References