Software Design Document Version 2

2/15/2026

Team Hive Thrive

Team Lead	**Elijah Sprouse**	**ebs233@nau.edu**
Member	**Andrew Velez**	**ajv359@nau.edu**
Member	**Benjamin Levine**	**bjl348@nau.edu**
Member	**Latisha Talayumptewa**	**lt537@nau.edu**

Mentor | Md Nazmul Hossain

Sponsor | Dr. Okim Kang

Contents

1. Introduction

Loneliness has become one of the most significant challenges facing young people today; studies show that nearly three out of four individuals between the ages of 16 and 24 regularly experience feelings of isolation or disconnection. For a generation that has grown up constantly connected online, this trend is both surprising and concerning. It represents more than just a social issue; it is a growing threat to mental health, motivation, and overall well-being.

Many tools claim to help people feel more connected, yet they often fall short of creating real change. Social media platforms and messaging apps make it easy to stay in touch, but they rarely provide the sense of community or support that people truly need. What is missing are digital tools that encourage healthy habits, promote consistency, and make positive behavior feel rewarding. Technology should not only allow people to communicate; it should help them grow.

Our team, Hive Thrive, is partnering with Dr. Okim Kang to create a progressive web application that helps users, especially teens and young adults, build and maintain better wellness routines. The system encourages small but meaningful actions, using features such as interactive teaching, gamification, virtual pet customization, progress and streak tracking, and more, to help engage users and make their goals achievable.

This Software Design Document will serve as a specification for the architectural design and the technical decisions we will be making throughout the development of our project. It will act as the blueprint for the final result we are aiming for. Due to the levels of depth we dive into for this document, we will find flaws and be able to fine tune our expectations.

This document will also be given to anyone who will work on this project in the future, whether it be to keep updates alive or to add new features. It will serve as an architectural guide that we planned and followed, and will hopefully be a good source of documentation for them.

2. Implementation Overview

This project builds upon our client's existing web-based wellness platform, Me Balanced, by adding new features in an already deployed system. Our planned implementation hinges on the current architecture, our development practices, and the need to deliver before the deadline, of course. As a result, the focus is extending and refining existing components and developing a select few new ones, rather than redesigning the system from the ground up.

Our overall solution can be summarized by providing users with a supportive and engaging wellness experience through daily interaction. The system should enable short check-ins through interactive surveys, deliver timely feedback based on user input, include a virtual pet, offer conversational support through AI, and if time permits, be accessible via mobile applications on Android and/or iOS. These additions are designed to feel responsive and approachable, while remaining consistent with the platform's existing structure and user flows.

From a technical perspective, the system follows a standard client-server web architecture. The frontend is responsible for all user-facing interaction, including survey interfaces, progress visualization, streak displays, and chat-style conversations with AI agents. The backend manages application logic, data persistence, and coordination with external services. This separation ensures that user interface updates can be developed independently from data processing and AI integration, while maintaining a clear boundary of responsibility between layers.

The technology stack reflects both project requirements and prior implementation decisions. The frontend is built using modern JavaScript frameworks and component-based UI design, allowing for dynamic rendering of survey elements and conversational interfaces. The backend is implemented using a server-side JavaScript runtime and web framework, providing REST-style endpoints for data submission, retrieval, and AI interaction. A document-oriented database is used to store user responses, progress data, and streak information, supporting flexible data structures that evolve as new wellness features are added. External AI services are accessed through backend APIs rather than directly from the client, ensuring centralized control over prompt handling and response filtering.

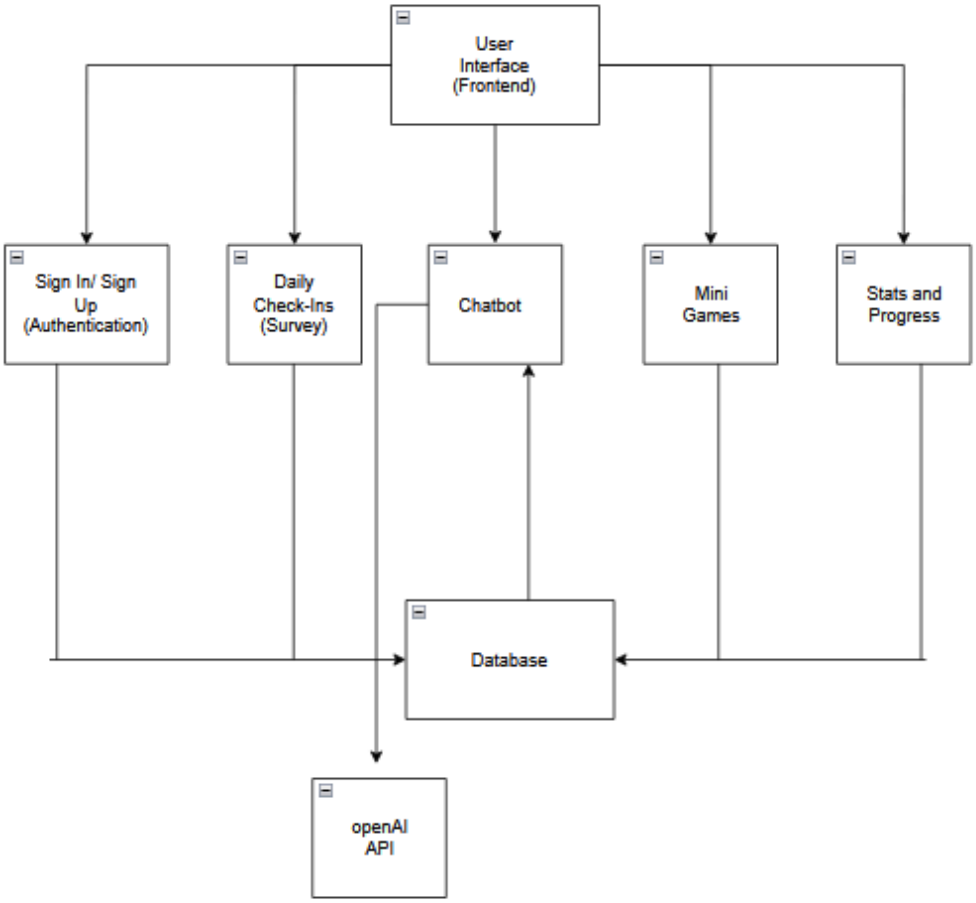
Any artificial intelligence features should be integrated cautiously and intentionally. The chatbot and virtual pet capabilities must rely on carefully processed AI requests, where prompts are constrained and contextualized before being sent to an external AI provider (OpenAI's ChatGPT). This design supports the enforcement of behavioral guardrails and limits the risk of unsafe or misleading responses, which is of utmost importance in a system monitoring mental health. We feel that by working with a popular and heavily maintained and regulated LLM like ChatGPT, and our own implemented guardrails on our backend, we can provide a safe, engaging experience for our users.

Development proceeds in incremental cycles that align with planned demonstrations and evaluation milestones. Each cycle introduces fully functional features that can be tested and demonstrated end to end, beginning with core interaction flows and expanding toward richer AI behavior and performance optimizations. This approach allows technical decisions to be validated early, reduces integration risk, and ensures the design remains closely aligned with the system as it is actually implemented.

Overall, this implementation approach emphasizes compatibility and consistency with the existing system. By grounding design decisions in realistic, timely constraints and phased development, the project maintains an executable blueprint that supports a successful implementation of planned features. In addition to this, the project will remain scalable for future development.

3. Architectural Overview

The architecture of our product prioritizes user usability and overall scalability. Our front end being built with HTML and CSS gives us different options for design and usability features, benefiting the users. Our use of nodeJS helps the backend functionality, and additionally adds to the scalability of the product. Using SQL for our database lets us scale our product as needed and is very easy to work with as a team.



System Architecture Diagram

A user is required to sign up or sign in to access the contents of our product. Thus, the connection between the sign in page and the database is a very important component. Once a user is added to the database or found in the database, they are given permission to access the features. When a user completes a survey, their data is sent to the database and stored. In addition, the AI Pet Chatbot can also access survey information. However, the way we have made it is so the AI can not connect the name of the user to the data received from the surveys. This is a way we plan to keep our users more safe while dealing with sensitive topics, and letting the AI help a user with their struggles without personal information being given. Once surveys

are completed, rewards are added to a user's account and that information is stored in the database. It is only through the use of HTML, CSS, and nodeJS that we can achieve such a great working system where all communication works smoothly while giving the users a great experience as well.

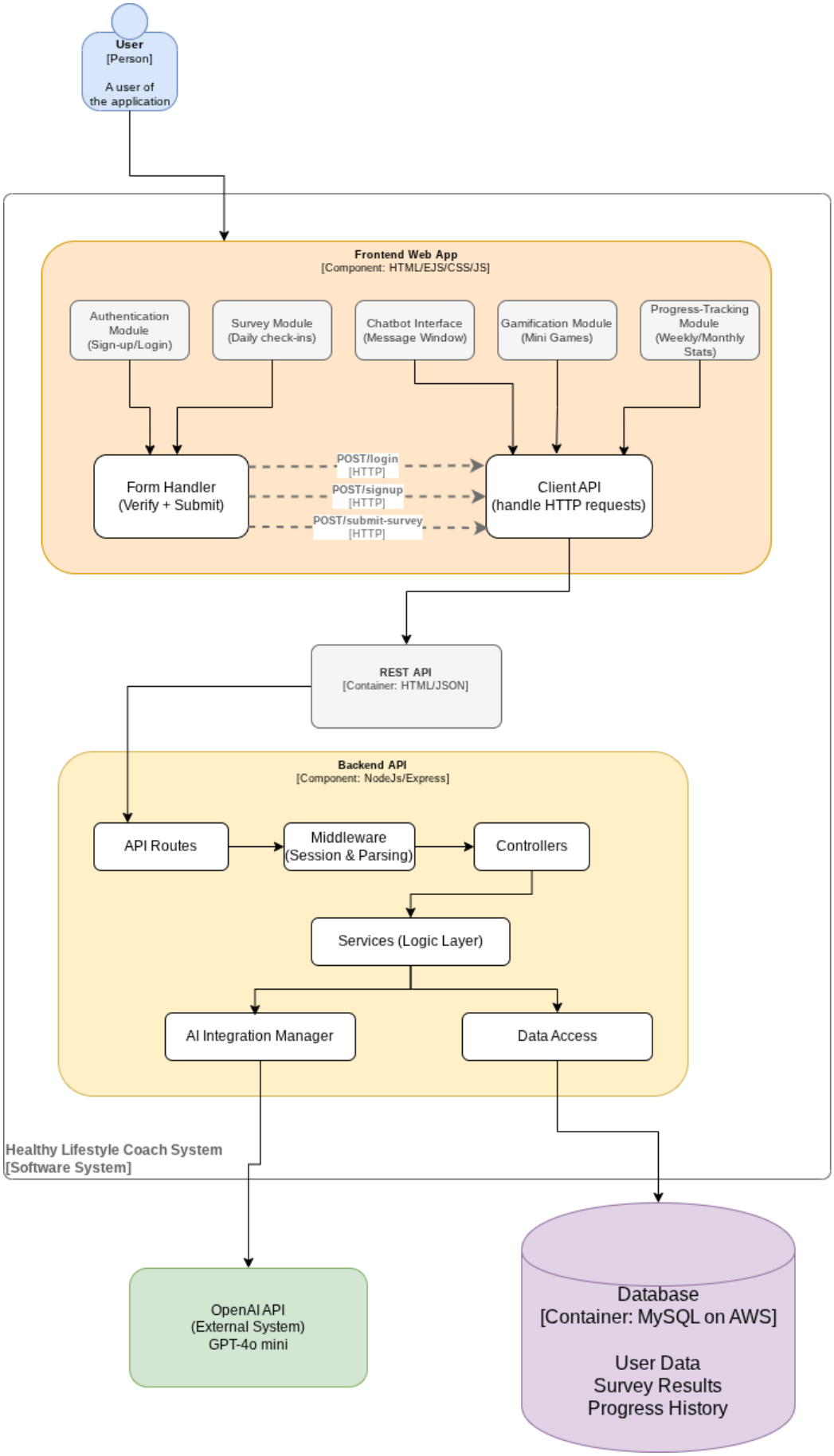
For our Chatbot and Pet Chatbot, we are utilizing an OpenAI API using GPT 4o mini. OpenAI is a leading company when it comes to the use of AI in chatbots, and thus their safety and guide rails align well with our desired method. In addition, our team has more experience with OpenAI and GPT than other alternatives. We have allowed our AI to read results from surveys without seeing identifying information, as to keep our users safe. We have overall stats and progress from the user's surveys that are stored as a page, so the users can see a graph of their progress. Additionally, there are mini-games that can be played on the website that have their data stored in the database, so users can try to beat their high scores.

4. Component-Level Design

4.1. User-Interface Component (Frontend) & Server-Side Component (Backend):

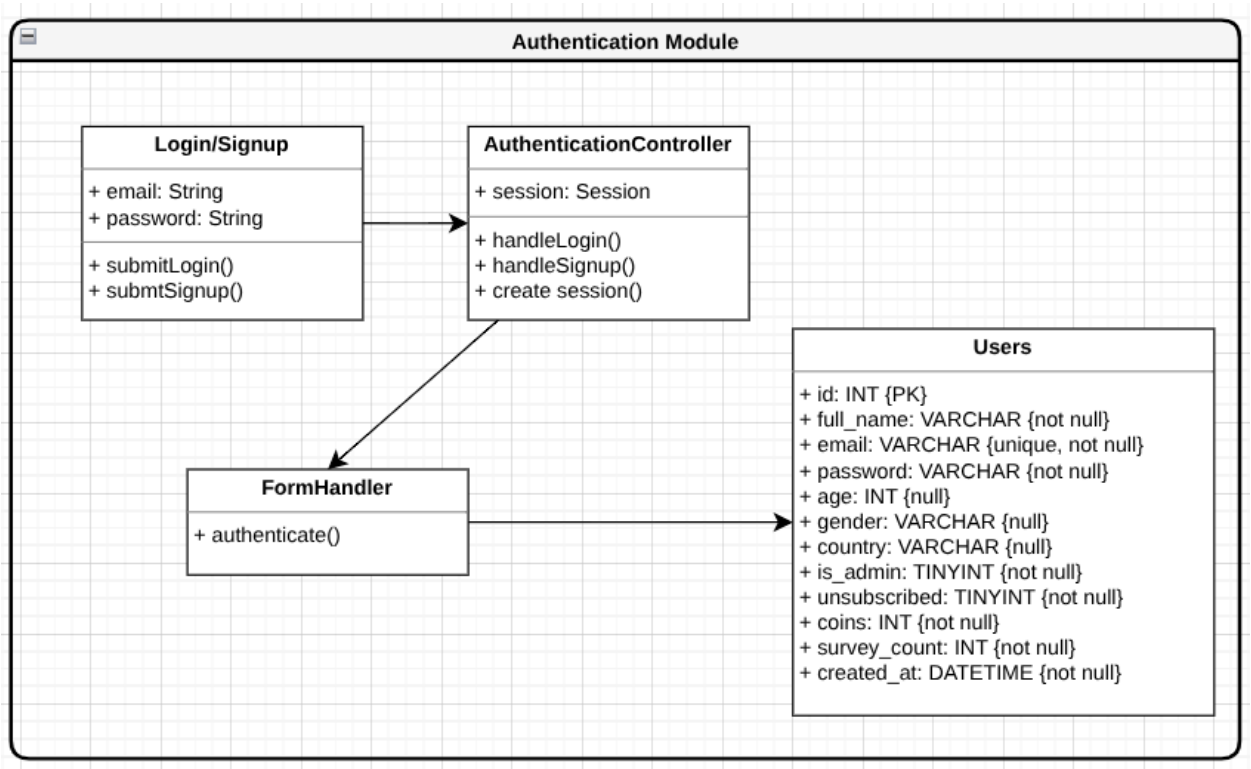
Our frontend component consists of the technical stack: HTML(EJS), CSS, and Javascript. This frontend technology stack is responsible for providing a dynamic, server-rendered interface, providing a seamless and interactive experience for users. The interface, for instance, allows the user to log into the system, visually interact with an AI chatbot, participate in daily check-ins with an active point system, engage with a gamification system which integrates rewards elements, and view weekly & monthly progress. The UI component modules handles the user input and proceeds to communicate with the backend through a middleware web-framework, handling both requests and responses.

The Server-Side Component acts as the controller and middleware of the application, utilizing the backend technology stack of Node.js and Express.js. The backend is responsible for organizing data flow and connection between the user interface, database, and the chatbot system. The module handles all business logic including user authentication, progress tracking, point rewarding, and session management. It ensures the frontend is only displaying content while the backend continues to maintain data and handle logic.


User
[Person]
A user of the application
Frontend Web App
[Component: HTML/EJS/CSS/JS]
Authentication Module (Sign-up/Login)
Survey Module (Daily check-ins)
Chatbot Interface (Message Window)
Gamification Module (Mini Games)
Progress-Tracking Module (Weekly/Monthly Stats)
Form Handler (Verify + Submit)
POST/login [HTTP]
POST/signup [HTTP]
POST/submit-survey [HTTP]
Client API (handle HTTP requests)
REST API
[Container: HTML/JSON]
Backend API
[Component: NodeJs/Express]
API Routes
Middleware (Session & Parsing)
Controllers
Services (Logic Layer)
AI Integration Manager
Data Access
Healthy Lifestyle Coach System
[Software System]
OpenAI API
(External System)
GPT-4o mini
Database
[Container: MySQL on AWS]
User Data
Survey Results
Progress History

Public Interface Description:

Authentication Module:

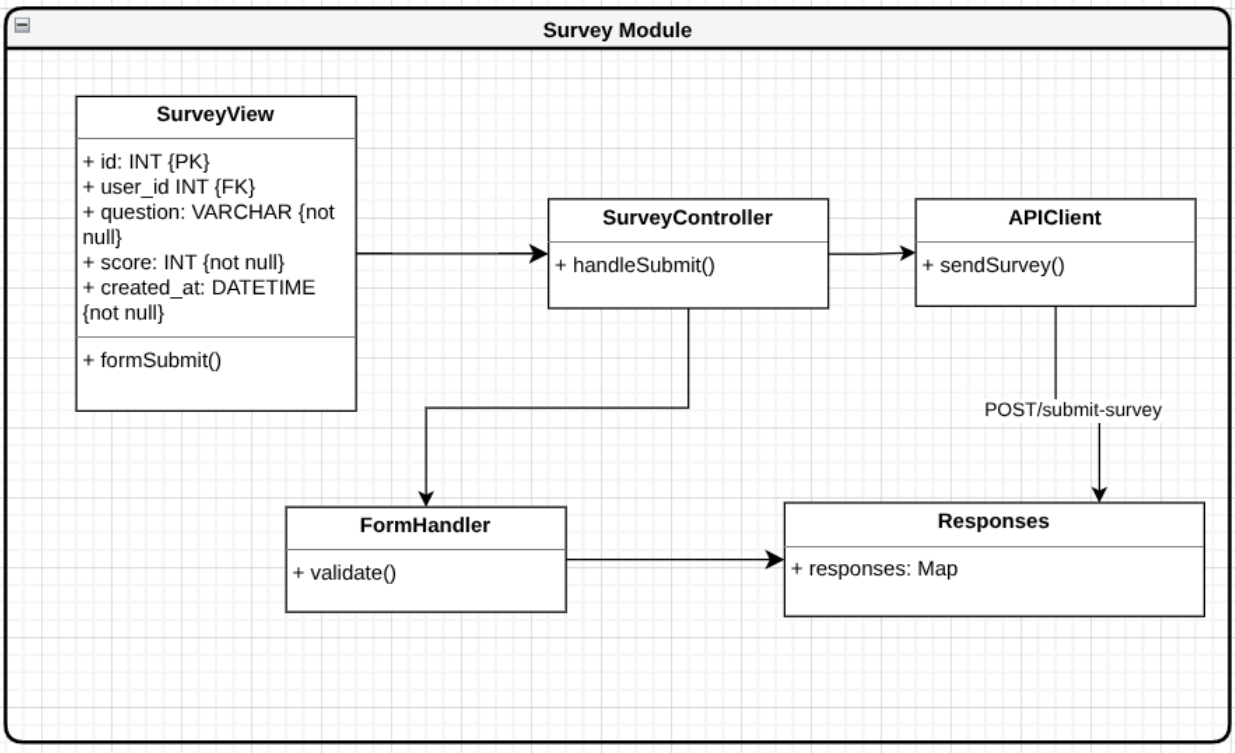


Service Provided: Handles user identity verification and sessions.

Inputs: User inputs desired “Email” and “Password”, then clicks “Login”/”SignUp”

Outputs: “Email” and “Password” are validated through Form Handler, ensuring correct formatting and calls are made through API connector to validate credentials.

Survey Module:

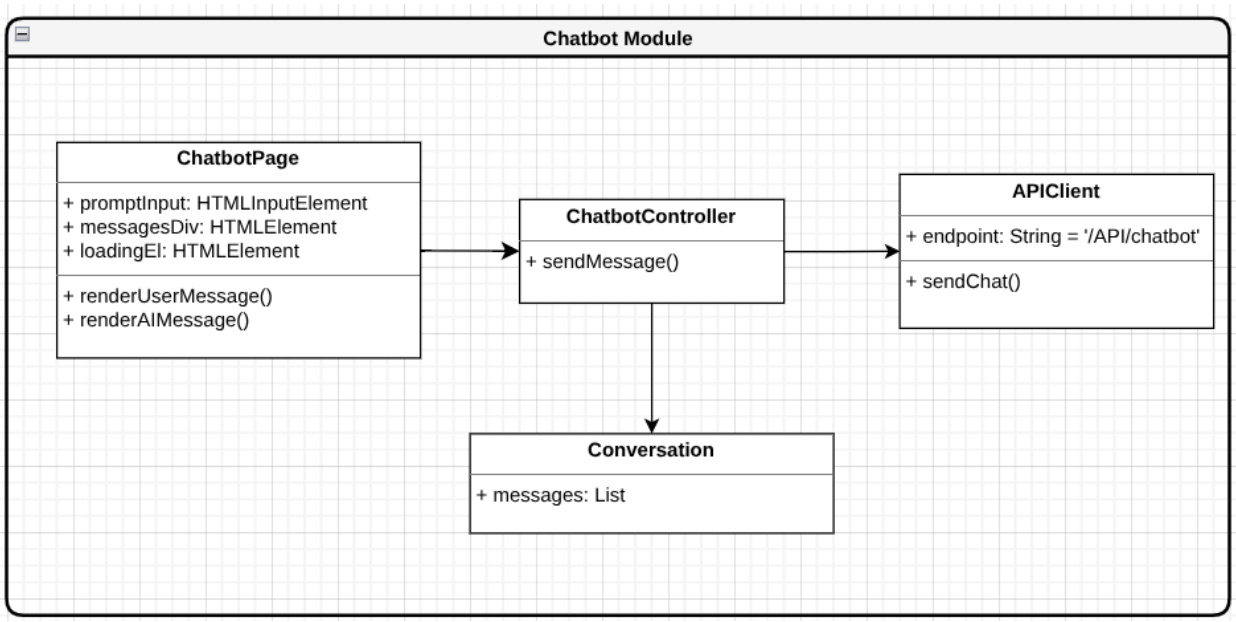


Service Provided: Capture user sentiment and activity data.

Inputs: User selects a response to the given health question.

Outputs: Survey calls on Form Handler to prevent incomplete submissions. Once form.submit() is called, JSON payload is sent through API connector.

Chatbot Module

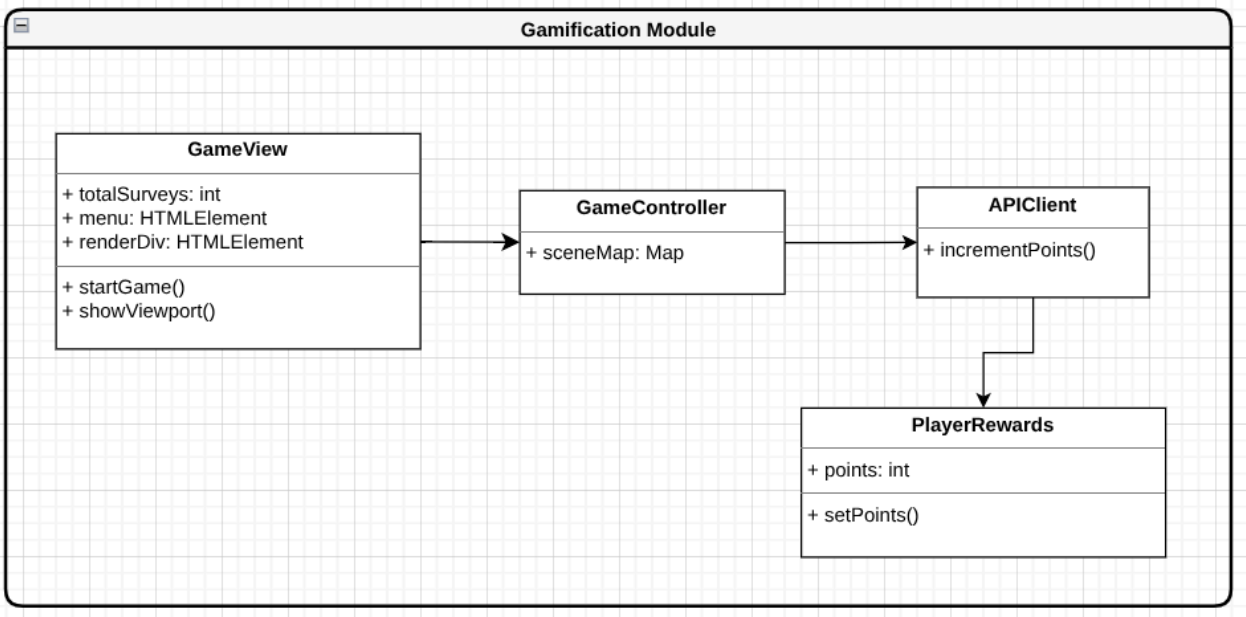


Service Provided: Provide window for AI communication.

Inputs: User types in input in the message composer.

Outputs: Connects to API connector to handle real-time message transmissions.

Gamification Module:

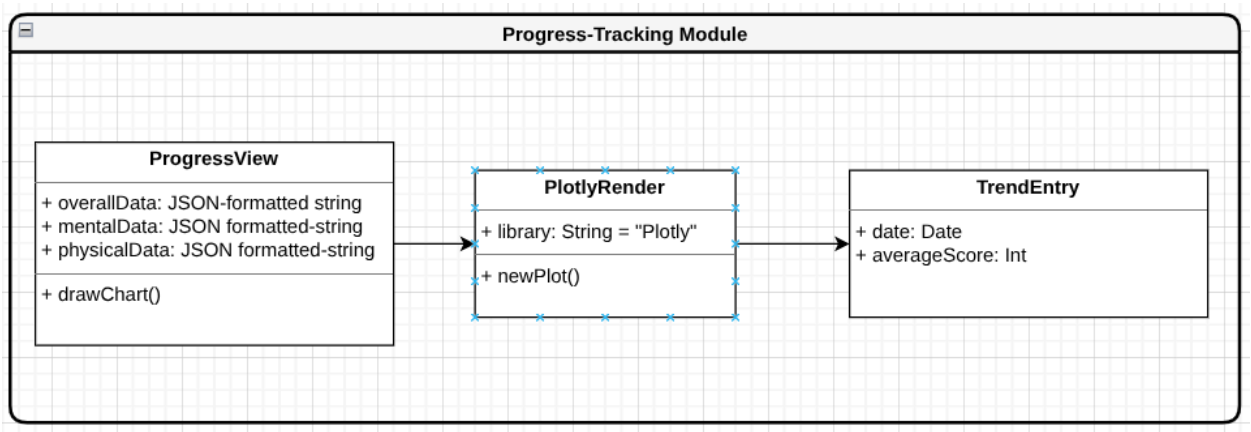


Service Provided: Manages user engagement through mini games and conducts point-system logic.

Inputs: User interacts through mouse swipes, clicks, etc. within the minigame viewport.

Outputs: The call to the API connector is made to update the user's total in the backend database once the game concludes.

Progress-Tracking Module:



Service Provided: Obtain survey data over time for user trend visualization, creating both a weekly and monthly trend graph.

Inputs: User clicks on either Weekly Progress or Monthly Progress to view the time-scale of the graph.

Outputs: Call to API connector is made to retrieve historical user data which is passed to a graphing library, and the Weekly/Monthly Progress graph is rendered.

Services Provided: The backend defines endpoints which the frontend uses to interact with the system.

Inputs:

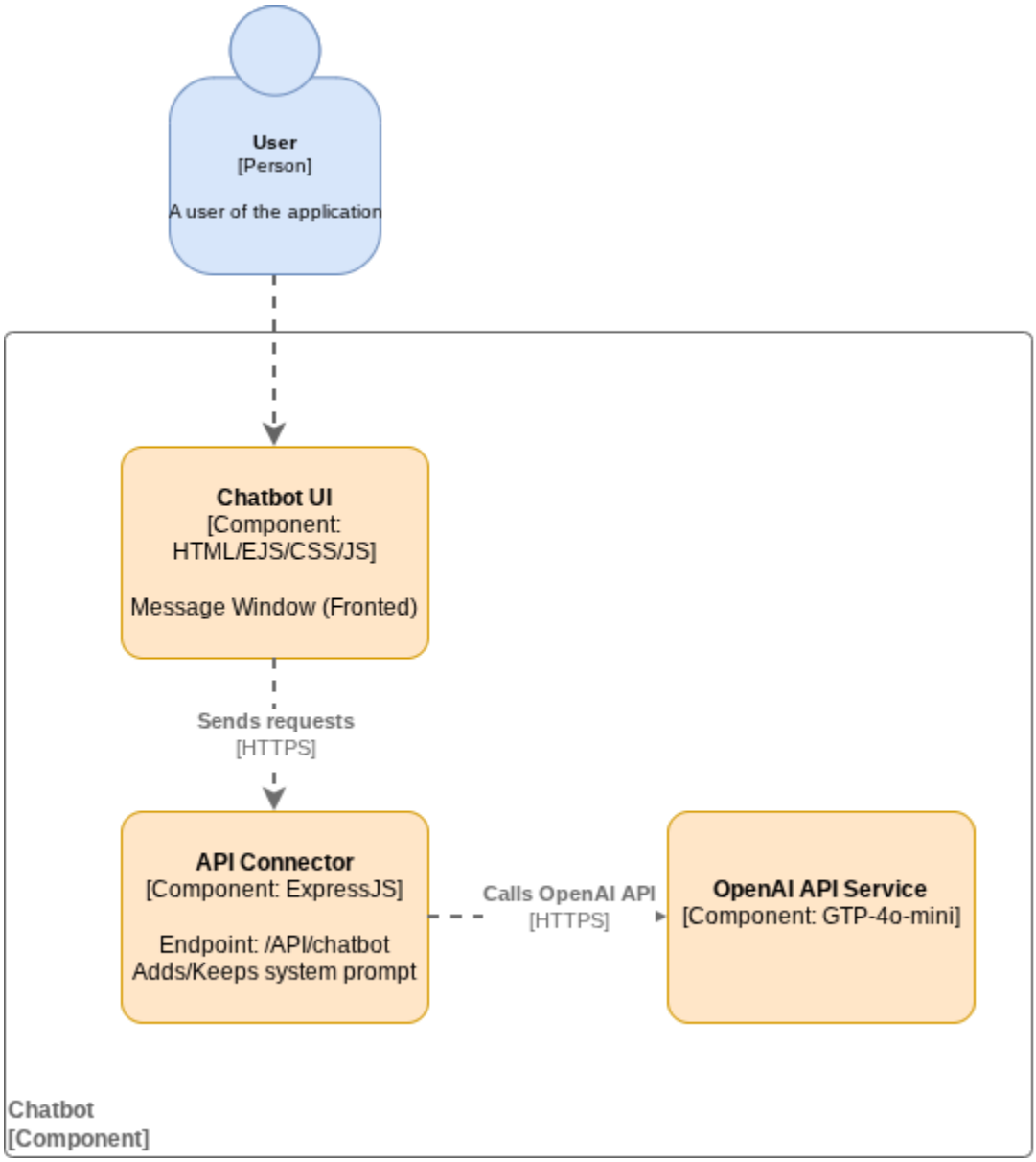
- POST/login or POST/sign-up accepts user credentials and a session token is provided.
- GET/progress retrieves the progress data of the user.

Outputs:

- Queries representing commands are sent to the database component to save, retrieve, or change records.
- Users create prompts to send to the AI chatbot, prompts are sent through a route to the chatbot component, and responses are being rendered.

4.2. Chatbot Component

The Chatbot Component is an external intelligence service that is supported by OpenAI API to provide supportive, dynamic feedback to the users based on their daily check-ins, progress, and previous history. Within the system architecture, this component is responsible for receiving user prompts, processes them, and returns a response.



Public Interface Description:

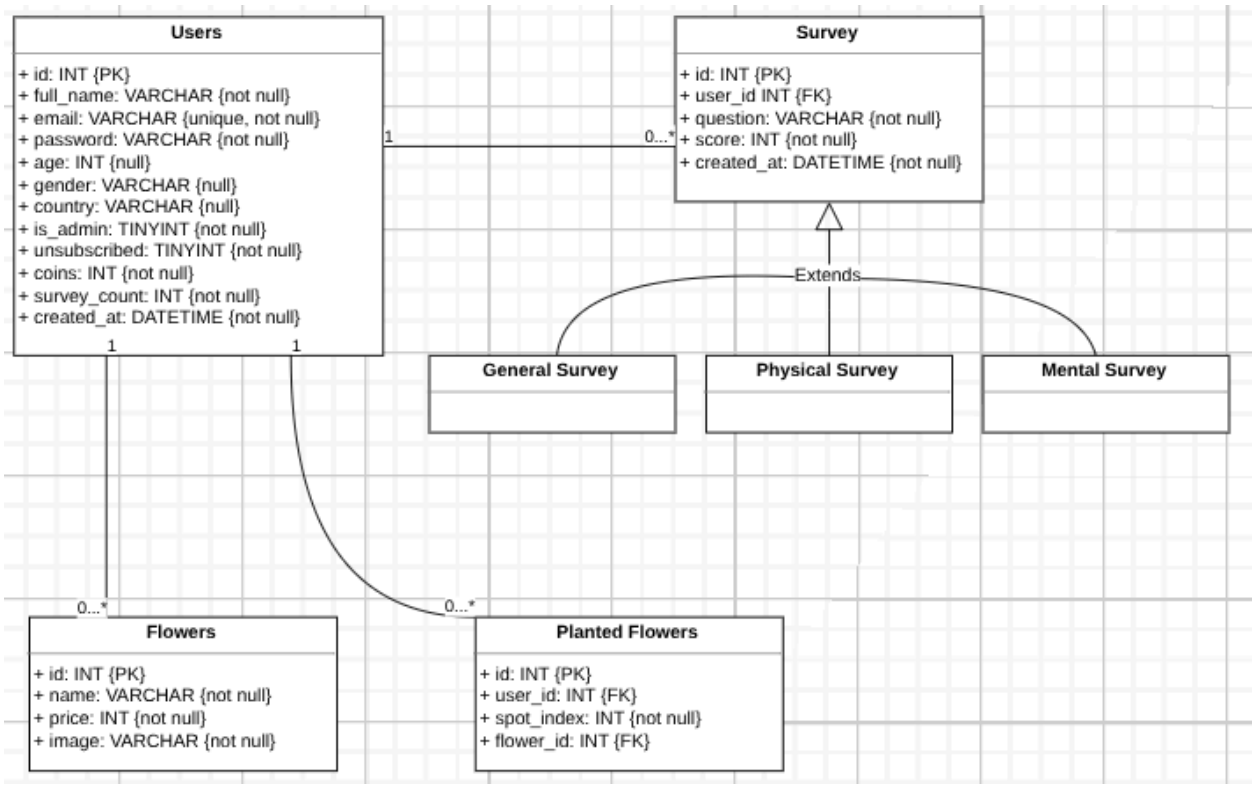
Services Provided: The chatbot serves as each user's personal wellness coach, providing feedback in response to user input.

Inputs: The API connector sends a POST request with the users input data.

Outputs: The response is returned, which is then passed back through the API connector and is displayed through the chatbot message window.

4.3. Database Component

The Database Component is responsible for data management, involving storage, organization, and retrieval of user data. The database keeps records of user information, user survey responses, progress history, point accumulation, and reward elements. It provides a reliable interface for the backend to perform CRUD operations. The module uses MySQL and AWS infrastructure to handle large amounts of data, as well as ensuring data integrity, security, and scalability.



Public Interface Description:

Services Provided: The database interacts with the system through a data access layer managed by the backend.

Inputs:

- The authentication module receives new user credentials through Sign-In to create an account.
- A user finishes completing a survey, the Submit button is clicked, and the user submissions are saved.

Outputs:

- User credentials are fetched by Sign-In or Login to verify the user.
- Historical user data is rendered by Progress Tracking for representation of user trends through graphs.

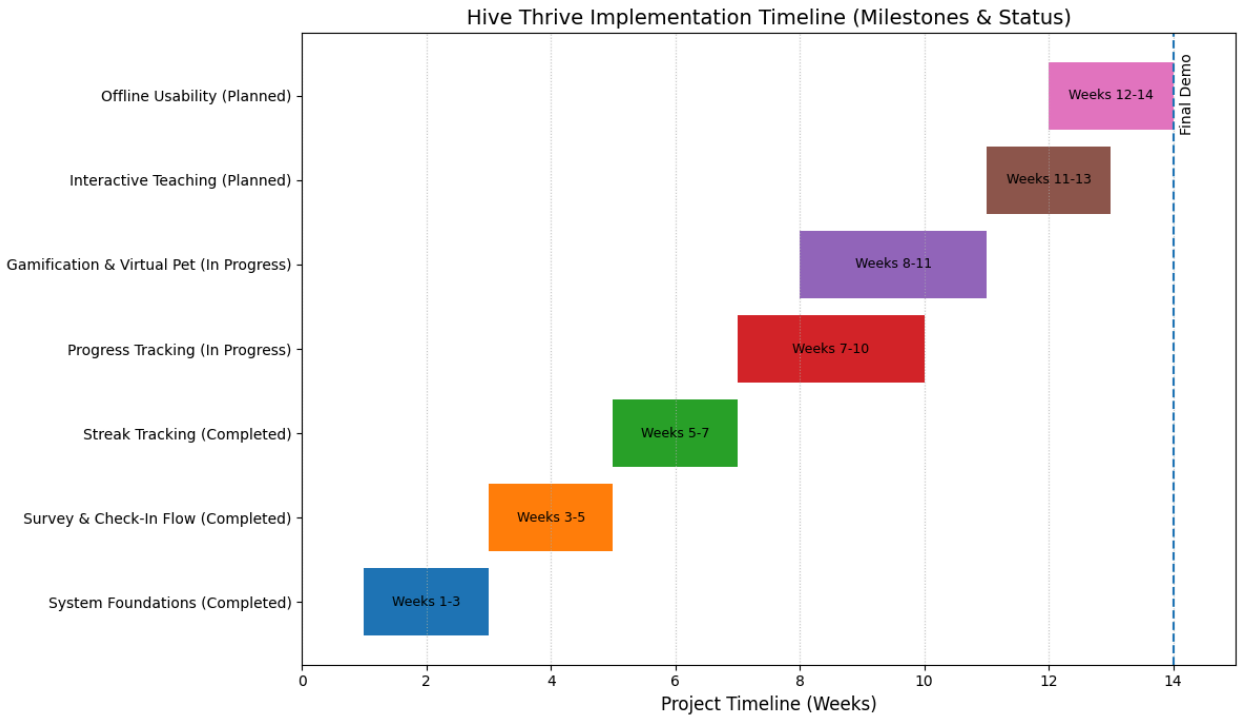
5. Implementation Plan

Chronological Implementation Overview

Hive Thrive is implemented using an incremental, system-oriented approach that prioritizes end-to-end feature slices. Development progresses in structured milestones, each delivering fully functional vertical slices spanning UI, backend logic, and database persistence. This ensures the system remains deployable and demo-ready throughout the semester.

Implementation proceeds in chronological order as follows:

Milestone Timeline



Milestone Breakdown

Milestone 1 – System Foundations

Status: Completed

- Repository setup
- Branching workflow defined
- Environment configuration
- Database schema + migrations
- Express server structure

This milestone established the architectural backbone required for all dependent features.

Milestone 2 – Survey & Check-In Flow

Status: Completed

- Icon-based survey UI
- Data persistence to MySQL
- Input validation
- Confirmation feedback

This milestone created the primary engagement loop and validated end-to-end data flow.

Milestone 3 – Streak Tracking System

Status: Completed

- Streak increment/reset logic
- Edge-case handling
- Persistent storage
- UI badge integration

This reinforced engagement and introduced behavioral continuity mechanics.

Milestone 4 – Progress Tracking & Feedback

Status: In Progress

- Historical aggregation queries
- Trend visualization
- Rule-based feedback engine
- Dashboard metrics

This milestone transforms raw data into actionable user insight.

Milestone 5 – Gamification & Virtual Pet

Status: In Progress

- Pet state mapping to streak data
- Visual state changes
- Reinforcement feedback loops

This layer deepens engagement without altering core logic.

Milestone 6 – Interactive Teaching (Rules-Based Coaching)

Status: Planned

- Conditional logic-based coaching
- Resource linking
- Non-clinical guidance generation

AI-assisted coaching remains optional and will only be added if stability allows.

Milestone 7 – Offline Usability Layer

Status: Planned

- Local storage queue
- Sync-on-reconnect
- UI sync indicators
- Failure-state handling

This milestone enhances reliability but is not required for MVP functionality.

Parallel Development Structure

After Milestone 2 stabilized the engagement loop, work proceeded in parallel:

Backend Stream

- Streak logic
- Feedback engine
- Data aggregation

Frontend Stream

- Dashboard visualization
- Virtual pet UI
- Survey UX refinements

Reliability Stream

- Testing
- Edge-case validation
- Demo preparation

Parallelism is dependency-aware: foundational architecture precedes concurrency, minimizing merge conflicts and architectural drift.

Technical Risks, Trade-offs, and Mitigation

1. Offline Data Integrity

Risk Meaning:
Offline check-ins introduce state synchronization complexity. Data conflicts, duplicate submissions, or failed merges could corrupt streak calculations or progress metrics.

Mitigation Strategy:

- Local write-queuing with timestamp validation
- Server-side duplicate detection
- Explicit sync confirmation indicators
- Controlled offline testing scenarios before release

2. Scope Creep

Risk Meaning:
Feature expansion (AI coaching, mobile wrapper, advanced gamification) could dilute focus from core engagement functionality and threaten delivery timelines.

Mitigation Strategy:

- Strict milestone prioritization
- AI treated as enhancement, not dependency
- Core engagement loop required before expansion
- Demo-readiness checkpoints before adding features

3. AI Reliability and Output Variability

Risk Meaning:
AI-generated coaching can produce inconsistent, inappropriate, or non-actionable responses, which introduces reputational and usability risk.

Mitigation Strategy:

- Default to deterministic rules-based coaching
- AI outputs gated behind structured prompts
- Clear disclaimers and non-clinical framing
- AI only enabled after core system stability

4. System Stability Under Growth

Risk Meaning:
As features stack (streak logic, pet state, progress trends), coupling between modules could introduce regression bugs or data inconsistencies.

Mitigation Strategy:

- Modular route separation
- Migration-based database updates
- Feature-branch merge discipline
- End-to-end acceptance testing before integration

6. Conclusion

This Software Design Document outlines the structure and direction for extending Me Balanced platform in a way that is realistic, scalable, and aligned with our project goals. It captures the major architectural decisions, component responsibilities, and implementation approach that guide how the system will be built and evolved throughout development.

Our design focuses on strengthening the existing system rather than rebuilding it. Core features such as daily check-ins, streak tracking, progress feedback, gamification, and AI-supported interactions are designed to work within the current platform while improving user engagement and overall usability. By clearly separating frontend interaction, backend logic, and data storage, the system remains easier to maintain, easier to test, and more flexible for future updates.

We also designed the system with implementation in mind from the start. Development is planned around complete, functional features that can be tested and demonstrated as they are built. This helps reduce integration issues and keeps progress visible throughout each development cycle. More complex features, especially those involving AI, are approached carefully so they enhance the experience without compromising safety, stability, or project scope.

Overall, this design provides a clear and achievable path toward completing the project successfully. It balances our technical goals with real constraints like time, complexity, and reliability, while still allowing room for future growth. As development continues, this document will be updated to reflect the system as it is built and to serve as a useful reference for future contributors working on Me Balanced.