Software Requirements Specification

for

AquaFocus

Version 1.0

Prepared by

Group Name: 4

|  |  |  |
| --- | --- | --- |
| Harun Işık | 2200004835 | 2200004835@stu.iku.edu.tr |
| Hamza Ali Polat | 2200004289 | 2200004289@stu.iku.edu.tr |
| Mehmet Köşetaş | 2100005201 | 2100005201@stu.iku.edu.tr |
| Mert Özcan | 2200004784 | 2200004784@stu.iku.edu.tr |
| Ahmet Furkan Güngör | 2200004766 | 2200004766@stu.iku.edu.tr |
| Yağız Efe Çelik | 2200005512 | 2200005512@stu.iku.edu.tr |
| Can Okutay | 2200004472 | 2200004472@stu.iku.edu.tr |
| Kerem Yılmaz | 2100003979 | 2100003979@stu.iku.edu.tr |

|  |  |
| --- | --- |
| Instructor: | Akhan Akbulut |
| Course: | COM 6064 – Software Engineering |
| Lab Section: | *A* |
| Teaching Assistant: | *Büşra Kocaçınar* |
| Date: | 11.05.2025 |

Contents

Contents ii

Revisions iii

1 Introduction 1

1.1 Project Purpose and Scope, and Objectives 1

1.2 Roles and responsibilities 1

1.3 Technical Assumptions and Constraints 1

1.4 Naming Conventions 1

2 Requirements 2

2.1 Functional Requirements 2

2.2 Non Functional Requirements 2

3 Other Requirements 3

4 System Architecture and Architectural Design 4

4.1 Logical View 4

4.2 Deployment View 4

4.3 Use Case View 4

5 Design and Implementation 5

6 Other Supporting Information 6

7 References 7

Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
| --- | --- | --- | --- |
| 1.0 | Hamza Ali Polat | Initial and final version. | 11/05/2025 |

# Introduction

## Project Purpose and Scope, and Objectives

**AquaFocus** is a gamified mobile productivity application that aims to improve focus, motivation, and time management skills of its users by combining the scientifically proven **Pomodoro Technique** with **real-time collaboration**, **visual feedback**, and **AI-driven personalized recommendations**.

The **purpose** of AquaFocus is to address the limitations of conventional Pomodoro timers, which are typically designed for solo use and lack engaging or adaptive features. AquaFocus enriches the user experience by allowing users to work together in synchronized teams (called "reefs"), track their progress visually via a growing underwater ecosystem, and receive tailored productivity advice generated by an AI engine.

The **scope** of this software includes:

* A cross-platform mobile application developed using **Flutter**, compatible with both Android and iOS.
* Backend services implemented with **Node.js** and **Express.js**, providing RESTful APIs and real-time synchronization via **Socket.IO**.
* A separate **FastAPI-based microservice** that analyzes user behavior and delivers motivational tips.
* A **MySQL database** supporting all operations including user management, team structures, session logging, and achievements.
* A fully interactive **coral reef visualization**, representing user and team progress based on Pomodoro session completion.
* Achievement unlocking and real-time feedback mechanisms that reinforce user engagement.

The main **objectives** of AquaFocus are:

* To increase individual and team focus through structured Pomodoro cycles enhanced by gamification.
* To allow real-time team collaboration through synchronized session timers.
* To provide actionable insights using AI-generated feedback based on session data and focus patterns.
* To motivate users with a visually evolving reef environment that reflects progress and milestones.
* To promote long-term habit formation by making productivity feel rewarding and social.

## Roles and responsibilities

**Hamza Ali Polat - Backend Developer**

Responsible for implementing and managing the backend architecture of the application. Developed RESTful API endpoints for user authentication, team (reef) operations, session tracking, and achievement management. Ensured secure data transactions using JWT authentication and integrated real-time synchronization using Socket.IO. Contributed to database design and maintained consistency across frontend-backend communication

**Mehmet Köşetaş – AI & Analytics Developer:**

Responsible for designing and implementing the AI-based productivity analysis module. Developed algorithms to compute user-specific focus metrics. Applied K-Means clustering to categorize user behavior patterns and generated personalized study recommendations. Integrated Gemini API to provide AI-powered guidance based on user data and contributed to the visualization of these insights within the app interface.

**Mert Özcan:**  
 Developed the MoreStatsScreen in Flutter using fl\_chart, implemented dynamic daily and weekly statistics visualizations, integrated backend API data into interactive graphs, and contributed to the preparation of the Software Requirements Specification (SRS) documentation.

**Harun Işık - Flutter Frontend Developer:**

**Responsibilities:**

As the primary Flutter Frontend Developer, Harun Işık was responsible for the complete design and implementation of the mobile user interface. This included:

* Developing all core UI components and screens (e.g., Statistics, Tasks, Growth, Survey)
* Integrating real-time communication using Socket.IO on the frontend
* Connecting frontend interfaces with backend REST APIs and AI microservices
* Ensuring dynamic state updates and data synchronization across modules
* Designing and maintaining responsive layouts and animations (coral reef, marine creatures)
* Managing session flow, input validation, and reactive architecture
* Collaborating with backend teams to align data models and socket event structures

His contributions ensured a seamless and interactive user experience across all modules of AquaFocus.

**Kerem Yılmaz - Cloud-Base MySQL Server Management**

* Designed a scalable and normalized MySQL database schema for managing users, teams, Pomodoro sessions, tasks, achievements, and gamified reef systems.
* Implemented entity relationships with proper use of foreign keys, cascading rules, and ENUM types to ensure data consistency.
* Deployed the database on a cloud server with automated backups, monitoring, and basic security configurations.
* Worked closely with backend developers to ensure seamless integration between application logic and database operations. Provided API-level support and contributed to query optimization for efficient data retrieval.

**Can Okutay – Frontend Developer:**

* Test scenarios were developed to validate the functionality of the achievements screen, including tab rendering, creature display, and details dialog operations.
* Real-time connection functionality was implemented using socket.io to enable dynamic interactions.
* Test failures concerning UI rendering, tap interactions, and bottom navigation bar functionality were addressed and fixed.
* Improvements were made to enhance the reliability of test cases, particularly in handling UI interactions.
* Optimization work was conducted to reduce frame skipping and improve performance.
* The reef was enhanced to improve its visual appeal and user experience.
* The achievements and rewards screen code was updated to improve its core functionality.

**Yağız Efe Çelik - Backend Developer:**  
  
As one of the Backend Developers, my primary responsibilities include designing and implementing the real-time server-side architecture using Node.js and Socket.IO. I am responsible for managing multi-user synchronization within shared reefs, optimizing data flow to ensure low latency, and maintaining consistent state across all connected clients. My role also involves integrating AI-powered productivity insights through OpenAI's API, optimizing session data storage, and enhancing the dashboard for real-time visibility. Additionally, I ensure robust error handling, perform quality assurance testing for all critical events, and contribute to future scalability planning.

* Design and maintain the real-time architecture of the application.
* Implement event-based communication using Socket.IO for real-time updates.
* Create and optimize endpoints (/api/reef/:reefId, /api/rooms, /api/rooms/delete/:reefId) for CRUD operations.
* Refactor and optimize session I/O operations to reduce latency and disk usage.
* Manage error handling across all events and ensure stability under load.
* Ensure real-time syncing between multiple users in the same reef as well as the state management.
* Optimize data flow to reflect instant updates in the dashboard.
* Implement consistency checks to prevent data mismatches.
* Integrate AI-powered productivity insights using the OpenAI API.
* Write test scenarios for critical socket events: joinReef, startPomodoro, endPomodoro, etc.

**Ahmet Furkan Güngör - Artificial Intelligence and Analytics Developer:**

In the Aqua Focus project, I contributed to the development of an AI-driven system that provides personalized productivity feedback. I was responsible for writing creative hints based on user data such as Pomodoro sessions, break times, and productivity scores. These hints were used to generate gamified, motivational suggestions. I also helped filter and format user data for the backend API. On the frontend, I implemented features like dark mode, button sound feedback, and personalized exit messages, and solved navigation issues to improve app stability. This work helped create a more personalized and interactive user experience.

## Technical Assumptions and Constraints

The development and deployment of the AquaFocus application are subject to the following technical assumptions and constraints:

* **Supported Platforms:**  
   The application is developed using **Flutter**, ensuring compatibility with both **Android** and **iOS** mobile devices.
* **Backend Technologies:**

The backend of AquaFocus is built using **Node.js** and **Express.js**, providing a lightweight and asynchronous runtime environment ideal for handling concurrent user interactions in real time. The backend exposes multiple **RESTful API endpoints** responsible for:

* User registration, login, and JWT-based authentication
* Team (reef) creation, joining, and member management
* Pomodoro session initiation, logging, and historical retrieval
* Achievement unlocking and progress tracking
* Communication with the AI microservice for personalized feedback

To enable real-time functionality, the system utilizes **Socket.IO**, which establishes WebSocket connections between clients and the server. This allows instant broadcasting of key events such as:

* startPomodoro and endPomodoro: trigger and synchronize timers across team members
* reefUpdated: notifies all team members when a reef grows due to a completed session
* aiInsights: pushes motivational tips directly tob users after session completion

The backend is structured into modular layers:

* **Route handlers** manage endpoint logic and request validation.
* **Controllers** delegate core operations and orchestrate services.
* **Service modules** interface with the MySQL database using parameterized queries to ensure security and efficiency.

Each API call and socket event is protected using **JWT tokens** to maintain session integrity and prevent unauthorized access. The backend also performs error handling, connection checks, and emits debugging logs for development traceability.

* **AI Feedback Service:**  
   A separate **FastAPI-based Python microservice** is used to process session data and generate motivational study tips using a rule-based engine and clustering logic.
* **Google Gemini API:**Content-generation endpoint invoked via HTTPS POST; requires API key management and rate-limit handling.
* **Prompt Engineering:** Dynamic prompt construction assumes stable JSON response from /stats endpoint with defined fields.
* **Database:**  
   The backend is powered by a **MySQL** database containing **17 normalized tables**, managing users, teams, sessions, achievements, and reef data.
* **Security Constraints:**
  + All requests require **JWT-based authentication** for session integrity.
  + Sensitive operations are protected against common threats such as **SQL Injection** and **unauthorized socket events**.
* **Deployment Assumptions:**
  + The backend and AI microservice are hosted on local or cloud-based Linux servers with internet access.
  + The application assumes stable internet connectivity for real-time synchronization features.
* **Tooling and Libraries:**
  + Development was carried out using **Visual Studio Code**, **Postman** (API testing), **MySQL Workbench**, and **Lucidchart** (for diagramming).
  + All frameworks and tools used are **open-source**.

## Naming Conventions

To ensure consistency, readability, and maintainability throughout the development process, the following naming conventions and standards have been adopted across the AquaFocus project:

#### **API Naming Conventions**

The AquaFocus backend follows **RESTful architectural principles**, and all endpoint URLs are named using the **snake\_case** convention to ensure consistency and readability.

This naming style was selected to:

* Maintain uniformity across all API routes
* Improve readability for developers working in multiple environments (e.g., frontend + backend)
* Prevent confusion with case sensitivity in some database or URL parsers

Each endpoint uses an **action-oriented and descriptive structure**, typically following the format: “HTTP\_METHOD /resource\_action”

**Examples:**

* POST /users/register – Registers a new user
* POST /users/login – Logins current user
* POST /users/pomodoro/complete – Completes a Pomodoro session
* GET /users/reef/:user\_id – Fetches the reef’s current visual growth state

All endpoints are versioned and prefixed under a base path (e.g., /api/v1/) to support future expansion and maintain backward compatibility.

#### **Code Naming Conventions**

* **JavaScript (Node.js/Express):**
  + Variable and function names use **camelCase**.  
     Example: getSessionData(), userId
  + Constants are written in **UPPER\_SNAKE\_CASE**.  
     Example: SESSION\_TIMEOUT = 1500
* **Python (FastAPI):**
  + Variable and method names use **snake\_case**.  
     Example: generate\_recommendation()
* **Flutter (Dart):**
  + Class names use **PascalCase**.  
     Example: PomodoroTimerWidget
  + Widget file names use **lower\_snake\_case.dart**.  
     Example: home\_screen.dart, session\_detail\_card.dart

#### **Database Naming Conventions**

* Table names use **plural snake\_case**.  
   Example: users, pomodoro\_sessions, reef\_creatures
* Column names use **snake\_case**.  
   Example: user\_id, session\_start, focus\_level

#### **File and Directory Structure**

* All directories are named in **lower\_snake\_case**.  
   Example: /models, /services, /screens
* Assets like images and icons are placed under /assets/ and referenced using relative paths.

# Requirements

## Functional Requirements

### **Functional Requirements**

AquaFocus system shall provide the following functional capabilities:

#### **User Management**

* The system shall allow new users to register by providing a **unique username**, **email address**, and **password**. Input validation shall be enforced on the frontend and backend to prevent empty fields, invalid formats, or duplicate entries.
* The system shall securely store user passwords using **bcrypt hashing** before saving them into the MySQL database.
* The system shall allow registered users to log in using their email and password credentials. Upon successful authentication, a **JSON Web Token (JWT)** shall be generated and returned to the client for subsequent requests.
* The system shall require JWT tokens to be included in the headers of all protected API requests. These tokens shall be validated on the server to ensure:
* The user’s identity is authentic
* The token has not expired
* The user has access to the requested resource
* The system shall include proper **error messaging** for invalid login attempts, missing fields, or expired tokens to guide user recovery.
* The system shall log all authentication events (e.g., login success/failure, token issues) for auditing and debugging purposes.

#### **Team (Reef) Management**

* The system shall allow users to **create a new team**, referred to as a “reef”, by entering a team name. Upon creation, the system shall generate a **globally unique reef ID (UUID)** which can be shared with others to join the same team.
* The system shall allow existing users to **join an existing reef** by entering a valid reef code (UUID). The backend shall verify the validity of the reef code before adding the user to the corresponding team.
* Each reef shall maintain a list of its members. When a user joins a reef, their user\_id shall be associated with the reef in the team\_members table of the database.
* The system shall allow users to view the current **team composition**, including:
* Active members in real time
* The reef’s current progress (e.g., number of completed sessions)
* The coral reef’s growth state based on team performance
* Real-time updates about reef activity (e.g., new member joins, session started, reef growth) shall be **broadcast using Socket.IO events** to all connected clients within the same reef.
* The system shall prevent users from joining multiple reefs at the same time. If a user attempts to join a new reef while already in another, the system shall prompt for confirmation to leave the current team.
* Reef creation and membership shall be **persisted in the database**, ensuring consistency across sessions and devices.

#### **Pomodoro Session Management**

* The system shall allow users to **initiate a Pomodoro session**, either individually or as part of a synchronized team. Users within the same reef can choose to work collaboratively, in which case the session will begin simultaneously for all active members.
* When a synchronized session is started by any team member, the backend shall emit a **startPomodoro Socket.IO event** to all connected users within that reef. Likewise, at the end of the session, an **endPomodoro event** shall be broadcast.
* The session duration shall follow the default Pomodoro configuration (25 minutes of focus, followed by a 5-minute break), though these values may be adjusted in future iterations.
* Each session shall be logged in the backend by inserting a record into the pomodoro\_sessions table. The stored metadata shall include:

session\_id (UUID), user\_id (UUID), reef\_id (UUID), session\_start and session\_end timestamps, focus\_level (1–5 self-reported scale), was\_distracted (boolean), Optional: task reference or session note

* Users shall be prompted to complete a **post-session survey** upon the end of each Pomodoro. Their responses shall be linked to the session and later processed by the AI module.
* The system shall ensure that **only one active session** can run per user at any given time. If a new session is initiated while another is ongoing, the system shall reject the request with an appropriate error message.
* All session events and timestamps shall be recorded in UTC to ensure **cross-timezone consistency**, especially for teams working globally.

#### **Gamification and Rewards**

* The system shall visualize team progress using a dynamically growing reef interface.
* The system shall unlock animated sea creatures (e.g., Clownfish, Octopus) based on cumulative Pomodoro completions.
* The system shall broadcast reef growth updates via Socket.IO events.

#### **AI Feedback and Session Reflection**

* After each Pomodoro session, the system shall prompt users with a self-assessment survey.
* The system shall send session data to the AI microservice and receive personalized tips.
* The system shall display AI-generated feedback on the post-session screen.

##### **FR-STAT-001: Fetch Daily Focus Time**

The system shall calculate the total "focus time" for the current day by summing session\_duration of all pomodoro\_sessions whose session\_start date equals today, and return it as a string in minutes.

##### **FR-STAT-002: Compute Aggregate Productivity Metrics**

The system shall compute per-user statistics including:

Average session duration, Average break duration ,Average productivity score, Average focus level, Completed Pomodoro count, Aggregated values are exposed via the Stats API.

##### **FR-STAT-003: Weekly Progress**

The system shall compute normalized average session durations for each weekday, map them to labels (M, T, W, T2, F, S, S2), and return a dictionary of float percentages.

##### **FR-STAT-004: Total Tasks Completed**

The system shall query the user\_tasks table, count entries with status="completed" for the user, and include the count in the StatsResponse.

##### **FR-STAT-005: AI-Generated Study Report**

The system shall generate a list of human-friendly recommendations based on clustering of user metrics (KMeans clusters) and thresholds for session duration, break duration, productivity score, and focus level.

##### **FR-UI-001: Stats Screen Integration**

The mobile app shall fetch StatsResponse from "/stats" endpoint and display:

Today's Focus Time, Completed Pomodoros, Focus Rate (avg\_focus\_level × 10)%, Weekly Progress chart, Performance Metrics (Avg Session, Avg Break, Productivity Score, Focus Level), Total Tasks Completed, Study Report section

#### **Statistics and History**

* The system shall track and display daily, weekly, and total focus durations.
* The system shall visualize productivity data using charts and diagrams.
* The system shall allow users to access historical Pomodoro session records.

#### **AI Explanation Page**

##### **FR-UI-002: AI Explanation Page**

The application shall implement an AiExplanationPage that:

Fetches current stats via the /stats API

Sends a structured prompt to Gemini API for personalized actionable suggestions

Subscribes to SocketService for real-time AI insights fallback

Displays up to 5 motivational recommendations

## Non-Functional Requirements

### Performance Requirements

* The system shall initiate a Pomodoro session within **2 seconds** of user input.
* Any API request shall receive a response within **1000 milliseconds** under normal load.
* Real-time socket events (start/end session, reef updates) shall propagate to all connected users within **500 milliseconds**.
* The AI feedback service shall return a response within **2 seconds** after session data is submitted.
* The system shall support at least **100 concurrent users** in a single reef without performance degradation.
* Reliability Stats API shall handle missing or zero-session days by returning "0 min" without failure. Gemini API errors shall be gracefully handled with user-friendly error messages.
* Performance API endpoints shall respond within 500 ms under typical load. Weekly progress and clustering computations shall complete within 100 ms per request.

### Safety and Security Requirements

To ensure data integrity, user privacy, and system protection against malicious activities, the AquaFocus platform incorporates the following security mechanisms:

* **Authentication and Authorization:**  
   All protected API endpoints shall require a **valid JWT (JSON Web Token)** to be included in the request header. The server shall verify the token's signature, expiration, and user identity before granting access to any sensitive resource. Unauthorized requests without a valid token shall be rejected with proper error messages.
* **Password Storage and Encryption:**  
   All user passwords shall be stored in the database using **bcrypt hashing** with a randomly generated salt. The system shall never store or transmit passwords in plaintext, ensuring protection against database breaches.
* **Real-Time Socket Security:**  
   Socket.IO communication channels shall include user authentication by passing the JWT token during connection handshake. The server shall validate the token before accepting or broadcasting events, preventing **event spoofing**, **session hijacking**, or **unauthorized socket listeners**.
* **SQL Injection Protection:**  
   All database interactions shall use **parameterized queries or ORM abstractions** to sanitize user input. This prevents SQL injection vulnerabilities where attackers might attempt to manipulate or access unauthorized data.
* **Access Control:**  
   Users shall only be able to access their own data or the data of the reef (team) to which they belong. The backend shall enforce **fine-grained access control** on all sensitive resources, ensuring cross-user data isolation.
* **Session Timeout and Token Expiry:**  
   JWT tokens shall have an expiration time to reduce the risk of token theft. The system may implement optional session timeouts for inactivity in future releases.
* **Logging and Monitoring:**  
   All authentication attempts, failed socket connections, and suspicious activities shall be logged for later analysis and potential threat detection.

### Software Quality Attributes

#### **Reliability** AquaFocus is designed to ensure high reliability in both individual and team-based usage scenarios. Socket.IO connections are monitored for disconnections, and automatic reconnection logic ensures continuity in real-time features. API requests are validated on both client and server sides to prevent malformed inputs or runtime failures. Errors are logged for post-mortem analysis.

#### **Portability** AquaFocus shall run seamlessly on both Android and iOS platforms using a shared Flutter codebase. No platform-specific features will compromise cross-platform functionality.

#### Maintainability The backend services are modular and follow a layered architecture (routes, controllers, services), allowing future feature additions or fixes to be applied independently.

#### Usability The user interface is designed using intuitive navigation, minimalist layout, and visual feedback (e.g., reef animation, badges) to enhance engagement. The average user should be able to complete core actions (start session, join team, see progress) with no more than 3 taps.

#### Testability All endpoints and socket events are unit tested using tools like Postman and Mocha. The system logs all session activities in JSON format for easy traceability and testing.

# Other Requirements

#### **Database Requirements**

* The system shall use a **MySQL** relational database.
* The database shall include **17 normalized tables** for managing users, teams, Pomodoro sessions, AI feedback, achievements, and reef data.
* Foreign key constraints shall be applied to maintain referential integrity between tables.

#### **Internationalization**

* Although the primary language is English, all frontend UI texts are stored as constants to support future internationalization (i18n) via localization files.

#### **Legal and Ethical Considerations**

* The application shall comply with **GDPR principles** for user data handling, including the ability to delete accounts and stored session data upon request.
* No personal data will be shared with third parties. All AI processing is performed within the application’s internal microservice infrastructure.
* User data is used only for productivity analysis and is not used for commercial or advertising purposes.

#### **Reusability Objectives**

* The backend API is designed to be **RESTful and stateless**, making it reusable in other platforms such as a future web or desktop version.
* The AI microservice and session analysis engine are implemented as independent modules, which can be ported or replaced without affecting the mobile app.

#### **Accessibility**

* The frontend design follows accessibility best practices such as:
  + High contrast UI elements
  + Text scalability with dynamic font sizing
  + Clear button labeling and color-independent interactions

# System Architecture and Architectural Design

AquaFocus is designed using a modular client-server architecture that separates the user interface, business logic, and data storage across three main layers: **Flutter frontend**, **Node.js backend with RESTful APIs and Socket.IO**, and a **FastAPI-based microservice** for AI-based productivity suggestions. The architecture supports scalability, maintainability, and real-time collaboration.

## Logical View

The system is organized into the following logical components:

* **Frontend (Flutter):**
  + Handles all user interactions
  + Manages Pomodoro timer, team management, reef visualization, and statistics
  + Communicates with backend through HTTP requests and listens to real-time updates via Socket.IO
* **Backend (Node.js + Express + Socket.IO):**

The backend layer is implemented using Node.js with the Express.js framework, and it is responsible for handling both API logic and real-time event management. It provides scalable and efficient access to application resources and coordinates communication between the client, session storage, and AI microservice for productivity insights.

Specifically, the backend is responsible for:  
  
**4.1.1.** Exposing RESTful API Endpoints that Support:

* Reef (Team) Management:
  + Creation, deletion, and modification of reef data.
  + Membership handling for users joining and leaving shared reefs.
* Session Management:
  + Start, stop, and track Pomodoro sessions within reefs.
  + Record session data and synchronize it with real-time updates.
* Achievement Tracking and History Retrieval:
  + Track user progress, total Pomodoros completed, and time spent in focus.
  + Maintain a history of sessions for analytics and insights.
* AI Insights Communication:
  + Interface with OpenAI's API to fetch productivity tips and recommendations.
  + Serve personalized insights to users based on their session history and focus habits.
* API Endpoints:
  + GET /api/rooms — Retrieve all active reefs.
  + GET /api/reef/:reefId — Retrieve a specific reef's data.
  + GET /api/rooms/delete/:reefId — Delete a specific reef.

**4.1.2.** Managing Real-time Events using Socket.IO: Real-time synchronization across multiple connected clients in a shared reef environment:

* reconnectRequest — Restores previous session data for returning users.
* joinReef — Allows users to join a specific reef, updating all members instantly.
* leaveReef — Removes a user from the reef and updates connected clients.
* startPomodoro — Triggers a synchronized countdown for all team members in the reef.
* endPomodoro — Broadcasts session completion and updates user statistics.
* reefUpdated — Reflects collective progress in the coral ecosystem for real-time feedback.
* aiInsights — Sends personalized feedback to users after each session.
* deleteReef — Deletes the specified reef from storage and broadcasts updates.

**4.1.3.** Interfacing with Local Session Storage for Persistence: Data is stored in sessions.json, acting as a local database to track:

* Reef Data and Metadata:
  + Reef names, users, progress, and session history.
* User Statistics and Session Information:
  + Individual Pomodoro counts, total focus time, and active session state.
* Total Duration Tracking:
  + Maintains accumulated focus duration for both users and entire reefs.

**4.1.4.** Error Handling and Session Management:

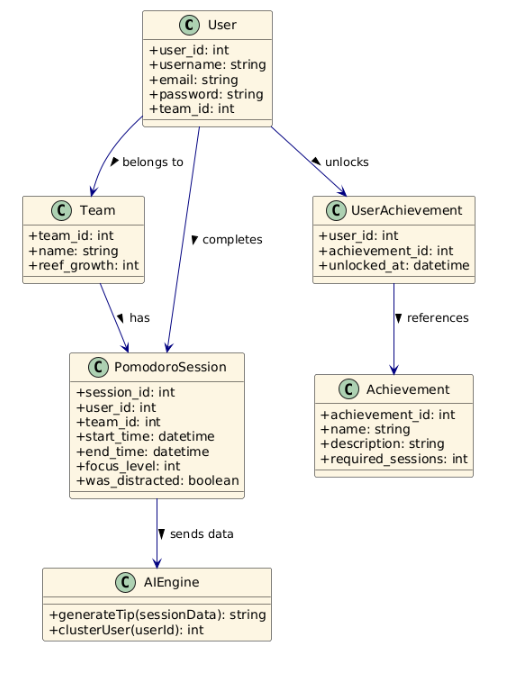
* Error Logging:
  + All socket events are wrapped with centralized error handlers to prevent application crashes and ensure consistent state.
* Session Consistency:
  + When a user joins or leaves a reef, the backend updates all connected clients to reflect the current state.
* Data Integrity:
  + Prevents duplicate user entries and enforces real-time synchronization across clients.

**4.1.5.** AI Microservice Integration:

* Productivity tips are fetched asynchronously from OpenAI's API.
* AI logic is modularized for maintainability and better integration.
* Caching logic prevents redundant API calls during short timeframes.

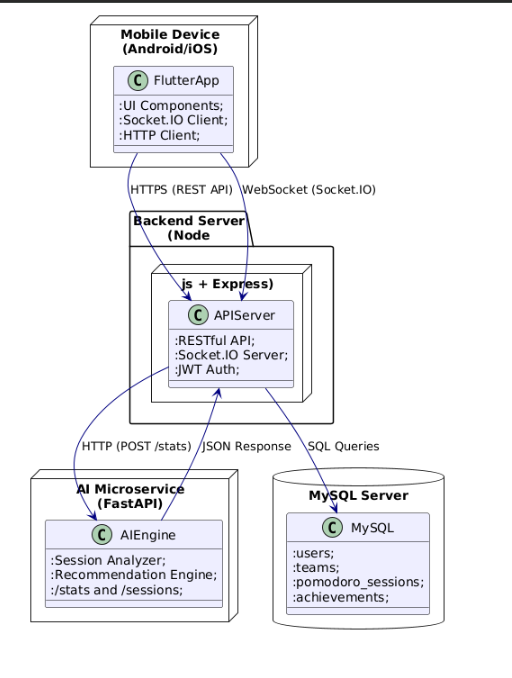
This backend architecture ensures that AquaFocus can operate smoothly under concurrent load, maintain secure stateful communication with multiple users, and scale easily as the user base grows.

* **StatsService (FastAPI Microservice)**
* **Data Access Layer**
  + Leverages SQLAlchemy to connect to the central MySQL database.
  + Fetches and parses both **pomodoro\_sessions** and **user\_tasks**, converting timestamps into native Python datetime objects.
* **Clustering Module**
  + Applies scikit-learn’s K-Means to group users into three performance tiers, based on:
    - Average Pomodoro session duration
    - Average break duration
    - Average productivity score
    - Average focus level
    - Total Pomodoro sessions completed
* **Statistics Endpoint** (GET /stats)
  + **Daily Metrics**: calculates today’s total focus time by summing session durations with a start date matching “today.”
  + **Aggregated KPIs**: returns completed Pomodoros, focus rate (normalized focus level), weekly progress breakdown, and all-time totals (avg session, avg break, productivity score, focus level, task completions).
  + **Recommendations**: alongside raw data, emits a “study\_report”—a concise list of actionable tips derived from cluster assignment and individual metrics.
* **AI Insight Service**
  + **Gemini API Integration**
    - Builds a dynamic prompt from the /stats payload (including focus rate, productivity score, Pomodoro count, etc.).
    - Invokes Google Gemini’s content-generation endpoint to produce 3–5 short, motivational, and practical recommendations.
  + **Real-Time Delivery (Socket.IO)**
    - Subscribes to a dedicated WebSocket channel for “AI insights” events.
    - Emits the freshly generated tips immediately to any connected Flutter client when a Pomodoro session ends or upon explicit request—enabling in-session guidance.
* **Real-Time Delivery (WebSocket)**  
   – Listens on a Socket.IO channel for AI-driven insight requests during live Pomodoro sessions.  
   – Pushes the generated recommendations immediately to the frontend when sessions end or on demand.
* **Database (MySQL):**
  + Stores users, teams, sessions, rewards, and statistics in normalized relational tables
  + Entity relations managed via foreign keys (e.g., user\_id, reef\_id)

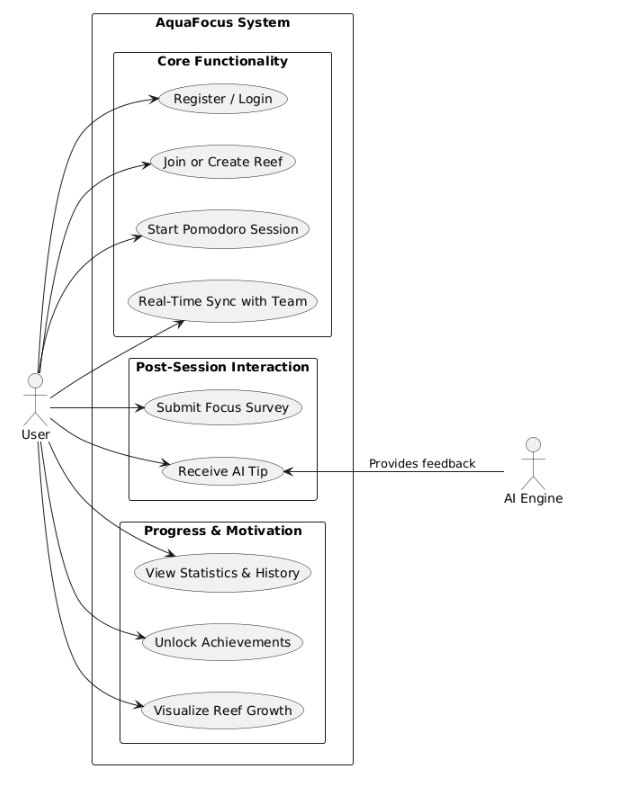


## Deployment View

The system is deployed using a client-server architecture. The **Flutter-based mobile client** interacts with the **Node.js backend server** over HTTPS and WebSockets. The backend handles both RESTful API requests and real-time synchronization via Socket.IO. A separate **FastAPI microservice** is used to provide AI-generated feedback by receiving session metadata from the backend. All persistent data, including users, sessions, and rewards, are stored in a **MySQL database** hosted on a remote server. Each component communicates over secure connections and operates independently for modularity and scalability.



## Use Case View



This diagram illustrates the main use cases of the AquaFocus system, organized into functional clusters: core operations, post-session interactions, and motivation-driven tracking. The primary actor, the **User**, interacts with all modules of the system, while the **AI Engine** acts as a secondary system actor, responsible for processing focus data and returning personalized tips. Each use case corresponds to a distinct functionality offered by the application.

# Design and Implementation

The AquaFocus system is implemented using a modular design that ensures scalability, testability, and separation of concerns. The system consists of three core layers: frontend (Flutter), backend (Node.js + Express), and AI microservice (FastAPI). Each module handles a specific responsibility and communicates through well-defined interfaces.

### **User Interface Design**

The user interface is built using **Flutter**, which provides cross-platform support for both Android and iOS devices. The design emphasizes simplicity, minimalism, and motivation through gamification. Key screens include:

* **Home Screen:** Displays Pomodoro progress, team status, and reef overview.
* **Team Management:** Allows users to join or create a team (reef) with a code.
* **Pomodoro Timer:** Includes start, pause, and session-end features. Works in synchronized mode when in a team.
* **Post-Session Survey:** Lets users submit self-reflection data for AI feedback.
* **Achievements & Reef View:** Displays unlocked rewards and animated reef progress.
* **Statistics Page:** Visualizes historical data (e.g., daily/weekly focus, average session time).

### **Backend Implementation**

The backend of AquaFocus is implemented using **Node.js** and **Express.js**, adopting a fully **RESTful architecture** that ensures modularity, scalability, and maintainability. The backend serves as the central orchestrator of all operations including user management, team coordination, session tracking, and real-time communication.

Core functionalities of the backend include:

* **User authentication and authorization** through secure JWT-based login and registration endpoints.
* **Team (reef) creation and membership management**, where users can create or join teams using unique reef IDs.
* **Pomodoro session handling**, including start/stop synchronization, session metadata logging, and self-reflection input capture.
* **Gamification logic**, such as achievement unlocking and reward tracking based on cumulative Pomodoro completions.
* **Data persistence** using a **MySQL** database, featuring a normalized schema of 17 interconnected tables.
* **Real-time communication** via **Socket.IO**, allowing instantaneous updates across all connected users within the same reef.

The backend is organized using a clean, layered architectural pattern:

* **Routes** handle HTTP endpoints and direct incoming requests.
* **Controllers** process business logic, enforce validation rules, and coordinate with services.
* **Services** manage database queries, often using parameterized SQL to prevent injection attacks.
* **Middlewares** handle JWT authentication, error responses, and reusable request utilities.

Security is enforced throughout the system using best practices such as:

* Input validation at both route and controller levels,
* Token verification on every protected endpoint and socket event,
* Hashed password storage with bcrypt,
* Logging of session and authentication events for debugging and auditing.

This structure ensures that backend logic is cleanly separated, easy to test, and extensible for future modules such as notification systems or external calendar integration.

### **Real-Time Synchronization**

Using **Socket.IO**, real-time Pomodoro session states are synchronized among team members. Events like startPomodoro, endPomodoro, and reefUpdated are emitted and listened to on both client and server side. Each event includes a timestamp, user ID, and reef ID for traceability.

### **AI Microservice**

A separate **FastAPI** Python microservice analyzes session metadata and provides productivity feedback. It uses:

* Pandas for data processing
* K-means clustering for user segmentation
* Rule-based engine for feedback generation

The microservice exposes two endpoints:

* POST /stats for returning full productivity summary
* POST /sessions for raw session data

### **Development Tools & Practices**

* Code Editor: **Visual Studio Code, Android Studio**
* Version Control: **Git + GitHub**
* API Testing: **Postman**
* Database: **MySQL Workbench**
* Diagrams: **Lucidchart**, **PlantUML**
* Team Communication: **Discord**, **Skype, Teams** **GitHub Projects**
* Web Service**: Render**

Code is modular, reusable, and documented. Error handling, socket disconnection recovery, and token validation are implemented for reliability.

# Other Supporting Information

The following information is provided to support the design and development of the AquaFocus application.

### **6.1 Sample Session Log Format (JSON)**

Each Pomodoro session is stored in a structured JSON format as shown below:

{

"user\_id": "4f54b290-4c71-11ee-be56-0242ac120002",

"username": "hamza ali polat",

"reef\_id": "c2a82134-5323-44b0-a6ff-8e0c3469fe1d",

"session\_start": "2025-05-08T14:30:00Z",

"session\_end": "2025-05-08T14:55:00Z",

"focus\_level": 4,

"was\_distracted": false

}

### **Sample Stats Output**

When a session is completed, the **Socket.io** service returns personalized feedback in JSON format:

{

"user\_id": 104,

"tip": "Try to take a 5-minute walk during your break to recharge."

}

### **Post-Session Survey Questions**

Users are asked the following questions after each Pomodoro session:

1. **How focused were you during this session?** (1 to 5 stars)
2. **Were you distracted?** (Yes/No)
3. **Would you like a tip for your next session?** (Yes/No)

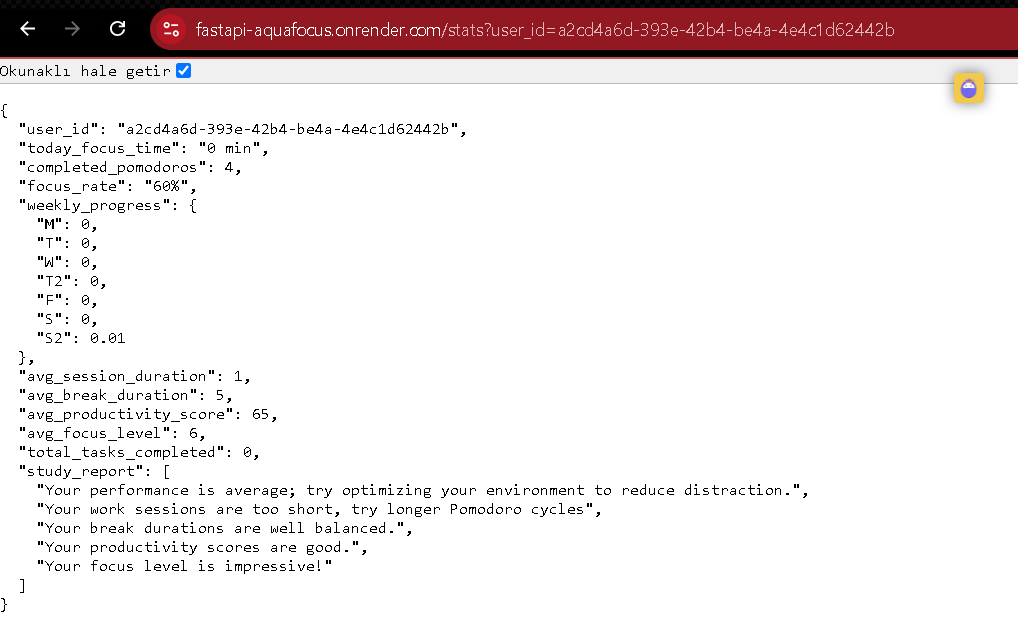
These answers contribute directly to the AI feedback and statistical tracking systems.

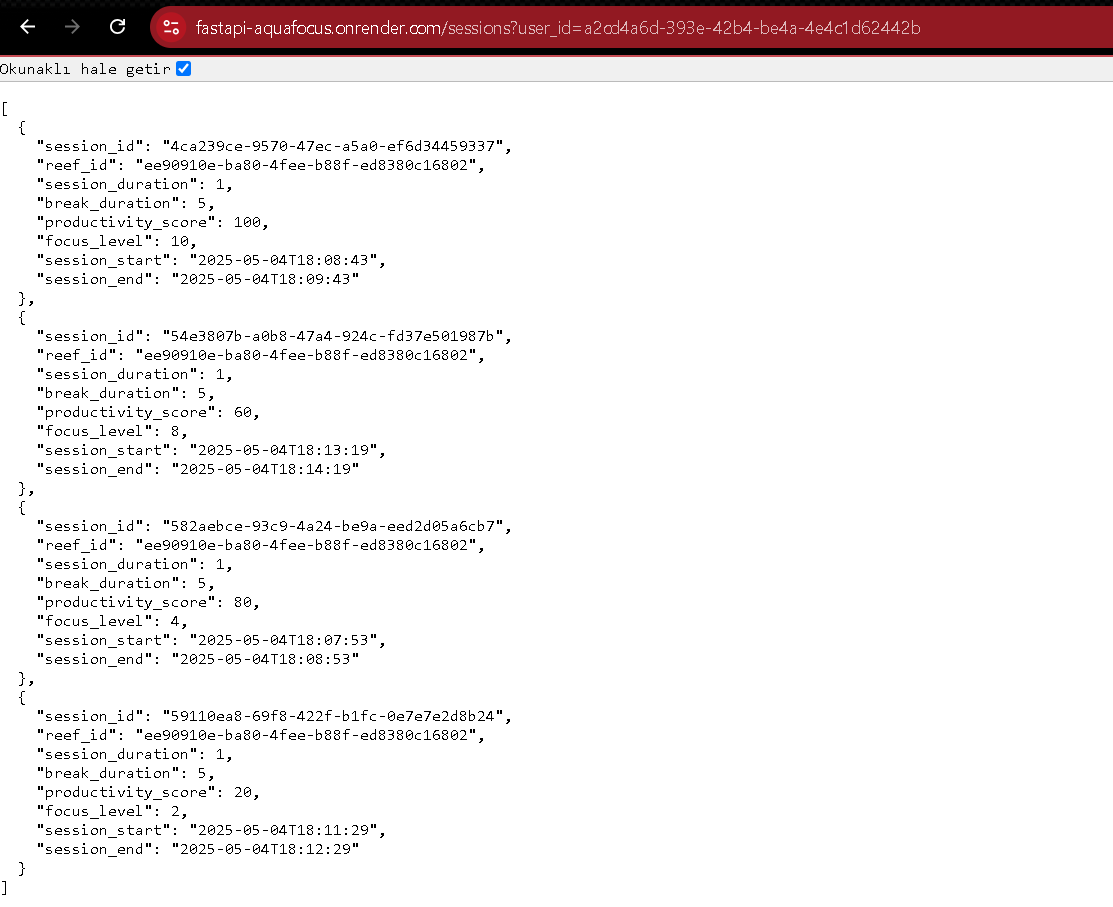
### **Design Considerations**

* The coral reef theme was selected for its symbolic representation of growth, calmness, and collective development.
* All development was completed without a commercial budget, relying entirely on open-source tools and collaborative teamwork.

### **Sample Fast API Responses**

**Stats Endpoint** (GET /stats?user\_id={user\_id}) returns a structured JSON object including metrics and recommendations, for example:

  
  
**Sessions Endpoint** (GET /sessions?user\_id={user\_id}) returns a JSON array of session objects, for example:



# References

< Identify each document by title, report number - if applicable - date, and publishing organization.

Specify the sources from which the references can be obtained.>

[1] F. Cirillo, *The Pomodoro Technique*, FC Garage, 2006.

[2] Forest App, *Stay Focused, Be Present*. [Online]. Available: [https://www.forestapp.cc](https://www.forestapp.cc/)

[3] Focus To-Do, *Pomodoro Timer and Task Manager*. [Online]. Available: [https://www.focustodo.cn](https://www.focustodo.cn/)

[4] OpenAI, *GPT Models and API*. [Online]. Available: [https://platform.openai.com](https://platform.openai.com/)

[5] W. Elmasry, *Flutter Development Guidance*, Istanbul Kültür University, 2025.

[6] Flutter Dev, *Build Apps for Any Screen*, Google. [Online]. Available: [https://flutter.dev](https://flutter.dev/)

[7] Socket.IO, *Realtime Application Framework (Node.js)*. [Online]. Available: [https://socket.io](https://socket.io/)

[8] FastAPI, *Modern, Fast (High-Performance), Web Framework for Building APIs*. [Online]. Available: [https://fastapi.tiangolo.com](https://fastapi.tiangolo.com/)

[9] Xwavesoft, *Be Focused App*. [Online]. Available: <https://xwavesoft.com/be-focused.html>

[10] Pomotodo App, *Pomodoro + Todo*. [Online]. Available: [https://pomotodo.com](https://pomotodo.com/)

[11] Focus Booster App, *Boost Your Productivity*. [Online]. Available: [https://www.focusboosterapp.com](https://www.focusboosterapp.com/)

[12] Tide App, *Focus, Sleep, Meditate*. [Online]. Available: [https://tide.fm](https://tide.fm/)

[13] R. E. Mayer, *The Cambridge Handbook of Multimedia Learning*, Cambridge University Press, 2005.

[14] M. Csikszentmihalyi, *Flow: The Psychology of Optimal Experience*, Harper and Row, 1990.

[15] S. McDaniel and K. Einstein, *Gamification in Education: What, How, Why Bother?*, Academic Exchange Quarterly, vol. 15, no. 2, 2011.

[16] Flutter Documentation: https://flutter.dev