Quack Time - Productivity Assistant
Software Design Specifications

Neal Kimchi (nk), Nikhar Ramlakhan (nr), Areyan Rastawan (ar), Abie Safdie (as)

6-03-2024 - v1.05

Table of Contents

1. SDS Revision History

Date	Author	Description
5-18-2024	nr	Created the initial document.
5-18-2024	ar	Completed the initial document.
5-19-2024	team	Created static and dynamic diagrams.
6-03-2024	nr	Added server-side section.
6-03-2024	nr, ar	Update diagrams to reflect developed software.
6-04-2024	team	Complete the final document

2. System Overview

The system, named Quack Time, aims to provide students with tools to enhance their productivity experience. Components for helping the student develop healthy work habits, while also measuring the student's productivity for their feedback, as well as a server component for storing student data.

3. Software Architecture

The software architecture of Quack Time is designed to promote efficient interaction between students and their tasks. It ensures that students are adopting healthy study techniques while tracking their progress to provide accurate feedback on their work. The architecture is composed of several key components, each serving a specific role in the system.

1. Set of Components:
- **User Login Interface:** Authenticates users and associates their activities with the correct user profile, ensuring secure access by verifying credentials and managing session data to provide a personalized experience.
- **Task Board:** Manages and displays a comprehensive list of tasks assigned to students, enabling them to view, organize, and prioritize their tasks effectively, including adding, deleting, and updating tasks.
- **Timer:** Implements the Pomodoro technique to enhance focus and efficiency, segmenting study sessions into intervals with regular breaks to optimize mental agility and sustained attention.
- **ProDUCKtivity Log:** Records and analyzes students' productivity ratings, providing a historical view of productivity patterns to track progress and identify areas for improvement, and offers analytical tools to highlight trends.
- **Data Storage Server:** Ensures the secure storage and retrieval of all user-related data, maintaining integrity and availability of user data, including task lists and productivity logs, and facilitates data synchronization and backup across various devices.

2. Functionality Provided by Each Component:
- **User Login Interface**: Acts as the gateway for all users, ensuring identity verification and a personalized experience by securely managing session data.
- **Task Board:** Functions as the central user interface, displaying a list of tasks and enabling efficient task management, which is crucial for organizing daily student activities and tracking progress.
- **Timer:** Supports the Pomodoro study method, optimizing focus and efficiency by dividing study time into intervals followed by short breaks, helping maintain concentration and assess productivity at the end of each interval.
- **ProDUCKtivity Log:** Collects and displays productivity ratings at the end of each study session, allowing users to input their productivity scores and providing detailed records of study sessions for trend analysis and performance optimization.

- **Data Storage Server:** Manages secure storage and retrieval of all application data, handling user authentication and maintaining data integrity, and ensures seamless communication across client-side components, making data consistent and accessible across user sessions.

3. Interaction Between Modules:
   - **User Login Interface:** Interacts with the Data Storage Server to authenticate user credentials and retrieve user-specific data, ensuring personalized and secure sessions.
   - **Task Board:** Continuously connects with the Data Storage Server to fetch and update user-specific tasks, ensuring current and synchronized task data across sessions, and integrates with the Timer to adjust task durations based on real-time user feedback.
   - **Timer:** Links with the Task Board to provide precise timing functionalities that influence task durations, receiving updates on task commencement and completion and adjusting tracking time accordingly, and communicates with the Data Storage Server to log time data for each task, ensuring accurate recording and storage of time spent on tasks.
   - **ProDUCKtivity Log:** Interacts with the Task Board and Timer to retrieve task details and associated time logs, compiling comprehensive records of study sessions to generate productivity trends and performance analyses.

4. Rationale for the Architectural Design:
   - The architectural design promotes modularity, scalability, and maintainability by decomposing the system into separate components, each responsible for a specific aspect of functionality, ensuring changes or updates to one component do not affect others.
   - The use of a server component allows for centralized data storage and management, facilitating collaboration and synchronization among multiple users. This architecture provides a solid foundation for developing a robust and feature-rich productivity assistant.

4. Software Modules

4.1. User Login Interface

Role and Primary Function

The User Login Interface module is responsible for rendering a login screen within the application. Its primary function is to display the login screen for users, allowing them to access their specific accounts and data.

Interface to Other Modules

- Interfacing with the task board module, it displays user tasks and the remaining time within each task.
- Communicates with the productivity logs, providing the correct log data corresponding to each user.

Static Model

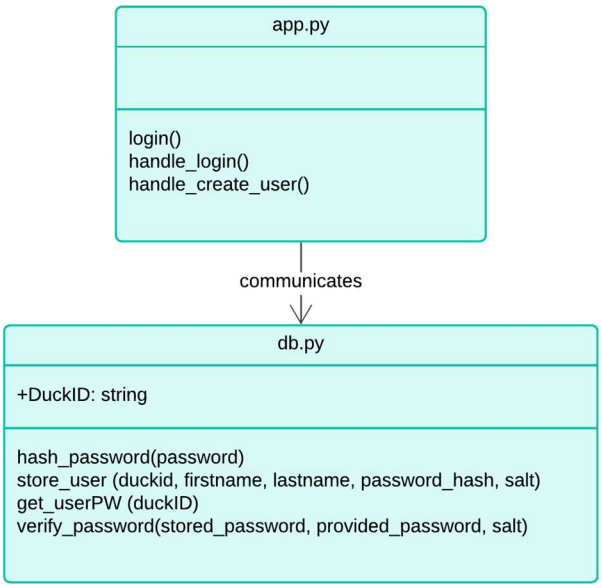

Figure 1

The static model (Figure 1) of the User Login Interface depicts its key components and their relationships. As a user starts a Quack Timer session, they either log in or create a new user.

Dynamic Model

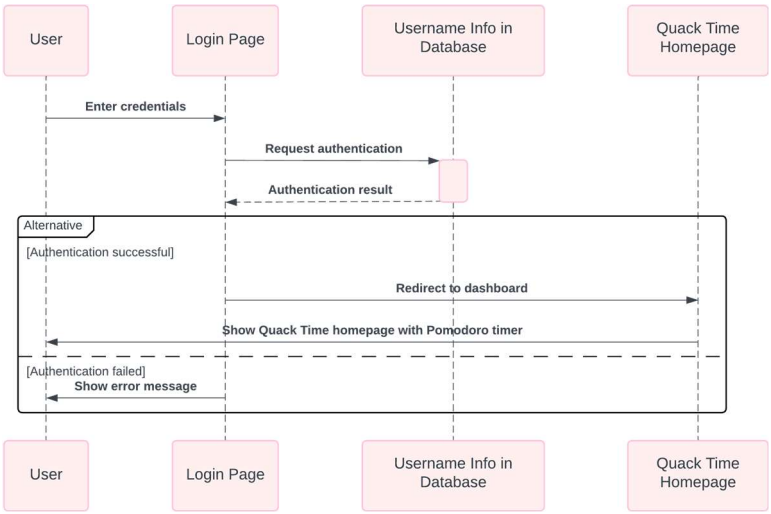

Figure 2

The dynamic model (Figure 2) illustrates the flow of control and data within the User Interface Model.

Design rationale

The User Login Interface of Quack Time ensures secure user authentication, enabling access to personalized accounts and data. This interface is designed for security, usability, efficiency, scalability, and seamless integration with other system components. It validates user input and interacts with the backend to verify credentials, ensuring that only authenticated users can proceed. The interaction flow involves the user entering credentials, the system verifying these credentials with the database, and redirecting the user to the Quack Time homepage upon successful authentication.

Alternative designs

An alternative design approach uses OAuth 2.0 for authentication, allowing users to log in with third-party services like Google or Facebook. This enhances security by leveraging the robust infrastructure of established providers, improves user experience by simplifying the login process, and reduces development overhead. The interaction flow involves the user selecting a third-party login option, authenticating with the provider, receiving an access token, and being redirected to the Quack Time homepage with a securely authenticated session**.** This approach replaces traditional username/password fields with OAuth tokens and updates the database interactions to map these tokens to user profiles.

4.2. Task Board

Role and Primary Function

The task board system module assumes responsibility for managing user-generated tasks. Its primary function is to facilitate the creation and organization of task sections and their respective tasks. Users are empowered to both add and delete tasks within sections, while also assigning estimated completion times to each task.

Interface to Other Modules

- Interacts with the timer to accurately calculate the time spent on each task.
- Connects with the database to store user tasks and records securely.
- Interfaces with the logs to ensure that each completed task is properly recorded within the log records.

Static Model

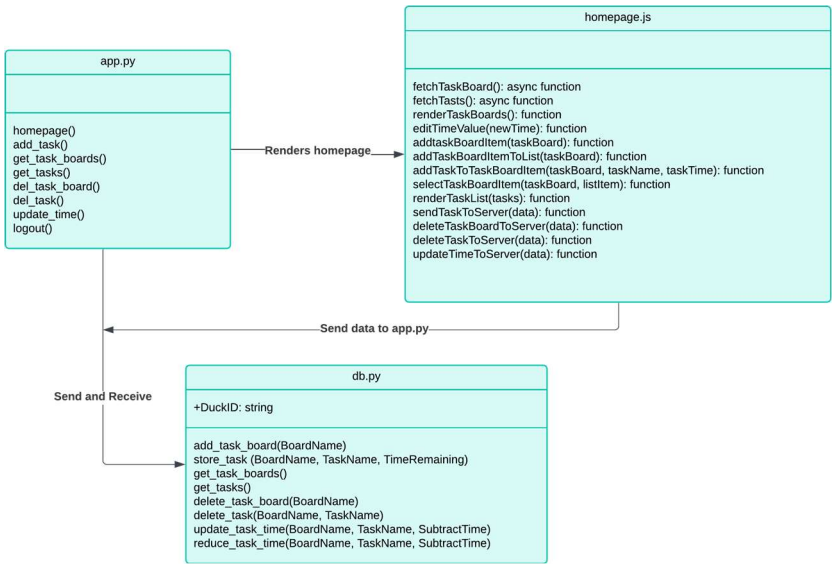

Figure 3

The static model (Figure 3) of the Task Board module outlines its components for managing the task board which facilitates categories and tasks within those categories as well as its usage with the timer.

Dynamic Model

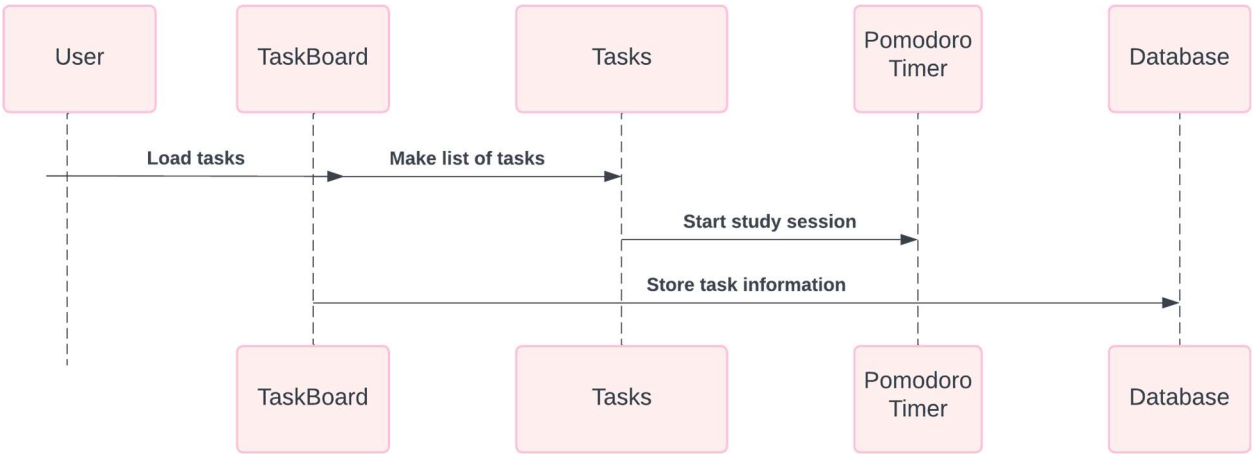

Figure 4

The dynamic model (Figure 4) illustrates the flow of working within the Task Board and how it integrates with the timer and database in the Task Board module. It demonstrates how user actions trigger updates across the platform.

Design rationale

The Task Board module is responsible for managing user-generated tasks, allowing users to create, organize, add, and delete tasks within sections while assigning estimated completion times. It ensures efficient task management, enhancing user productivity by providing a clear and organized interface for task tracking. The Task Board interacts with the timer to calculate the time spent on each task, connects with the database to securely store user tasks and records, and interfaces with the productivity logs to ensure each completed task is properly recorded.

Alternative designs

An alternative design approach involves integrating AI-based task management. Instead of manually organizing tasks, the AI-driven Task Board suggests task prioritization based on deadlines, user behavior, and historical data, providing personalized task management recommendations. This approach enhances efficiency by leveraging machine learning algorithms to predict optimal task sequences, improving user productivity with minimal manual intervention. The AI-based Task Board will still interact with the timer, database, and logs but will incorporate additional AI components for data analysis and task prioritization, offering a more dynamic and adaptive task management system**.**

4.3. Timer

Role and Primary Function

The timer assumes the crucial role of orchestrating the Pomodoro study exercise for the user. Its primary functionality lies in adhering to the principles of the Pomodoro study method, ensuring maximum efficiency in study sessions.

Interface to Other Modules

- Interfaces with the application to provide user identification.
- Communicates with the Data Storage Server module to store data with respect to each user.

Static Model

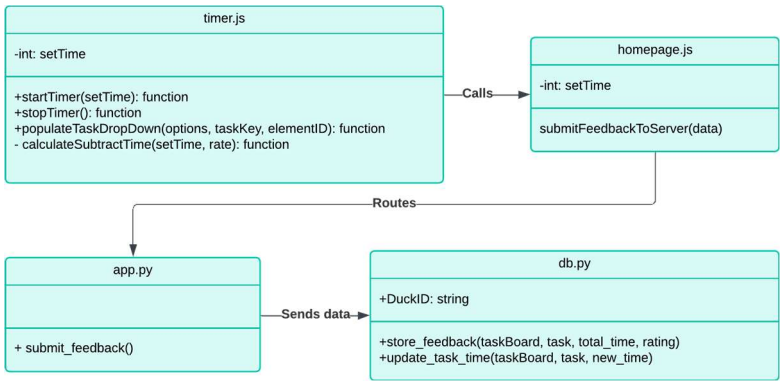

Figure 5

The static model (Figure 5) of the Timer outlines how the module interacts as a standalone model and and triggering the pop-up window.

Dynamic Model

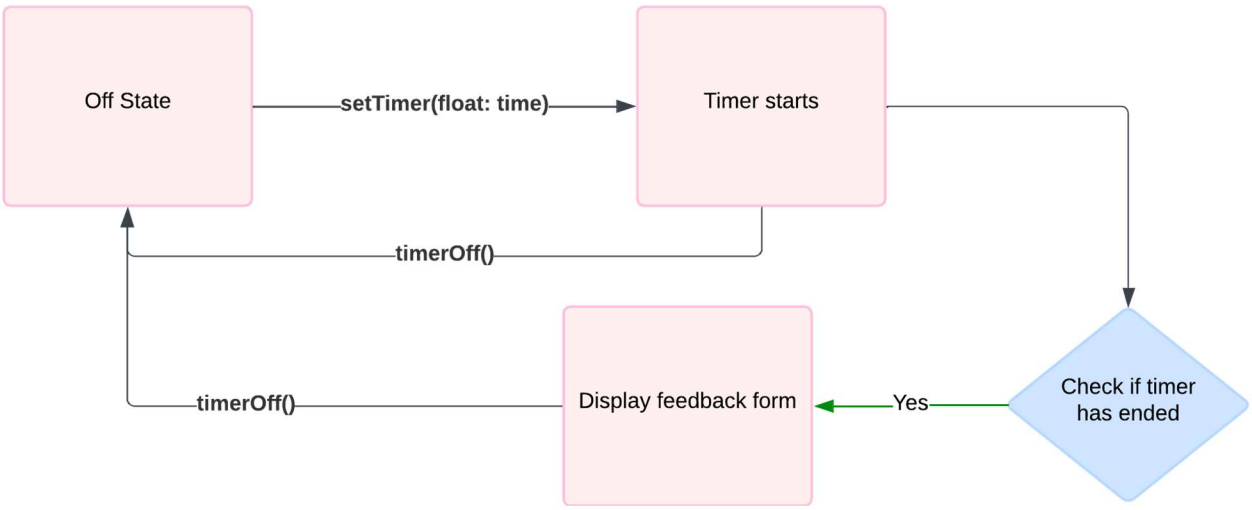

Figure 6

The dynamic model (Figure 6) illustrates the behavior of the Timer module and how it transitions between on and off stages and sounds the alarm.

Design rationale

The Timer module orchestrates the Pomodoro study exercise, enhancing user efficiency during study sessions by segmenting study time into focused intervals followed by breaks. Its primary function is to adhere to the Pomodoro technique principles, ensuring maximum efficiency in study sessions. The Timer interfaces with the application to provide user identification and communicates with the Data Storage Server to store data for each user.

Alternative designs

An alternative design approach involves integrating adaptive timing features. Instead of fixed Pomodoro intervals, the adaptive Timer adjusts session lengths based on user performance and preferences. This dynamic adjustment could increase productivity by tailoring the timing to individual needs, improving focus and reducing burnout. The adaptive Timer would still interact with the application and Data Storage Server but would include additional components to analyze user data and adjust session lengths accordingly, providing a more personalized study experience.

4.4. ProDUCKtivity Log

Role and Primary Function

The primary function of the productivity log is to compile the history of all sessions, providing users with a comprehensive log to showcase trends between their sessions.

Interface to Other Modules

- Collaborates with the task board to retrieve task data and remaining time after each session.
- Interacts with the timer to log each session upon completion of the study session.

Static Model

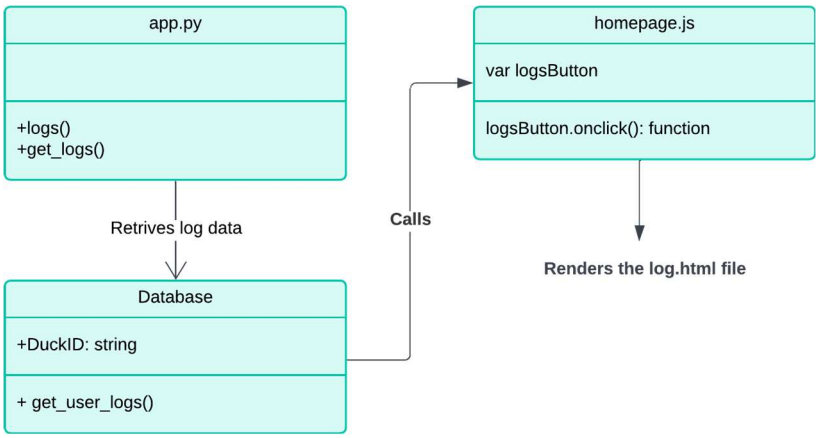

Figure 7

The static model (Figure 7) of the ProDUCKtivity Log module outlines the structure and build of the logging system and its reliance on the Database and, indirectly, the pop-up window that is displayed after each pomodoro session is completed from the timer module.

Dynamic Model

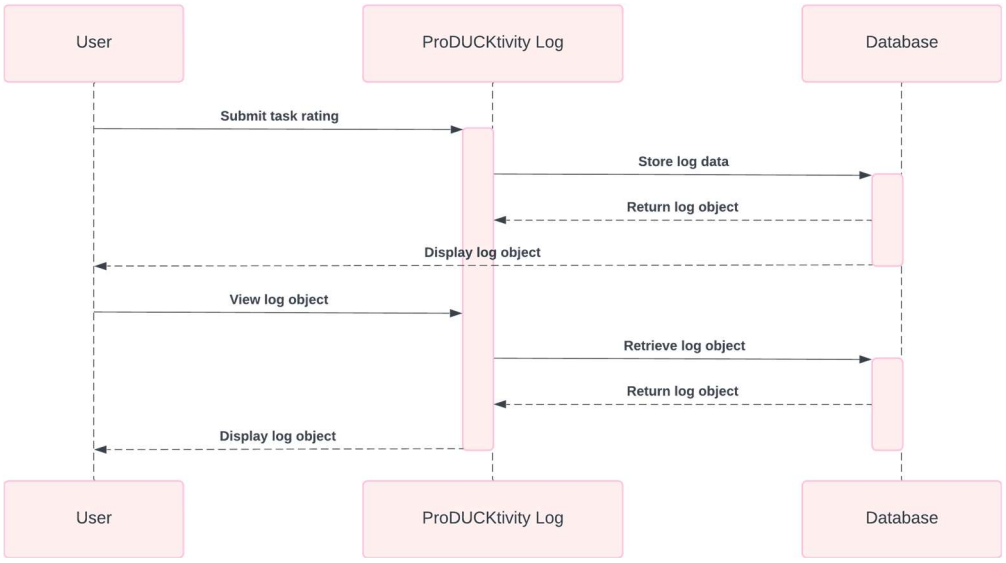

Figure 8

The dynamic model (Figure 8) illustrates the flow of data within the ProDUCKtivity log and how its behaviour is reliant on both the user and Database. Users are required to rate their session which is stored into the Database. The ProDUCKtivity log then retrieves the logs upon access.

Design rationale

The ProDUCKtivity Log is designed to help users track their productivity over time, providing insights into their study habits and allowing them to identify areas for improvement. By integrating with the Task Board and Timer, it ensures that all relevant data is captured accurately and presented in a user-friendly format.

Alternative designs

An alternative design for the ProDUCKtivity Log could include advanced data analytics and machine learning capabilities. Instead of simply logging and displaying historical data, this enhanced log would analyze patterns in user behavior and provide personalized recommendations to optimize study sessions. It could suggest optimal study times, identify frequently procrastinated tasks, and recommend breaks based on user productivity levels.

4.5. Data Storage SQL Server

Role and Primary Function

The Data Storage Server module is responsible for managing application data, including user accounts, logs, and tasks. Its primary function is to provide a centralized repository for storing and retrieving data, ensuring data integrity, security, and scalability.

Interface to Other Modules

- Interfaces with the Taskboard module to hold the sections and task the student has added
- Interfaces timer to add tasks to the logs file at the end of the study session

Static Model

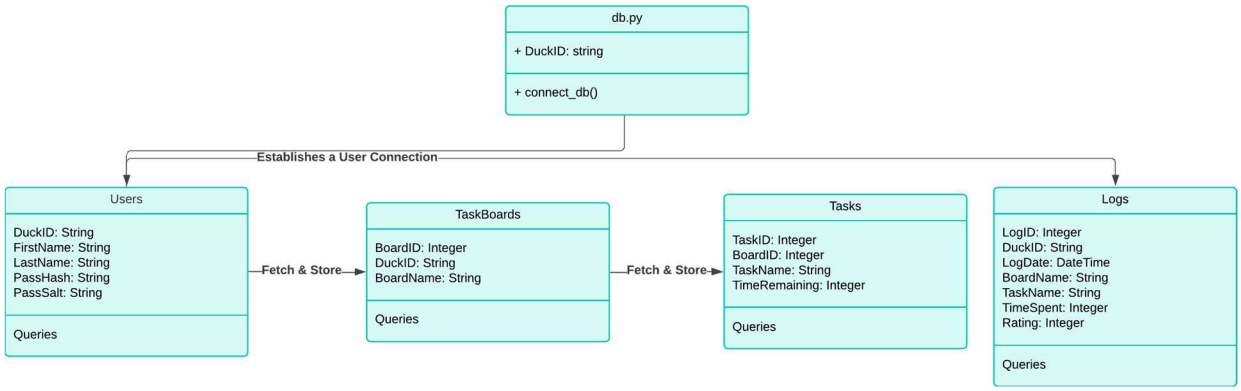

Figure 9

The static model (Figure 9) of the Data Storage Server module outlines the structure and build of the Database used within the system which is a Database containing tables for users, task boards, tasks and logs. These tables are processed across three major components. The Login Interface, Task Board and ProDUCKtivity log.

Dynamic Model

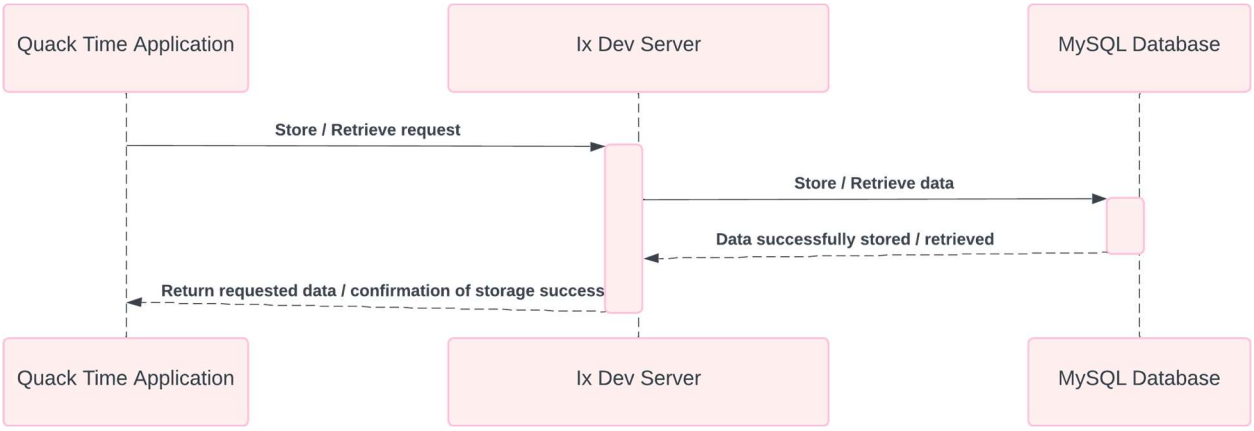

Figure 10

The dynamic model (Figure 10) illustrates the flow of data within the Data Storage Server module, demonstrating how users' requests are processed, and served, and how data is retrieved from and stored in the database and the server.

Design rationale

The Data Storage Server is designed to ensure reliable data storage and retrieval, maintaining data consistency and security across the application. By centralizing data management, it supports seamless synchronization and backup, enhancing the overall user experience and ensuring that data is always up-to-date and accessible.

Alternative designs

An alternative design could involve using a distributed database system, such as Apache Cassandra or Amazon DynamoDB, to handle data storage. This would enhance scalability and fault tolerance by distributing data across multiple nodes, ensuring high availability and performance even under heavy loads. Additionally, integrating real-time data processing capabilities could enable instant updates and analytics, providing users with immediate feedback on their productivity.

5. Dynamic Model of Operational Scenarios

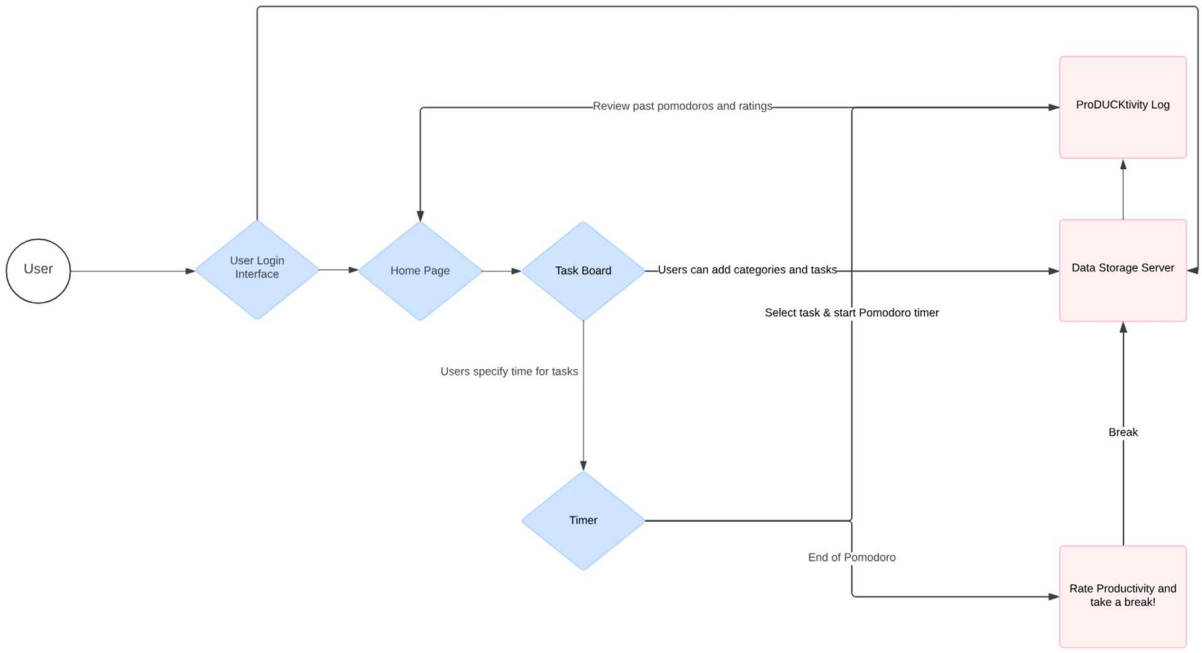

Figure 9

Figure 11 illustrates the primary use case scenario for Quack Timer. The diagram highlights that the initial step involves the user logging in to the system, being directed to the home page which displays their task board and the timer. The diagram shows processes that occur between the modules and other components that are crucial to the design of Quack Timer.

6. Server Side

A MySQL server has been set up the University of Oregon's Computer Science department server, ix-dev. The database structure for the Quack Time application is designed to efficiently manage user data, task boards, tasks, and logs.

The database for the Quack Time application consists of four main tables: Users, TaskBoard, Tasks, and Logs. Each table is designed to store specific types of information and is linked to other tables through key relationships. The following sections outline the tables used in the database, their fields, and the relationships between them.

6.1. Users Table

The Users table stores information about each user in the application.

- **DuckID**: A unique identifier for each user. It is a string of exactly 9 characters.
- **FirstName**: The user's first name. It can be up to 255 characters long.
- **LastName**: The user's last name. It can be up to 255 characters long.
- **PassHash**: The hashed version of the user's password for security purposes. It can be up to 255 characters long.
- **PassSalt**: The salt used in hashing the user's password to enhance security. It can be up to 255 characters long.

6.2. TaskBoard Table

The TaskBoard table stores information about task boards created by users.

- **BoardID**: A unique identifier for each task board. It is automatically incremented for each new board.
- **DuckID**: The unique identifier of the user who owns the task board. It references the *DuckID* in the *Users* table.
- **BoardName**: The name of the task board. It can be up to 255 characters long.

6.3. Tasks Table

The Tasks table stores information about individual tasks within task boards.

- **TaskID**: A unique identifier for each task. It is automatically incremented for each new task.
- **BoardID**: The unique identifier of the task board to which the task belongs. It references the *BoardID* in the *TaskBoard* table.
- **TaskName**: The name of the task. It can be up to 255 characters long.
- **TimeRemaining**: The amount of time remaining to complete the task. It is stored as an integer representing minutes.

6.4. Logs Table

The Logs table stores log entries that track user activities related to tasks and task boards.

- **LogID**: A unique identifier for each log entry. It is automatically incremented for each new log entry.
- **DuckID**: The unique identifier of the user who made the log entry. It references the *DuckID* in the *Users* table.
- **LogDate**: The date and time when the log entry was created.
- **BoardName**: The name of the task board associated with the log entry. It can be up to 255 characters long.
- **TaskName**: The name of the task associated with the log entry. It can be up to 255 characters long.
- **TimeSpent**: The amount of time the user spent on the task during the logged session. It is stored as an integer representing minutes.
- **Rating**: The user's rating of the task for that session. It is an integer value between 1 and 5.

6.5. Relationships

- **Users and TaskBoard:** A one-to-many relationship where each user can own multiple task boards.
- **TaskBoard and Tasks:** A one-to-many relationship where each task board can contain multiple tasks.
- **Users and Logs:** A one-to-many relationship where each user can have multiple log entries.

6.6. Constraints and Validations

- **DuckID** in the Users table is exactly 9 characters long.
- **Rating** in the Logs table is an integer value constrained between 1 and 5.

This structure ensures organized storage and retrieval of user data, task boards, tasks, and activity logs, providing a robust foundation for the Quack Time application.

7. References

Faulk, Stuart. (2011-2017). CIS 422 Document Template. Downloaded from https:// uocis.assembla.com/spaces/cis-f17-template/wiki in 2018. It appears as if some of the material in this document was written by Michal Young.

IEEE Std 1016-2009. (2009). IEEE Standard for Information Technology—Systems Design—Software Design Descriptions. https://ieeexplore.ieee.org/document/5167255

J, Anthony Hornof. “Software Design Specification [Template].” 6 May 2019.

Sommerville, I. (2000) Software Engineering. Addison-Wesley, Harlow.

8. Acknowledgements

This document is built with reference to the SDS template provided by Prof. Anthony Hornof. Additionally, it builds on a document developed by Stuart Faulk in 2017, and on the publications cited within the document, such as IEEE Std 1016-2009.