Design Document

Group Details

Project Name: Scrum Daddy
Group Number: 31
Members: Matthew Zlatniski, Victor Dollosso, Can Kucukkurt, Patrick Barry, and Zachary Wu

Purpose

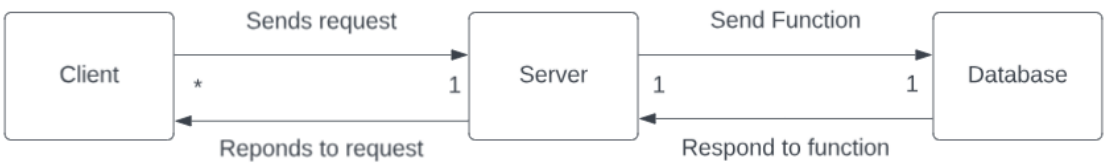
Scrum has become a popular framework for organizing teamwork. The core values of Scrum are learning through experience, self-organization, and reflecting to improve continuously.

This project aims to provide a centralized platform to integrate all project documents seamlessly into a user-friendly interface. Our solution will exclusively focus on Scrum methodologies, providing a streamlined experience without the clutter of unnecessary features. This simplicity ensures teams can effortlessly track projects and monitor progress efficiently.

Design Outline

High Level Overview
This project will develop a web application that is designed to streamline the scrum process for software development teams. This allows users to use our product from any browser, which promotes flexibility and eases collaboration. We will utilize the Client-Server model, where one server handles a large number of client access simultaneously. The server will access a database where all the user profiles, and project information is stored.

Client Server Model



Client

- Components: The dashboard will display an overview of ongoing sprints, task statuses, individual assignments, and anything else needed. The scum board will be a visual representation of the task flow, allowing users to view/update tasks. The backlog management is an interface for adding and prioritizing upcoming tasks. Other features like a document to create a project charter, product backlog or design document.
- Purpose: The client serves as the primary interface for users to interact with the application. It allows team members to create, view, update, manage and complete the scrum framework requirements.

Server

- Components: The API endpoints facilitate communication between the client and server, handling requests such as task creation, updates, user management. The authentication service manages user authentication and authorization, ensuring secure access to the application.
- Purpose: The server handles business logic, data processing and database interactions. It ensures data consistency and processes client requests.

Database

- Components: The user tables store user credentials and profiles. The task tables contain task descriptions, status, assignments, etc. The sprint tables hold sprint schedules, durations, associated tasks.
- Purpose: Stores all persistent data, including user profiles, task details, sprint schedules and project configurations.

Scrum features

- Components: Sprint planning, task board, backlog management
- Purpose: allows for easy project management

Report generation

- Components: Burndown Charts, progress bar
- Purpose: To get an insight into how a project is going along

Interactions

User interacts with the UI

- Creates, assigns, tracks tasks
- Manages deadlines

UI interacts with the server

- Sends requests for storage, retrieval

Server interacts with database

- Stores and retrieves data, user information, task information

Design Issues

Functional Issues:

1. Signing up
    Option 1: Email and password
    Option 2: Name, Email, Password
    Option 3: Name, Email, Phone, Password

We chose Option 2. User authentication is a big issue in this project. In this scrum framework task manager, we will have email notifications so we need users' email but not necessary for their phone number. This also allows for another form of verification. The email will be needed to reset the password.

2. User role management
    Option 1: Have everyone have access to everything/ Single role
    Option 2: Multiple roles with different access levels (Developer, Scrum master, Admin/Project owner)

We chose Option 2. In a scrum framework, different users have distinct responsibilities. Providing role-based access control allows for better management of tasks, sprints, and permissions, enhancing security and efficiency. Project owners can configure teams and projects, manage priorities etc. Scrum masters can manage sprints and tasks, while development members can update task statuses and break up tasks into smaller subtasks.

3. Task dependency handling
    Option 1: No explicit task dependency
    Option 2: Option for tasks to be dependent on other tasks

We chose Option 2. Task dependencies are critical in project management as some tasks cannot start until others are completed. By allowing users to define dependencies, the system can help prevent scheduling conflicts and improve task prioritization.

4. Task prioritization

Option 1: Manual prioritizing
Option 2: Semi automatic prioritizing
Option 3: Fully automatic prioritizing based on dependencies and deadlines

We chose option 1 because manual prioritizing ensures that tasks critical to the project that might not seem critical can be prioritized. This allows the owner or scrum master to prioritize between tasks if necessary and allows full control over which tasks are more important.

5. Notification system
Option 1: No notifications for any reminders
Option 2: Email notifications for every reminder/updates
Option 3: Email notifications for critical alerts

We chose option 2 and 3. An email notification system provides immediate updates to users during the sprint enhancing real time collaboration. Emails for every task update or document change is a bit much, so given the option to change to just critical alerts, so those important updates won't be missed, like being assigned to a task.

Nonfunctional Issues:

1. What language/framework should we use for frontend?
Option 1: html/javascript
Option 2: React
Option 3: Angular

We chose option 2 because of the component-based architecture and the familiarity from our project. The component-based architecture allows for the development of reusable UI components that manage their state, leading to a more organized codebase. Although our team members are inexperienced in web development, React is the most familiar.

2. What language/framework should we use for the backend?
Option 1: NodeJS
Option 2: Django
Option 3: Spring Boot

We chose NodeJS because we are already using Javascript in the frontend, so using javascript in the backend would be best. NodeJS also offers significant advantages in terms of development efficiency, performance, scalability, and support.

3. What are we going to do with testing?
Option 1: Manual testing

Option 2: Automated Testing

Option 3: Continuous Integration/ Continuous Development (CI/CD) with automated Testing

We are going to start out with manual testing, but if we have time, we are going to try and integrate CI/CD, or option 3.

4. Which database are we going to use?

Option 1: MySQL

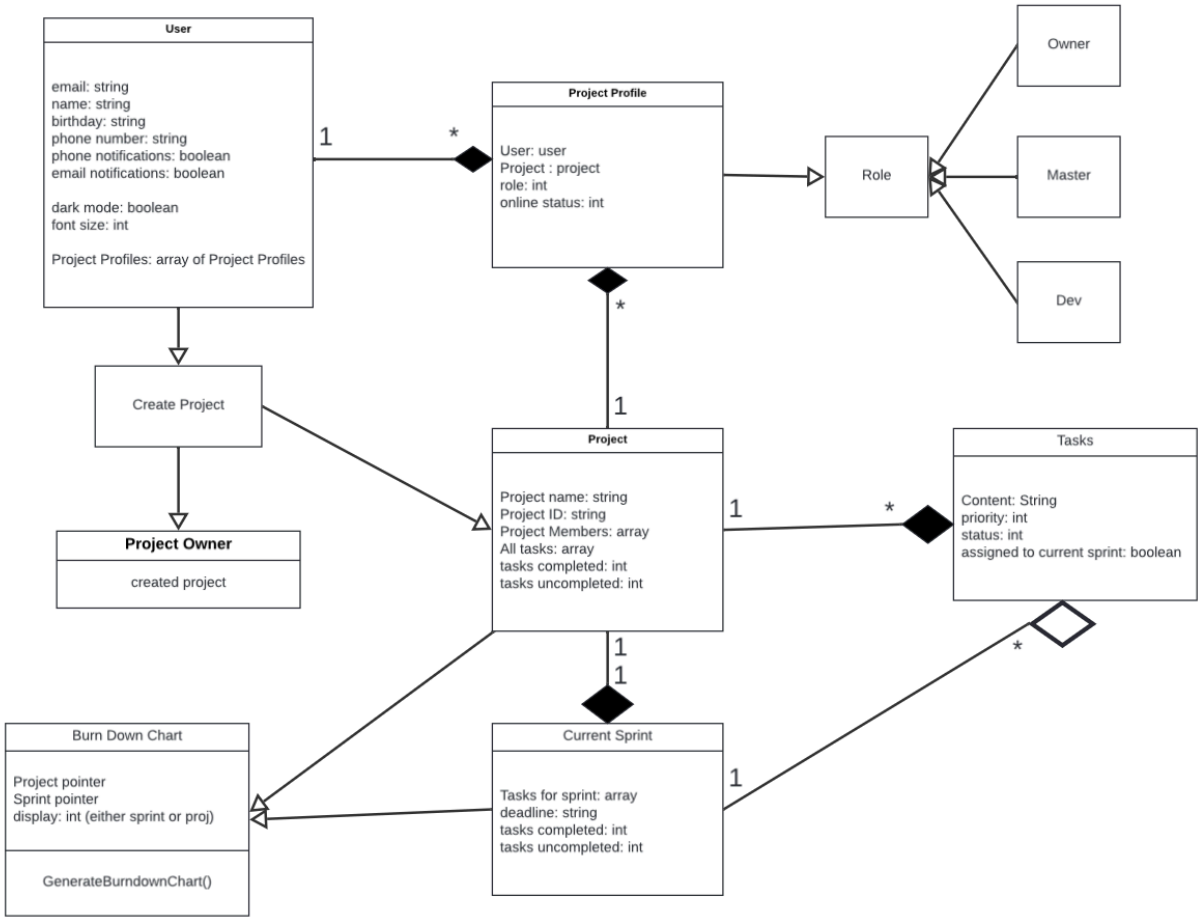
Option 2: MongoDB

Option 3: Google Spanner

We chose to use MySQL because our application relies on structured data, such as the user information, tasks, sprints, etc. This ensures data interactions are organized and consistent. Also another reason is the security features. MySQL comes with a strong set of security features which ensures safe and secure data.

Design Details

UML Class Diagram



Description of interactions between classes and interactions between classes

User

- A User object is created when someone signs up
- Each user has unique id
- Each User will enter their name, birthday, email, password
- Each User has the option to choose preferences such as font size and light/dark mode
- Each User will have a project profile which shows all the projects they are in. They will have the option to choose from one or create another project.
- Each role(owner, scrum master, developer) has his own set of actions.

Project Profile

- A project profile object is created once you make an account
- When a user creates or get added to a project, it updates your project profile
- Each project profile contains the user, list of the users projects, list of the users roles on the projects, and their online status

Project

- When a user creates a project, a project object is created
- Each project is assigned a unique project id
- Each project has its own name, list of members, list tasks, count of completed tasks
- Each project will have a message board to allow collaborates to chat

Tasks

- Each project consists of multiple task objects
- A task contains the content, priority, the status, whether or not it is in the current sprint
- The list of tasks also belong to the current sprint

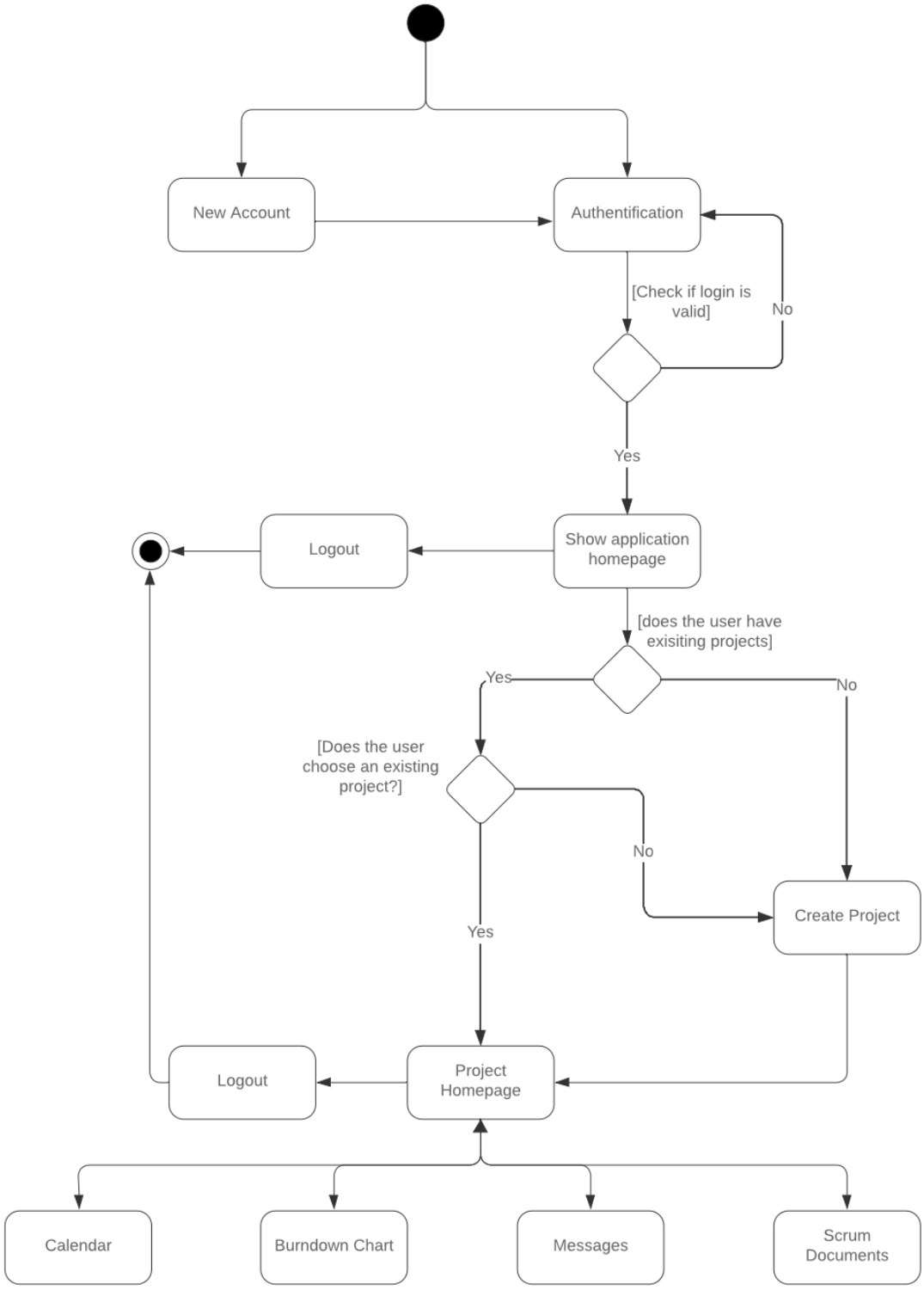
Burndown Chart

- A burndown chart object is generated for the entire project as well as the current sprint
- The burndown chart has a status that indicates whether the project or current sprint chart is generated
- Calls the functions GenerateBurndownChart()

Current Sprint

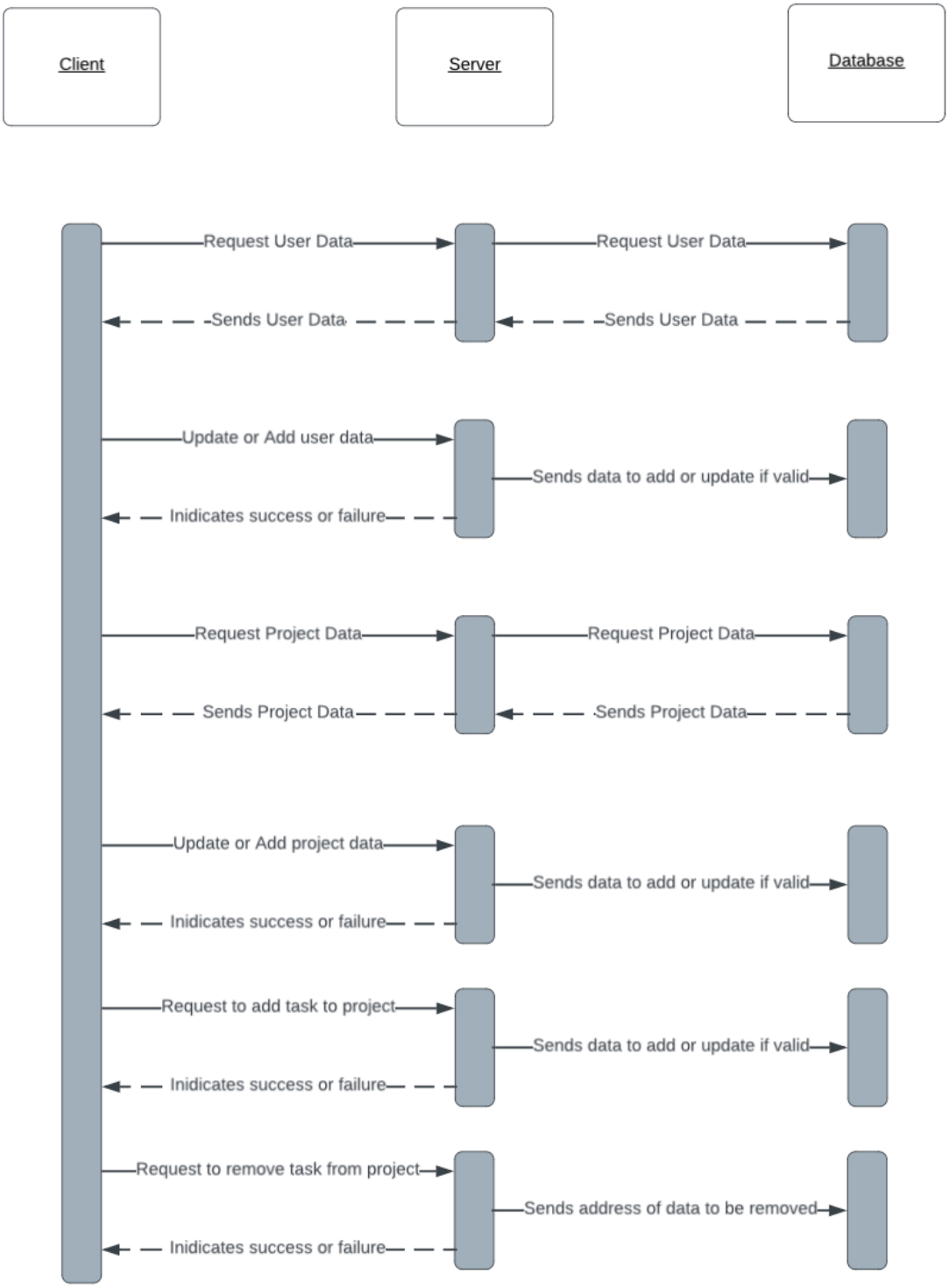
- Each project has exactly one current sprint object
- The current sprint object has an array of task objects, a deadline, and integers to count how many tasks are complete and incomplete
- This maps to the burndown chart as well as task objects

Activity Diagram

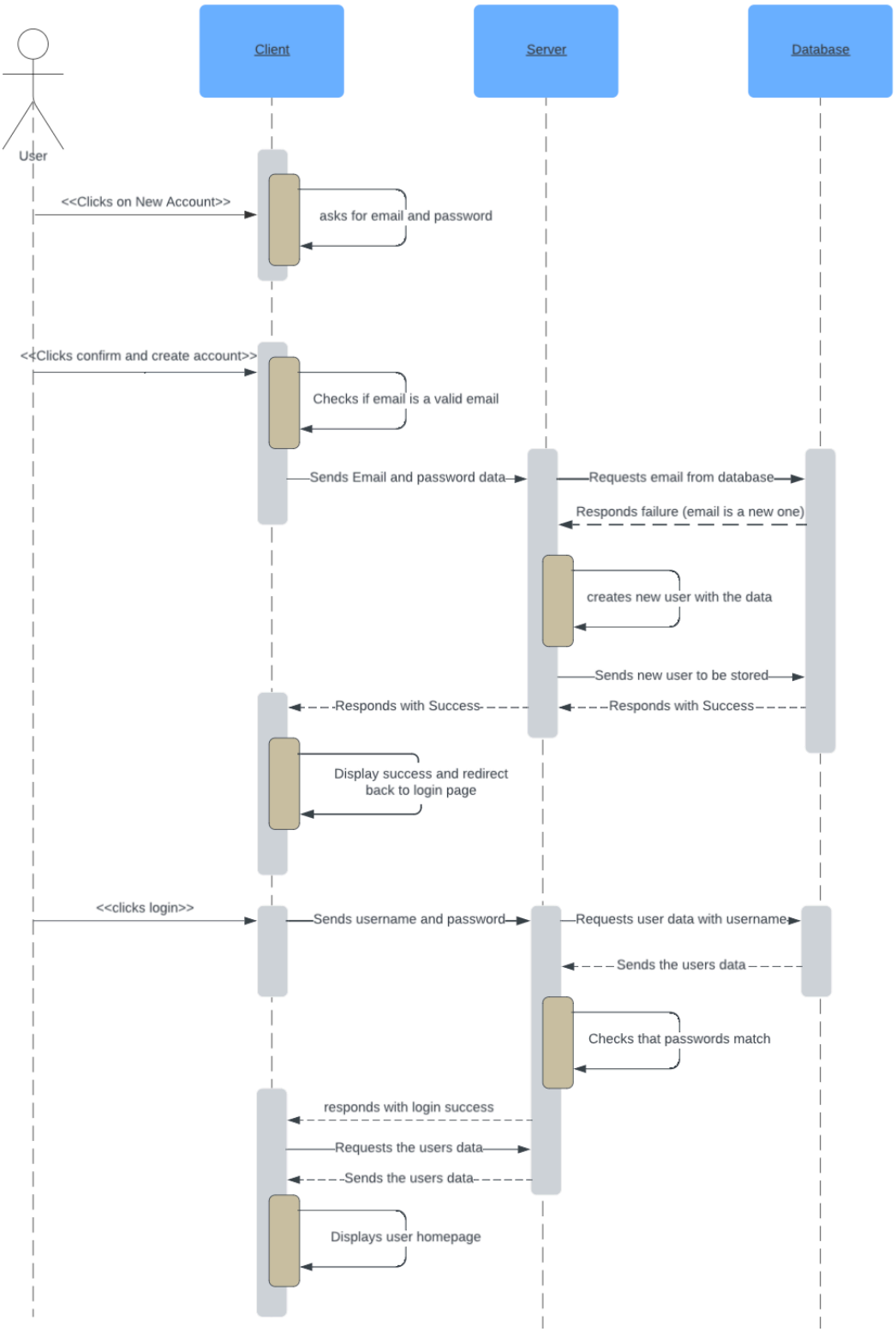


Sequence Diagrams

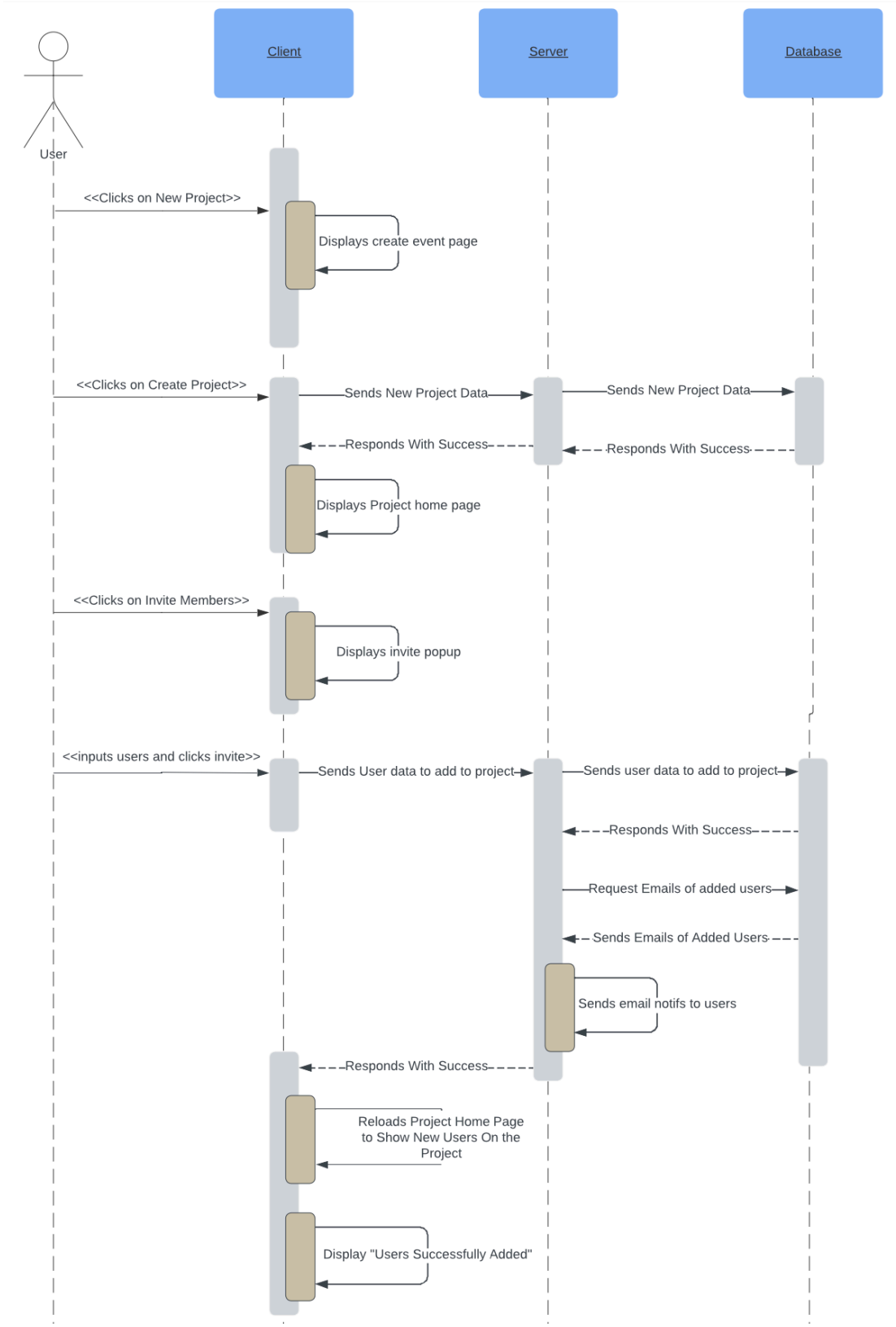
Sequence of Events Overview



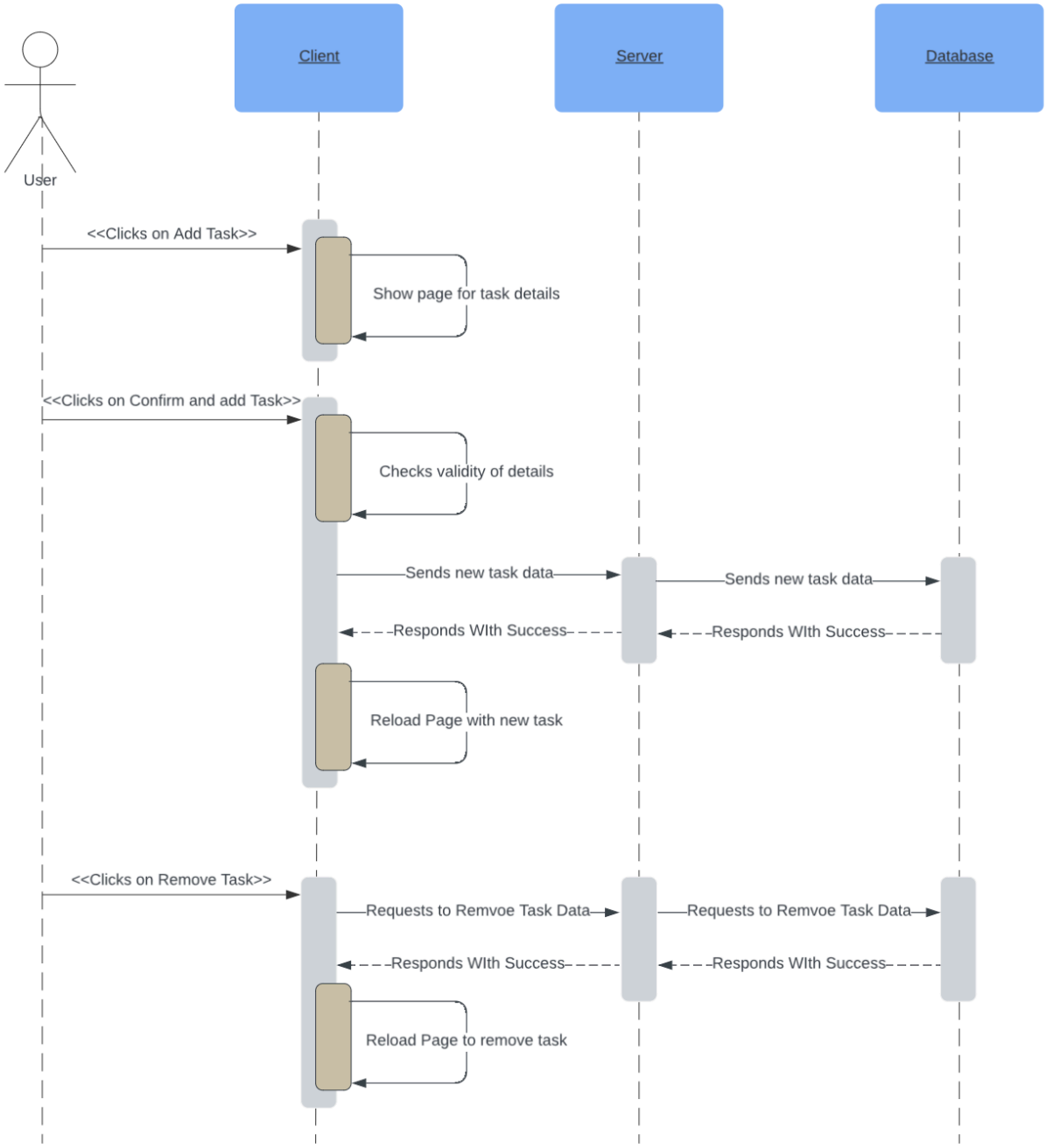
Login/Create Account Sequence of Events



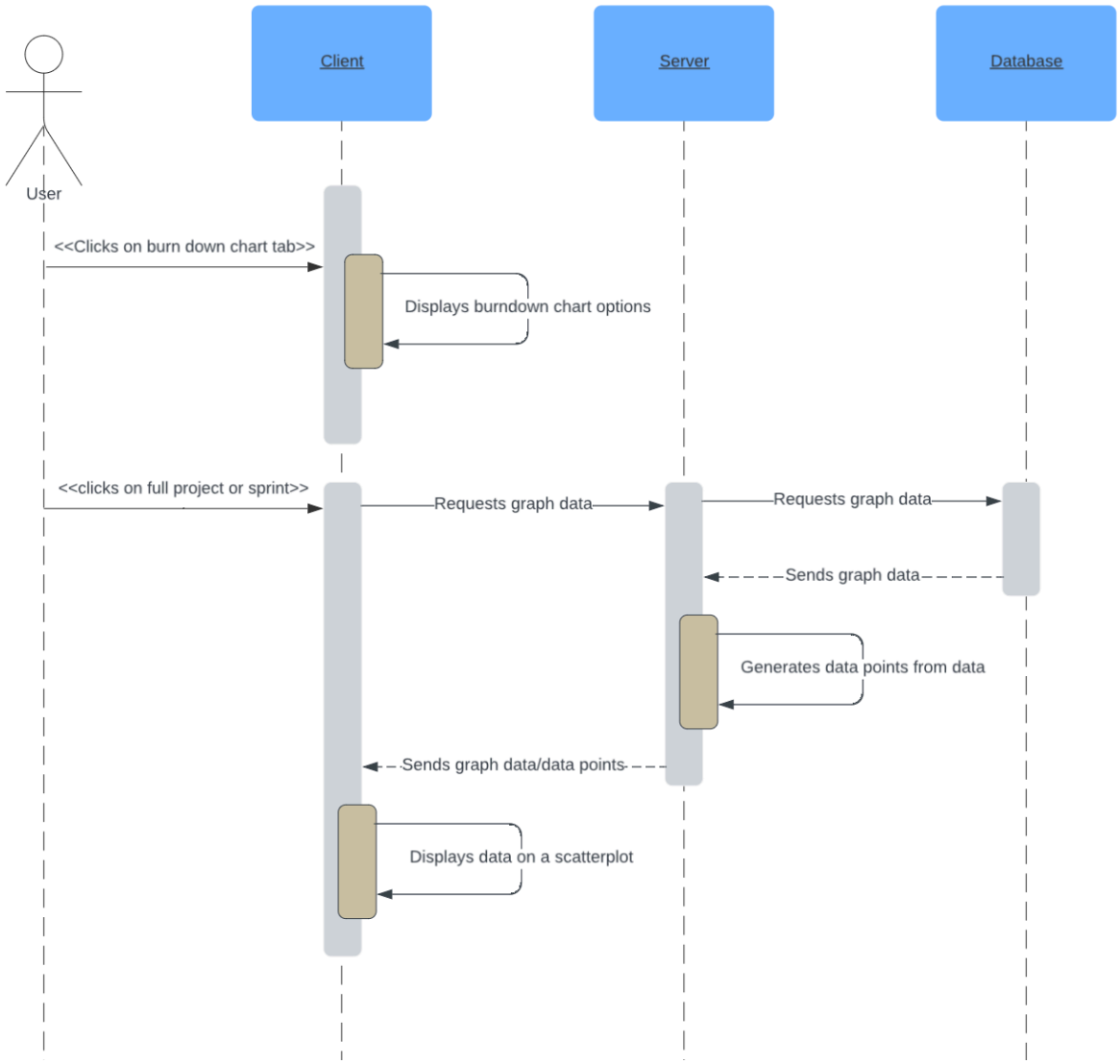
Create Project Sequence of Events



Add/Remove Tasks Sequence of Events

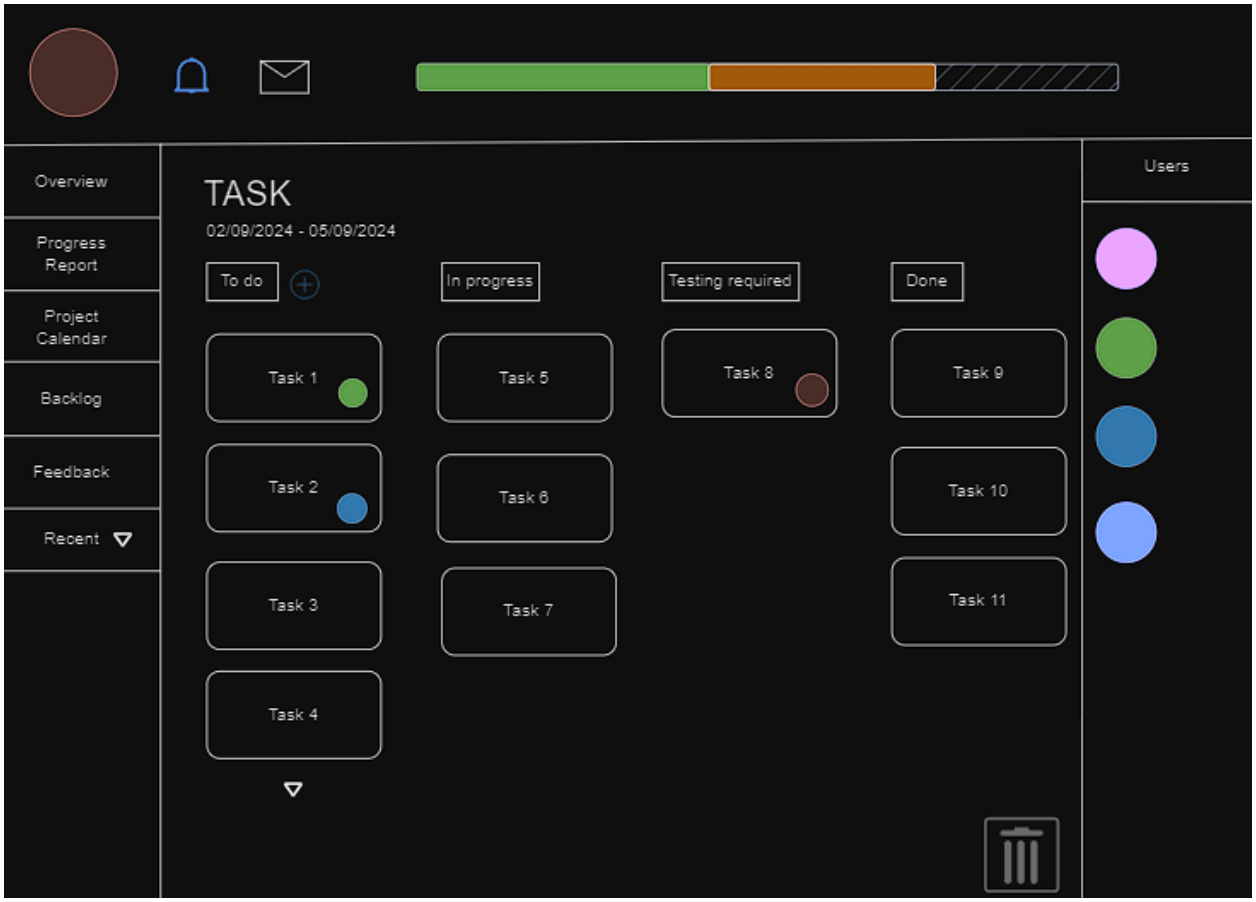


Automated Burn Down Chart Sequence of Events



UI Design


Overview
Progress Report
Project Calendar
Backlog
Feedback
Recent
OVERVIEW
16 Done
5 Needs Review
NEW
2 New
To Do
Done
%45
In Progress
Chatbox
Users


Overview
Progress Report
Project Calendar
Backlog
Feedback
Recent
TASK
02/09/2024 - 05/09/2024
To do
In progress
Testing required
Done
Task 1
Task 2
Task 3
Task 4
Task 5
Task 6
Task 7
Task 8
Task 9
Task 10
Task 11
Users