Design Document

September 19, 2022

Haoyu Zhang

Jim Ning

Yunhan Huang

Ziyang Fang

Index

Purpose

Students throughout their entire schooling career have always experienced a group member who can't pull their own weight. Measurements in individual group participation have always been delegated to peer evaluations. There is a demand for a group project manager that can provide a more accurate description of what each individual contributed to the group project. Additionally, current project managers have very general user interfaces where every individual within the group views all the information for the project. However, once the individuals are working on the project, not all the tasks and information are pertinent to the user. This can create an interface that is less efficient for the user, and inhibit the productivity and collaboration between the group members in finishing the project.

ClickUp provides a user interface where group members can create new projects and tasks to collaborate and monitor the progress of the project. In our research, these applications offer services that are often too abstract and do not show enough information on the project progress details. Additionally, they do not offer personalized interfaces that display information pertinent to each user.

Goh will generate a personalized user interface that will display tasks that users need to work on. While other information will be available to the user, we believe that focusing on pertinent tasks will increase overall efficiency and project management. Also, by using user task time completion we can provide an analytical report of the various contributions of each user. Through these two added functions, we believe that we can provide a superior project manager to models such as ClickUp.

Functional Requirements

*** User(Member, Leader, and Guest)**
*** Member (Project and task editing privilege)**
*** Leader (Owner)**
*** Guest (Task metadata and transfer between users privilege)**

1. **User can create/modify the account**
   As a user,
   a. I would like to register for a Goh account.
   b. I would like to login to my account.
   c. I would like to be able to change my account information (reset password, change avatar, username, linked email, and etc.)
   d. I would like to have role tags (e.g. CS project: frontend, backend, logo designer, and etc.).

2. **Project leader can create/modify the project/task**
   As a project leader,
   a. I would like to create the project.
   b. I would like to create tasks for my project.
   c. I would like to modify project content.
   d. I would like to modify task content.
   e. I would like to show what is primary task
   f. (if time allows) I would like to share my project and its tasks with other members.

3. **User/project manager have different access to the task**
   As a User,
   a. I would like to be able to take on a task.
   b. I would like to search for a specific task (Ctrl-F functionality).
   c. I would like to be able to easily navigate between my projects and tasks.
   d. I would like to comment on group members' tasks.
   As a project manager,
   a. I would like to be able to delete tasks.
   b. I would like to set the status of tasks (multi user or single user).
   c. (if time allows) I would like to pin/highlight features (pin tasks to the top).

4. **User/project manager can view all the tasks on the calendar**
   As a user,
   a. I would like to see the due dates for each of my tasks (calendar system).

b. (if time allows) I would like to export the task calendar. So, I can write the specification of my project.
c. (if time allows) I would like to export the task calendar.

5. **Users should be notified in those circumstances.**

As a user,

a. I would like to be notified when I am assigned to a small group.
b. I would like to be notified when one of my group mates comments on me.
c. I would like to be notified when a task is assigned to me.

As a project leader,

a. I would like to notify all the group members at once.

6. **Chat System (If time allows)**
   a. As a project leader, I would like to assign specific group members into a small group.
   b. As a project leader, I would like to have group communication.
   c. I would like to see what task my group mates are working on.
   d. I would like to see other members' contact information.

Nonfunctional Requirements

1. **Client requirements:**

As a developer,

a. I want the server to be available for any area.

2. **Server requirements**:

As a developer,

a. I want there to be both a development and production database.
b. I want there to be both a development and production server.
c. I want the server to be tested for both development and production.
d. I want there to be a production interface.

3. **Design requirements:**

As a developer,

a. I want both the client and server to share a set of class model to be sent and received via the API.

b. I want there to be build scripts and use the git repository to deploy the development and production perspectives.

4. **Performance requirements:**

   As a developer,

   a. I want the server to be able to handle adequate users at the same time to support one hundred people at max.
   b. I want the server to be able to respond to all user requests.
   c. I want the server to gracefully handle all the errors or other components that the user interacts with.
   d. I want the server to withstand enough testing before showing it to clients.

5. **Appearance requirements:**

   As a developer,

   a. I want the server to be enjoyable, striking enough and easy to use.
   b. I want the server to be productive and efficient for the user.

6. **Security requirements:**

   As a developer,

   a. I want all the data we collect from the user such as username, password, contact information to be properly secured on the user's device and our system.

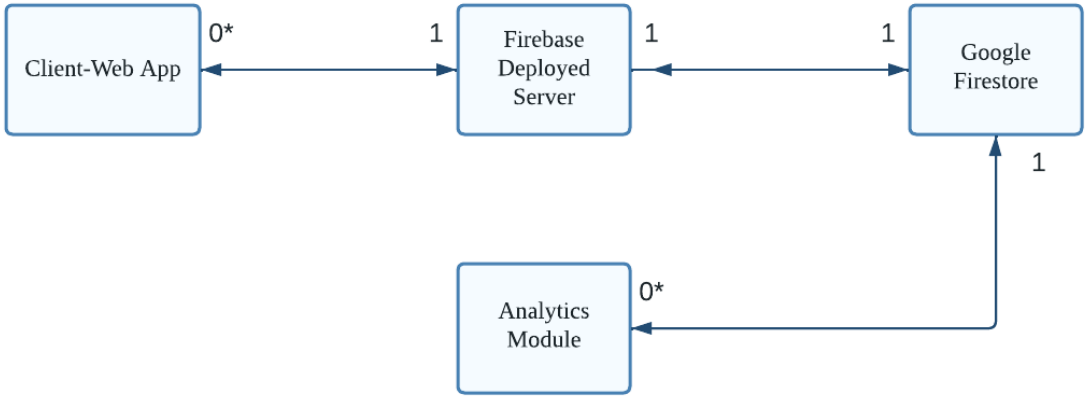
Design Outline

Our application provides a dedicated interface where users can see the task that is currently executing, tasks that are ready to execute, tasks that are waiting for input, and tasks that are completed. Our implementation will utilize a Client-Server model. The server will access data stored in a database and allow for multiple concurrent web-browser clients. We will use a modular service architecture to populate the database with information

1. Client-Web Application
    a. The Web-Browser client will be available on any modern, javascript supporting, web browser (i.e. Google Chrome, FireFox, Microsoft Edge, Safari, etc) and will provide the interface for Goh.
    b. The client will send and receive data to our database via HTTP requests and user JSON objects.
    c. The data taken from the database will be used to display on our web client. Examples include user's projects, tasks, and account information.
2. Server
    a. The server will provide the abstraction layer on top of our database to make it easy for users to request and send information.
    b. The server will accept HTTP requests, query to access the database and then return the requested information.
3. Database
    a. The database will store all of the data used in Goh.
    b. The database provides a real time database in NoSQL and can keep all the client web-applications in sync. This can be done through real time listeners and offline support for the clients.
    c. The data stored in the database will be the relationships between users, projects, and tasks. Additionally, it will include user time taken on each task.
4. Analytics Module
    a. The analytics module will take data from the database and render an analytical report of the users' time taken on each task at the end of the project.

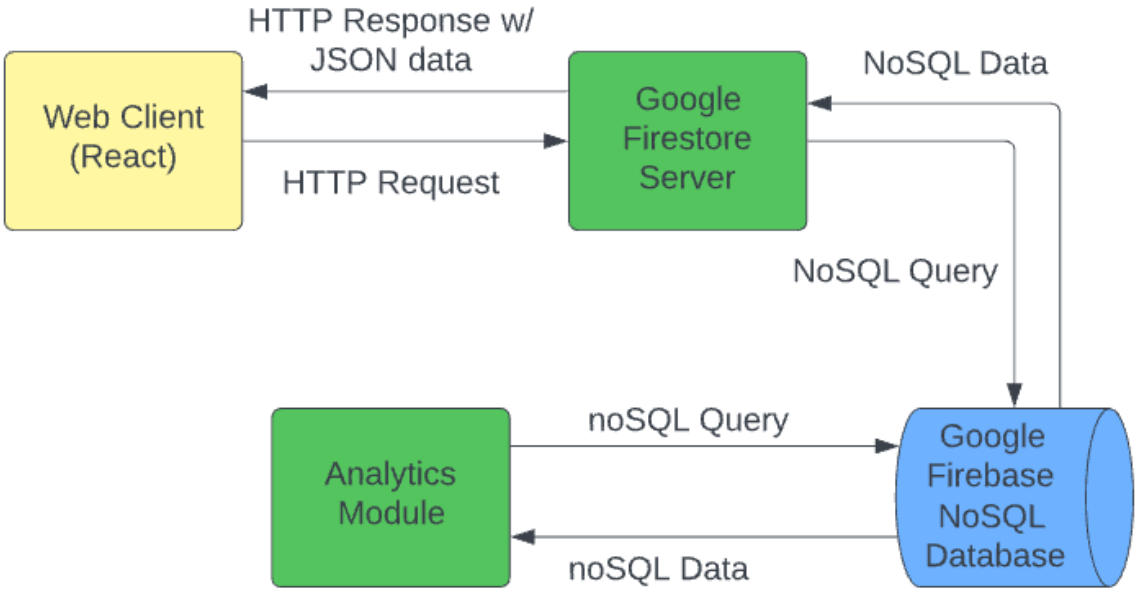
High Level Overview of The System

The Web-Client and Database form a many-to one client-server architecture. The Web-Client will send queries to the database for information, and the database will fulfill these queries and send relevant data back to the client. When storing, the Web-Client will write the information into the database. The analytics module will calculate and analyze data from the database and save the results back into the database. This can be done independently and allow for the data to be up to date.

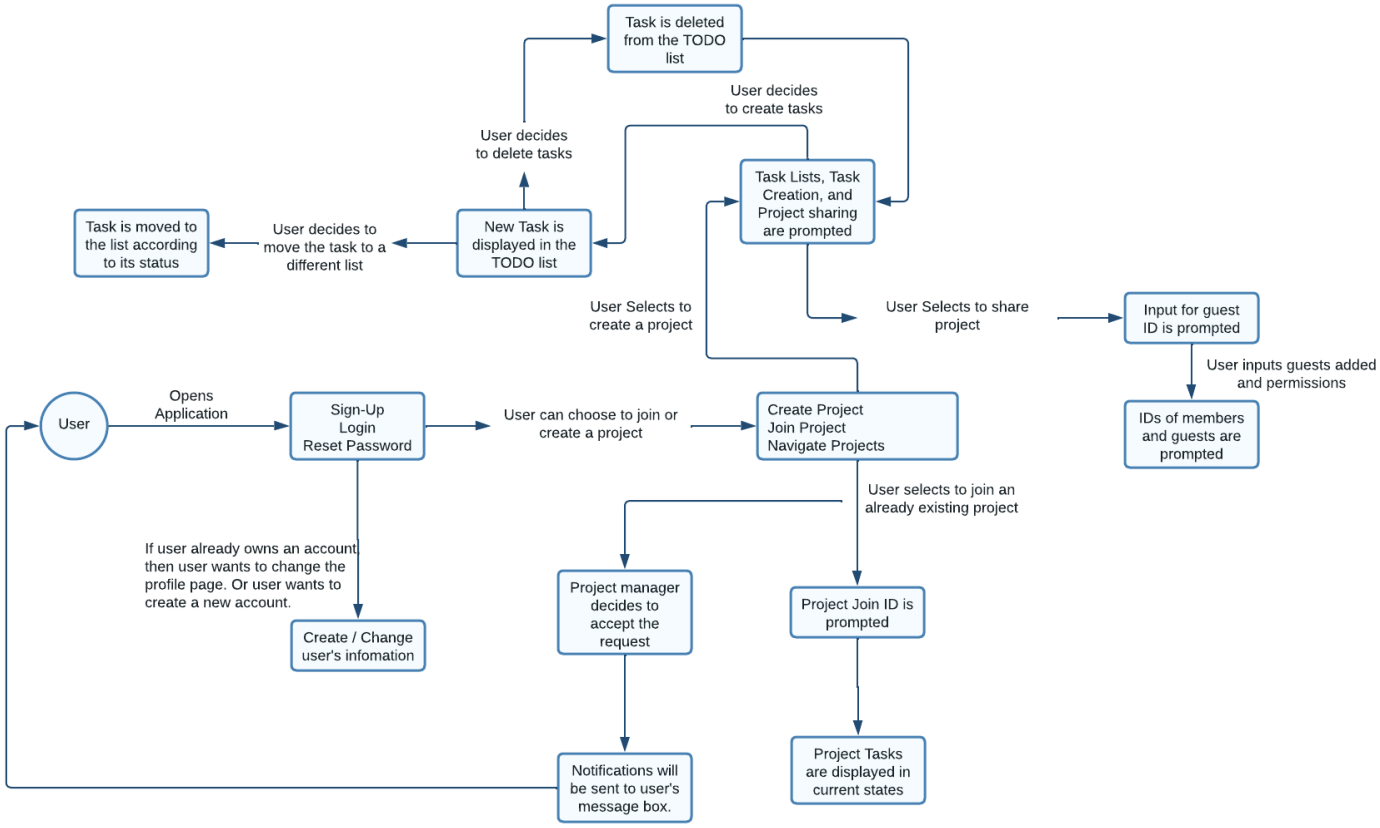


Protocol, Service, and Hosting Overview

The Web Client will interact using HTTP requests. Using HTTP requests, we can communicate and pass data in a standard JSON format to and from the database. Our backend service architecture includes the database and the analytics module that communicate using NoSQL queries.



Activity/State Diagram



Design Issues

Functional Issues

1. What do users need to login to use the application?
   Option 1: No, users can use it as guest
   Option 2: Yes, username required
   Option 3: Yes, both username and email required

   Decision: We chose to implement a user login system but for now email is not required because when we store user data into our database, we would like to keep the database clean and well-organized. From users' side, they would like to see all their projects each time they come back to the application. In conclusion, the username login system is crucial for our application.

2. What should be the association between user roles and task status?
   Option 1: No, only project manager can change the status of tasks
   Option 2: Yes, both members and project manager can change the status of tasks
   Option 3: Yes, but members can only mark tasks as "todo", "in progress", and "in review"

   Decision: We chose to only grant the user partial access to change the status of tasks, and the project managers will have all the access to make the changes. The reason we make this decision is because if the project manager does all the job for status checking and modifying, the usability of the application as users will decrease. But if the users have all the access to making changes on task status, it is potentially going to cause chaos in the project. In conclusion, we decided to give users partial access to task status modification.

3. How should different components in the application be displayed?
   Option 1: All components keep on one view-window
   Option 2: Different components locate in different view windows

Decision: For different components in the application, for example, the calendar view, the task board, the account management, and etc, we decided to put them in different view-windows. The main reason we decided to separate view-windows is that we want to offer high-quality user experience and when all the functions/features are crowded in only one view-port, it will be difficult for users to navigate through.

Nonfunctional Issues

1. How are we going to host our backend services?
   Option 1: Google Firebase
   Option 2: Amazon Web Services

   Decision: We chose Google Firebase for hosting our backend services. Compared to AWS, Google Firebase provides a great number of built-in functions, which lessen the time needed for the programmers to implement and maintain backend databases. As a newly developed application, Google Firebase provides a cheaper price. Moreover, projects on Google Firebase can be deployed on AWS easily if we have the needs.

2. Which languages are appropriate for implementing our backend services?
   Option 1: JavaScript
   Option 2: C++
   Option 3: Java

   Decision: We decided to use JavaScript as our main developing language because it is the best fit for single page application development. Another reason for using JS is because Google Firebase has covered most of the backend development for our project which means there is no specific need for us to use languages like C++ and Java.

3. What type of database is most appropriate for our data?
   Option 1: MySQL
   Option 2: SQLite
   Option 3: NoSQL (Google Firestore)

   Decision: NoSQL is the best fit for our application because we chose to deploy our single page application on Google Firebase which provides built-in functions and packages for NoSQL database operations including fetching and storing. The other two options will be alternatives in later upgrades only if we decide to move the application to a platform like

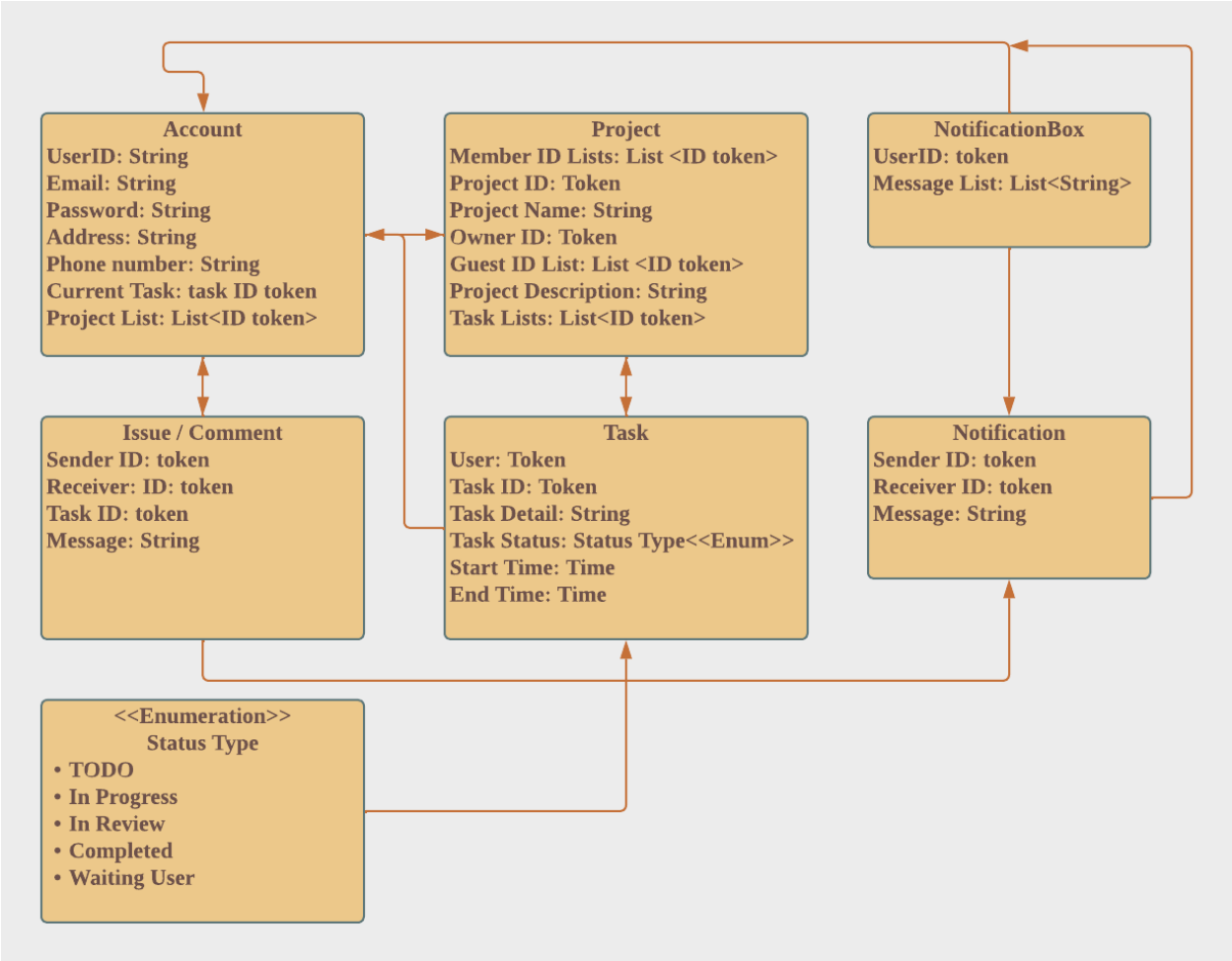
AWS. Overall, NoSQL is the best option for us at this point.

4. How can we optimize keeping the Web-Client up-to-date with the most current information?
   Option 1: ReactJS
   Option 2: VueJS
   Option 3: AngularJS

   Decision: For our application, we decided to use ReactJS. We found that due to ReactJS' Componentized design allows the web application to render the up-to-date information in real time. Additionally, ReactJS allows for many other built-in functionalities that all allow for extremely efficient page rendering and component updates. Compared to the other two frameworks, ReactJS is easy to learn, use and provides powerful functions.

Design Detail

Data Class Overview



Data Class Detailed Explanation

Our data class design is based on the relationship between users and projects and tasks.

User:

- Represents a member who is enrolled in this project.
- Create when anyone creates his/her account.
- Contains different kinds of positions (user/project manager)
- Contains user ID, contact information, projected enrolled ID, and tasks taken.

Projects:

- Represents a collaborative assignment that is planned to achieve a particular aim
- Created by the project leader.
- Contains project ID, project name, project owner's ID, project members ID list, project description, and task ID list.
- This will be the main interaction place for a user on the website, as they can take, edit, and view the project/task.

Tasks:

- All the tasks will be composed of a project.
- Created by the project leader.
- User will take the task to finish the project
- Contains User ID(user who takes this task, empty if no one takes it), tasks ID, task status, task detail, task starting time and ending time, and the total time spend on this task.

Task Status:

- Showed the task status to the user.
- Created by the project leader.
- Modified by the user.
- Status includes TODO, in progress, in review, and complete.

Comments:

- Any user's opinion on any task can write as a comment shown under the task.

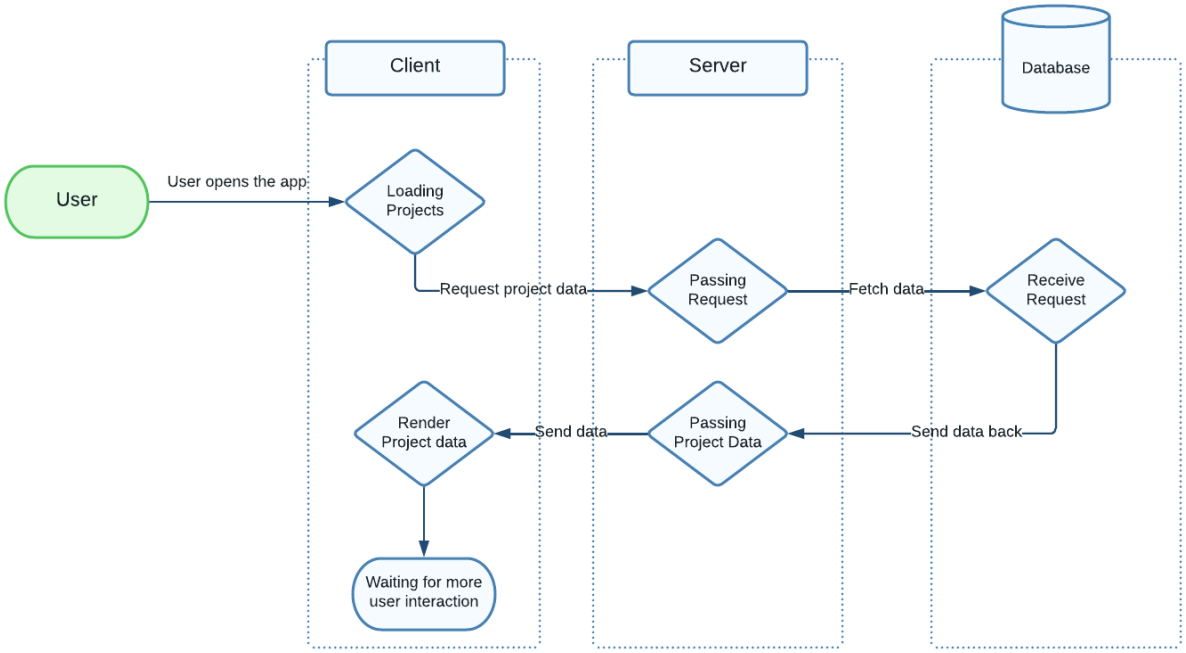
- Created by the user/project leader
- Contains sender ID, receiver ID, task ID, and comment message

Notifications:

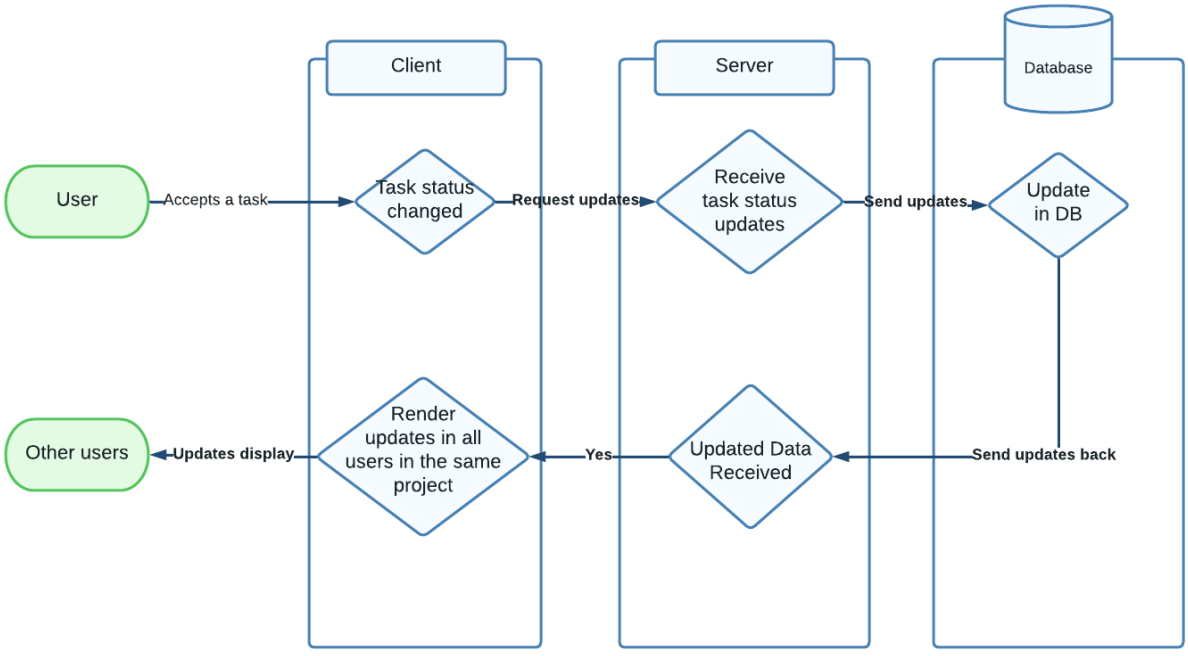
- Notifications on all the messages except the comments on tasks.
- Notifications will all be stored in a notification box.

Interaction between Components

- **User starts the application**

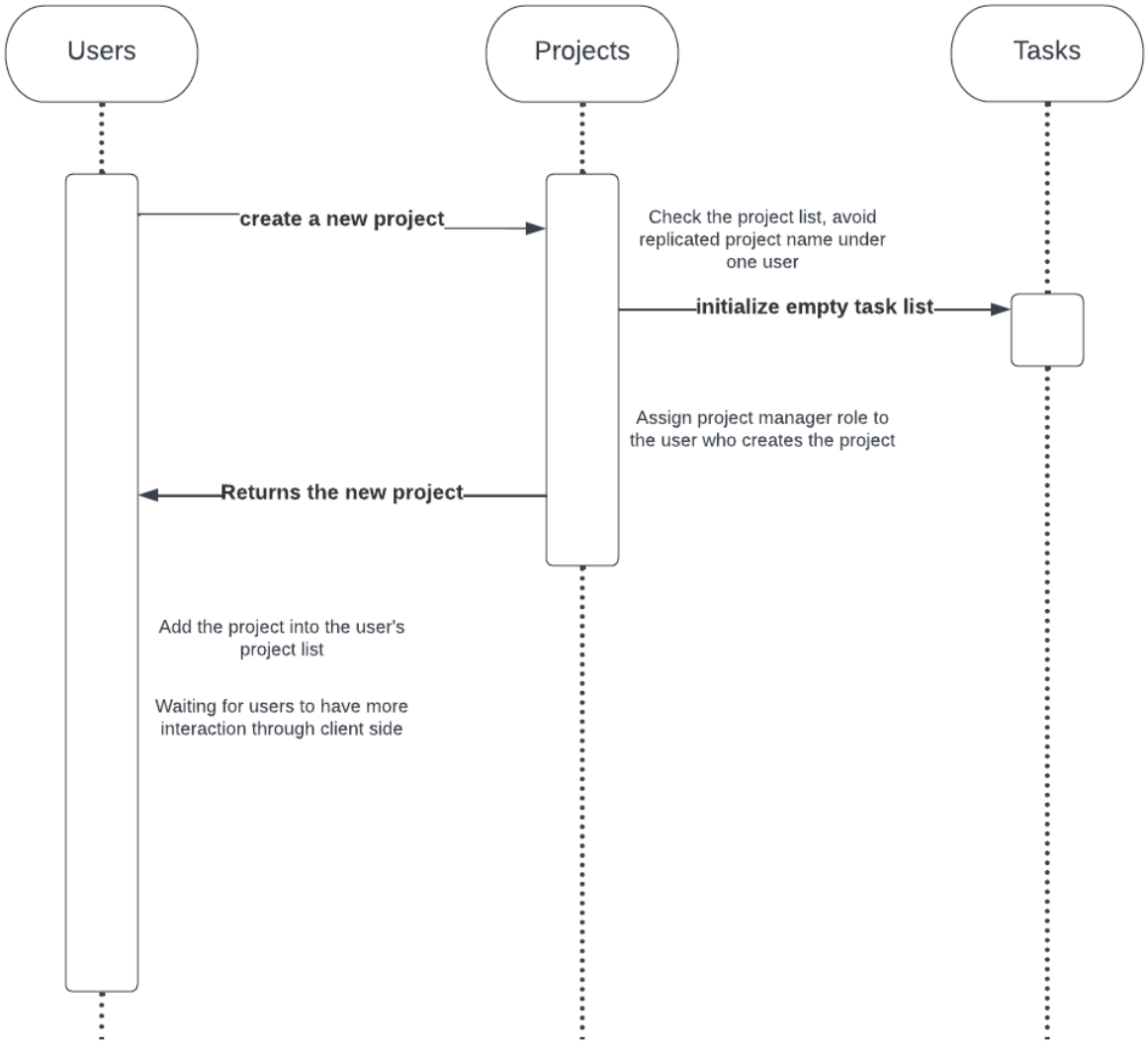


- **User update task status**

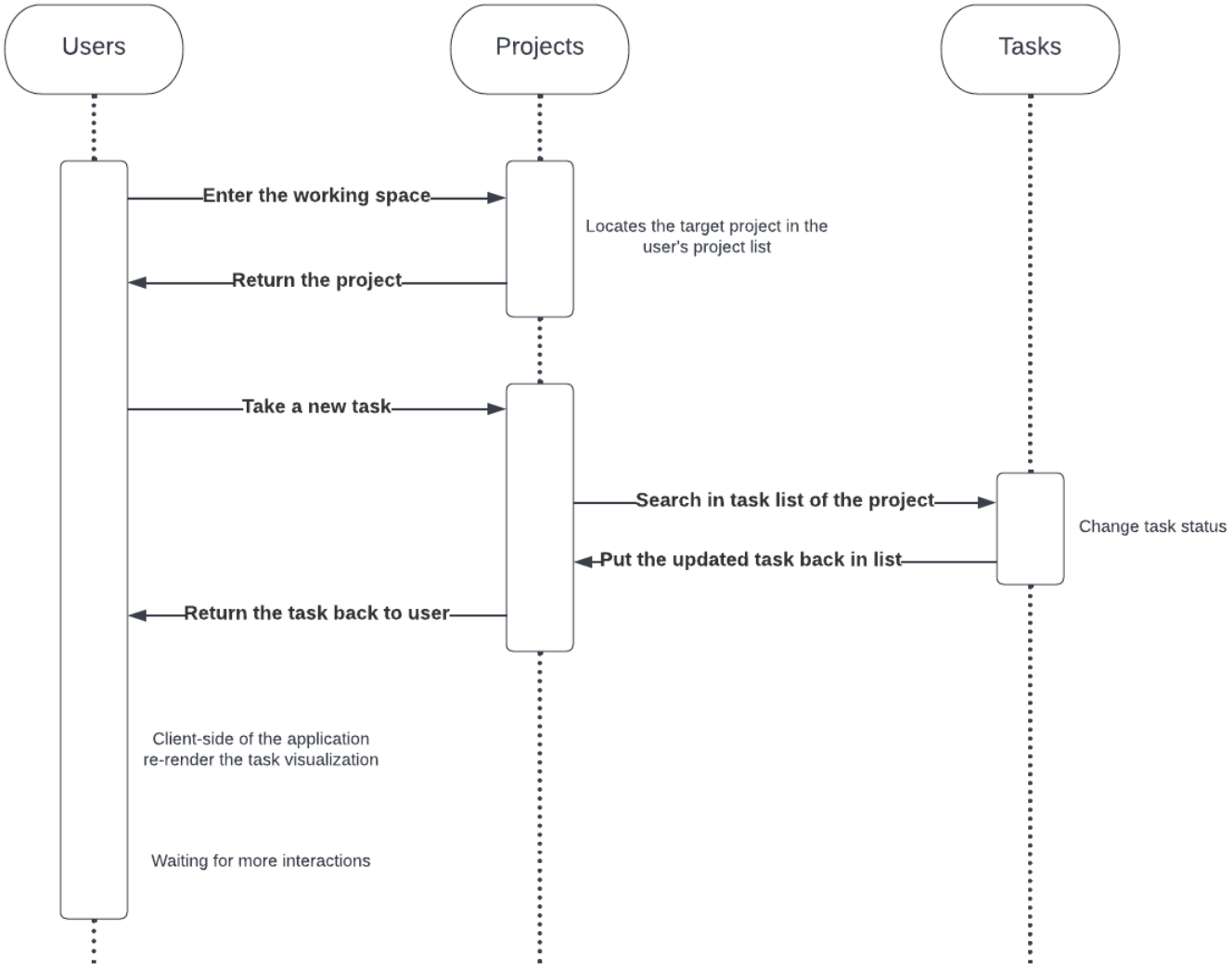


Sequence Diagram

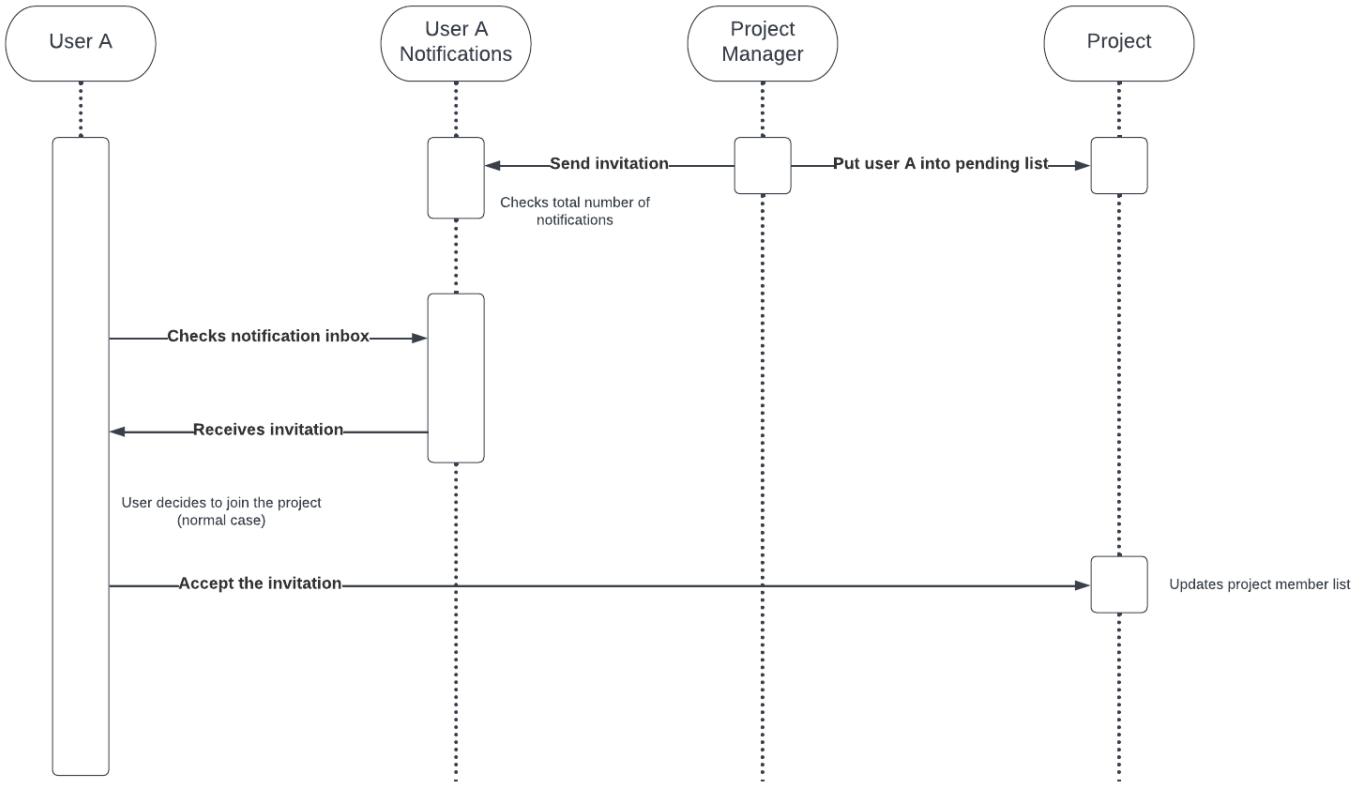
- **User create a new project**



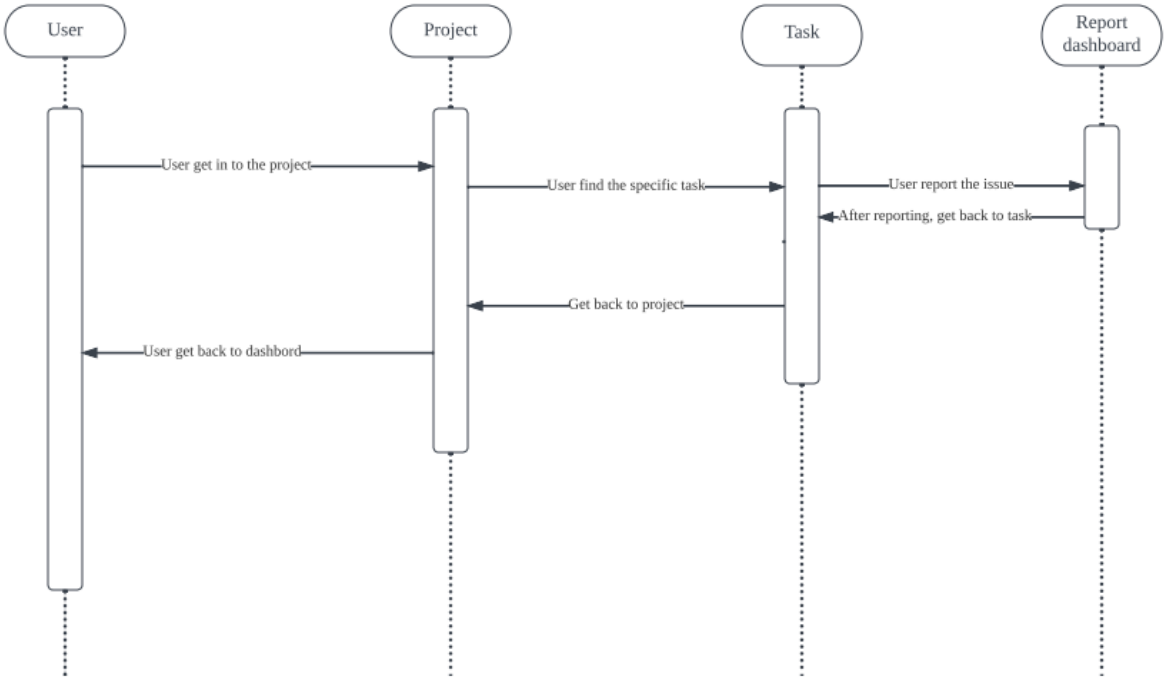
- **User takes a new task from a project**



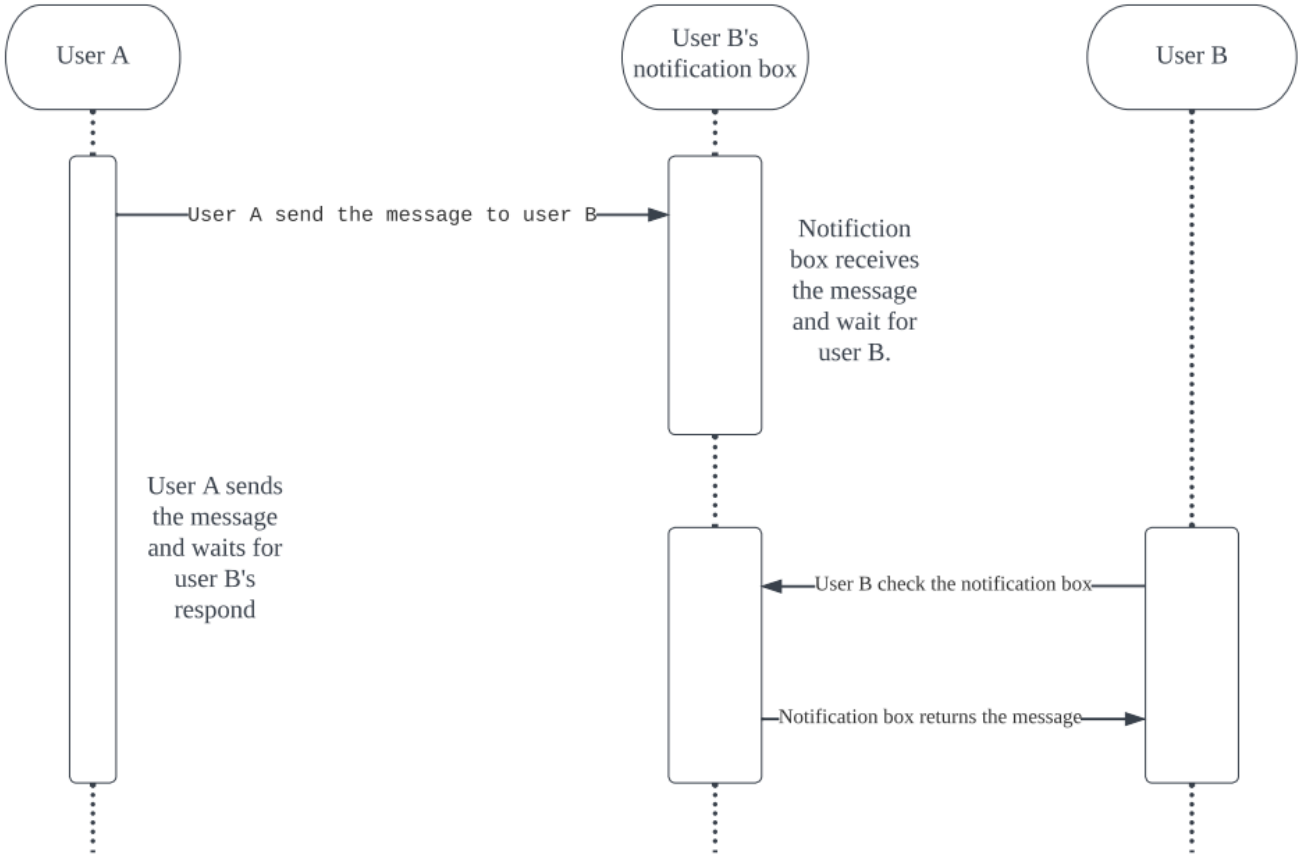
- **User join a project by invitation**



- **User report an issue about a task**

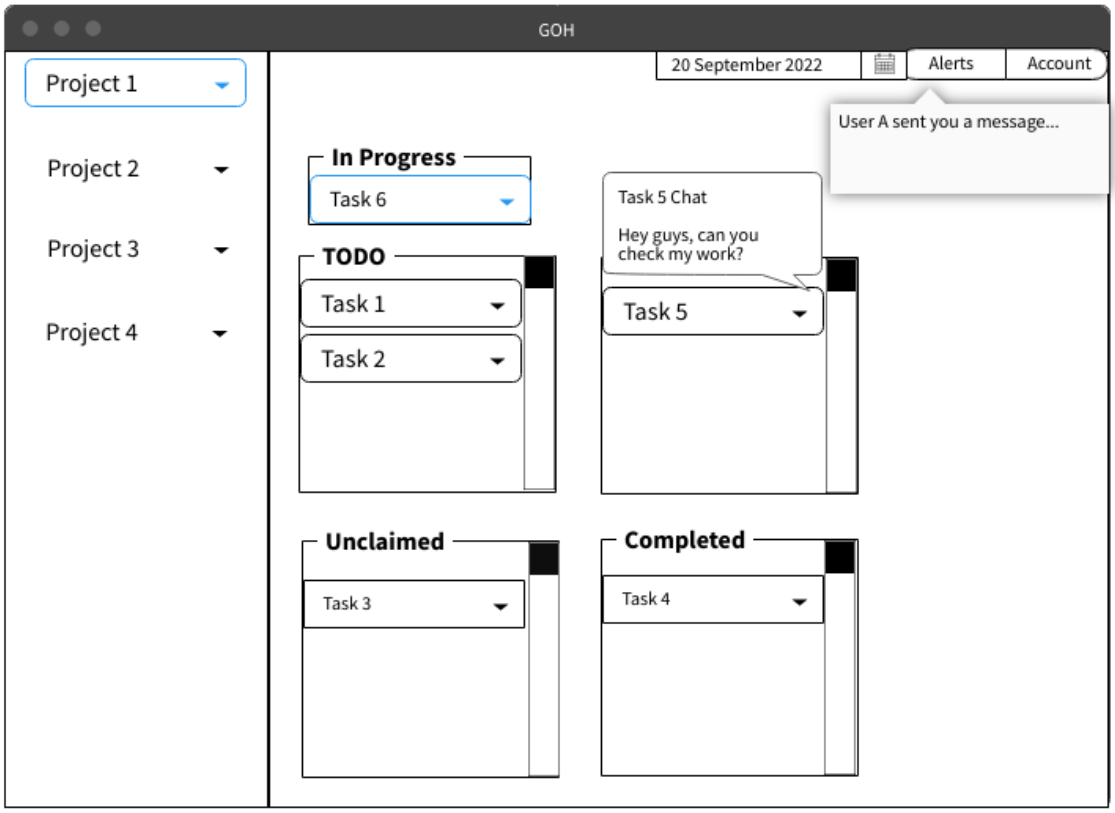


- **User send a message to another user**

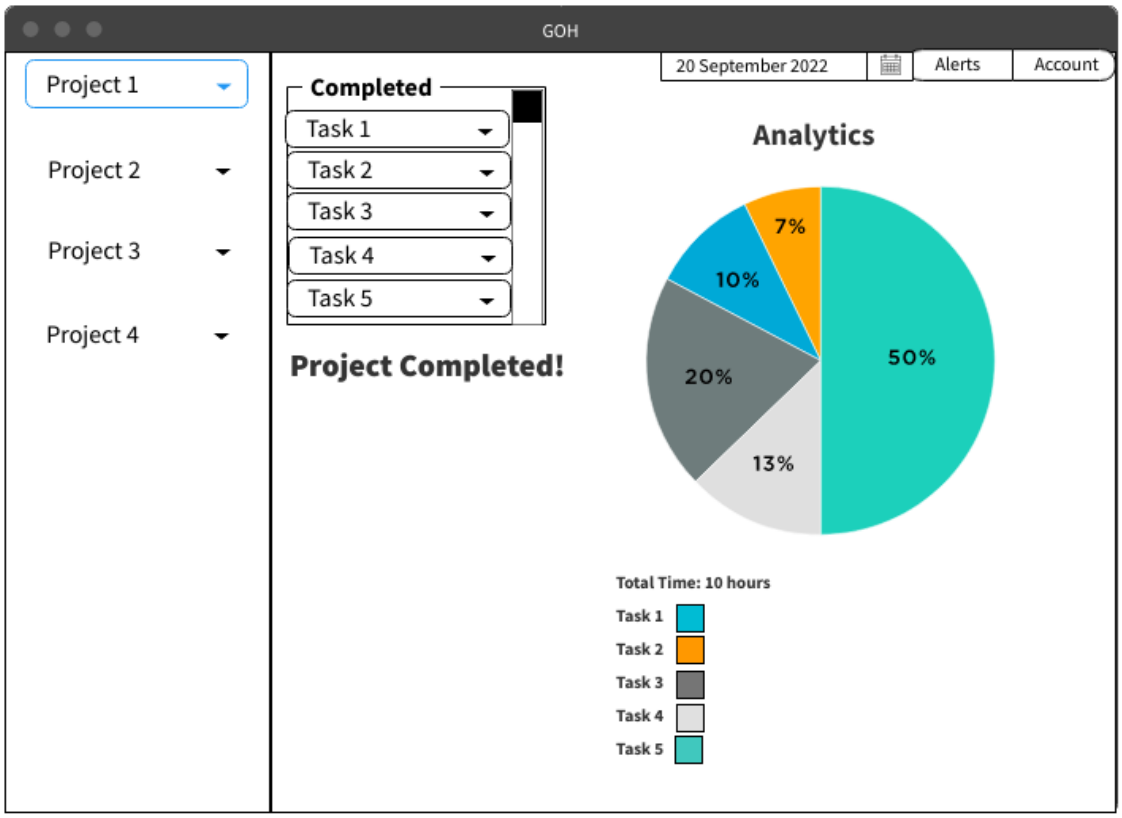


UI Mockup

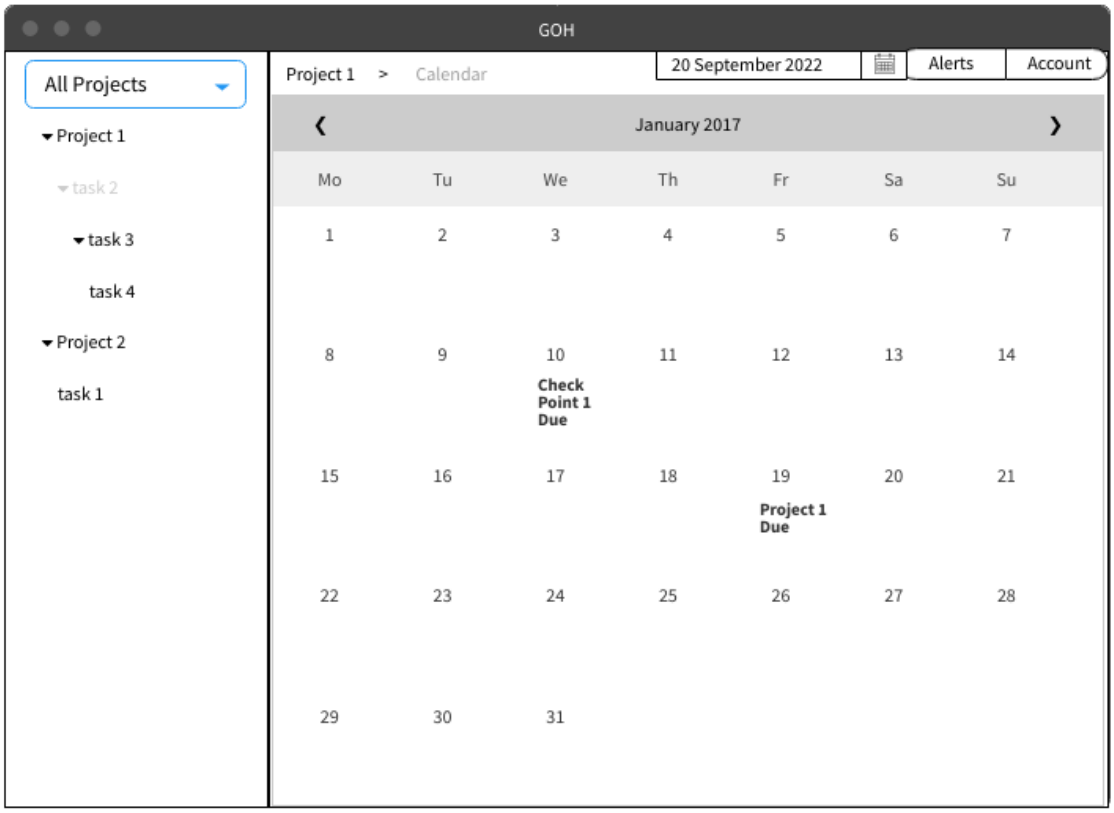
- **Project Home Page**



- **Analytics Page**

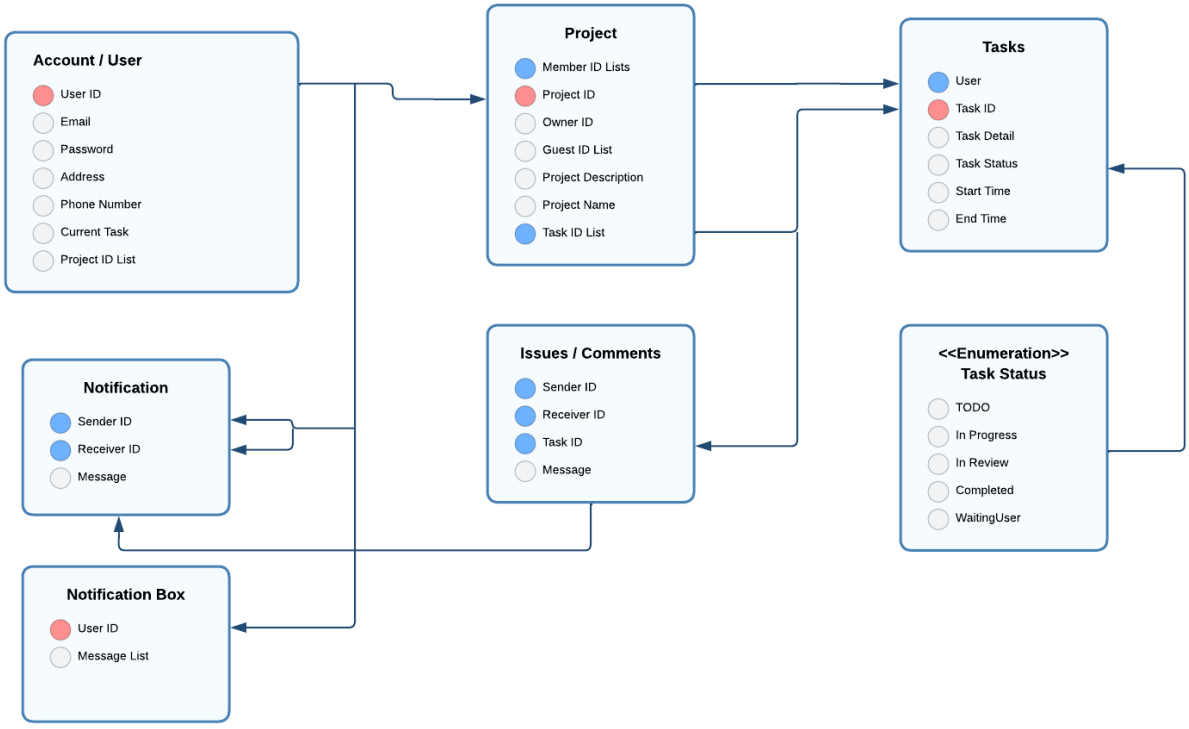


- **Calendar Mockup**



Database Design

We are using a noSQL database that is hosted by the Google Firebase service. Within the database, there will be data classes that are identified in our domain. Within each table, there is a foreign key, represented by a blue circle, and a primary key, represented by the red circle. In addition, we will implement various Views, triggers, and store procedures that will allow the database to run more efficiently.



API Routes

The API Routes are used to define the communication from user end to server and database end. It defines a clear boundary between the low-level data and user interaction on the high level user interface. Each route represents an object or set of objects that can be provided by the server (via GET requests), or created and updated by the client (via PUT, POST, and DELETE requests).

Route	**Supported Methods**
/user	GET
/user/{id}	GET
/user/{id}/{email}	GET, POST, PUT
/user/{id}/{password}	GET, PUT
/user/{id}/{phoneNumber}	GET, POST, PUT
/user/{id}/{address}	GET, POST, PUT
/user/{id}/current_task/{id}	GET, POST, PUT, DELETE
/user/{id}/project_list/{projects}/{id}	GET, POST, PUT