

Enjoyagoals

Design Document

Team 39:

- Prahas Pattem
- Nick Norton
- Karim Mammadli
- Nabi Nabiyev
- Ryan Chang

Index

Purpose:	4
Functional Requirements:	4
Non-Functional Requirements:	6
Design Outline:	7
High Level Overview:	7
Sequence of Events Overview:	7
Design Issues:	8
Functional Issues:	8
Non-Functional Issues:	10
Design Details:	12
Class Design:	12
Description of Classes and Models:	13
Sequence Diagram:	15
Navigation Flow Map:	19
UI Mockups:	20
Registration	20
Main UI	20
Leaderboard (detailed view)	22
Task Creation Interface	23
Pending Tasks View	24
My account info	25
General Chat	26

Purpose:

As humans, we have made almost every facet of life as enjoyable as possible—except for organizing and working towards goals. Well-defined goals and planning have always been at the foundation of society. As society has progressed, we have developed ways to make structuring projects *easier* but not necessarily more fun. Why not? Nearly everything in people's lives requires some amount of planning. Although there are existing services that allow you to organize your goals, they fail to take advantage of people's desire to receive instant gratification and to have fun.

Enjoyagoals is unique because it is based on the idea that people like games and reward systems. This application motivates people to be productive because it makes tasks or projects feel like a game, capitalizing on our desire for immediate rewards, pleasure, and feedback. This may sound unsettling, but the result is that the environment in Enjoyagoals feels much friendlier compared to others.

Functional Requirements:

- **As an individual user, I would like to be able to:**
 1. register for an Enjoyagoals account.
 2. login and manage my Enjoyagoals account, and reset my password.
 3. register using my google account if I do not have an Enjoyagoals account.
 4. easily set a custom profile picture and customize my profile page's theme and banner.
 5. display my email account and phone number on my Enjoyagoals account alongside other personal information for others to see.
 6. view all of the tasks I need to finish in a comprehensive list after logging in
 7. easily check the details of each individual task (description, date created, deadline, point value).
 8. easily view all of my completed tasks in a list along with my total points accumulated and number of tasks completed in a graphic visualization.
 9. join, switch between, or withdraw from different project rooms at any time after signing in.
 10. purchase exclusive profile pictures or emotes with points that I have earned from tasks to modify my user experience.
 11. view a calendar with the project and task deadlines listed clearly.
 12. receive notifications through email and/or text for approaching deadlines and disable them if I want.
 13. view my level based on the number of tasks I have completed.
 14. easily join a team with either a designated team leader delegating tasks to others or a team without a designated leader democratically deciding what tasks should be done.
 15. view a progress bar visually showing progress made towards the end goal.
 16. (if time allows) view a rough estimation of the time to complete a task.
- **As a team member, I would like to be able to:**

1. view tasks to be completed, their point values, and their status.
 2. indicate that I am currently working on a task and then add notes if I feel like I need to share information with my team.
 3. upload files from my computer to my Enjoyagoals group(s) when I have completed a task.
 4. indicate that I have completed a task.
 5. view the number of the tasks that have been completed.
 6. view the tasks that are being completed by other team members.
 7. view the details of each completed task, such as the date & time marked as complete, who completed the task, the point value, etc.
 8. communicate with other team members through a messaging system.
 9. start and participate in a voting system in case the team needs to kick a member.
 10. view the leaderboard that shows the other team members' levels & rankings.
 11. query the leaderboard.
 12. generate a pdf file that includes a progress report for every team member.
 13. (if time allows) implement a coup d'état system to remove the leader.
- **As a team member with a team leader, I would like to be able to:**
 1. request that the team leader reviews my work & provides either points or feedback.
 - **As a team member without a team leader, I would like to be able to:**
 1. propose tasks to be completed as well as point values for the tasks to be completed.
 2. vote on each task proposal and each point value proposal.
 3. request that the rest of the team review my work.
 4. receive feedback and points for the tasks I have completed.
 5. view a leaderboard showing everyone else's contributions, levels, and rankings.
 - **As a team leader, I would like to be able to:**
 1. invite team members to my team at any point.
 2. create tasks to be completed with the details of each task clearly displayed, such as the date & time marked as completed, who completed the task, the point value, etc.
 3. assign point values to each task created.
 4. provide feedback for every task completed.
 5. view a task's status.
 6. forcibly remove a team member from a task they are currently working on.
 7. easily review any finished tasks to ensure proper completion.
 8. view a leaderboard graphically visualizing each team members' contributions, levels, and rankings.
 9. create announcements that will notify all of the members in my room through email and/or text.
 10. easily transfer my leader role to another member if I leave the room.
 11. assign moderators with less authority than me but more than the other members to my rooms.

12. edit pending tasks assigned by the rest of the team members before approving or ignoring them.
13. (if time allows) deduct points from team members if a task was not completed properly.

Non-Functional Requirements:

- **Architecture and Performance:**

- The application will have a completely separate backend and frontend. This will allow us to work on the backend and the frontend simultaneously so we can avoid compatibility issues and distribute work efficiently.
- The backend will implement a RESTful API (Representational State Transfer) in Node.js and Express. Node.js is a robust framework that offers multiple features to make our web application scalable. Additionally, it is an asynchronous language and is extremely fast compared to most other languages. Express is used to handle various HTTP requests and to write responses to specific URLs. When Express and Node.js are used together, the response time for the application is low.
- The frontend will be developed using React and will extract data from the backend by triggering API endpoints. Moreover, development in React is very fast because it uses virtual DOM and reusable components that developers can use. We will be using React's Axios library to send asynchronous HTTP requests to REST API endpoints on the backend.
- If the frontend and the backend are independent of each other, the application can always be rewritten in other languages or frameworks and expanded onto other platforms.

- **Security**

- Making the application secure and safe is critical as we will store sensitive user information. MongoDB allows us to store data with authentication, access control, and encryption features. This will ensure that sensitive data is stored securely and cannot be exploited easily. We will also ensure that users can only view the project rooms they have access to, not someone else's. Additionally, if a team decides to have a team leader, they will be responsible for adding or removing tasks and determining whether a team member has completed a task. Every request to trigger an API endpoint will be authenticated to ensure user safety.

- **Usability**

- The interface should be simple and intuitive while having a fun design reminiscent of a video game. Features should be easy to group into modules, so the interface should not be cluttered with buttons or fields. The leaderboard inside the room should be reasonably straightforward to understand and interact with. Looking at certain rooms' leaderboards while not in any room is done by simply hovering over the room on the room selection interface. Our application will be usable on all devices with no significant change in design.

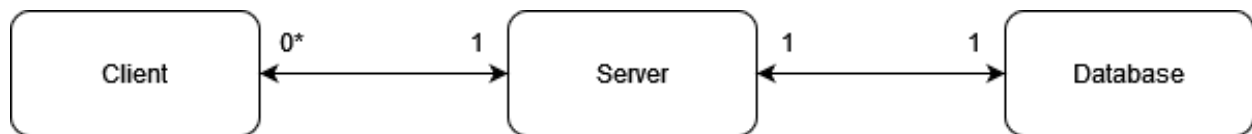
- **Hosting/Deployment**

- The backend and frontend will be developed independently at the same time, and they will also be deployed independently. The node server on the backend will be deployed on Heroku, and the frontend React will be deployed on Netlify.

Design Outline:

High Level Overview:

Our project is an application that helps either an individual or a team to accomplish their goals using gamification. Our application will follow a Client-Server model. Multiple users will be allowed to concurrently communicate with the server, and the server will access the database for any data that is required.

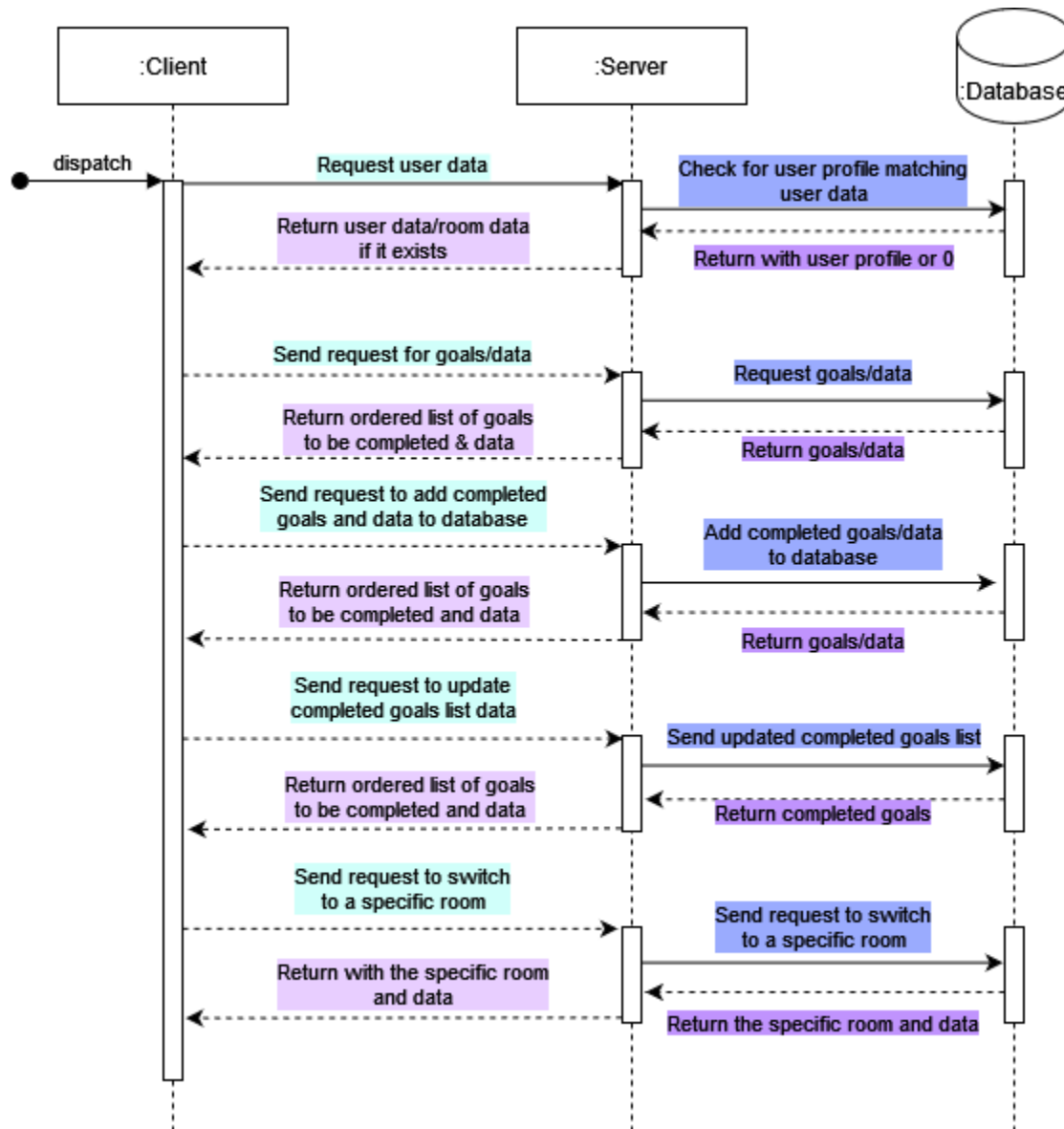


- **Web Client:**
 - The interface to our system will be a client accessed through a web browser.
 - The client will retrieve and send data from the server using HTTP requests and JSON.
 - The client will parse, format, and display the data received from the server.
- **Server:**
 - The server mediates between the database and the client, providing a layer of abstraction between the user and the database.
 - The server will handle HTTP requests from the client, and respond with an HTTP response that has data from the database in JSON format.
- **Database:**
 - A relational database stores user data, task data, etc.
 - Database responds to requests from the server and sends the requested data back.

Sequence of Events Overview:

The sequence diagram below shows the typical interaction between clients, the server, and the database. The sequence is initiated by a user launching the web app. When the user tries to log in, the client sends a request for user data to the server. The server handles the request, queries the database for a user profile matching said data, and receives the corresponding data from the database. Once the user successfully logs in, they can send requests to the server for other actions, like managing their profile, joining/switching rooms, viewing and adding tasks, etc.

To respond to these requests, the server queries the database for related data. The database responds with updated data after receiving update requests. Then, the server returns the requested data to the client.



Design Issues:

Functional Issues:

1. What details are required in order to create an account?
 - Option 1: Password only
 - Option 2: Password and username only

Option 3: Username, password, and email address

Option 4: Use your Google account to login

Choice: Option 3 and Option 4

Justification: We need a unique username for every account. As the website is more solely based on teamwork and collaboration, having a username makes it easier for people to find and keep track of you. A working email address is required for verification and in order to stop spamming of account creation. At the same time, the email will be used to reset the password. To reduce the burden of login for users, we also decided to implement a Google account sign in which is a secure authentication method.

2. How will the point system work for each user?

Option 1: Static point increase

Option 2: Point increase based on the complexity of a task

Option 3: Option 2 with point deductions if the task is not completed on time

Choice: Option 2

Justification: A proper point system is important for users to compete with one another and advance their rank. The website becomes more gamified and engaging by implementing a point system based on the complexity of a task because it requires users to think critically about whether it would be more effective to work on a difficult task with a higher reward or to complete multiple easy or medium level problems to get equally advanced on a leaderboard. We also decided not to use a point deduction system because it would blur the website's purpose of being both a fun and competitive environment for users.

3. How should we allow the users to communicate with each other?

Option 1: Real time chatroom

Option 2: Discussion board

Choice: Both

Justification: The purpose of allowing in app communication is to make completing projects more gamelike and more fun. Many games have a real time chat room, so incorporating one into our app would serve to make communication between potential team members more efficient and give our app a more game-like feel. However, adding certain aspects of a discussion board to the list of tasks would serve to make clarifying issues on tasks easier than having to vfy for attention in the group chat and should not be too difficult to implement. Therefore, we decided to do a bit of both.

4. How should tasks be assigned to users in a group?

Option 1: Each user claims tasks in first come first served system

Option 2: Tasks are delegated to each member by the moderator or by other group members

Choice: Option 1 (with elements of Option 2)

Justification: Making tasks first come first served heightens the sense of competition between group members by incentivizing them to take on more tasks more quickly and is easier to implement than the alternative. However, if any one member takes on too many

at once and leaves no remaining tasks for other members, they could be forcibly removed from their task(s) by the group leader or the rest of their group members.

Non-Functional Issues:

1. What web service should we use?

- Option 1: AWS
- Option 2: Azure
- Option 3: Heroku
- Option 4: Netlify

Choice: Azure

Justification: Integrate API Gateway from Azure to handle user authentication and rate limiting. Secure data lake for high-performance analytics. Azure also has a vast amount of features and products that are high in performance and are highly efficient.

2. What backend language/framework should we use?

- Option 1: Django
- Option 2: Flask
- Option 3: ExpressJS
- Option 4: Node.js

Choice: ExpressJS & Node.js

Justification: We will be utilizing the MERN stack for our web app. Express.js is an open-source backend web application framework that is used for building REST APIs along with Node.js. Express.js makes it very easy and fast to design and build web applications. Moreover, Node.js is event-driven and asynchronous, so our application will not die while waiting for data from the server.

3. What frontend language/framework should we use?

- Option 1: React
- Option 2: Vue
- Option 3: Angular
- Option 4: raw HTML & Javascript

Choice: React

Justification: React is used everywhere in web development thus there are a great number of resources such as tutorials, documentations, which can help us with developing the web app. Instead of creating the entire application line by line, it has pre-built functions that can be used like basic elements to develop quick and flexible projects in less time.

4. What database should we use?

- Option 1: MongoDB
- Option 2: MariaDB
- Option 3: Firebase

Option 4: MySQL

Choice: MongoDB

Justification: MongoDB's high speed and higher flexibility are its crucial benefits.

MongoDB is a document-based database meaning information can be embedded inside a single document, which significantly increases its speed compared to other databases. It's also quite simple to use it due to its user-friendly UI.

5. Which API should we use to access location data?

Option 1: REST

Option 2: SOAP

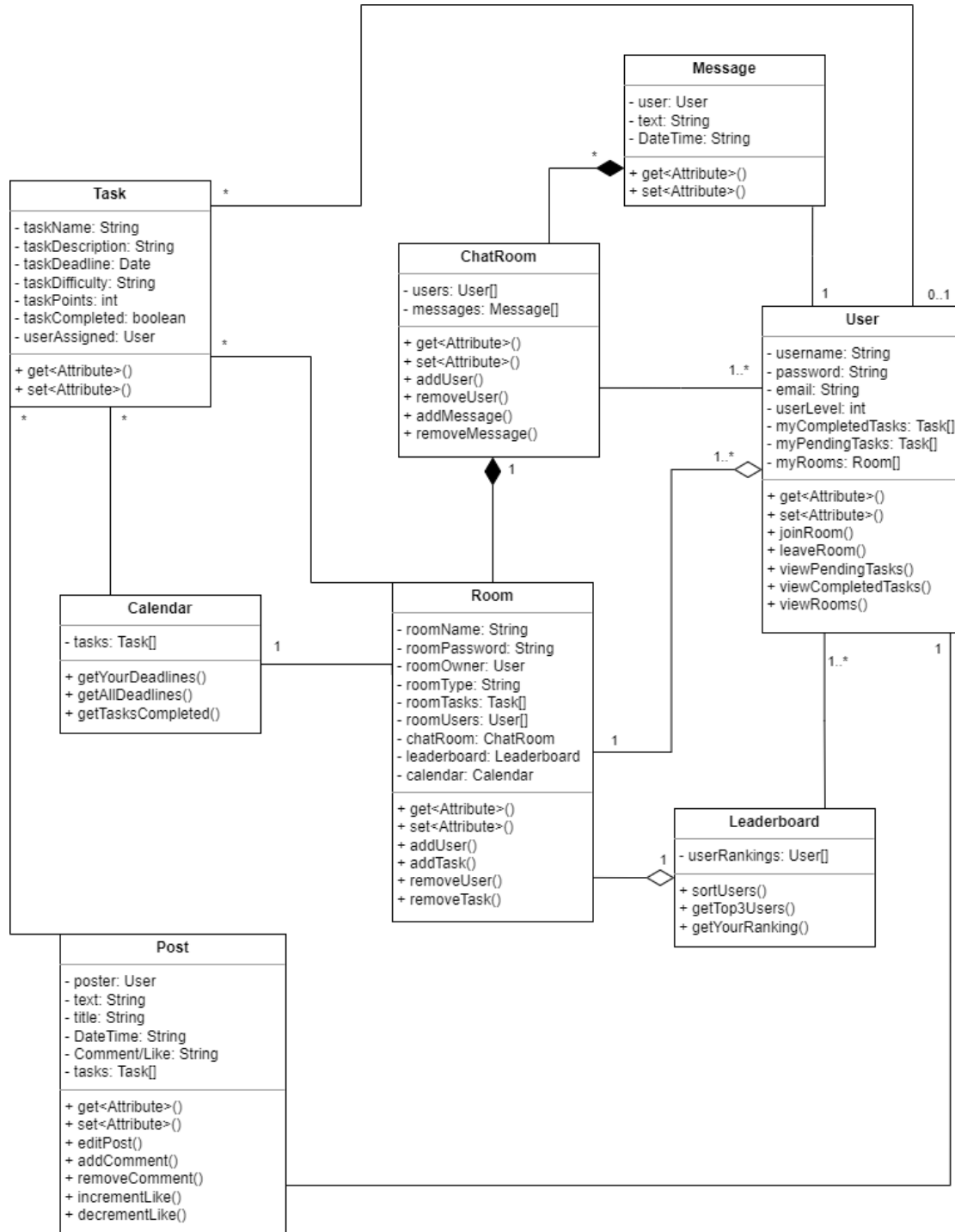
Option 3: RPC

Choice: REST

Justification: It becomes quite simple for developers to store or alter data on the server side while displaying information on the client side with REST API. REST is extremely flexible and integrates very easily with MERN stack since it uses JSON. Moreover, the basic REST HTTP requests are POST, GET, PUT, DELETE, which make the process of create, read, update, and delete (CRUD) very easy to execute since we plan on implementing these functionalities in our application.

Design Details:

Class Design:



Description of Classes and Models:

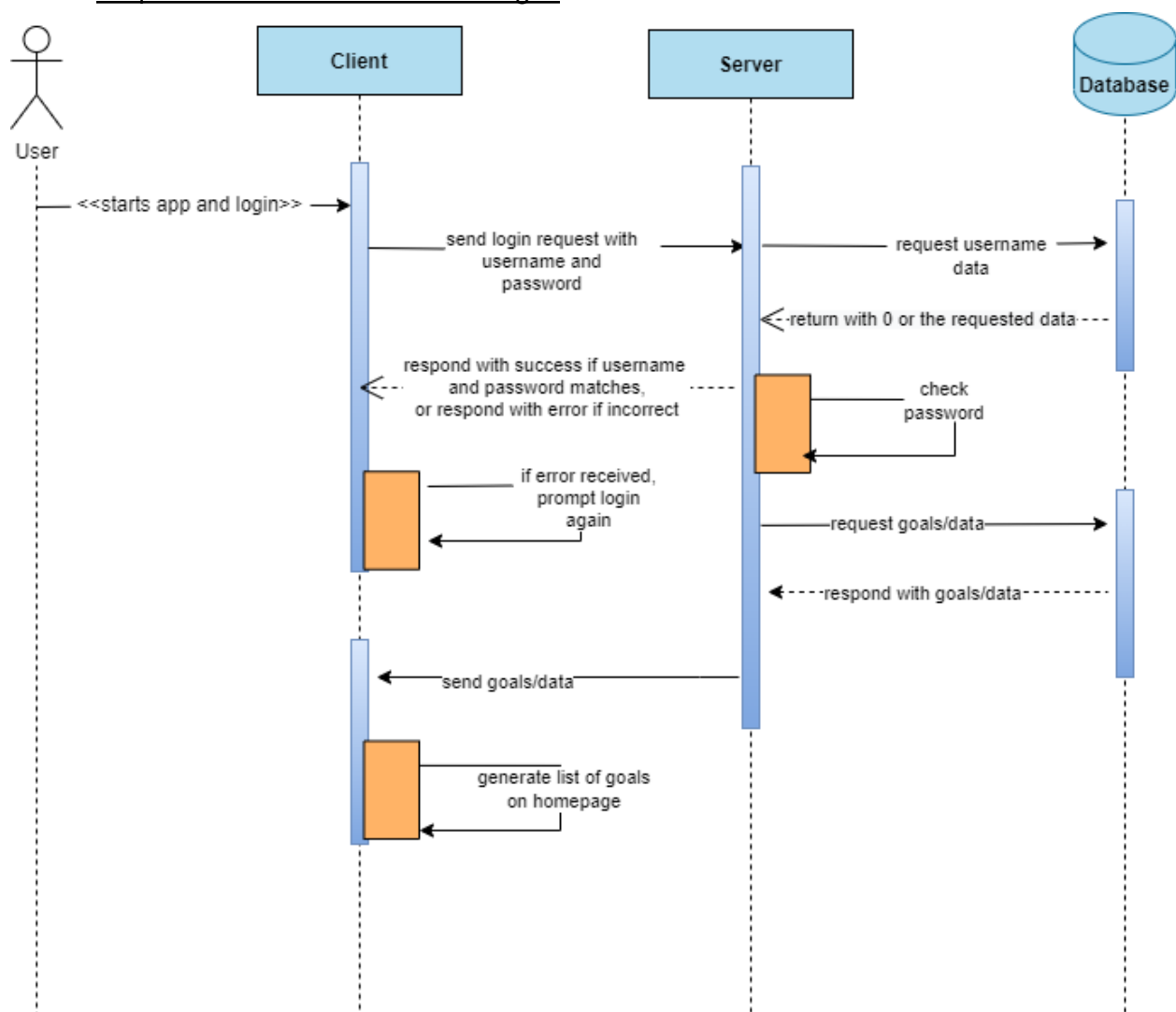
- User:
 - The user object is created upon signing into Enjoyagoals.
 - Each user has a username, password, and email string to login.
 - Each user has a list of tasks they have currently accepted and finished. These pending and completed tasks can appear on the user's homepage, but can be filtered out if they wish.
 - Each user has a rating based on the number of points they have earned from successfully completing tasks.
 - In rooms with a defined room leader, the owner can perform certain actions unavailable to the rest of the room. For instance, they can forcibly remove another member from a task that they are working on, or determine whether a team member adequately completed a task they submitted for review.
 - Each user has a list of rooms they are currently participating in.
 - Users can join or leave rooms at will.
- Task:
 - Task object created when a user creates and proposes a task.
 - Each Task is assigned a name, difficulty, deadline, and point value.
 - Each Task will have a description containing a brief summary of what it entails written by the member creating it along with the above strings.
 - Each Task will be assigned a userAssigned string only when another user has claimed it.
 - Each Task has a boolean indicating whether it is complete or not.
- Post:
 - Post object is created when a user's task proposal is accepted by either the team leader or the rest of the group, but not when the proposal is rejected.
 - The object is what is displayed on the main UI.
 - Each Post has a User, text displaying the Task's description and title, the date and time which it was created, and comments, likes, and upvotes.
 - Posts can be edited by Users.
- Room
 - Room object is created when a user creates a room.
 - Each Room has a name, password, and owner.
 - Each Room has a designated Owner associated with the User who created the Room.
 - Each Room has a list of Tasks associated with it.
 - Each Room has a list of Users associated with it.
 - The leader of a room can add or remove other users from the room.
 - Members of the room can add tasks to the room's list.
 - Tasks are removed from the list when their deadlines expire.
 - Each Room has one ChatRoom tied to it.

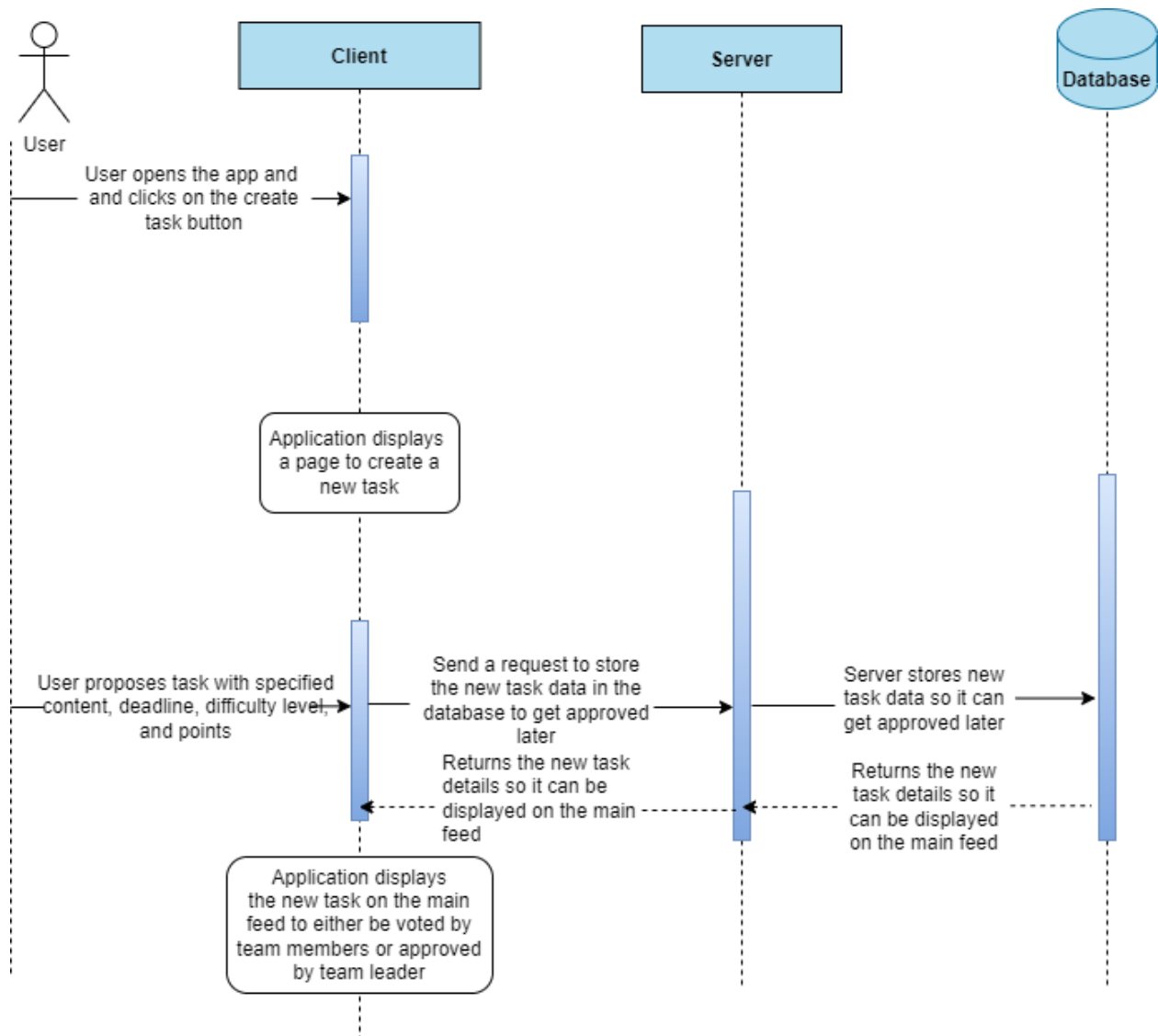
- Calendar
 - Calendar object is created with the creation of a Room object.
 - Calendar displays deadlines for tasks you accepted and the tasks for everyone in the Room it is associated with.
- Leaderboard
 - Leaderboard object is created with the creation of a Room object.
 - Contains a list of users.
 - Leaderboard automatically sorts users by descending points, with the user with the highest point values displayed at the top.
- ChatRoom
 - ChatRoom object is created with the creation of a Room object.
 - Every user in the same Room who has been added to a ChatRoom can freely add or delete Messages from it.
 - Only one ChatRoom for each Room can exist at a time.
 - Displays Message Objects in real time.
- Message
 - Message Object is a composition of the ChatRoom.
 - It is created when a user sends a message to the rest of the ChatRoom.
 - Each Message contains a user, text, and the date and time at which it was sent.

Sequence Diagram:

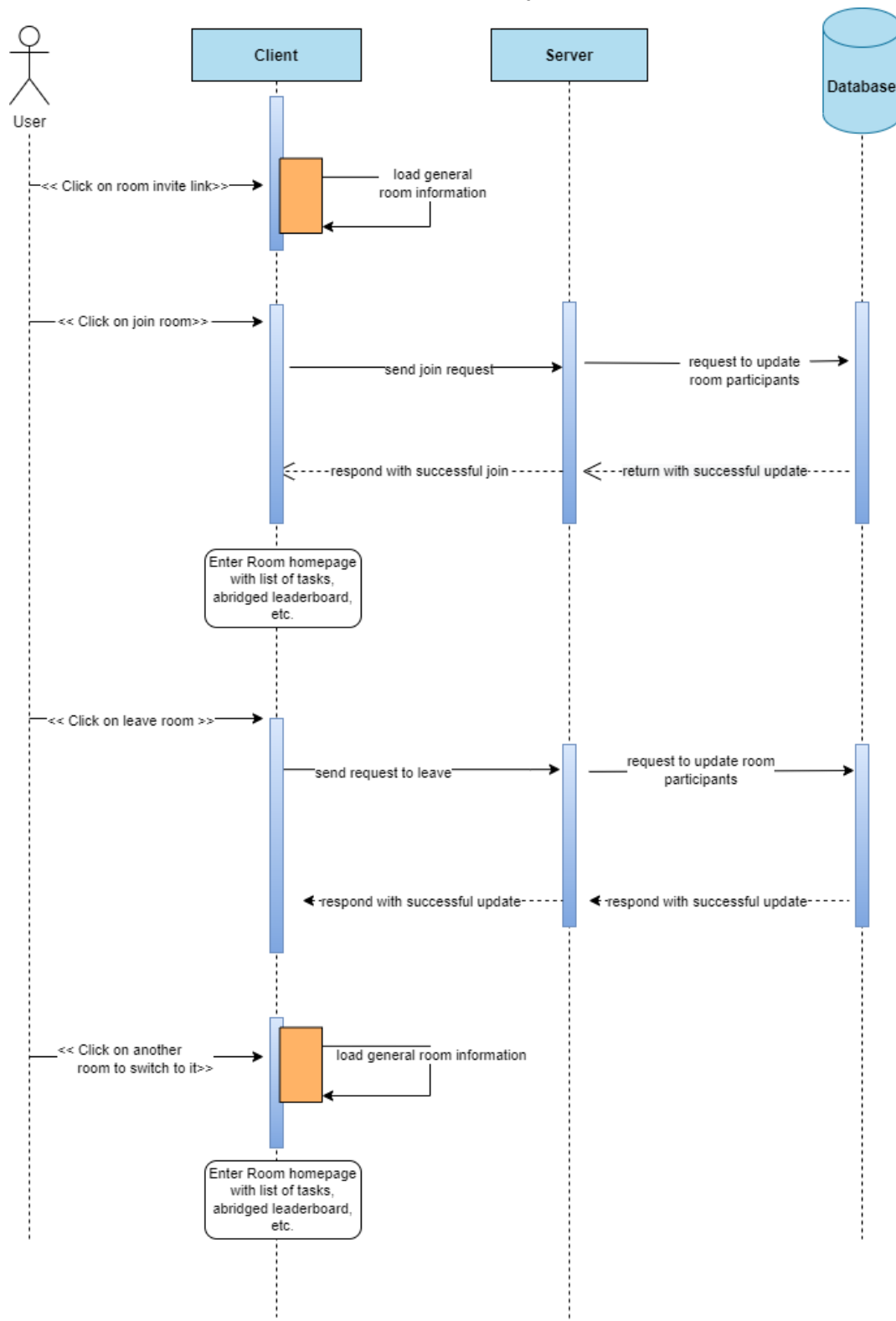
The following diagrams portray the sequence of events for the major functionalities of our application. The sequences we decided to depict are user login, creation of a new task, all possible sequences for rooms (joining, leaving, and switching between rooms), and sending a message in the chatroom. All the sequences follow the client-server model. The client sends a request to the server, and the server will pull data from the database. The server can then process this data and send it back to the client. This data can either be further processed by the client or it can be directly displayed on the UI.

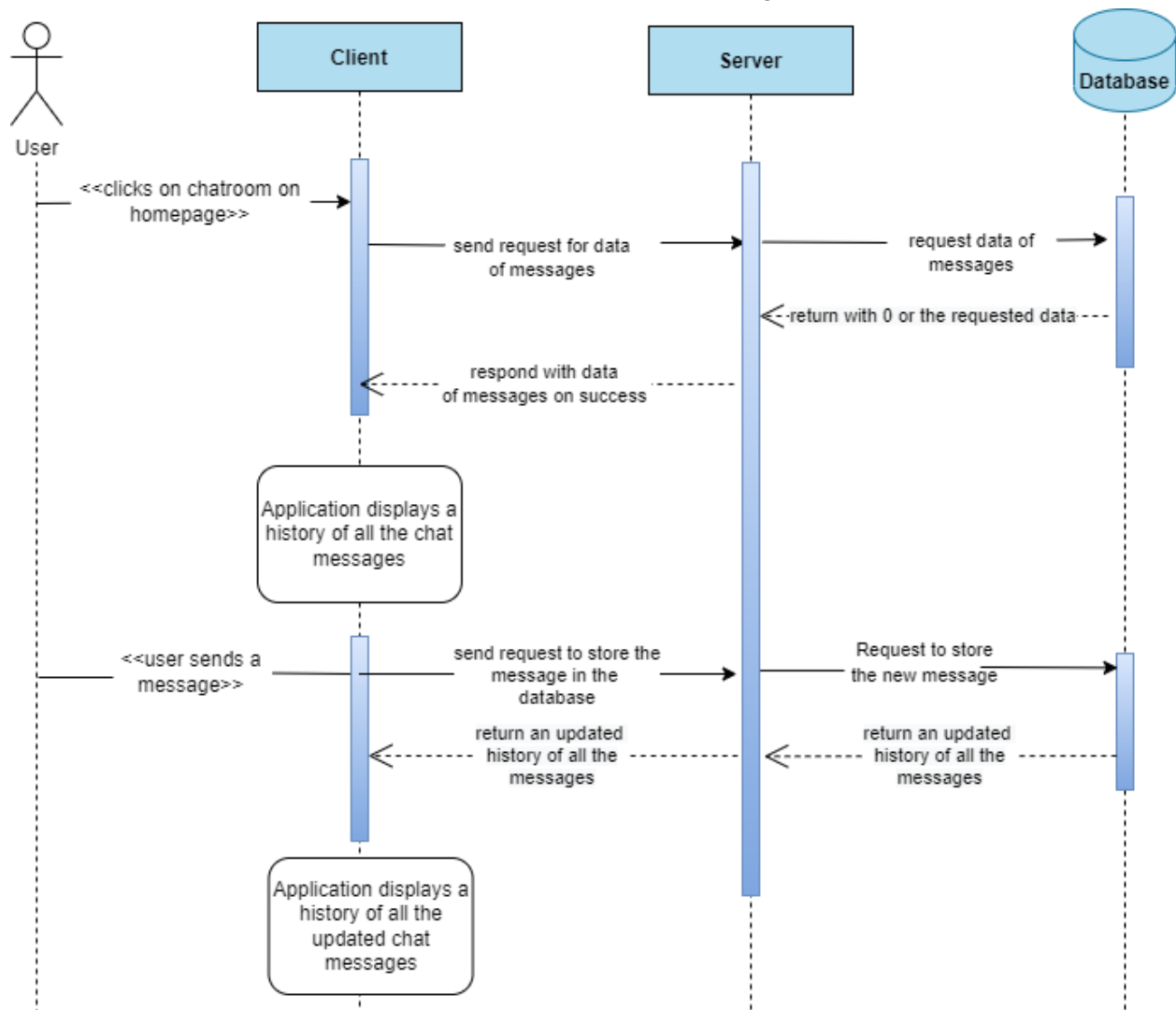
1. Sequence of events when users log in



2. Sequence of events when users wants to create a new task

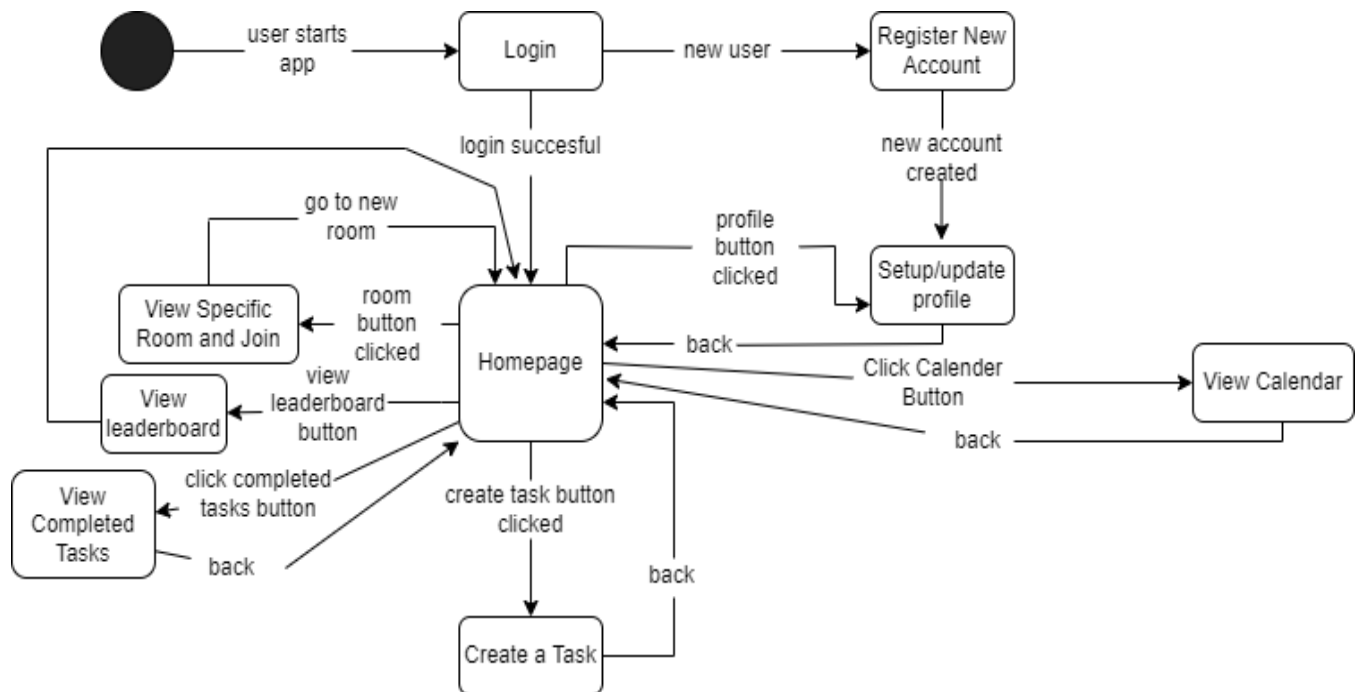
3. Sequence of events when users want to join rooms



4. Sequence of events when users want to send a message in the chatroom

Navigation Flow Map:

Our application was designed to make it easy for users to navigate the application with ease. As soon as the user starts the application, they can login and go to the homepage where they are free to switch to any room of their choice. If the user does not have an account, they can register for an account, after which they are redirected to the homepage. On the homepage, there are multiple buttons the user can click on. The user can click on their profile button and view their profile information and update their credentials if they want to. Or they can click on the new task button and create a new task which they can later send in for approval. The user can also click on the leaderboard button when they are at the homepage to view the entire leaderboard. The completed tasks list can also be viewed by the users by clicking on the completed tasks button.



UI Mockups:

Registration

Welcome to EnjoyaGoals!

Username

Email

Password

Confirm Password

Already have an account? Log in

Create an account

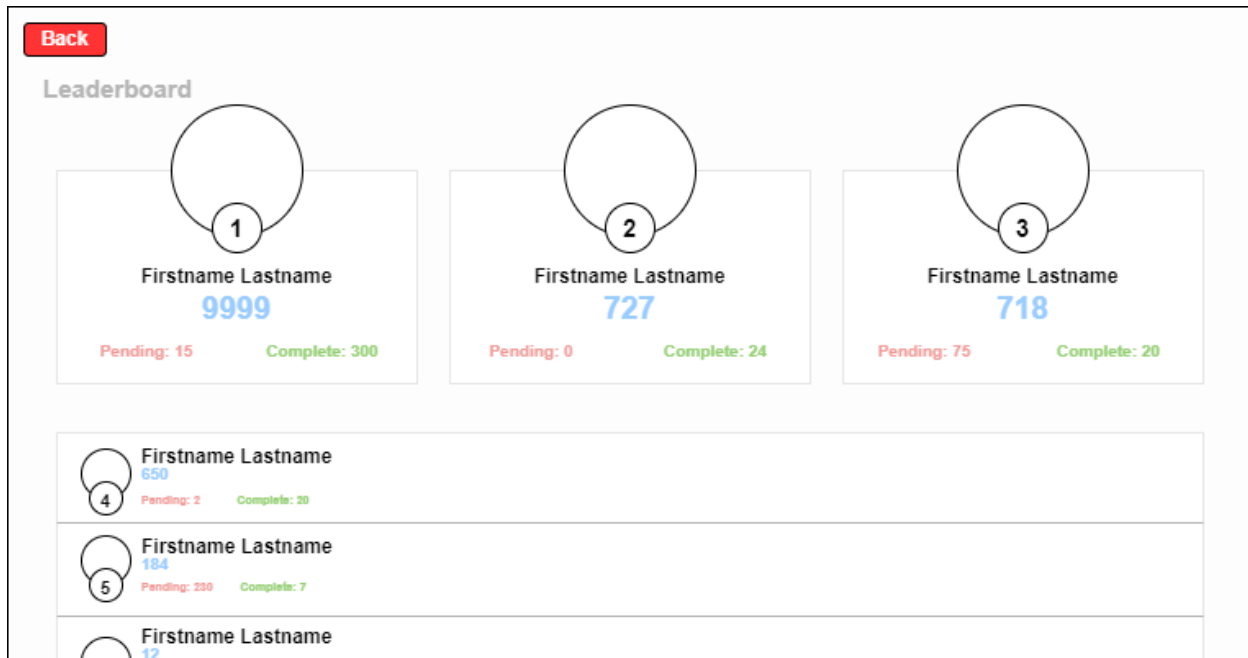
The design above is the signup page where the user is required to include a username, password, and email address. If you already have an account, the button at the bottom left allow the user to go to login page if you already have an account, while the button at the bottom right allows the user to create an account.

Main UI

				Room Name	01/01/2022		Leaderboard
		name 09/20/2022			1		name
					2		name
					3		name
		name 09/20/2022			4	name	
					5	name	
					6	name	
					7	name	
					8	name	
					9	name	
					10	name	
					37	name	
					56	name	
				Type Here	View Leaderboard		


This is the homepage a user sees after signing into the application. The user feed takes up the most space on the homepage and here the user will be able to view information that is specific to the room they are in - such as tasks completed by other users, upcoming deadlines, etc. On the top, the user will be able to view their completed tasks, their pending tasks, and all the tasks that need to be completed in the room. The top 3 rankings of the room are displayed on the right, and the full expanded leaderboard can be viewed by clicking the button on the bottom right. The users will also be able to switch between rooms. It is also possible for users to view notifications pertaining to the room's tasks by clicking on the bell at the top, or access a store where they can buy items with their points obtained from completing tasks, or modify personal room settings by clicking the hamburger sign. The room's general chat is present on the bottom of the homepage where users can send messages to all the members of the room. The buttons above the general chat can be used to carry out those respective functions.

Leaderboard (detailed view)



This is the main UI after pressing the “View Leaderboard” button on the main UI. The blank circles are placeholders for profile pictures. The red back button will return the user to the main UI. Each user will have their point total, current pending tasks, and completed tasks listed beneath their names.

Task Creation Interface

<div>Add a title</div> <div>Add a description</div>	<div>Difficulty</div> <div>Easy</div> <div>Medium</div> <div>Hard</div>
<div>Deadline</div> <div>01 January 2023</div> <div>11 : 59 AM</div> <div>Propose </div>	<div>Points</div> <div>1 - 10</div>

This is the UI that will appear after pressing the “Create Task” button on the Main UI. There are 3 text fields for attributes to assign to the task that is to be created: one for a title, one for a description, and then one for a point value. In the top right corner, there is a menu for selecting a difficulty from 3 options: {Easy, Medium, Hard}. In the bottom left corner, there is an interface to set a deadline for the task. Finally, also in the bottom left corner, there is a button to Propose the task, as well as a button to Delete the task draft if the user does not want to Propose it.


Pending Tasks View

				Team Name	01/01/2022		Leaderboard
 	Pending Tasks					Create Task	1 name
						Completed Tasks	2 name
						Pending Tasks	3 name
						Team Tasks	4 name
						Chat	5 name
						<input type="radio"/> name message <input type="radio"/> name message	6 name
					Type Here	7 name	
							8 name
							9 name
							10 name
							37 name
							56 name
							View Leaderboard

This is the main UI after pressing the “Pending Tasks” button directly to the left of the abridged leaderboard. All of the tasks that the user has accepted but not completed will be clearly listed out for the user to view with the difficulty, point value, and deadline listed next to their names.

My account info

Account Info

Avatar


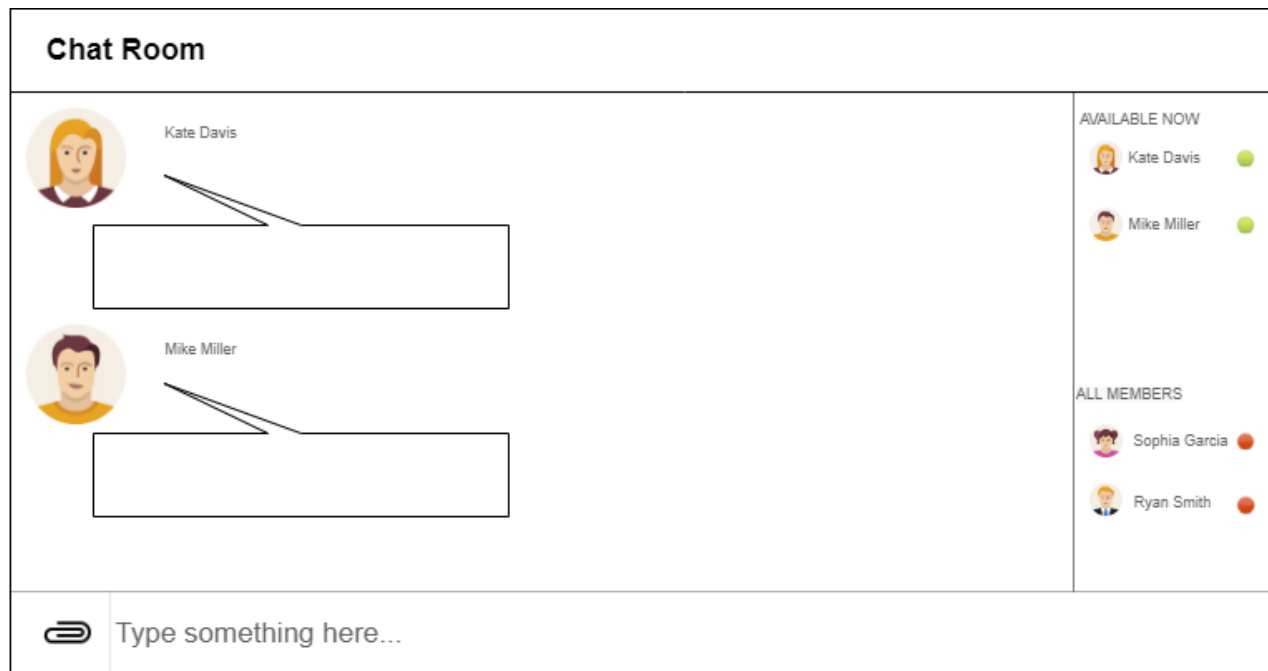
Username

Email

Password

This is the account information page, accessible by clicking the user's profile picture in the top left corner of the main UI. This page allows the user to change their default profile picture, username, which email they connected, and password. On the bottom middle of the design, there is a button to save the changed details.

General Chat



This is an expanded view of the chatroom. Users can see the online/offline status of other members, scroll through a log of previous chat messages, send messages of their own, and attach external documents to their messages.