DevMatch

Design Document

Team 17

Brian Qi

Roberto Healy

Aditya Menon

Ling-jet Chee

Jeff Chen

Auden Huang

Index

1. Purpose **2**
1.1 Purpose statement 2
1.2 Functional Requirements 2
1.3 Non-Functional Requirements 6

2. Design outline **8**
2.1 High Level Overview 8

3. Design Issues **9**
3.1 Functional Issues: 9
3.2 Non-Functional Issues: 11

4. Design Details **13**
4.1 UML Class Diagram 13
4.2 Sequence of events when users make an account 14
4.3 Sequence of events when users login 15
4.4 Sequence of events when user creates project 16
4.5 Sequence of events when user requests recommendations 17
4.6 Sequence of events when user opens a chat room or a direct message 18

1. Purpose

1.1 Purpose statement

Currently, when starting new projects, either academic or personal, it can be difficult to find teammates with matching skills and interests. Additionally, deciding how to manage that project afterward can be a hassle. To accomplish these tasks, teams typically need to utilize many different platforms at once, which can be very annoying to deal with due to things like needing to spread your attention across multiple platforms, or lack of cohesiveness between platforms due to different purposes.

The purpose of our project is to make the process of forming and operating in teams seamless. Our platform matches users with the right people for their project and provides tools to help manage the project after a team has been formed. We aim to provide an all-in-one project service that not only helps with finding team members but also includes document editing, messaging, calling, and task management features that simplify the project creation and organization process by eliminating the need to connect on several different platforms to achieve different tasks.

Some similar products to our proposed service include things like Discord, Jira, or CATME. Discord is an excellent service for communication and organizing conversations about different topics, but it does not include things like task management or team matching. Jira is excellent for organizing and assigning tasks when working on projects, but like Discord, it is a specialized service that only does that. CATME is another specialized platform, which only facilitates team matching and reviews, but also does not include automatic team matching based on skills and project requirements and details. The disadvantage of all of these platforms is that they are all specialized for one purpose.

Our proposed service of DevMatch has advantages to all these services because it combines the main functionality of these services into one platform, as well as providing automatic team and project matching, which solves the problems presented in other similar products and services mentioned in the previous paragraph and makes DevMatch perfect for project and team coordination.

1.2 Functional Requirements

1. Account Creation/Management

<u>As a user,</u>

a. I want to create an account (so that I can use the application).
b. I want to add details such as skills and interests to my account.
c. I want to be able to reset/ change the password
d. I want to be able to log in and log out of my account
e. I want to be able to set a profile picture
f. I want to be able to deactivate my account
g. I want to be able to update the information on my profile whenever I want

2. Project Creation/Management

<u>As a user,</u>

a. I want to create a project (so that I can find other teammates).

<u>As a project owner,</u>

a. I want to edit the project (so that I can add details such as what it is and what I am looking for).
b. I want to get recommended people who would fit in for the project.
c. I want to be able to invite other users to join my project.
d. I want to be able to review the resumes/ skill of the applicants, so I can reject or accept the application
e. I want my project to be hidden from other users other than the ones in my team once it has gathered the number of members that I have specified
f. I want to be able to see who is in my project
g. I want to be able to customize my project's dashboard (Project profile photo, project banner)
h. As a project owner, I want to be able to promote members in my project to have the capabilities of a project owner (transfer ownership, or have co-owners)
i. As a project owner, I want to be able to demote members and take away certain capabilities
j. I want to be able to create tasks that can be marked as completed (like Trello)
k. I want to be able to set the project's milestones
l. I want to be able to mark off the project's milestones and approve other members' requests for marking off any of the milestones

m. I want to be able to see what projects the user has worked on through this application
n. I want to be able to create different channels for different tasks
o. I want to be able to change the project name that'll be displayed on the dashboard of all members
p. I want to be able to set a limit to the number of members I am looking for.
q. I want to be able to remove users from my project if things don't work out

3. Social Function

<u>As a user,</u>

a. I want to be able to search for other users.
b. I want to be able to connect to other users.
c. I want to be able to go between different chats with potential teammates or project owners
d. I want to be able to message other users
e. I want to be able to submit ratings for other users I have worked with
f. I want to be able to report other suspicious/malicious users
g. I want to be able to leave ratings on team members that I've worked with in the past
h. I want to be able to see a list of users that I have worked with in the past

4. Finding Projects

<u>As a user,</u>

a. I want to be able to join projects
b. I want to have projects recommended to me
c. I want to be able to look for a specific type of project: a project for a school module, a project for research, or a project for fun
d. I want to be able to adjust my preferences for what kind of projects are recommended to me. (Preferences such as languages, frameworks, distance from me, time willing to spend, etc.)
e. I want to be able to decline projects that I am not interested in being apart of
f. I want to be able to sort/filter projects by most recently posted, most relevant to me, project owner rating, etc

5. Finding Project Members

<u>As a project owner,</u>

a. I want to be able to invite other users to join my project
b. I want to be able to specify the type of project I'm posting. Either it is for my research, one of my modules, or just for fun
c. I want to be able to adjust my preferences for what kind of users are recommended to me. (Preferences such as languages, frameworks, distance from me, time willing to spend, etc.)

6. Project Tools

As a user and project owner,

a. I want to be able to be in a chat room with all the members within the same project (so that I can discuss the project with my teammates as a group)
b. I want to have a text editor that I can use to drop down notes
c. I want to be able to see the projects I'm in
d. I want to be able to keep track of the project's milestones
e. I want to be able to set & remove team reminders
f. I want to be able to set up voice calls with my team (if time allows)
g. I want to be able to pin & unpin messages in a chat room
h. I want to be able to mark off project milestones, and the owner of the project must approve it so that it'll be marked as accomplished on the tracker)
i. I want to be able to write on a blackboard/ whiteboard while on a call. (If time allows)
j. I want to be able to set a recurring call
k. As a user, I want to be able to cancel a recurring call
l. As a user in a project, I want to be able to share my screen while on a call with my teammates. (If time allows)

As a project owner,

a. I want to be able to create collaborative text documents to share
b. I want to be able to set & remove team reminders

7. Dashboard/UI

As a user,

a. I want to be able to see the past project I've completed

b. I want the completed project to be in a different section than my ongoing project on the dashboard (Like in two different drop-down lists, similar to those on Wikipedia)
c. I want to be able to adjust the order of the projects shown on my dashboard. (Like on Discord you can tap and drag on the server icon to change the order your servers display on the sidebar)
d. I want to be able to customize the UI (dark mode, light mode)

1.3 Non-Functional Requirements

1. Security

As a developer,

a. I want to encode/ encrypt user information such as account password, and message history in the database
b. I would like the website the support two-factor authentication at the login
c. I would like the user to have limited access to functions and features based on their role

2. Usability

As a developer,

a. I want the web page to be a platform that combines several different commonly used project management materials along with a user-matching system
b. I want the web page interface to be easily understood, eliminating user confusion.
    i. I want the user to be able to seamlessly switch between different features without any hassle
    ii. I want the web page to have entirely different dashboards for each unique feature, making it clear which area of the platform a user is viewing.

3. Performance

As a developer,

a. I want the server to be able to accommodate at least 2,200 users at once since that is the estimated number of Purdue undergraduate students in Computer Science.
b. I would like the user to get their recommendation within 5 seconds

4. Scalability

As a developer,

a. I want the webpage server to be expandable, this will allow us to scale up the webpage to handle more users in order to include students in other majors or even students in other Universities if we have time.

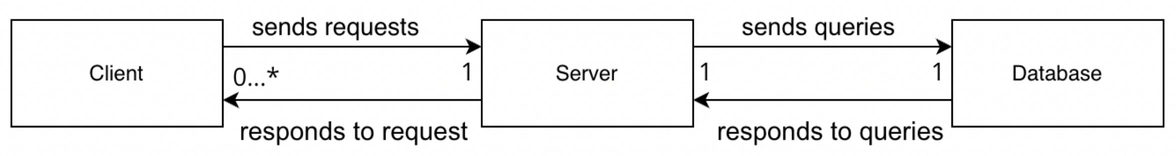
5. Architecture

As a developer,

a. I want to develop the front and the backend of the website separately
b. I want to use a backend model of RESTful APIs, with different testing for each API
c. I want the backend to have different testing and backlog items for each API call.
d. I want to develop the backend using Java and/or NodeJS
e. I want the backend to have connections without a database
f. I want to develop the database with NoSQL
g. I want to develop the frontend using NextJS along with React and Tailwind, so I can create a modern webpage with full interactivity for devices

2. Design outline

2.1 High-Level Overview

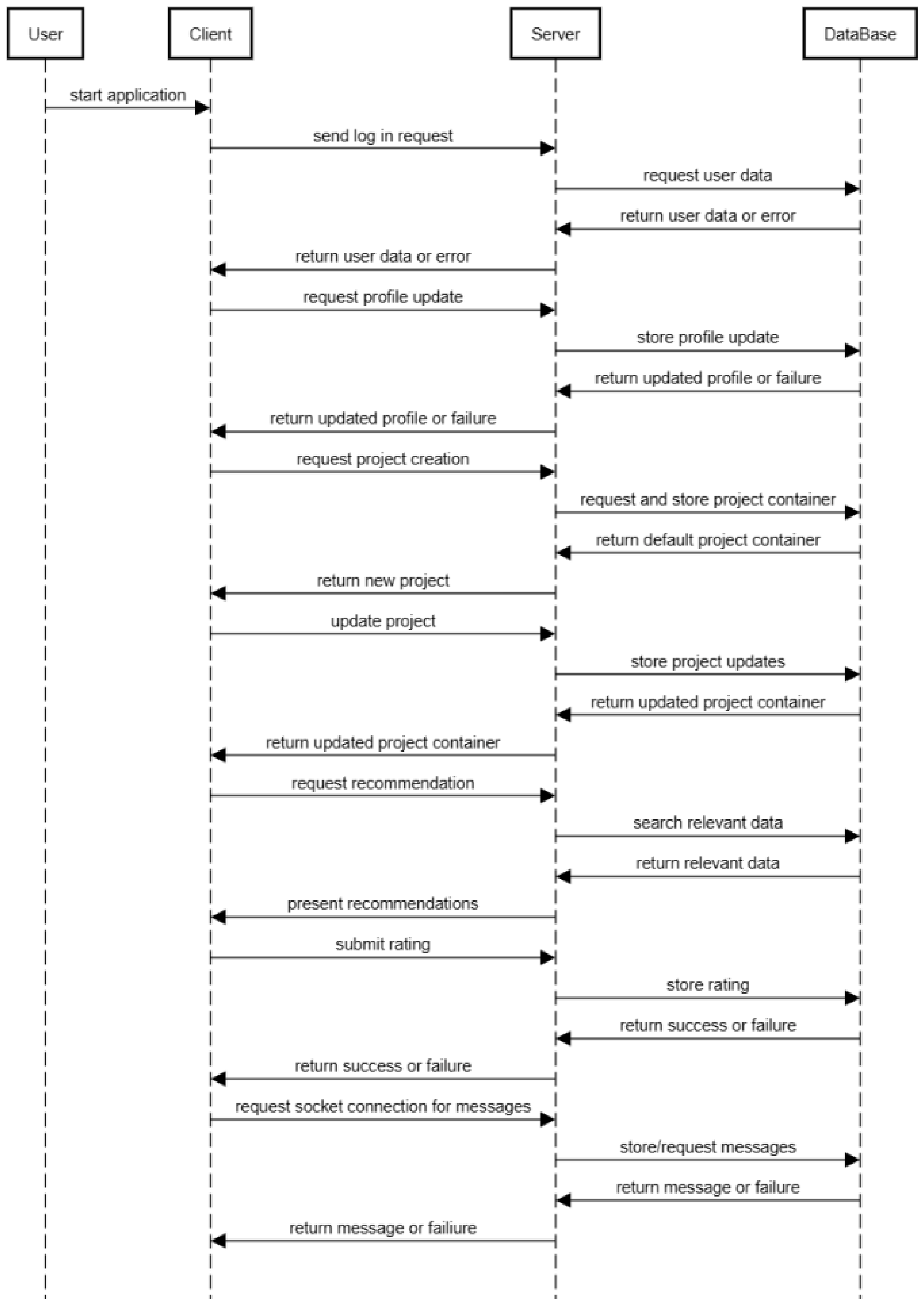


Our project will be a web application that allows users to post either their project ideas or their own profiles to be matched with a team based on their and other users' skills, preferences, and interests. Our application will follow the client-server model. Our server will handle requests from clients using our Golang backend. The server will accept client requests, access or store data in the database, and feedback to the client(s) accordingly.

1. Client
   a. Client provides an interface for users to interact with our system
   b. Client sends requests to the server
   c. Client receives responses from the server and displays them to users as needed
2. Server
   a. Server receives and handles requests from client
   b. Server checks requests and sends queries to the database
   c. Server creates responses to send to client
3. Database
   a. Database stores all user data such as profiles, projects, and messages
   b. Database responds to queries from server and adds, modifies, or gets data

2.2 Sequence of Events Overview

The below diagram shows how our program handles interactions between user, client, server, and database. Typically, the first thing the user will do is login to our platform. The user logs in, and the client sends the request to the server. The server handles the request, queries the database, and then fetches the data from the database. Following the user login, the client continues to send requests to the server for actions such as: creating new projects, messaging other users, requesting user matches, and updating user profiles. For these actions, the client sends requests to the server, the server either queries the database for the relevant data or stores new updated data depending on the task, and the database returns the relevant data to be displayed and updated for users.


User
Client
Server
DataBase
start application
send log in request
request user data
return user data or error
return user data or error
request profile update
store profile update
return updated profile or failure
return updated profile or failure
request project creation
request and store project container
return default project container
return new project
update project
store project updates
return updated project container
return updated project container
request recommendation
search relevant data
return relevant data
present recommendations
submit rating
store rating
return success or failure
return success or failure
request socket connection for messages
store/request messages
return message or failure
return message or failiure

3. Design Issues

3.1 Functional Issues:

1. What information would we require from users to register a new account?

- Option 1: Username and password
- Option 2: Username, password, PUID
- Option 3: Username, password, school email address
- Option 4: Username, password, email address, phone number

Choice: Option 3

Justification: Choosing the third option allows our application to have an additional level of security that is provided by students having unique emails. We realized that it would be necessary to have some sort of proof of identification as well as a way to communicate with the user outside of the application. Having a school email satisfies both of those wants.

2. How would we recommend projects to users?

- Option 1: Based on needed skills matching with user skills
- Option 2: Based on matches between the project description and user interests
- Option 3: Using a machine learning model

Choice: Option 1

Justification: Matching user skills with the project's needed skills would provide for a more straightforward implementation, while also providing accurate results. This is due to how we plan to implement our user and project classes, which will have fields representing skills and skills needed, respectively. Option 2 would also be a good option, as users would want to work on things they are interested in as well if this differs from their skills. Ideally, we would use both of these options to match projects with users.

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

- Option 1: Private DM's between individual people, and project group chats
- Option 2: Discussion post forum

- Option 3: Just project group chats

Choice: Option 1

Justification: Having the ability to direct message users allows users to first get to know candidate teammates and see if they would be a good fit. After deciding to collaborate, users can be added to the project group chat. That way, there is a method to communicate before fully joining the team.

4. How will we distinguish user types?

- Option 1: Have two different login systems for regular users and project owners
- Option 2: Have one type of user, but tag project owners so they have access to respective UI dashboards

Choice: Option 2

Justification: We felt it was easier to allow users to both be project owners and prospective teammates at once. That way, users can easily switch between modes. If the user only wants to be one type of user, then they can simply indicate as such and will just not be shown the other recommendations.

5. How do we authenticate users and avoid spam?

- Option 1: Using Purdue authentication (email, PUID)
- Option 2: Only allows text messages with users you are connected with
- Option 3: Use some authentication method such as sending a picture of driver's license

Choice: Option 1

Justification: Since our target audience is primarily Purdue students, we felt that it made the most sense to use an authentication method that already applies to Purdue students. Using Purdue emails and PUID’s would make authentication relatively easy but also very secure as each student only has one of each.

3.2 Non-Functional Issues:

1. What web service should we use for hosting?

- Option 1: Firebase
- Option 2: AWS
- Option 3: Heroku

Choice: Option 1

Justification: We plan on using the free tier “spark” that allows 1 GB total storage, 20K daily writes / deletes, and 50K daily reads. One of our group members is familiar with Firebase, and since we have also decided on using Firebase for the database, we felt it made sense to keep it all on one platform that some of our group has experience with.

2. What backend language/framework should we use?

- Option 1: Go
- Option 2: Flask
- Option 3: Node.js

Choice: Option 1

Justification: Golang is a faster alternative to other backend options. It also provides easier scalability and the possibility of including concurrency.

3. What frontend language/framework should we use?

- Option 1: React
- Option 2: HTML + JS
- Option 3: Angular
- Option 4: Vue

Choice: Option 1

Justification: React is a straightforward platform with a lot of resources for creating user interfaces. We felt this made the most sense as it seems very practical as well as easier to learn than some of the other options.

4. What database should we use?

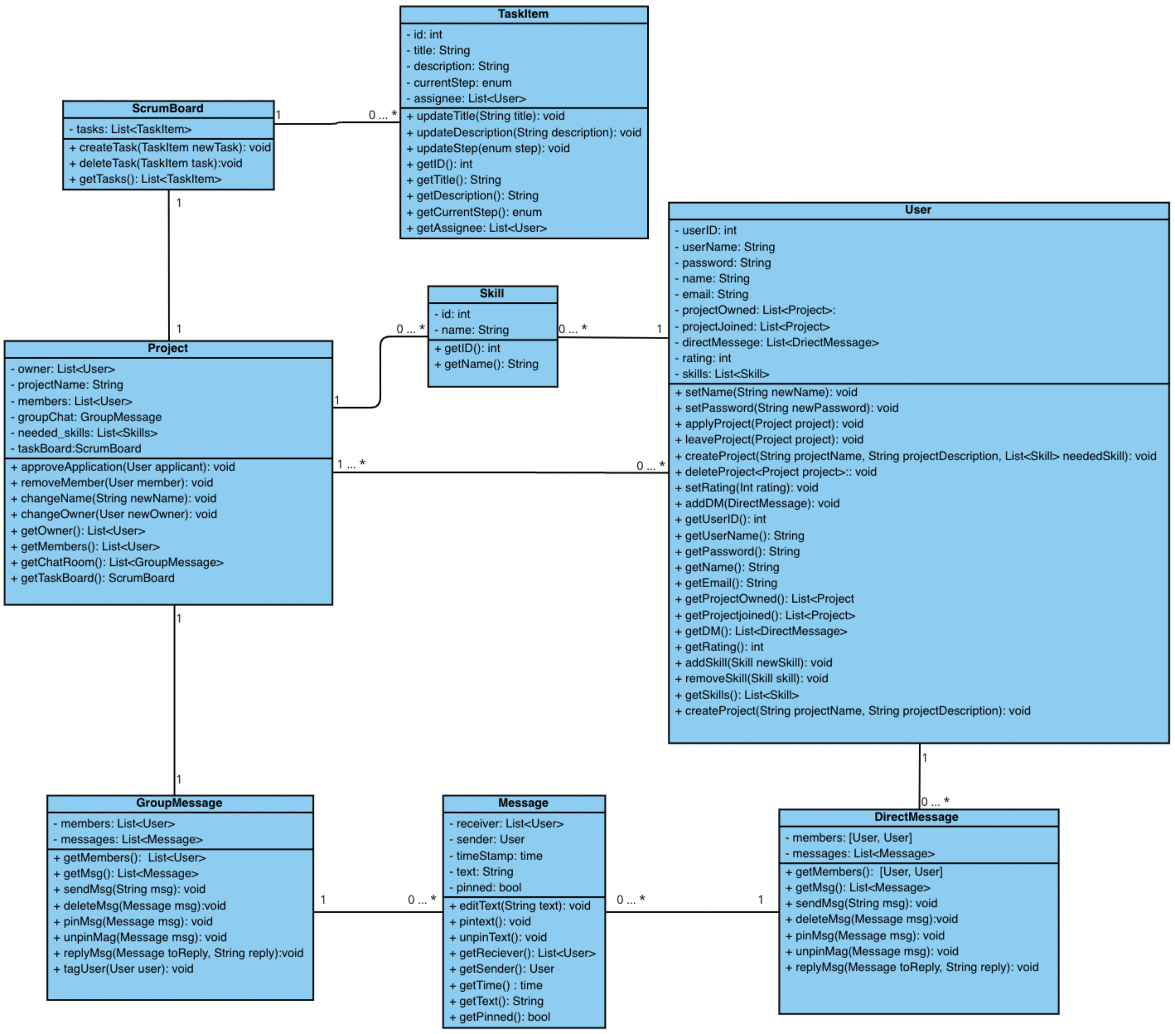
- Option 1: Firebase
- Option 2: MongoDB
- Option 3: MySQL

Choice: Option 1

Justification: Firebase provides a streamlined and accessible service that satisfies the needs of our application. This is beneficial as the data we collect is not especially intricate. This will also reduce the amount of time required to learn it, making it far more practical.

4. Design Details

4.1.1 UML Class Diagram


TaskItem
- id: int
- title: String
- description: String
- currentStep: enum
- assignee: List<User>
+ updateTitle(String title): void
+ updateDescription(String description): void
+ updateStep(enum step): void
+ getID(): int
+ getTitle(): String
+ getDescription(): String
+ getCurrentStep(): enum
+ getAssignee: List<User>
ScrumBoard
- tasks: List<TaskItem>
+ createTask(TaskItem newTask): void
+ deleteTask(TaskItem task):void
+ getTasks(): List<TaskItem>
Skill
- id: int
- name: String
+ getID(): int
+ getName(): String
User
- userID: int
- userName: String
- password: String
- name: String
- email: String
- projectOwned: List<Project>:
- projectJoined: List<Project>
- directMessege: List<DriectMessage>
- rating: int
- skills: List<Skill>
+ setName(String newName): void
+ setPassword(String newPassword): void
+ applyProject(Project project): void
+ leaveProject(Project project): void
+ createProject(String projectName, String projectDescription, List<Skill> neededSkill): void
+ deleteProject<Project project>:: void
+ setRating(Int rating): void
+ addDM(DirectMessage): void
+ getUserID(): int
+ getUserName(): String
+ getPassword(): String
+ getName(): String
+ getEmail(): String
+ getProjectOwned(): List<Project
+ getProjectjoined(): List<Project>
+ getDM(): List<DirectMessage>
+ getRating(): int
+ addSkill(Skill newSkill): void
+ removeSkill(Skill skill): void
+ getSkills(): List<Skill>
+ createProject(String projectName, String projectDescription): void
Project
- owner: List<User>
- projectName: String
- members: List<User>
- groupChat: GroupMessage
- needed_skills: List<Skills>
- taskBoard:ScrumBoard
+ approveApplication(User applicant): void
+ removeMember(User member): void
+ changeName(String newName): void
+ changeOwner(User newOwner): void
+ getOwner(): List<User>
+ getMembers(): List<User>
+ getChatRoom(): List<GroupMessage>
+ getTaskBoard(): ScrumBoard
GroupMessage
- members: List<User>
- messages: List<Message>
+ getMembers(): List<User>
+ getMsg(): List<Message>
+ sendMsg(String msg): void
+ deleteMsg(Message msg):void
+ pinMsg(Message msg): void
+ unpinMag(Message msg): void
+ replyMsg(Message toReply, String reply):void
+ tagUser(User user): void
Message
- receiver: List<User>
- sender: User
- timeStamp: time
- text: String
- pinned: bool
+ editText(String text): void
+ pintext(): void
+ unpinText(): void
+ getReciever(): List<User>
+ getSender(): User
+ getTime() : time
+ getText(): String
+ getPinned(): bool
DirectMessage
- members: [User, User]
- messages: List<Message>
+ getMembers(): [User, User]
+ getMsg(): List<Message>
+ sendMsg(String msg): void
+ deleteMsg(Message msg):void
+ pinMsg(Message msg): void
+ unpinMag(Message msg): void
+ replyMsg(Message toReply, String reply): void
1
0 ... *
1
1
0 ... *
1
0 ... *
1
1 ... *
0 ... *
1
1
1
0 ... *
1
0 ... *
0 ... *
1
1
0 ... *

4.1.2 Description of Classes and Interaction Between Classes

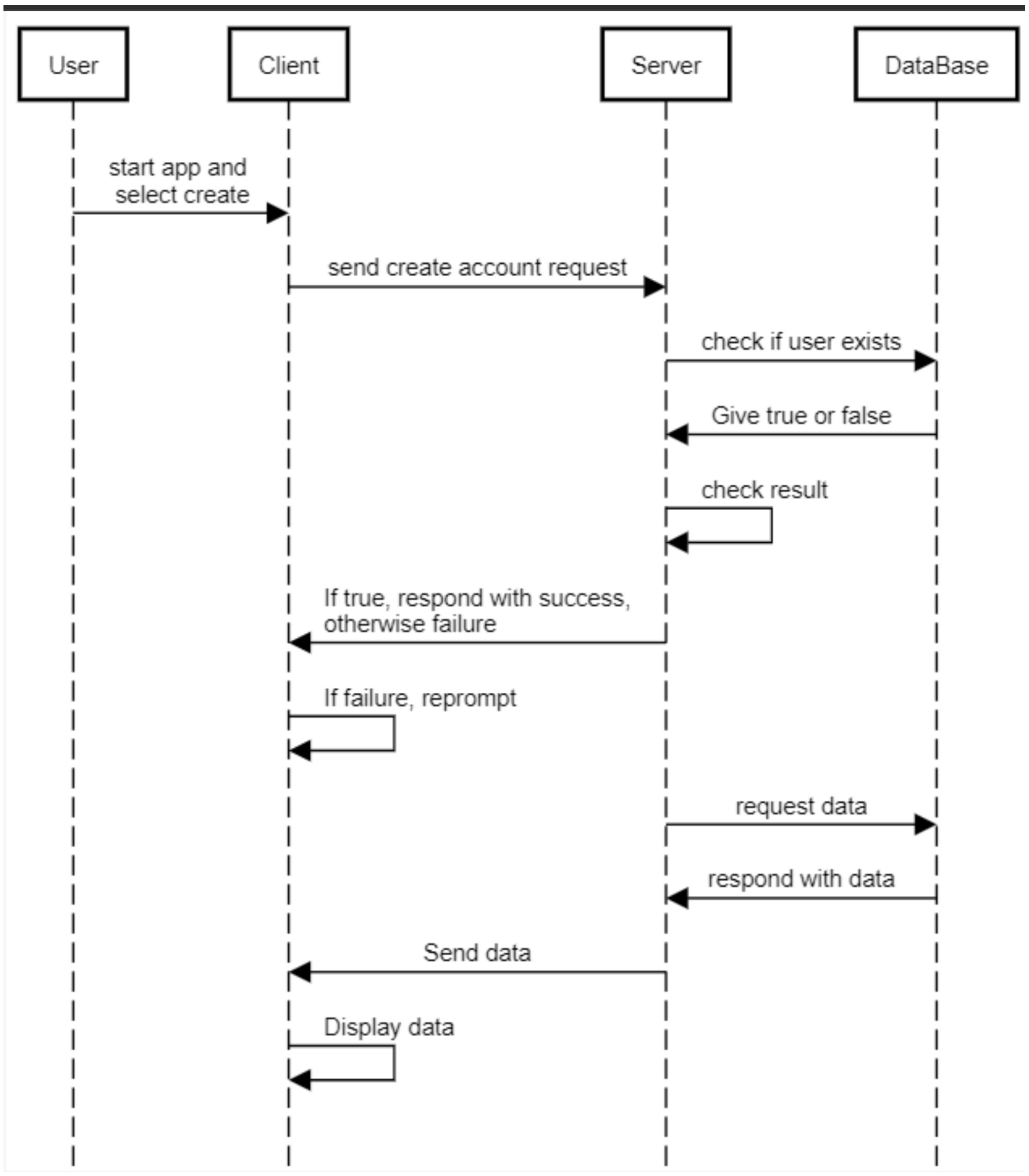
- **User**
  - Each user has userID for an easier and faster way of accessing users on the database
  - Each user has userName and name which will be the info the user displays
  - Each user has a password and email which will be used for login
  - Each user has projectsOwned which will be a list of the class Project, specifically those projects created and owned by the user
  - Each user has projectsJoined which will be a list of the class Project, specifically those the user is a member of

- Each user has directMessages which is a list of class DirectMessage, this will be a list of all direct conversations with other users
- Each user has a rating which will be a way to show the score of past ratings
- Each user has skillList, which is a list of type Skill, the skills are selected when creating the account and can be changed at any point

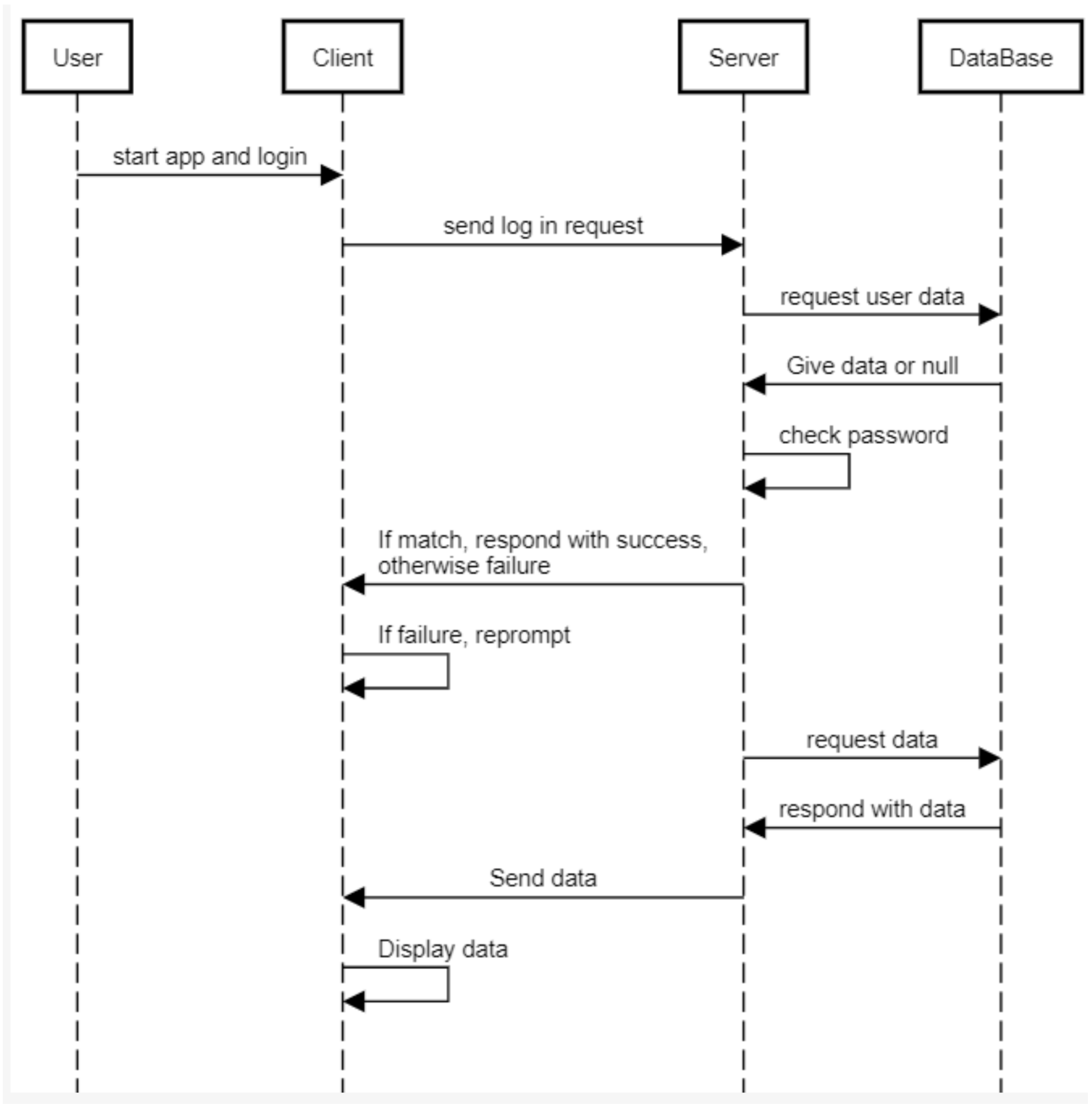
- **Skill**
  - Each skill will have an id, which will be used to identify quickly
  - Each skill will have a name, which will be used as the text description of the skills
  - Skills will be one of the main attributes when filtering and matching projects and users
- **Project**
  - Each project has owners, which can be one or multiple users
  - Each project has members, which can be one or multiple users
  - Each project has a projectName, which will be the public display name
  - Each project has a projectID, which is a non-public way to store the projects and access them in the database
  - Each project has needed_skills, which has a list of type Skill, these skills will be used to match people with the project
  - Each project has a groupChat, where all the members will be able to chat
  - Each project has a scrumboard, where the team members can view the different tasks and their progress
- **ScrumBoard**
  - Each ScrumBoard has tasks, which will be a list of type TaskItem
  - The ScrumBoard has control of creating and removing TaskItems
- **TaskItem**
  - Each TaskItem has an id, for quick operations such as delete
  - Each TaskItem has a title and description which will be displayed
  - Each TaskItem has a currentStep which will have a value of an enum classifying the different steps in the scrumboard
  - Each TaskItem has assignees, which can be a single or multiple users
- **Message**
  - The message class is a simple register of a text, it is used by both DirectMessage and GroupMessage, but cannot be used by itself
  - Each Message has a sender, single user, and receivers, single or multiple users
  - Each Message has a timestamp, which will record at what time the message was sent
  - Each Message has a pinned attribute of type bool that defaults to false but can be changed by other classes
- **DirectMessage**
  - The DirectMessage stores multiple Message types to form a conversation between two users
  - Each DirectMessage has functions to send messages and to pin messages in a conversation
- **GroupMessage**
  - The GroupMessage stores multiple Message types to form a conversation with multiple users

- Each GroupMessage has functions to send messages and to pin messages in a conversation

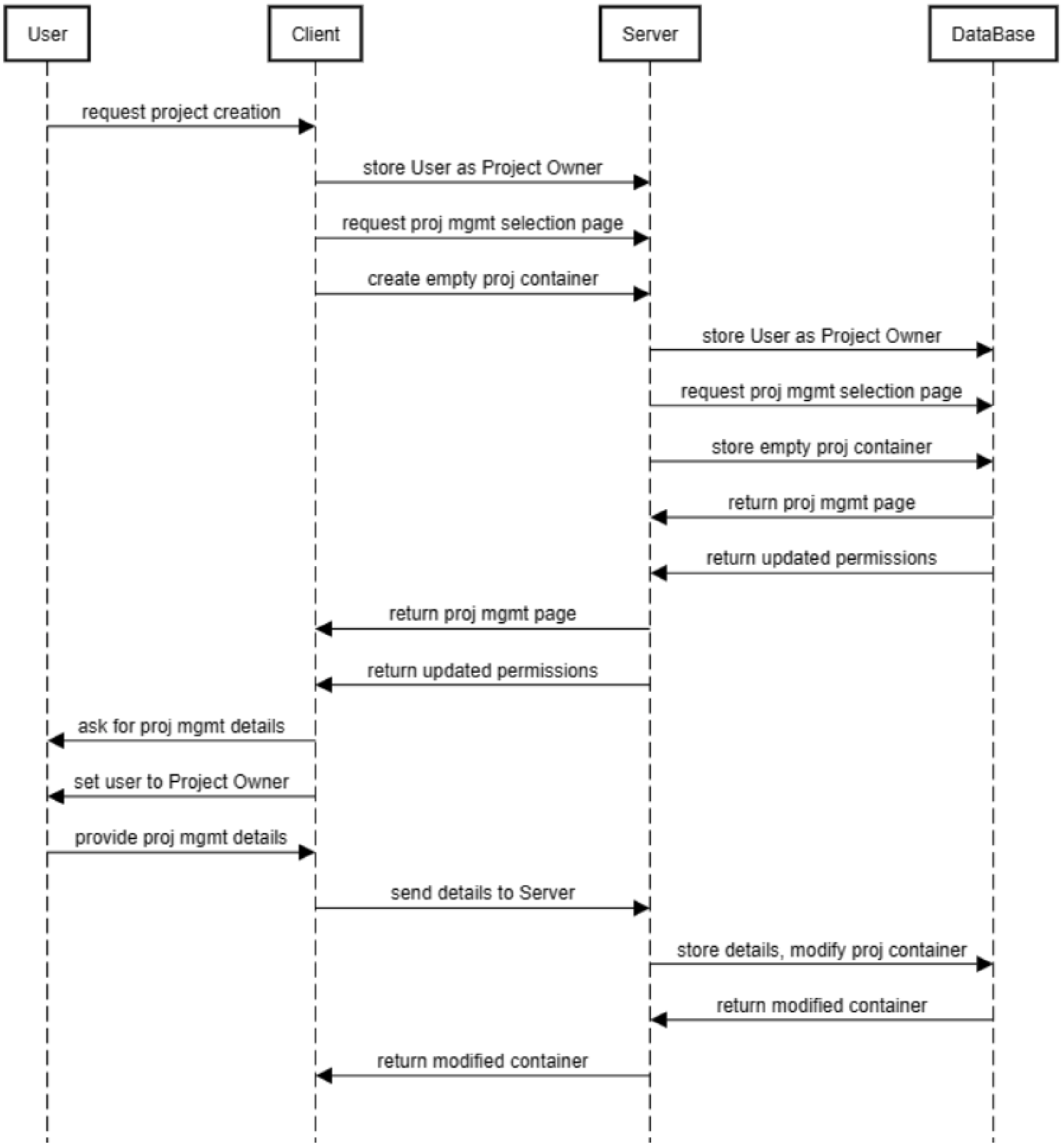
4.2 Sequence of events when users make an account



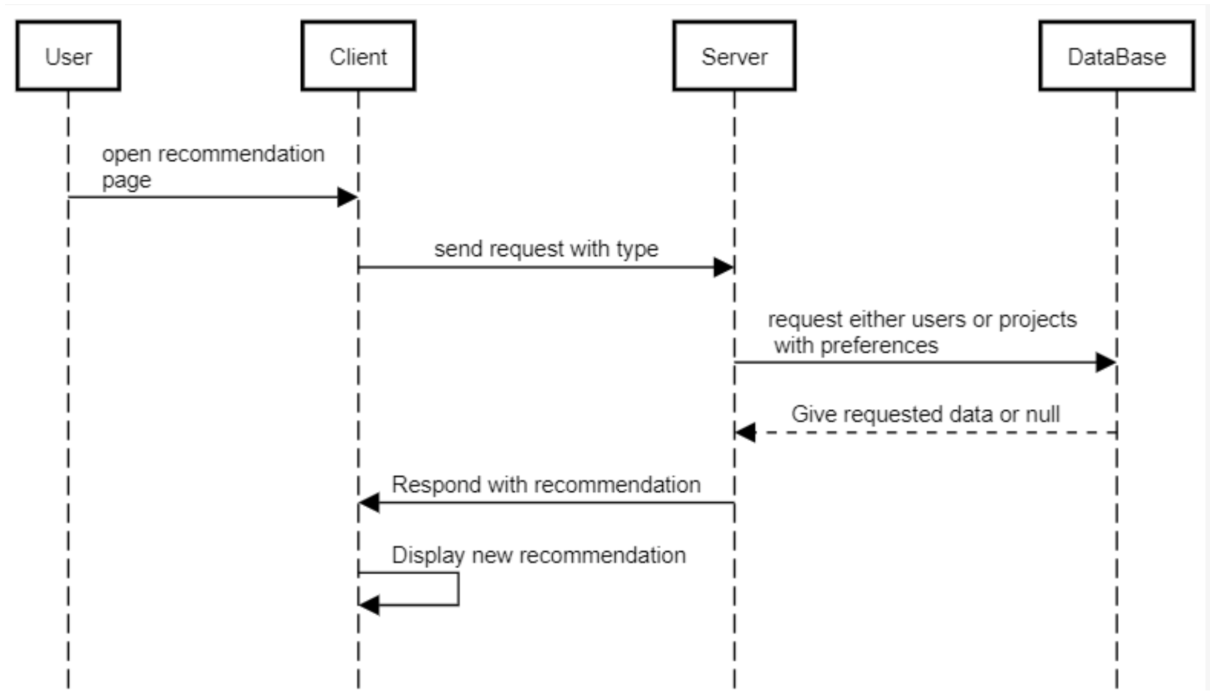
4.3 Sequence of events when users login



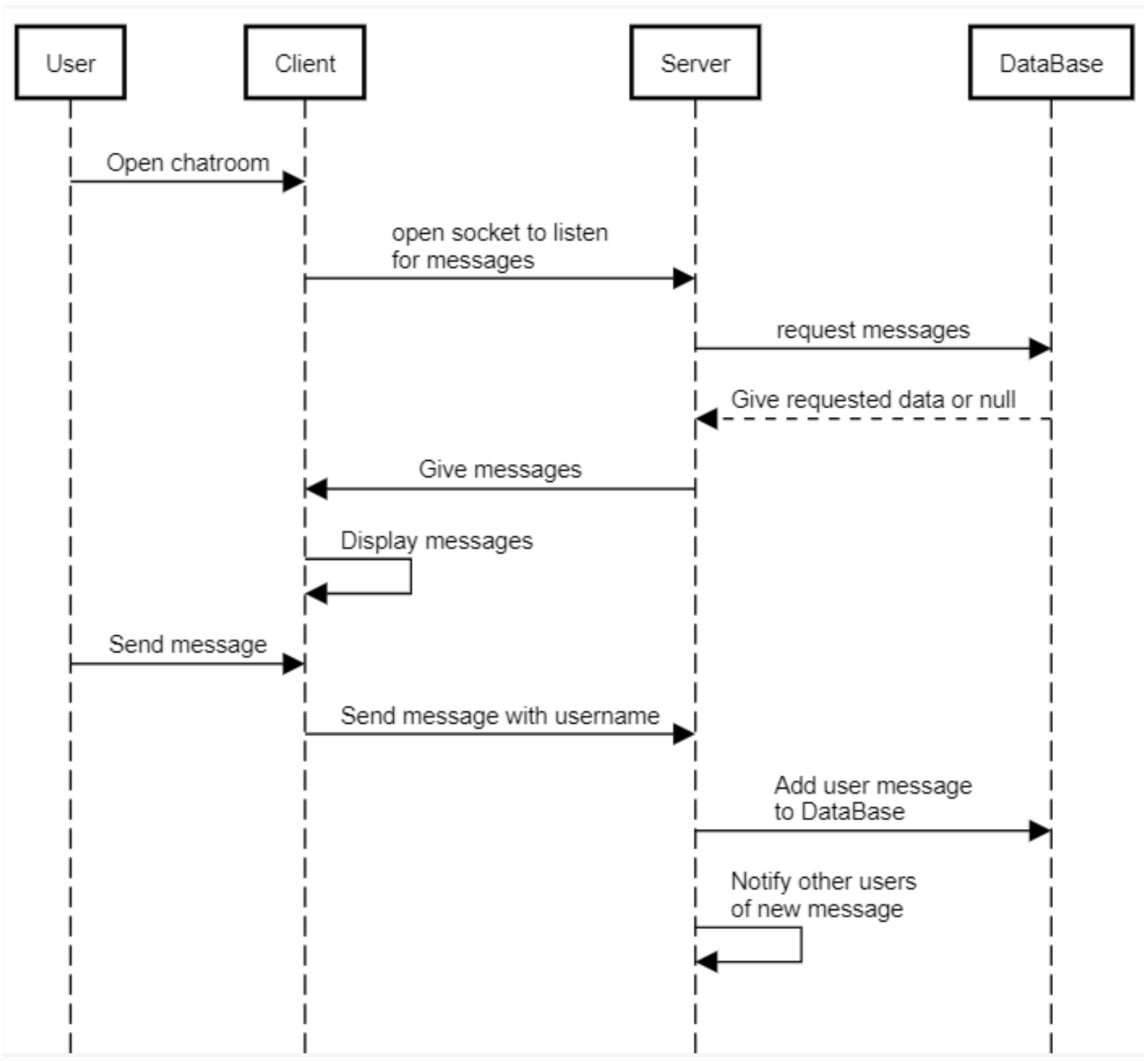
4.4 Sequence of events when user creates project


User
Client
Server
DataBase
request project creation
store User as Project Owner
request proj mgmt selection page
create empty proj container
store User as Project Owner
request proj mgmt selection page
store empty proj container
return proj mgmt page
return updated permissions
return proj mgmt page
return updated permissions
ask for proj mgmt details
set user to Project Owner
provide proj mgmt details
send details to Server
store details, modify proj container
return modified container
return modified container

4.5 Sequence of events when user requests recommendations


User
Client
Server
DataBase
open recommendation page
send request with type
request either users or projects with preferences
Give requested data or null
Respond with recommendation
Display new recommendation

4.6 Sequence of events when user opens a chat room or a direct message


User
Client
Server
DataBase
Open chatroom
open socket to listen for messages
request messages
Give requested data or null
Give messages
Display messages
Send message
Send message with username
Add user message to DataBase
Notify other users of new message