CS 307
DESIGN DOCUMENT

Team 28
Chiho Song, Erin Joy Kramer, Shreyash Bairi, William Quinn

Index

Purpose

Teams bring together individuals with different perspectives, skills, and experiences, making it possible to solve problems and make decisions more effectively. There is a need for management apps that provide a centralized platform for teams to coordinate and exchange ideas as well as communicate in real time. Current apps provide similar solutions yet, some apps may have a cluttered or confusing interface, making it difficult for users to navigate and find what they need. In addition, some apps may not integrate well with other tools and platforms, making it difficult for users to work efficiently.

The purpose of this project is to design a group discussion and management app that not only mitigates these issues but improves upon them with additional features. By having both a to-do list and group management features in one app, users can easily manage their tasks and team projects in one place, improving their workflow and saving time. Our app can allow team members to track progress, and exchange feedback, leading to better collaboration and teamwork. The app can provide an overview of both individual and team tasks, helping everyone to see what needs to be done and how progress is being made. With to-do list features, users can create and prioritize tasks, set deadlines, and track progress, improving their task management and increasing productivity. In addition, our app incorporates real-time communication features, allowing team members to communicate and collaborate in real time, reducing the need for email or in-person meetings.

With the inclusion of a calendar, our app can serve as a centralized platform for team members to view and manage their schedules and events as well as keep track of events. The ability for managers to add events that reflect on all team members' calendars can improve coordination and ensure everyone is on the same page. This feature can also eliminate the need for multiple calendar apps, simplifying the scheduling process. By integrating with other tools and platforms, our app aims to eliminate the need for multiple apps and make it easier for users to work efficiently. With a clean and user-friendly interface, our app can provide a better user experience overall, making it easier for users to find what they need and work productively.

Functional Requirements

To-Do List

1. As a user, I would like to have a personal to-do list
2. As a user, I would like for my to-do list and calendar contents to be saved for the next to open the app, even on another device
3. As a user, I would like to have a group to-do list or group board that is for tasks for all team members, that is separate from the personal to-do list
4. As a user, I would like to see a more detailed view of a task such as description, deadline, and users working on
5. As a user, I would like to apply priority to tasks
6. As a user, I would like to view completed tasks for each to-do list
7. As a user, I would like to sort and filter tasks in a to-do list by team, type, the person working on
8. As a user, I would like to assign deadlines to tasks
9. As a user, I would like to update the status of a task

Calendar and Events

10. As a user, I would like to have a calendar that displays my events as well as group events
11. As a user, I would like to have a detailed and easy-to-use UI for creating a new calendar event
12. As a user, I would like to assign detail and labels to calendar events, such as labeling an event as personal or assigning it to a specific team so that a manager just sees it as personal time and can not see the details
13. As a user, I would like to view my calendar in different modes such as a 7-day week, 5-day week, or month
14. As a user, I would like to select which groups or personal calendars to focus on by choosing which ones to display

User Profile and Customization

15. As a user, I would like to have a profile page that allows me to see my personal information, such as roles, and user info
16. As a user, I would like to have an appealing main page on which will include the features like calendar and to-do list.
17. As a user, I would like to create a zoom link for an event by having a connection to zoom built into the event I am creating as an option
18. As a user, I would like to be able to personalize the colors
19. As a user, I would like to sign up, log in, and logout
20. As a user, I would like to change the size of sections so that they better fit my needs(like photoshop or IntelliJ) (IF TIME ALLOWS)

21. As a user, I would like to be able to personalize the amount of information shown for events and tasks with different views (ie clean compact view or detailed) (IF TIME ALLOWS)

Notifications

22. As a user, I would like to set reminders for tasks that send notifications
23. As a user, I would like to have a tab/section for a list of all notifications
24. As a user, I would like to have notification settings, allow notifications to pop up or make noise, and notifications can be sent via email
25. As a user, I would like to have access to a settings page where I can put in my personalization and notification preferences

Teams and Communication

26. As a user, I would like to create a new team which I am the manager of
27. As a user, I would like to accept invites to teams through notifications as well as send requests to join a team
28. As a user, I would like to be able to switch between teams, some of which I am a team member of and others that I am a manager of
29. As a user, I would like to participate in a group chat with the team
30. As a user, I would like to dm individual users
31. As a user, I would like to “poke” other team members to request a meeting with them
32. As a user, I would like to see a list of team members within a team
33. As a user, I would like to be able to see the job role of members of my team (IF TIME ALLOWS)
34. As a user I would like to share files (in a resource tab or in the chat) (IF TIME ALLOWS)

Manager Privileges

35. As a manager, I would like to have a separate UI framework specific to manager permission
36. As a manager, I would like to be able to see team-specific events on team member’s calendar
37. As a manager, I would like to add events to team members' calendars
38. As a manager, I would like to send invites to users to join a team, admit users requesting to join, and remove users from a team
39. As a manager, I would like to assign tasks to individual team members or the team as a whole
40. As a manager, I would like to see times that are blocked off as unavailable by team members so that I can better schedule a meeting (IF TIME ALLOWS)

41. As a manager, I would like to receive additional notifications about team members' tasks completed or status
42. As a manager, I would like to receive a notification when a team member leaves the team
43. As a manager, I would like to see the tasks and team events of a team member that leaves or is removed from a group on the group to-do list
44. As a manager, I would like to post polls and announcements
45. As a manager, I would like to create custom roles for team members that provide them with different permission of the team such as posting polls or announcements or having the ability to assign tasks (IF TIME ALLOWS)
46. As a manager, I would like to create different projects within teams that team members can view based on their role permissions (IF TIME ALLOWS)
47. As a user, I would like to be able to turn on a timer (IF TIME ALLOWS)

Nonfunctional Requirement

1. Client Requirements

1.Usability

a. The interface of the software should be very intuitive to use and not very different from what most users are already familiar with. There should not be a steep learning curve
b. Features, icons, and descriptions must be obvious and easy to understand so as to prevent any interruption in workflow. Interfaces should also be optimized for the task being performed.
c. Since the platform is central for all kinds of users, students, and management; the UI must reflect the changes in permissions and privileges. The application must be executable on all screen sizes, resolutions, and browsers.

2.Security

a. Our application includes personal information and corporate data that may be critical when leaked.
b. Our software will authenticate every user so that everyone has the credentials to be on the team.
c. Users will be able to choose what others want to see from their calendars and profile.
d. Joining projects will be only available through invites, along with requests to the manager to verify every member of the team.

2. Performance Requirements

1.Latency

a. The response time for any action taken by a user in our application should be under 5 seconds.
b. Though we do intend to use third-party data storage, the latency of our servers should not have a noticeable impact on the functionality of our program.
c. The loading time for the User Interface should also follow these same standards. The uptime will be approximately 95%.

2.Scalability

a. Software should be made with the idea that updates and new features can and will come along.

b. The server-side software should have the capability to simply and easily incorporate more users and data. Although the server may be hosted for small use at first it should easily be scaled up to account for a growing user base.

3. Design Requirements

1.Architecture

a. We plan to use MERN stack, a JavaScript based framework that utilizes React, Express.js, Node.js, and MongoDB in tandem to build web applications. React is becoming increasingly common in industry and is an excellent tool for creating a scalable application and designing a frontend.
b. There is also a large community surrounding it so there is a lot of potential for maintainability and future development. Express.js and Node.js are utilized in the backend as servers that are largely unopinionated and are very powerful for URL routing and HTTP requests.
c. MongoDB is a database that allows for persistent memory that is stored in the form of JSON documents. Overall, MERN is the best choice for our full stack application.
d. We chose MERN stack because it has a growing community with lots of resources for us to consult and is also extremely versatile and reliable for building web applications.

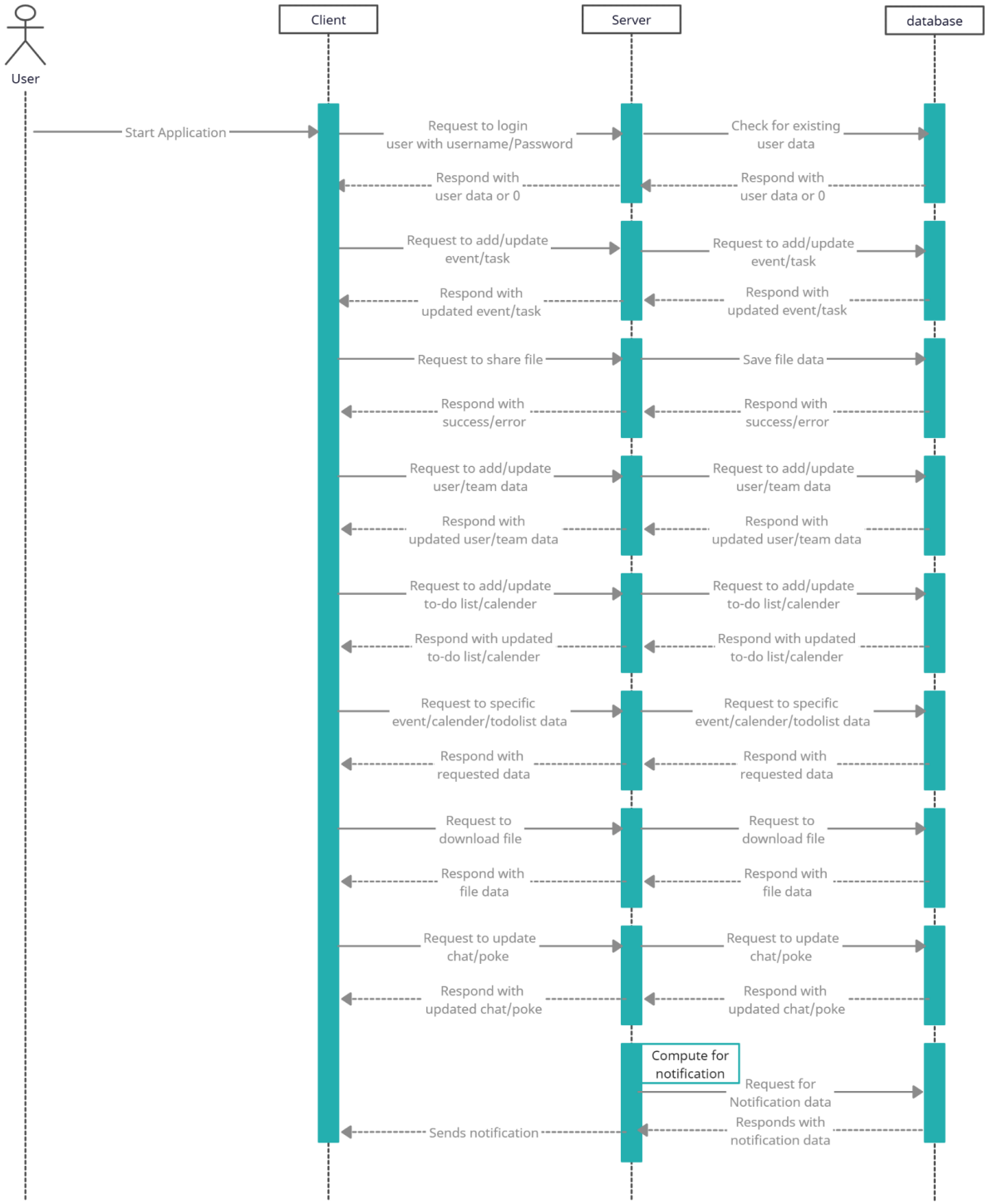
Design Outline

High level Overview

Our software will be a team management tool application on the web. It provides users with functionality such as team to-do lists, calendar and sharing files. It will use a client-server model, communicating with different sets of data provided by users and databases through the server. Client will present data it retrieved from database to users and any changes made in client, such as creating events or finishing to-do list will be saved in database through server constantly. Client will update its UI, based on response from the server.

Sequence of Events Diagram

This diagram shows an overview of interaction between clients, server and database. As the user starts a web application and provides their sign in information, the client will send user data to the server, which will check for existing users in the database and return user information if a match is found. Once logged in, the user can stay on the client requesting/adding/updating data to the server which then interacts with data to retrieve or update data. Also, clients may request for share/download files or send messages or poke other clients. Server will handle requests by saving data into the database, then respond to the client with results to save . Lastly, the server will send notification to the client based on the existing database.


User
Client
Server
database
Start Application
Request to login
user with username/Password
Check for existing
user data
Respond with
user data or 0
Respond with
user data or 0
Request to add/update
event/task
Request to add/update
event/task
Respond with
updated event/task
Respond with
updated event/task
Request to share file
Save file data
Respond with
success/error
Respond with
success/error
Request to add/update
user/team data
Request to add/update
user/team data
Respond with
updated user/team data
Respond with
updated user/team data
Request to add/update
to-do list/calender
Request to add/update
to-do list/calender
Respond with updated
to-do list/calender
Respond with updated
to-do list/calender
Request to specific
event/calender/todolist data
Request to specific
event/calender/todolist data
Respond with
requested data
Respond with
requested data
Request to
download file
Request to
download file
Respond with
file data
Respond with
file data
Request to update
chat/poke
Request to update
chat/poke
Respond with
updated chat/poke
Respond with
updated chat/poke
Compute for
notification
Request for
Notification data
Responds with
notification data
Sends notification

Design Issues

Functional Issues

1. What information do we need to store a user's account?
    - Option 1: teams, username, password
    - Option 2: teams, username, password, email, phone number
    - Option 3: teams, username, password, email, preferences

    Choice: Option 3

    Justification: We determined that we would like to give the user the option to receive notifications from their event calendar and to-do list via email if they choose. That is why we decided to include an email and preferences on top of the general information such as team, username, and password. We determined that to protect user privacy as well as the fact that our application is largely meant to be used within some kind of workplace environment, storing mobile numbers was unnecessary.

2. What filters should we allow users to have access to when filtering tasks and events?
    - Option 1: None
    - Option 2: Team, Personal
    - Option 3: Team, Personal, Category, Project

    Choice: Option 2

    Justification: This decision was made largely on the big picture idea that we have for this project. Our goal is to compartmentalize time and scheduling across multiple teams and within a user's own personal life, so while we may end up having different categories and projects for the user's own ease of assigning and prioritizing their tasks, filtering based on a specific project as a subgroup of a team is out of the scope of our goals this semester.

3. How many labels should we allow a user to put on one event or task?
    - Option 1: 1 label for team
    - Option 2: 3 labels for team, time
    - Option 3: 3 labels for team, time, and a description

    Choice: Option 3

    Justification: Tasks and Events should be as descriptive as possible giving users all the information they need/want to put in tasks and events. Users do not have to fill everyone of these attributes but by giving them the option they are not limited.

4. What display options should users have for their calendar?
    - Option 1: 7 day calendar
    - Option 2: 7 day or 5 day/weekday calendar
    - Option 3: 7 day, 5 day/weekday, and 1 day calendar

Choice: Option 2
Justification: For the purposes of having a clean and readable user interface and user experience, having a 1 day calendar was deemed to not be a good option. On the other hand, because we plan on our application being heavily used by working teams, having a 5 day/weekday calendar option to account for the normal working week makes logical sense.

5. How should features be different between manager and team member user interfaces?
    - Option 1: Same user interface just allow managers to have extra edit permissions
    - Option 2: Manager has unique options and buttons

    Choice: Option 2
    Justification: Users should have a unique manager UI for teams that they are the manager of because it will create a clear differentiation to the user that they are a manager, as well as giving them the unique permissions. By having a separate UI a manager will have additional settings and options for the team that would be hidden to team members, such as additional buttons and menus.

6. What is the maximum number of people that should be allowed on a team?
    - Option 1: 5 people
    - Option 2: 20 people
    - Option 3: No maximum

    Choice: Option 2
    Justification: By having a maximum of 20 ppl per team, users will not find that their teams are too restrictive in their sizing. There must be a maximum for storage purposes as well as runtime demands. In the future if Rome gets scaled up the maximum may be increased.

Non-Functional Issues

1. How and where should we store data so that users can access the same information through their account, even on another device?
    - Option 1: MongoDB
    - Option 2: SQL

    Choice: Option 1
    Justification: Some members of our team already have a basic introduction to MongoDB. Also, since we have opted to use ExpressJS as our server and React for frontend interaction, using MERN stack logically follows as it is the most compatible database for these frameworks, not to mention being free and easy to learn.

2. What framework and languages should we use for the backend?
    - Option 1: ExpressJS

- Option 2: Spring Boot

Choice: Option 1

Justification: ExpressJS works well with the frontend, React, that we have opted to use. It is also relatively easy to learn with a large community of developers and resources. ExpressJS is also boasted by most sources to be the best source for backend web development with Spring Boot being slightly more tedious and difficult to manage. We delegated seriously on the benefits of multithreading that Spring Boot has that ExpressJS does not, but ultimately decided that ExpressJS had far more benefits than Spring Boot or any other option.

3. What framework and languages should we use for frontend?
    - Option 1: Angular
    - Option 2: React
    - Option 3: Vue

    Choice: Option 2

    Justification: We have elected to use React over the also popular frameworks Angular and Vue for several reasons. The biggest reason is that members of the team are interested in learning the language, because it is the most popular frontend framework in the industry right now with a huge and still growing developer community. It very easy to learn on top of offering reusable components, increased efficiency, and efficient debugging, all of which will be vital to use during our sprints.

4. If we decide to host, what hosting service should we use?
    - Option 1: IONOS
    - Option 2: AWS
    - Option 3: Azure

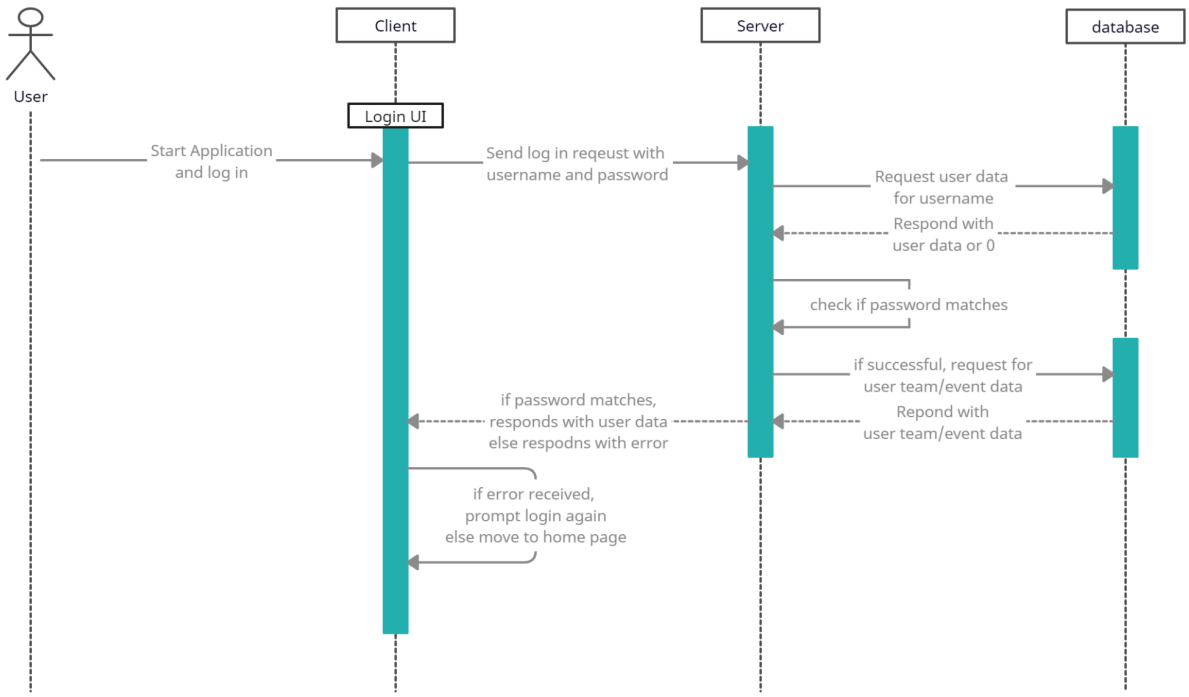
    Choice: Option 2

    Justification: We narrowed our options down to AWS and Azure for hosting purposes because they are both very popular among developers with a reliable reputation to fall back on as well as up-to-date maintenance and access to technical support. From there, we looked at what databases we have already committed to using and realized that on top of AWS generally offering easier deployment, it also works better with the NoSQL database, MongoDB.

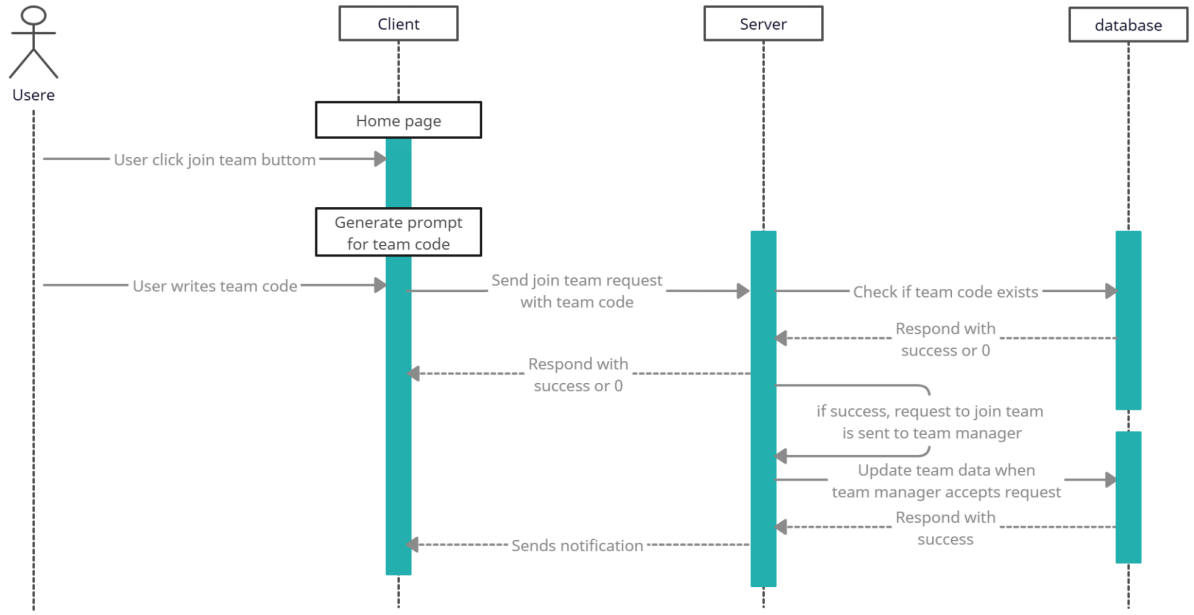
Design Details

Sequence Diagram

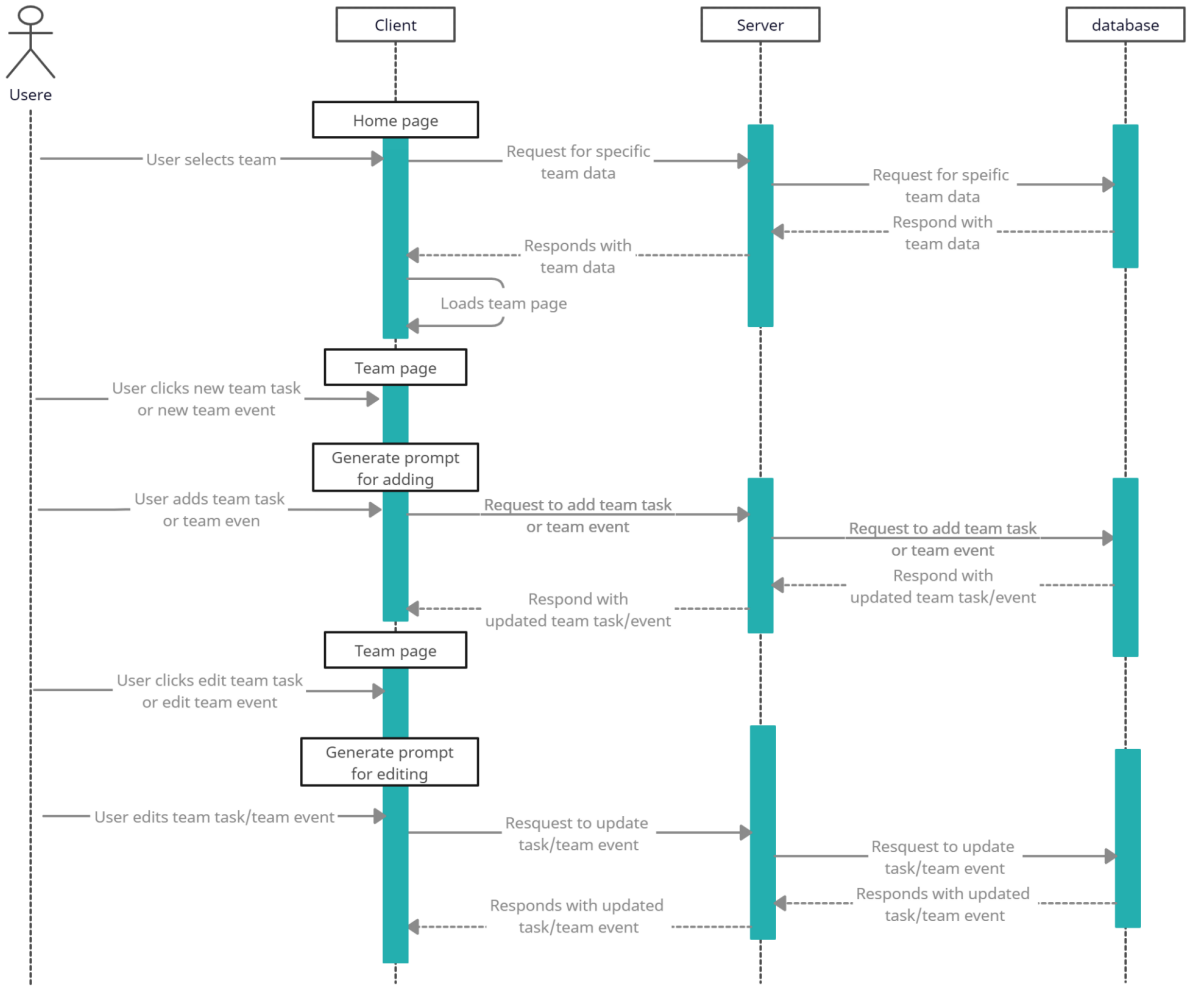
1. Sequence of event when user log in



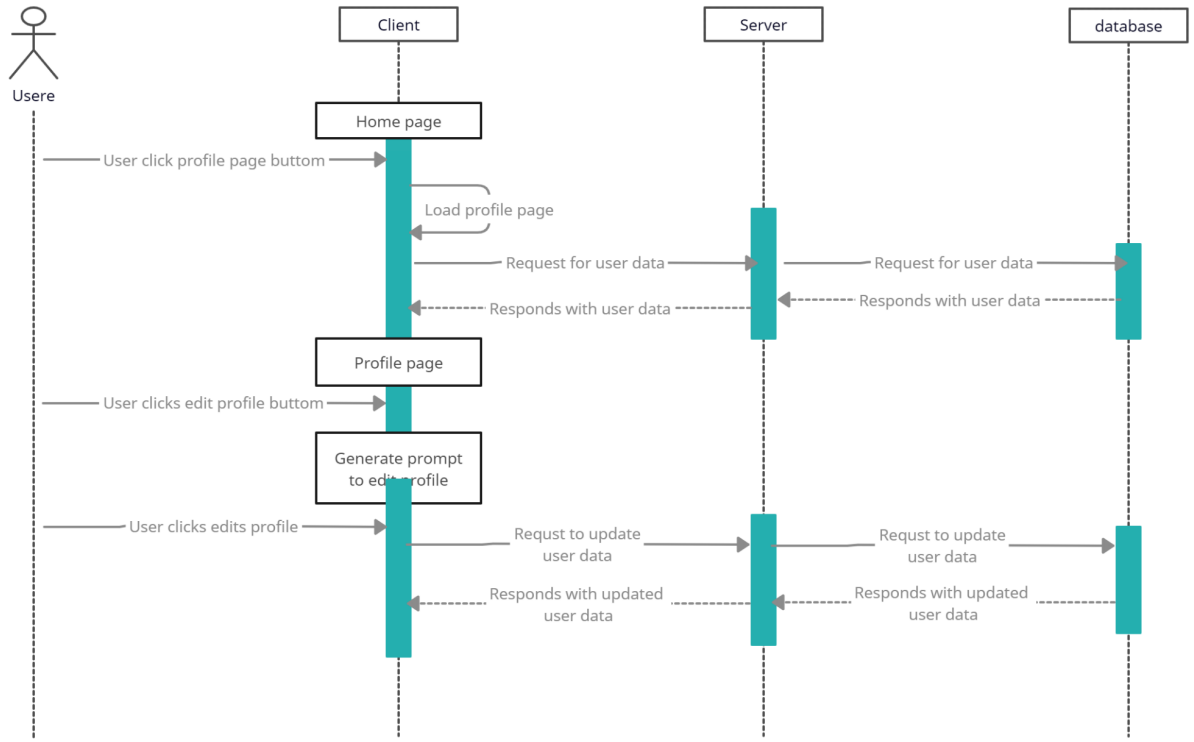
2. Sequence of events when user tries to join team



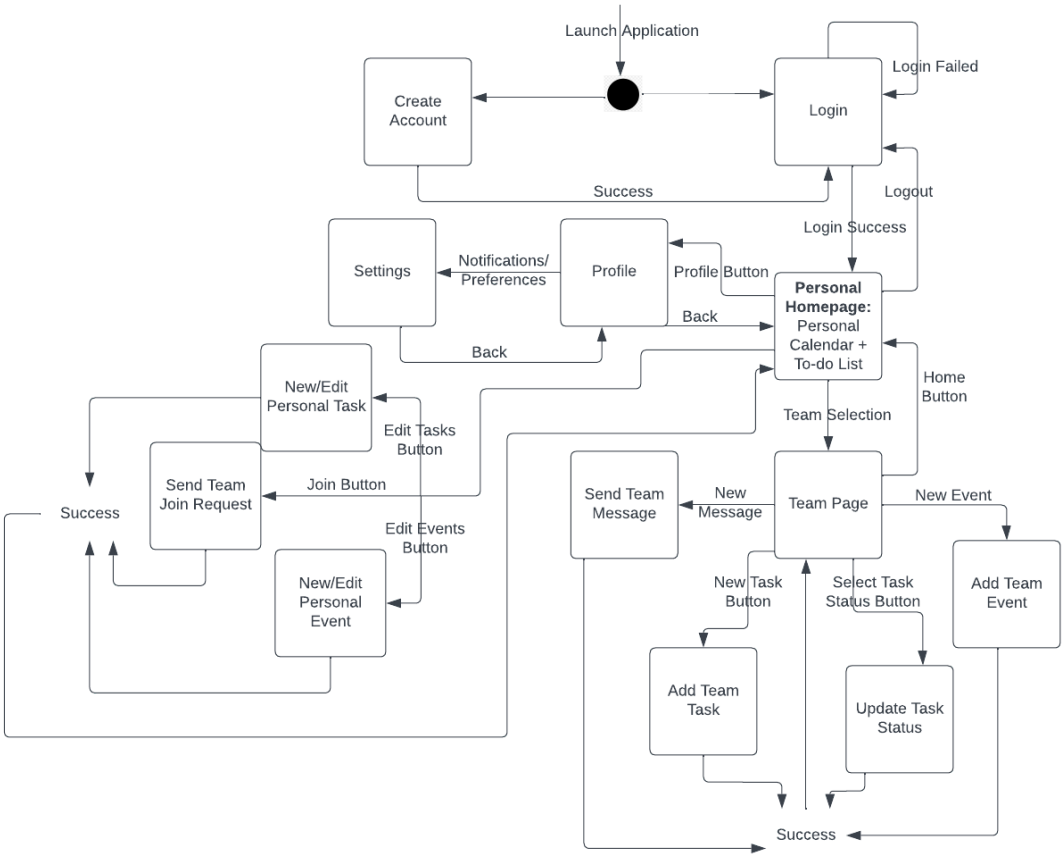
3. Sequence of events when user add/edits team data



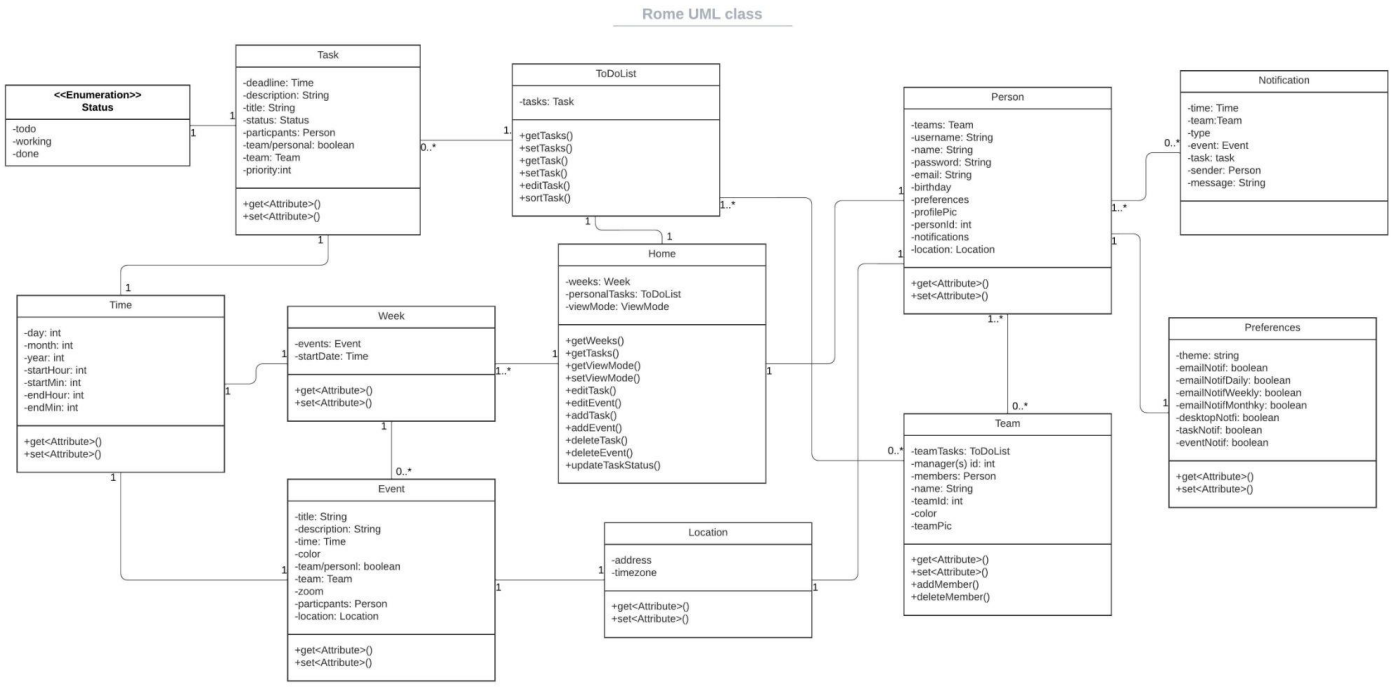
4. Sequence diagram when user edits their profile



Navigation Flow Map



Class Design



Person

- Person object is created when a new user creates an account
- Each Person has a unique username, and a password used to login in and track users
- Each Person also has a unique person id used to track user permissions and ownerships, this will allow for more efficient comparison when searching for a user as person id is an integer and you will not have to compare using strings if only using the username
- Each Person will also fill in there personal information such as birthday, name, location, profile pic, and email
- Each Person will also have a list of teams that they are members of
- Each Person has a preferences object that stores all of their settings and personalizations

Preferences

- Preferences is a subclass of Person and is used to store preferences information
- These preferences include personalization and notification settings
- This is a separate class as it will only need to be graded from the server when the user opens preferences page

Home

- The Home class is used as the container for all of the user interaction with the rest of the class structure from the home page
- It holds the list of weeks that will be currently loaded onto the client side for their calendar
- Additionally Home hold a ToDolist object that stores all personal tasks for the user
- Mostly Home is used to store methods that the user will invoke by interacting with the UI

Team

- A Team object represents all of the information for any given team
- A Team object can be created by any Person who will become the manager of the team
- A Team stores the Person ID of the manager
- A Team also has a name, team pic, and color
- A Team contains a list of Tasks for the entire team
- A Team also contains a list of members for all People in the team

Task

- A Task object represents each individual task on a Person or Teams to-do list
- A Task can be created by any Person
- A Task can be created for the your personal to-do list as well as a team to-do list
    - When created on the home page the Task will be a personal
    - When created on a Team page the task will be for that given Team
- Each Task object must have a title
- Each Task also has a team/personal boolean that determines if the task is a personal task or a team task. By having this boolean it can easily determine the relevance of other attributes.
- Each Task object can also have a description, a deadline, a list of participants, a priority, and a Team it is associated with
- Tasks exist as listed objects with ToDoList

Event

- An Event task represents the individual events that will be with-in the calendar
- Any user can create an event on their own personal calendar
- A manager can create team events that get added to team members calendar
- Each Event will have a title and time
- Each Event also has a team/personal boolean that determines if the task is a personal task or a team task. By having this boolean it can easily be determined the relevance of other attributes.
- Additionally each event can have a description, a location, a team, and a participants list

Notification

- A Notification exist in each person notification list
- All notifications are automatically created by the software when a person is notified. These situations include
    - A message is sent by another user
    - A deadline is approaching or has been reached for a task
    - Am event is coming up soon or has started
- Notifications store the time in which the notification was sent, a message, and a type
- The type is used to differentiate the notification as a message, task deadline, or event reminder
- Depending on the type, a Notification can have
    - A sender, or the person who sent you a message\
    - A task, the task in which a deadline is referring to
    - An event, the event in which the notification is reminding you of

Time

- The time object is used to easily stored all types of time data for deadlines, event times, and notification sent times
- A Time object has a day, month, year, start hour, start minute, and am/pm boolean
- Depending on the use of the Time object it may also have end hour and end minute. These may be used for an event as it has an elapsed time that it takes place so it need a start and end time
- A Time object may not use the end attributes if it is for a task or notification that just has one given time and does not need to take any time but be one given time

Location

- The Location object is used to store all location information
- Stores address information as well as timezone
- Timezone can be used to synch time if members are across multiple time zones
- Location object exist as an attribute in the Person object
- Events can also have a location associated with them

Week

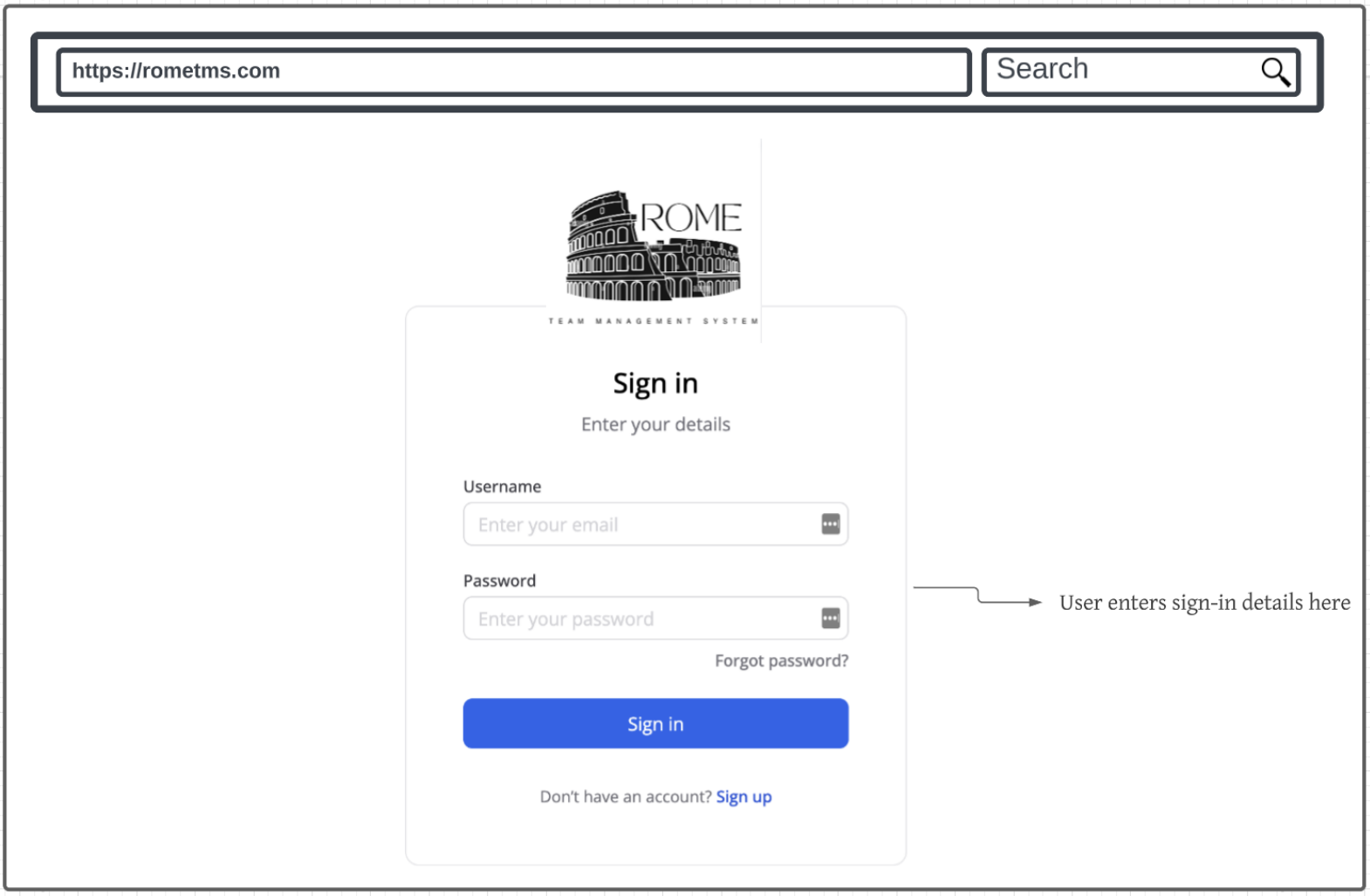
- The Week object is used to store all of the events in a given week
- The Week object has a list of events, as well as a start date. Only a start time is need as weeks are a set length
- The Week object is useful as the calendar will be traversed in weeks at a time so by consolidating all of the events into week objects it is easier for client-server communication than individually requesting all of the events of a given week.

ToDoList

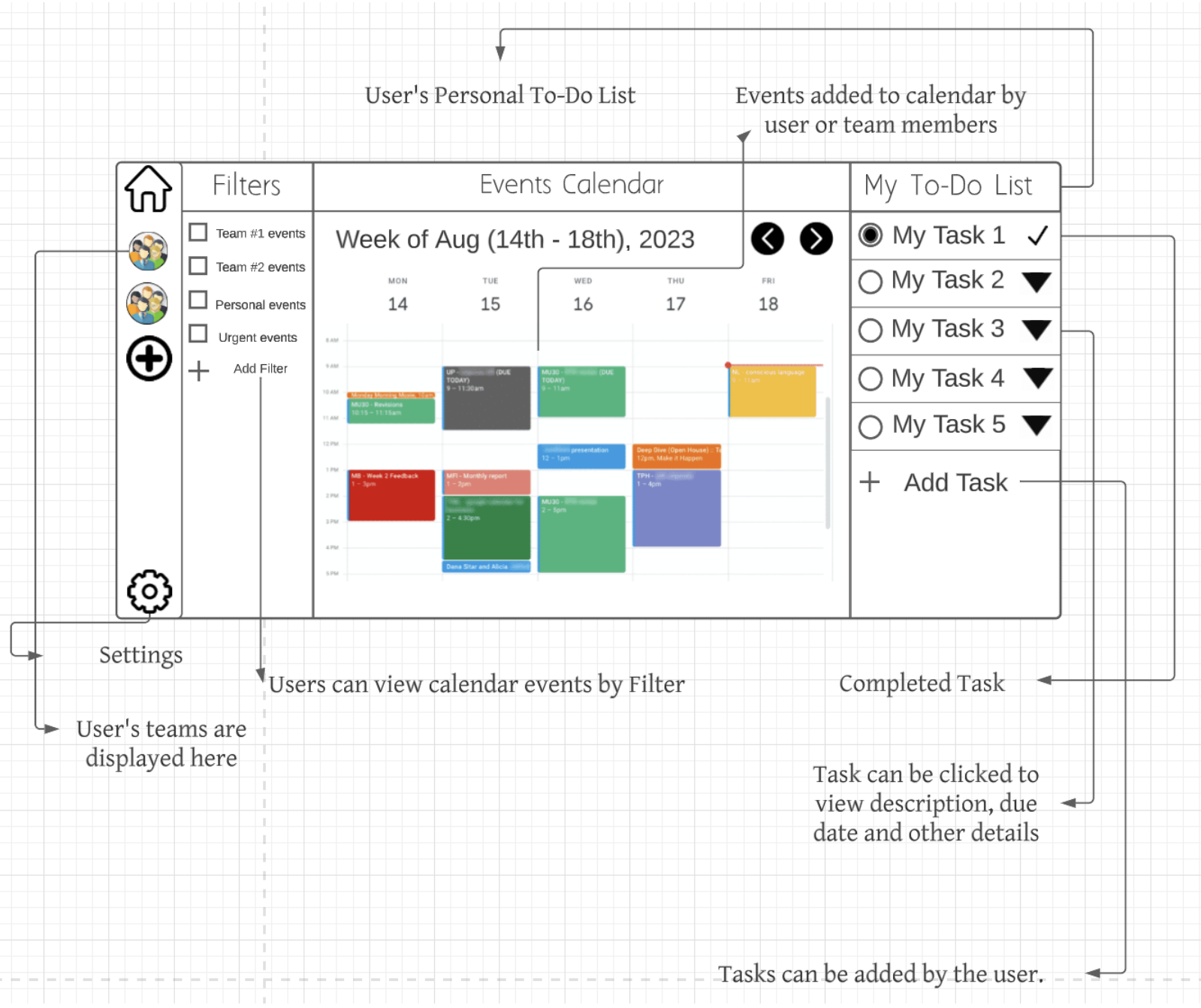
- ToDoList is a simple class that just holds a list of Task object
- The rationale for this class is to have a list object that has methods for easily manipulating the list as a whole

UI Mockup

The UI is clean, simple and intuitive. This is the Login page. It contains fields for the user to enter their username and password. There is also a button in case the user forgot their password and the option to sign up for a new account.



This is the first page the User sees when they login. The main page is a calendar that displays events based on the filters selected from the filters panel. The calendar can be navigated using the left and right arrows. On the right side, the user's personal tasks are displayed as a to-do list. Tasks can be checked/marked as completed. Tasks can also be clicked to view further details like description, due date, etc. The user can also add tasks. On the left-hand side, there is a navigation panel that has the home option (leads to this page), and the teams the user is a part of. There is also an option to add a team, either join or create your own, indicated by the '+' sign. The settings button indicated by the gear at the bottom of this panel allows the user to change their preferences.



This is the group tasks board. This is what is displayed when the user clicks on a team from the left panel. The tasks displayed are the ones for the team the user selected and is a part of. The tasks are distributed into 3 containers: To-do, In Progress and Done. The tasks have their due date, description and a user associated with whoever in the team is working on one. On the right hand side, the messenger option allows users to communicate with individual teammates or teams.