Final Project for SW Engineering Class
CSC648-848 Section 01 Summer 2024

Team 05 - Concat
TeamMate - Personnel Management

Name	E-Mail	Role
Krishna		Team Lead
		Frontend Lead
		Frontend Dev
		Backend Lead
		Backend Dev
		Database Lead
		Database Dev
		Github Master

Demo Link

http://ec2 compute-1.amazonaws.com:3000/

Milestone 5
August 01 2024

Revision	Date Submitted
M1	13 June 2024
M1V1	20 June 2024
M1V2	09 July 2024
M2V1	09 July 2024
M2V2	23 July 2024
M3V1	23 July 2024
M3V2	30 July 2024
M4V1	30 July 2024
M4V2	01 August 2024
M5	01 August 2024

Table of Contents

Product Summary

TeamMate is a comprehensive personnel management solution designed for small to mid-sized groups, such as non-profits and community organizations. It consolidates essential management functions into a single, user-friendly platform, enabling easy registration, member tracking, event management, and award issuance.

TeamMate’s standout and superior feature is its Automated Recognition and Progress Tracking System. This system automatically tracks member participation, awards achievements, and updates progress, providing timely recognition and broad overviews of individual and group accomplishments. This holistic view is valuable for performance reviews, planning, setting future goals along with motivating the members of the organization.

Discord focuses on communication and lacks recognition and structured progress tracking, **Airtable** and **Google sheets** require extensive manual setup to achieve similar functionality, whereas **TeamMate** offers an out-of-the-box solution that simplifies administrative tasks. **PERSCOM** provides good tracking but demands more manual effort for recognition, while **Rippling HCM** caters to larger organizations with complex and costly systems, making it less suitable for smaller groups.

Committed Functions

Account

- Users shall register for a new account.
- Accounts shall have unique emails.
- An account shall have a username.
- An account shall have a password.
- An account shall have secure passwords as per modern standards.
- Users shall log into an existing account.
- An account shall create a new organization.
- Accounts shall join an existing organization.
- Accounts shall view all organizations they have joined.

Member

- Member permissions shall be members, group leaders, or admins.
- Admins shall assign member and group leader permissions.
- Admins shall be able to issue awards to members.
- A member's page shall show their information for that organization, along with all attributes, qualifications, awards, etc as a dashboard.

Organization

- An organization shall have an owner.
- An owner can assign other members to be admins.
- An organization shall have a roster page that shows all members in the organization separated by groups.
- Users shall click on any member on the roster to access their member page.

Awards

- Admins shall create award types.
- Admins shall update award types.
- Admins shall delete award types.
- Awards shall have a title.
- Awards shall have a description.
- Admins shall directly assign awards.
- Award instances shall have a date received.
- Award instances shall state the issuing member.

Qualifications

- Admins shall create qualifications types.
- Admins shall update qualifications types.
- Admins shall delete qualifications types.
- Qualifications shall have a title.
- Qualifications shall have a description.
- Admins shall directly assign qualifications.
- Members shall view individual qualifications.
- Qualification instances shall have a date received.
- Qualification instances shall state the issuing member.

Events

- Admins shall create events.
- Group leaders shall create events.
- Admins shall assign members to an existing event.
- Group leaders shall assign members to an existing event.
- Members shall be able to join an event.
- Events shall have a title.
- Events shall have a description.
- Events shall have a start datetime.

Application

http://ec2 compute-1.amazonaws.com:3000/

Milestone Documents

Attached below are the version 2 pdfs of all the milestones up to milestone 4.

SW Engineering CSC648-848-05 Summer 2024

TeamMate
Personnel Management
Team 05 - Concat

Team Members

Name	E-Mail	Role
Krishna		Team Lead
		Frontend Lead
		Database Admin
		Docs-Editor
		Github Master
		Backend Lead

Milestone 1
July 09 2024

Version History

Revision	Date Submitted
Initial Draft	13 June 2024
Revision 1	20 June 2024
Revision 2	09 July 2024

Table of Contents

Executive Summary

TeamMate is a comprehensive personnel management solution specifically tailored for small to mid-sized groups. Recognizing that smaller non-profit organizations and community groups often lack the resources or need for full-scale professional HR services, TeamMate fills this gap by providing an accessible and user-friendly platform. Designed for less formal environments, TeamMate offers a range of customizable features that also make it adaptable for professional settings. Instead of juggling multiple services to handle various aspects of personnel management, TeamMate consolidates these functions into a single, streamlined platform.

TeamMate is designed to provide leadership with a comprehensive overview of their organization's members, enabling them to quickly assess activities and track progress throughout each member's tenure. The platform showcases individual accolades and accomplishments, making them visible to other users within the organization, while allowing administrators to issue said accolades. TeamMate includes robust event management capabilities, allowing administrators to easily set up and manage various events such as meetings, training sessions, workshops, and other organizational activities. The system tracks attendance, ensuring that all participation is documented and easily accessible for future reference. Additionally, TeamMate provides tools for monitoring and managing qualifications and certifications. Administrators can keep track of who holds specific certifications and when they were obtained.

TeamMate empowers a diverse range of smaller communities. Whether it's tracking a student's progress, a gamer's achievements, a volunteer's service hours, or a non-profit member's contributions. TeamMate not only simplifies management but also enhances transparency and accountability within the group. The system fosters a culture of recognition and appreciation by highlighting individual achievements and contributions. This recognition encourages members to take pride in their efforts and motivates them to strive for excellence. By using TeamMate, organizations can create a motivating environment that drives continuous improvement and engagement.

Main Use Cases

Actors: Dave (Customer), Chris (Dave's Friend), Scouts (Users), TeamMate (Company)

Assumptions:

Dave has information about his scouts in separate offline platforms.

Dave, Chris, and Scouts all have reliable internet access.

Use Case #1

Dave is part of the leadership for Boy Scouts of America in the silicon valley. During COVID, the Boy Scout troops were hardly able to go outside and do their activities, Dave is worried that such events will negatively affect troop motivation and participation. A platform to display scout progress and achievements online appeals to him, as it would still allow the scouts to take pride in their achievements and showcase them without needing in-person meetings. Dave creates an account and an organization on TeamMate for all troops in the Silicon Valley. He uses the groups category to ensure each Troop has their own section, and imports all awards. As the scouts create their own account and join the organization one by one, Dave starts assigning them their awards and groups, but realizes that this is tedious work for one person. Dave gets his friend Chris to also create and account and join the organization, where he gives Chris administrator permissions. Now, Dave and Chris both assign groups and awards to the incoming scouts, and Dave is relieved that the workload is much more manageable. On the dashboard screen, Dave can see all the scouts in his organization, separated by Troops; he can click on a scout to open a page that displays information about them and their awards.

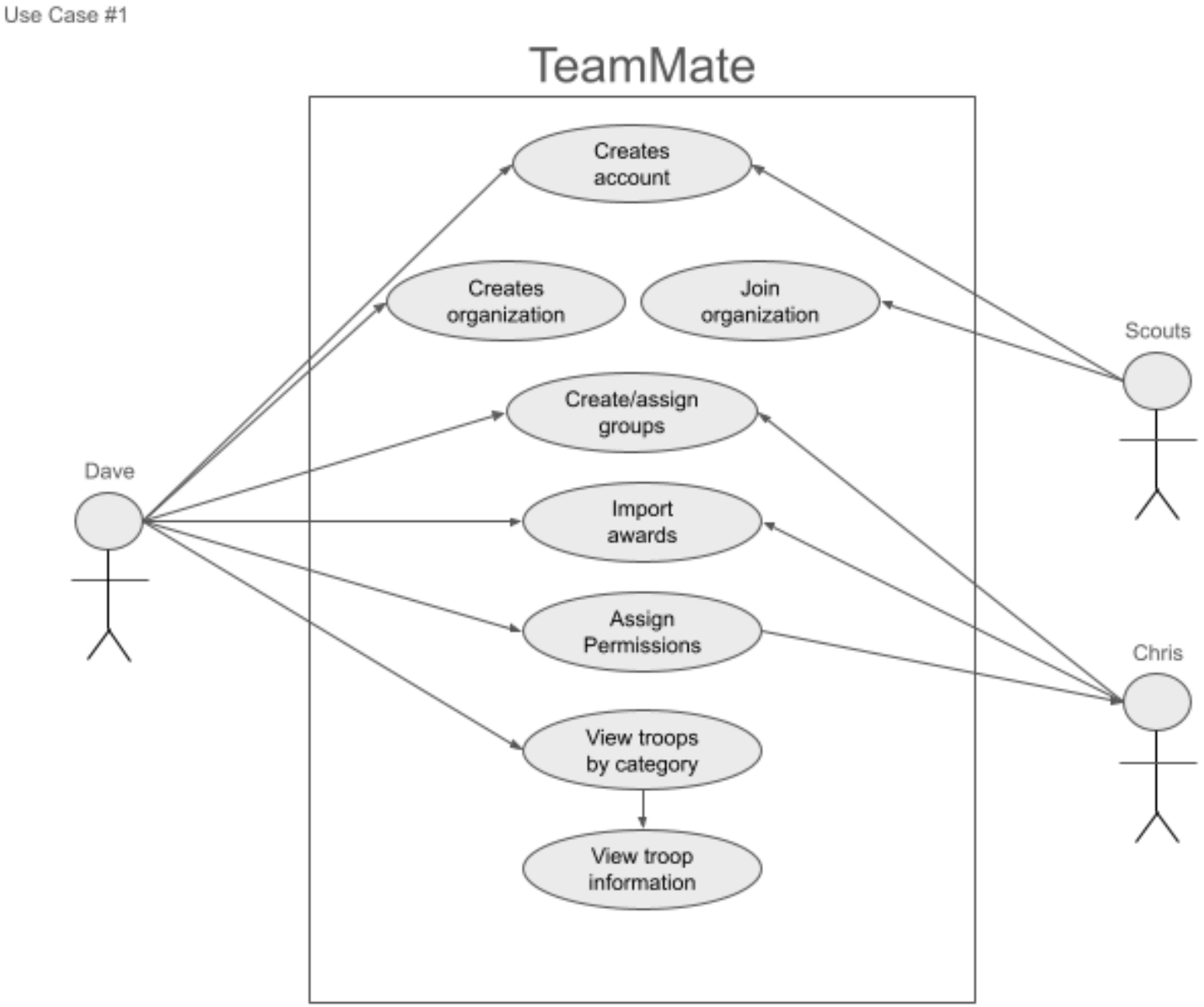
Benefits:

Dave and Chris can track all their scouts on one central platform.

Scouts, and their friends and family, can quickly check their progress.

Scouts feel recognized and motivated to participate more in their community.

Diagram


Use Case #1
TeamMate
Creates account
Creates organization
Join organization
Create/assign groups
Import awards
Assign Permissions
View troops by category
View troop information
Dave
Scouts
Chris

Actors: Jake (Customer), Specialists (Users), TeamMate (Company)

Assumptions:

Jake is keeping track of his specialists certifications separately.

Jake and his Specialists have access to the internet.

Use Case #2

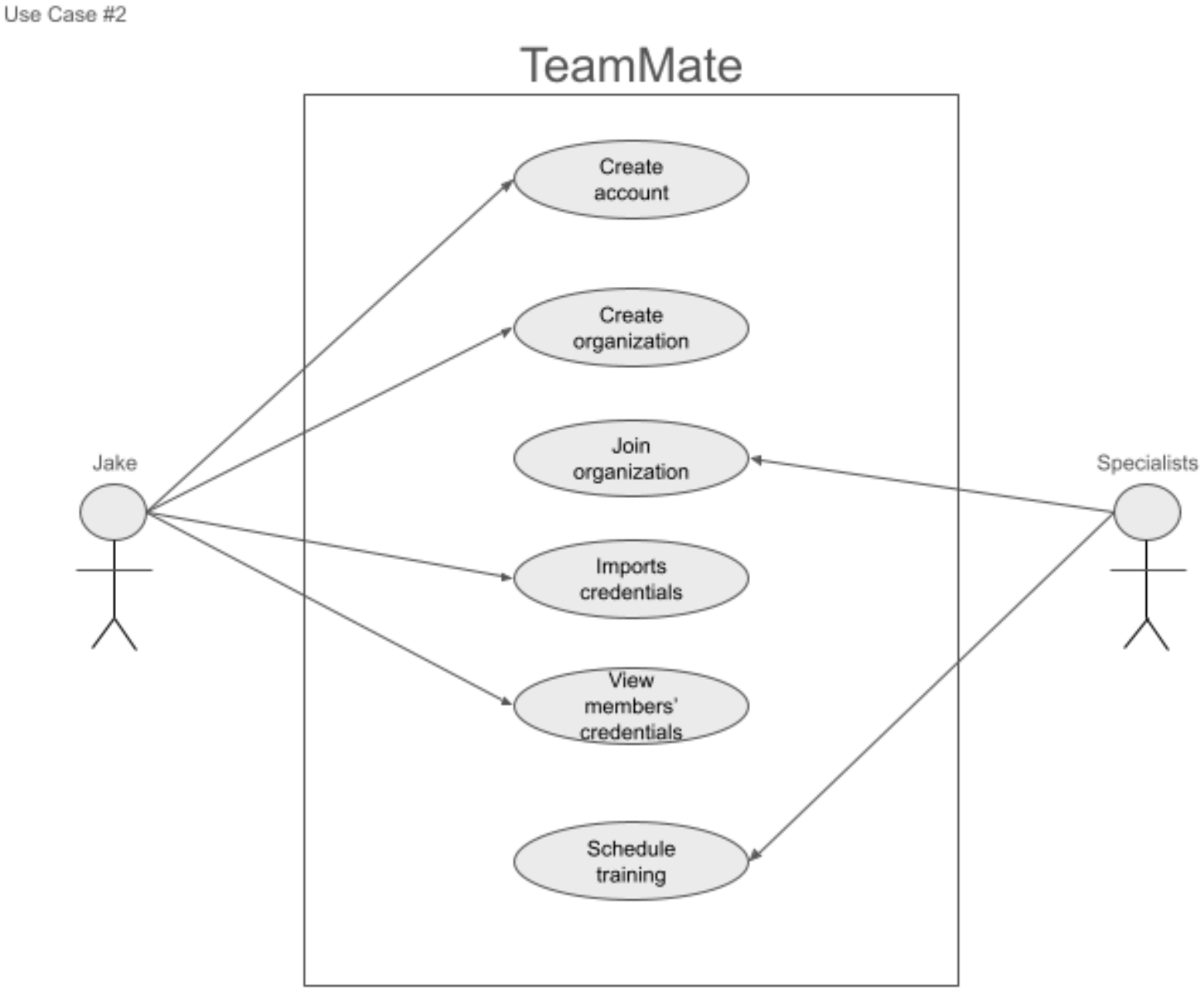
Jake is the leader of a professional group in a field with many qualifications. His specialists need these qualifications to be considered for work by potential customers. However, each of these certifications use their own system and platform. Jake finds it tedious and time consuming to track all of his subordinates' qualifications through the various platforms they acquired them from. In order to improve productivity and keep everything centralized, Jake creates a TeamMate account and organization. Jake's Specialists also create TeamMate accounts and join his organization. Jake then imports his Specialists' qualifications, certifications, and training into the system in order to see all of them at a glance. Jake can also schedule training for certifications, see who attends them, and update their qualifications accordingly. This centralized dashboard relieves Jake, since he no longer has to track many different systems and applications, but can instead have all required information on just TeamMate.

Benefits:

Jake can quickly check if his team has the qualification a customer asks for.

Jake can check which of his team members are attending the training events.

Diagram


Use Case #2
TeamMate
Create account
Create organization
Join organization
Imports credentials
View members' credentials
Schedule training
Jake
Specialists

Actors: Dr. Sarah (Supervisor), Researchers (Alice, Bob), Lab Assistants (Eve, Frank), Graduate Students (Heidi, Jose)

Assumptions:

Dr. Sarah has access to the TeamMate platform

Researchers, Lab assistants, and graduate students have created their accounts on TeamMate and joined the lab organization.

Use Case #3

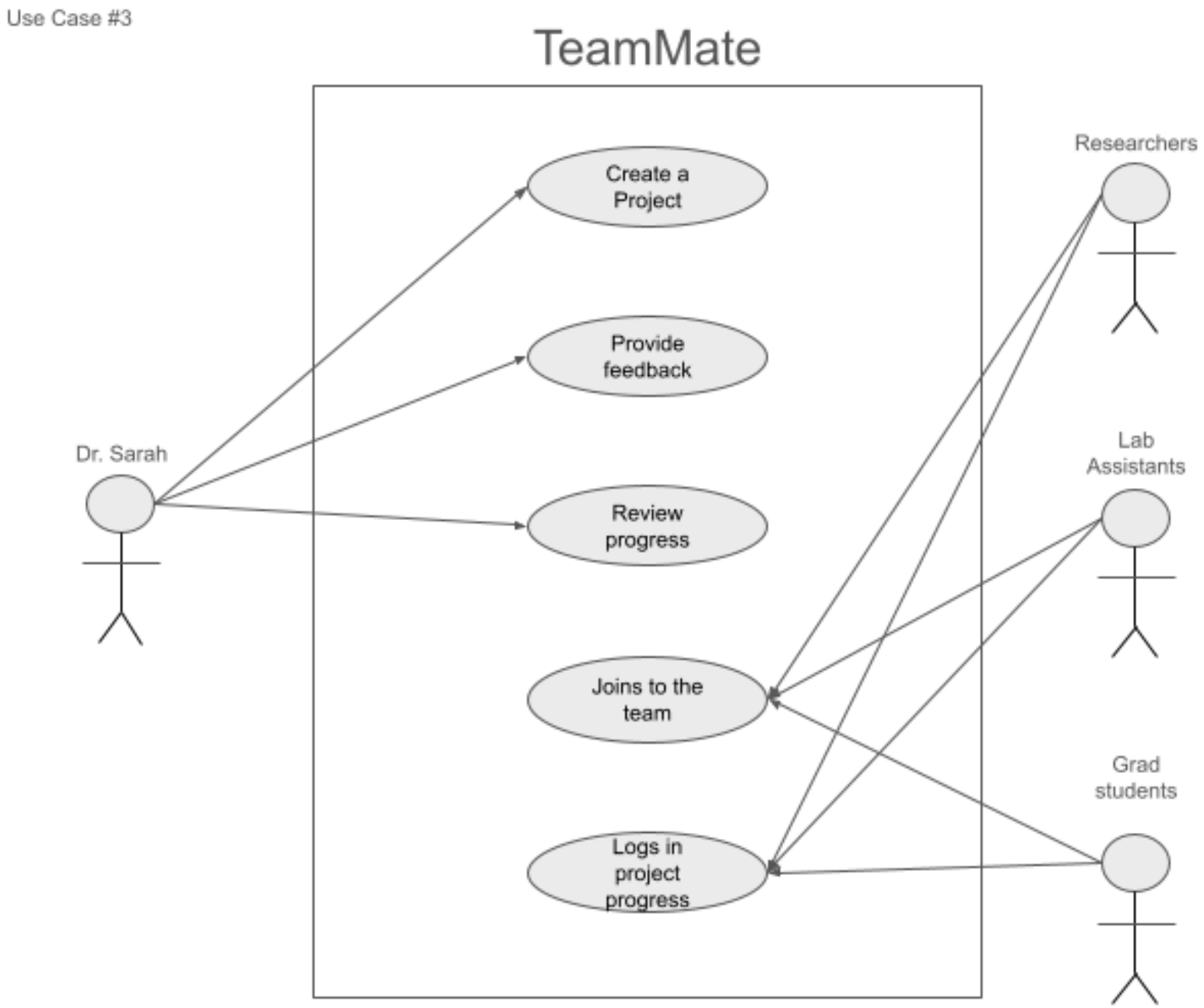
Dr. Sarah logs into the TeamMate platform and navigates to the custom “Project Management” section. She inputs the details of the new project, including objectives, timelines, and assigned researchers. Researchers log their progress and update project milestones regularly. Lab assistants support researchers by updating their progress on the tasks. Graduate students input their progress and updates on their specific research tasks. Dr. Sarah reviews the progress updates and provides feedback or adjusts timelines as needed. The system generates a progress report summarizing the status of all ongoing research projects and confirms the progress updates are saved and the report is generated successfully.

Benefits:

Dr. Sarah can efficiently monitor the progress of multiple research projects.

Researchers, Lab assistants, and graduate students receive timely feedback and support from the lab supervisor.

Diagram


Use Case #3
TeamMate
Create a Project
Provide feedback
Review progress
Joins to the team
Logs in project progress
Dr. Sarah
Researchers
Lab Assistants
Grad students

Actors: Dr. Susie (Supervisor/Lab Manager), Technicians(Ingrid, Jack), Maintenance Staff (Nancy, Billie)

Assumptions:

Dr. Susie has access to the TeamMate platform

The team members have created their accounts individually on TeamMate and joined the lab organization.

Use Case #4

Dr. Susie logs into the TeamMate platform and navigates to the custom “Equipment and Supplies Management” section. She inputs the details of the lab equipment and supplies, including quantities of them, locations, and maintenance schedules. Lab technicians log equipment usage and report any issues or maintenance needs. Maintenance staff schedules and performs maintenance based on the logs and reports from the lab technicians. Dr. Susie reviews the equipment usage logs and schedules maintenance as needed. The system tracks inventory levels and alerts Dr. Susie when supplies are low. Dr. Susie places orders for new supplies and updates the inventory accordingly. The system confirms the equipment and supplies logs are updated and inventory levels are tracked regularly.

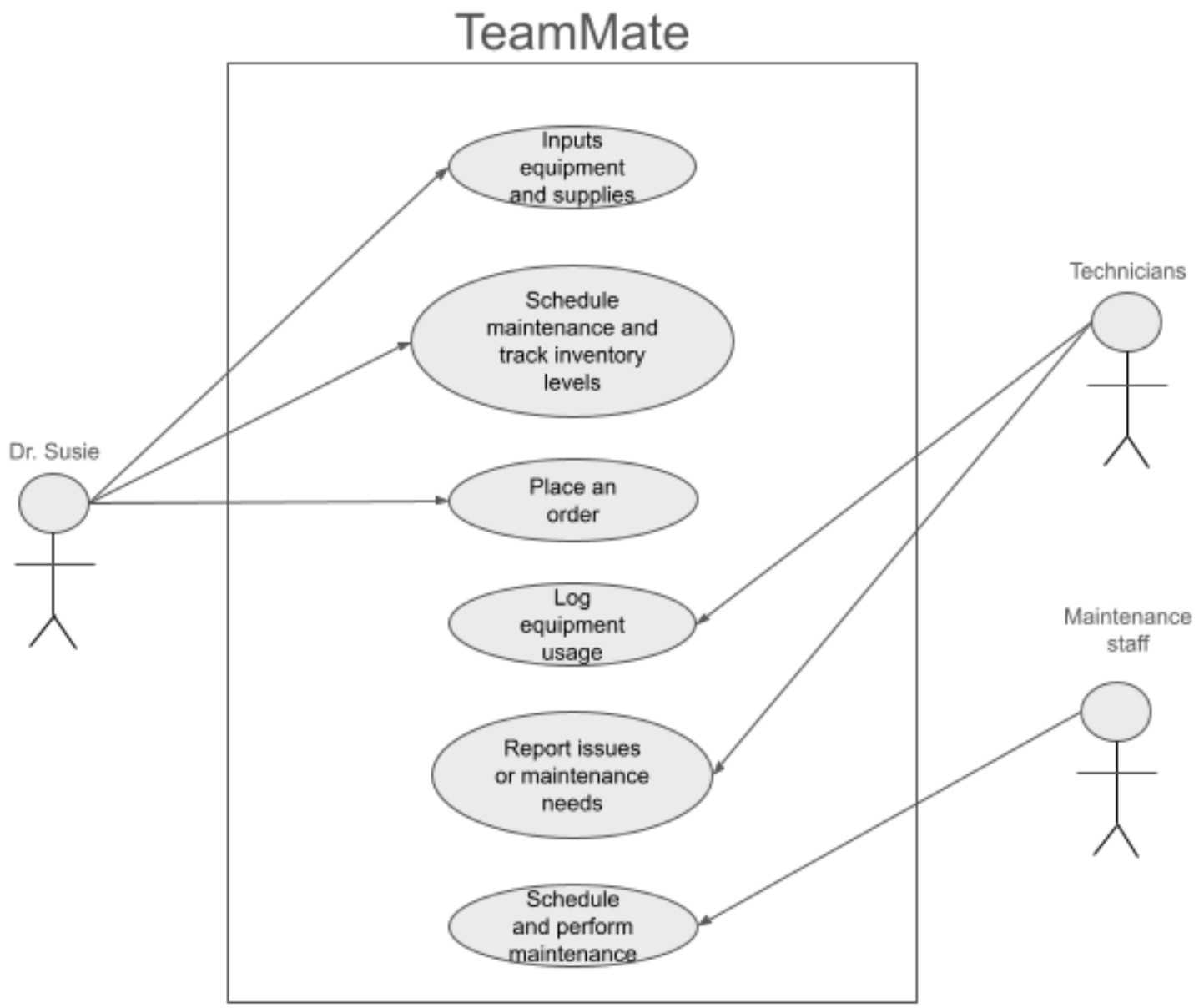
Benefits:

Dr. Susie can efficiently manage lab equipment and supplies in one centralized platform.

Equipment maintenance is timely and well documented.

Diagram

Use Case #4



Actors: Steve (Manager), XYZ staff (User), TeamMate(Company)

Assumptions:

Steve is knowledgeable in the usage of applications on websites and smartphones.

Steve and his staff have access to the internet

Use Case #5

Steve is a manager at a non-government organization, XYZ, and he's enthusiastic about recognizing activities and workflow in the organization because mostly, the body of the workforce is volunteers. At the same time, he is concerned that everyone in the company is at the same pace. In reality, it was difficult to monitor, encourage, and recognize everyone at the same time.

Meanwhile, he found out that an online platform TeamMate can track and provide a comprehensive overview of all the members, making sure no one is left behind by recognizing their activities. He can issue digital accolades to the XYZ staff to recognize their contributions, which are visible on their profiles. The TeamMate would suggest Steve or HR manager to issue digital badges when volunteers complete a certain amount of tasks or projects they were assigned. He can also share the recognition to the public via social media.

Benefits for Steve

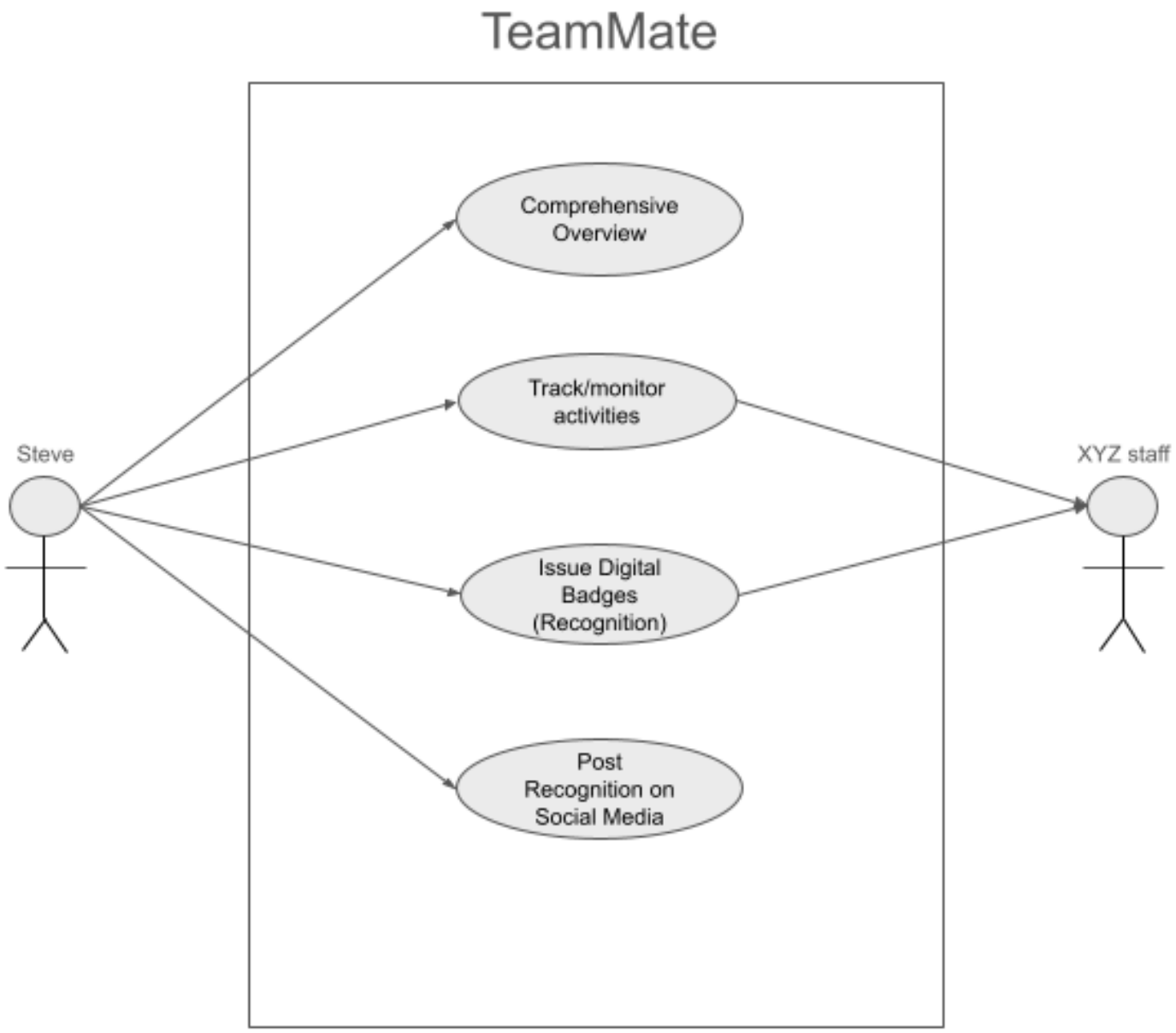
The app saves Steve time and energy by tracking workflow and issuing digital accolades to recognize volunteers' contributions.

The volunteers get motivated by the culture of public recognition to continue their efforts.

Steve is happy because the way the platform transforms the volunteers more organized, motivated, and dedicated to their tasks, missions.

Diagram

Use Case #5



Actors: Bob (Founder of Inferno), TeamMate (Company), Teams/ players (Inferno community)

Assumptions:

Bob is willing to try the platform(TeamMate).

All the teams create accounts on TeamMate, and reliable internet is accessible.

Use Case #6

Bob is a CS graduate from SFSU and he founded a gaming community called Inferno. After one year, the community can host gaming events across the Bay Areas with active 100 members. As members grow, it is challenging to host events, organize teams, track individual achievements, and post announcements. He could have hired more people, but he doesn't want to raise membership fees. He is worried that he could not manage teams and events perfectly.

One of the members introduces him to an online team/event/organization management platform (TeamMate) that can set up events, organize teams, check qualifications, track streamlined activities, recognize achievements, and a few to name. Now the gaming community creates events and tournaments with features of signing up more teams without schedule conflicts. The platform records attendance, ensuring not just teams, but individuals have detailed profiles that include biography, participating history, level, rewards, and achievements. The administrators can also make announcements on the platform.

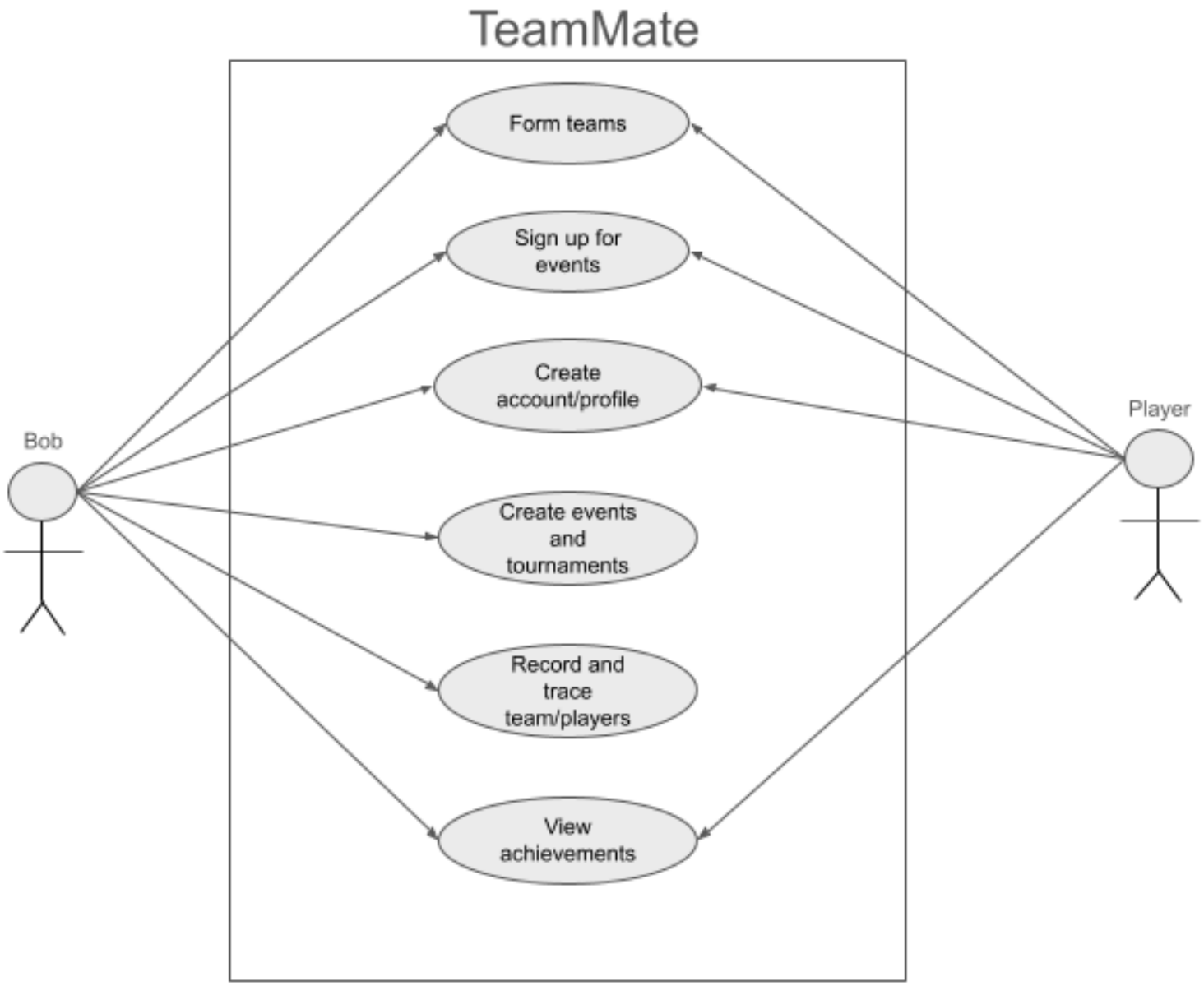
Benefits for Bob:

The online platform saves Bob resources and energy to handle teams, events and organization effectively.

Bob can access the platform anywhere and anytime.

Diagram

Use Case #6



Actors: Dan (Construction Company manager) Luigi, Angel, Joshi (Construction workers, users) Danliland (Construction Company), and TeamMate (App Company)

Assumptions:

Dan is currently having a difficult time interviewing new prospects for his construction company and knowing why other are failing to do the requested work

Dan is aware of the TeamMate App knows to use it and has introduced it to his workers

Use Case #7

Dan serves as a construction company manager at the Danliland construction company. As manager he employs at least 20 workers in his company. Some of his workers are Luigi, Angel and Joshi, as construction workers Dan asks them to complete a training survey in order to track what the employers need to learn about construction, what they already know, but also emphasize what they have learned in the past about construction working with other companies. Dan uses the TeamMate App to create an account for each worker so they can login and complete the survey within it. Having the name of each worker, the email, and experience in construction keep track of who has completed the survey and who has not, also track what needs to be learned while the workers work. With the app the construction manager is also able to manage when the survey was opened or completed. Depending on survey results some workers will have to meet with Dan

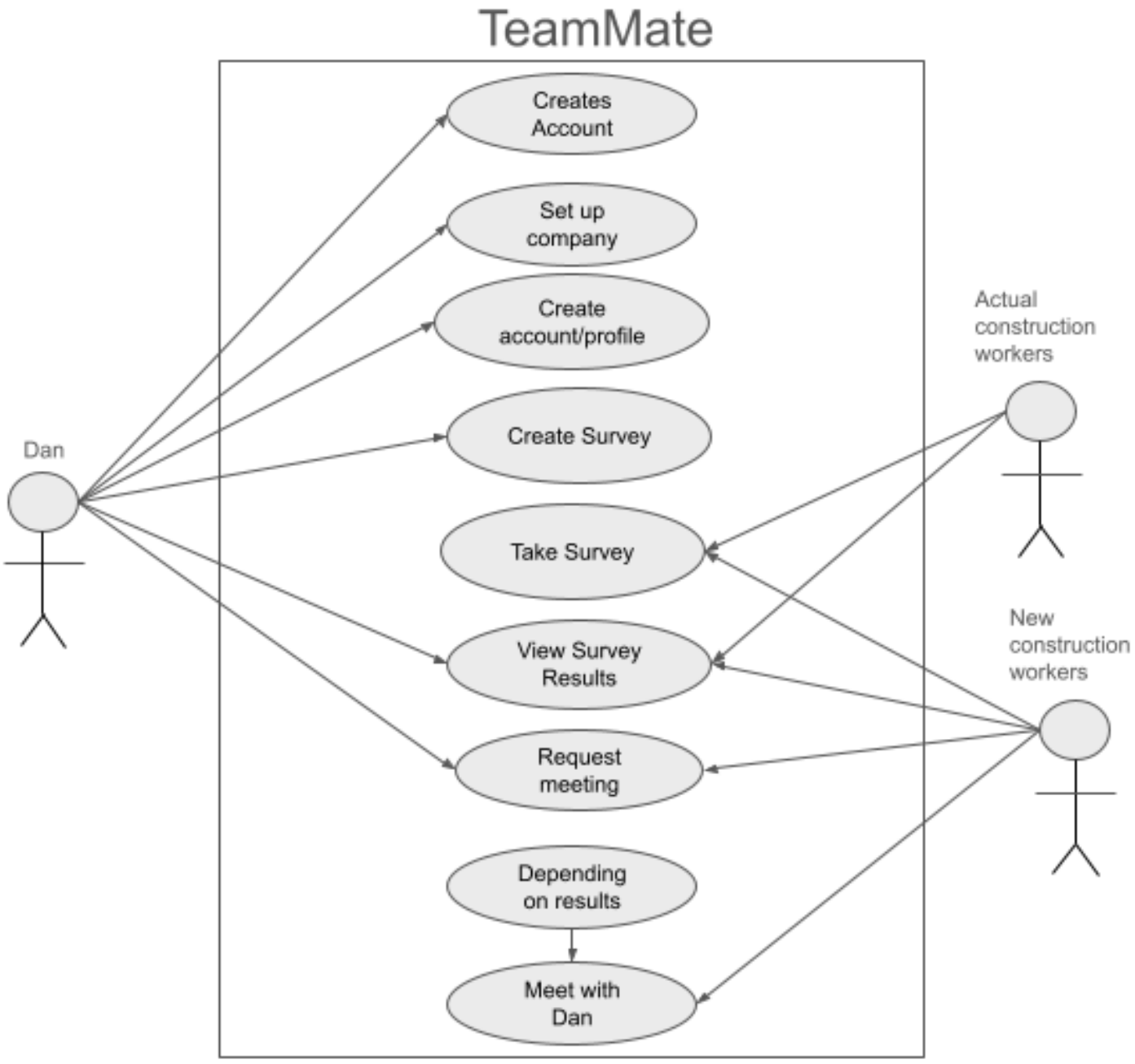
Benefits:

The TeamMate App is going to help Dan have a more organized life when he asks new and current employers to complete the work survey because he will keep track of what each worker needs to learn about construction, what they already know about construction, and what knowledge they plan to take out while working in a construction company

The TeamMate App can help Dan learn more about each of his workers (background wise) in relation to the experience they currently have in construction

Diagram

Use Case #7



Actors: Mary (Customer), Volunteers (Users), TeamMate (Company)

Assumptions:

Mary is currently tracking her volunteer activities and hours manually or using multiple tools

Mary and her volunteers have reliable internet access

Use Case #8

Mary works as a coordinator at a local non-profit organization that relies on volunteers to carry out and support its mission. She finds it overwhelming and challenging to keep track of the volunteers' hours, activities, and accomplishments using tools such as spreadsheets and manually keep track of these processes. In desperate need for a solution that can help streamline this process and provide a clear overview of each volunteer's contribution, her search begins, and she comes across the website, TeamMate. Mary creates an account on TeamMate and sets up her organization. Mary then is able to import historical volunteer data which includes their hours worked, tasks completed, and any awards they achieved. TeamMate allows her to easily assign new tasks to volunteers, track their hours in real time, and manage schedules for upcoming events. Volunteers can create an account, log their hours directly into the system, review their progress, and receive recognition for their contributions. Mary can also generate reports to showcase or highlight the volunteers' efforts which help recognize their hard work and secure future funding for the organization.

Benefits:

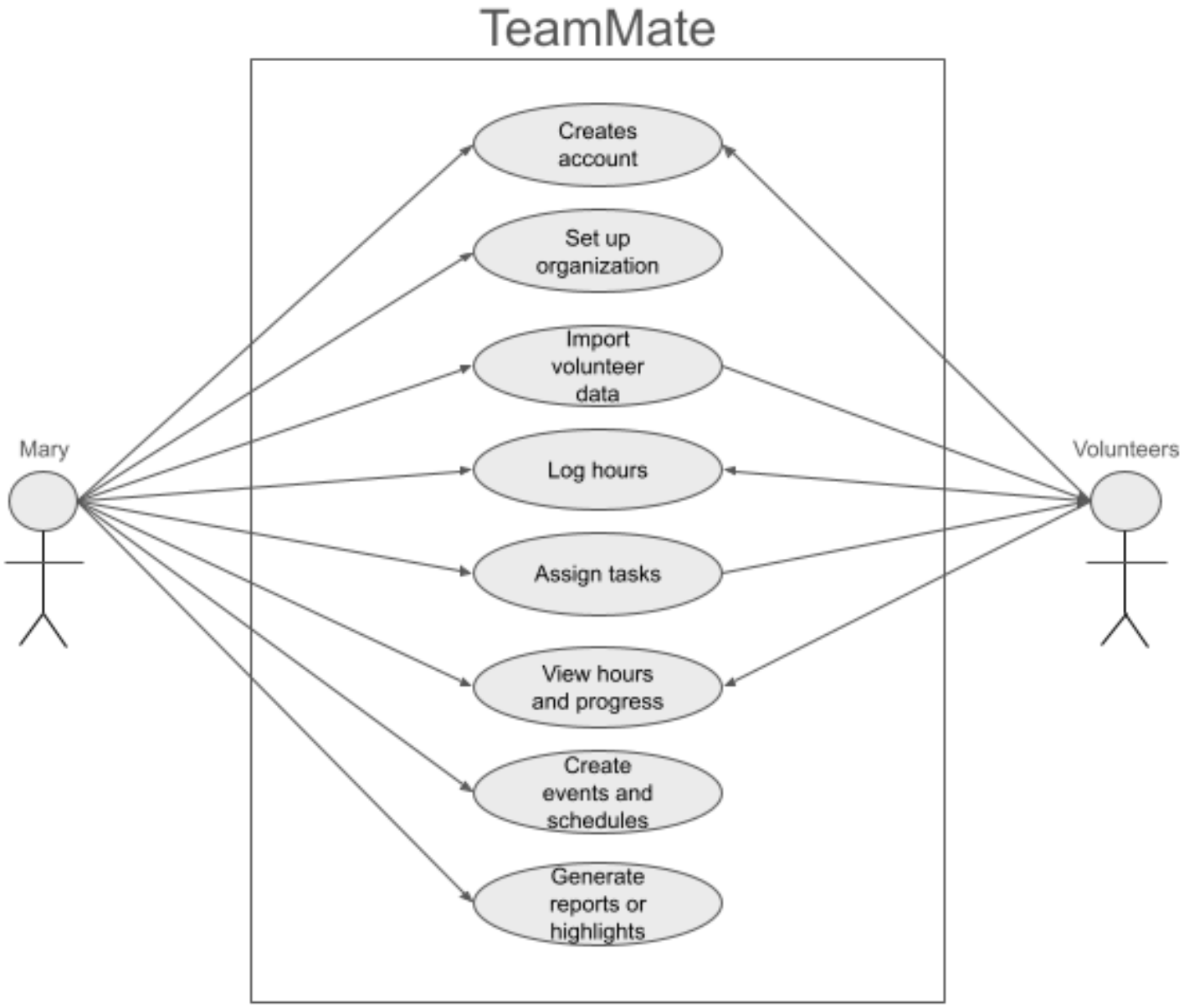
Mary is able to efficiently manage all volunteer activities and hours in one centralized platform.

Volunteers can easily log their own hours and view their contributions which can promote transparency and recognition.

The organization within TeamMate can provide detailed reports or highlights on volunteer activities that can further aid funding or grand opportunities.

Diagram

Use Case #8



Actors: Alex (Customer), Students (Users), TeamMate (Company)

Assumptions:

Alex is responsible for managing the school's gaming club

Alex and the Students have reliable internet access

Use Case #9

Alex is a teacher who is in charge of running a gaming club at a local high school. The club has recently grown in popularity, and many students participate in various games and competitions. However, it is becoming increasingly complex to manage the club with the increase of participants in the club. Alex needs to organize teams, track the players' rankings, and continuously find new opponents for the club to compete against.

Alex comes across TeamMate in his search for a tool to streamline the management of the gaming club. He creates an account on TeamMate and sets up the club's organization. Students create their own TeamMate accounts and join the gaming club. Alex can then create teams, assign players to these teams, update scores, and track individuals and team rankings based on performance in competitions. This platform also allows Alex to schedule matches, find new teams to compete against, and organize internal tournaments. The students can view their rankings, see their team assignments, and sign up for upcoming competitions. They are also able to communicate with their teammates and receive updates about matches and events. This centralized system allows Alex to efficiently manage the club while the students can stay informed and motivated to participate.

Benefits:

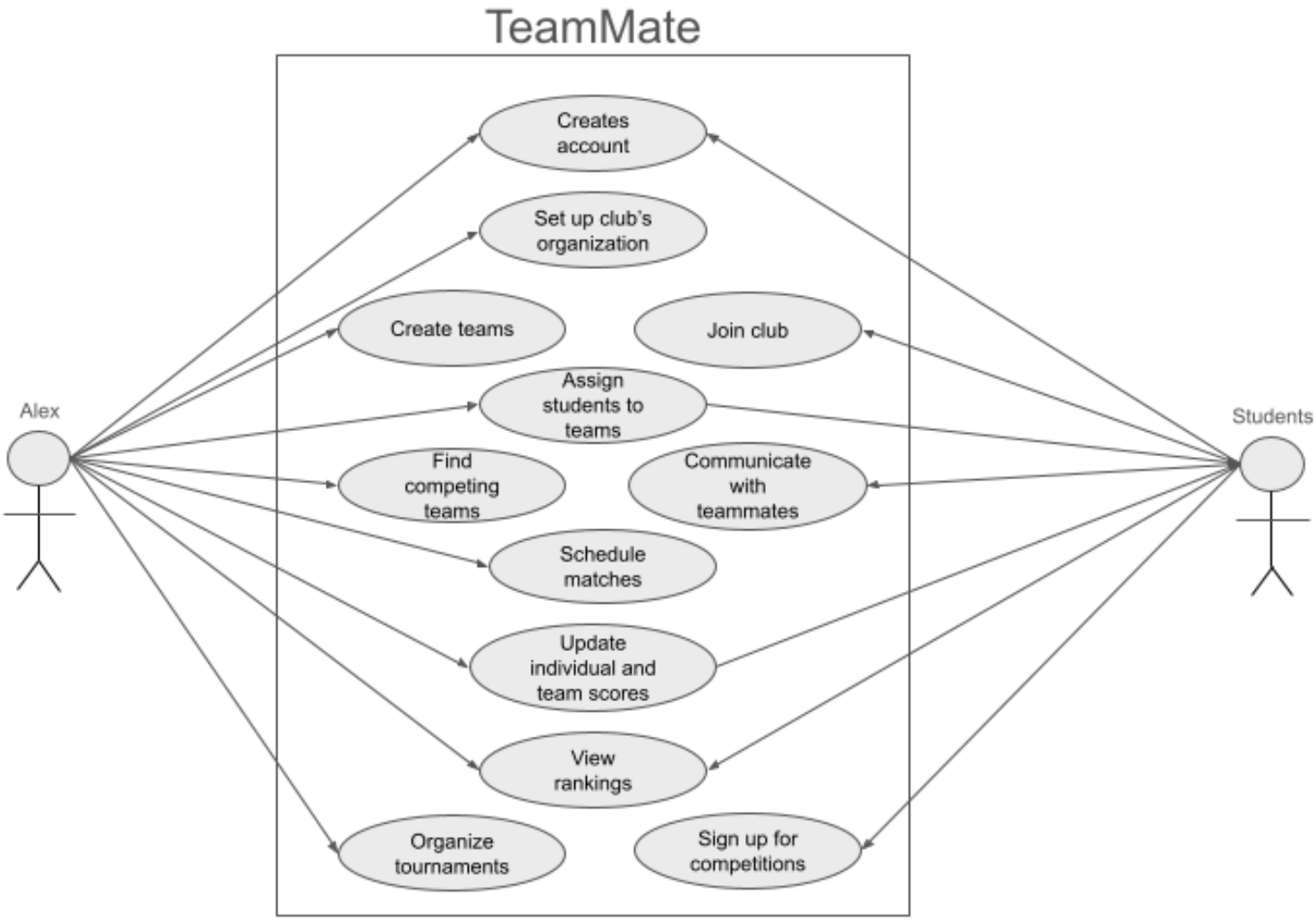
Alex can efficiently manage team assignments, track rankings, and organize competitions in one platform.

Students can easily view their ranking, team assignments, and view upcoming competitions.

The club can find new teams to compete against, enhancing the overall gaming experience.

Diagram

Use Case #9



Actors: John (Customer, a sport team leader), John's Teammates (Customers, sport team teammates), TeamMate (Company)

Assumptions:

John is a full-time basketball team leader, but he has a lot of part time jobs so it makes it hard to use the little time he has to manage the team he has.

John and his teammates have access to the internet and the TeamMate platform.

Use Case #10

John is a full-time basketball team leader, who is trying to get his team in the professional league. But right now he's running multiple part times to make sure the team has enough fundings. While the team is benefited by the extra money they have, John is having an extra hard time managing the team. Such as, to manage the team activities, setting up a training time and locations, keeping track of everyone's physical condition or even having group communications.

One member of the basketball team introduced John TeamMate, it was an app to make managing more convenience that promises to be a life changer. John created an account on this app, and setted up his organization. His teammates always create an account and join the organization. Then he can conveniently assign tasks for his team to complete, keeping track of the team progress, making announcements, or even rewarding the players who actively complete tasks. With the help of this app, John and his team feel more connected to each other. Players within the teammate will compete with a positive environment, also encouraging players to be more engaged..

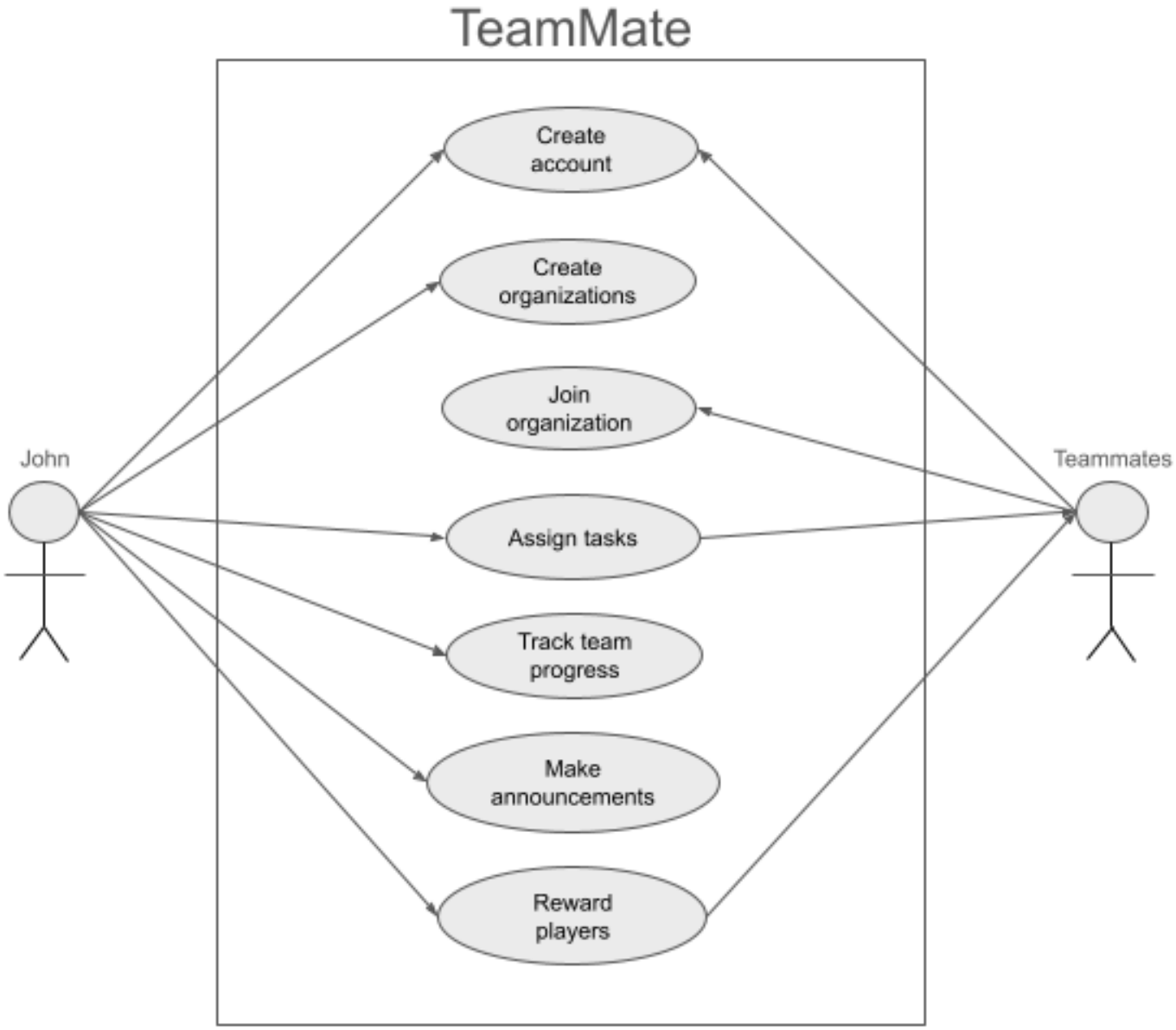
Benefits:

John can use TeamMate to manage the whole group, which it was hard to do with one app before. Increasing efficiency.

The features in the app keeps the team more connected to each other, which can potentially increase morale.

Diagram

Use Case #10



Actors: Eric (An orphanage Director, Customer), Faculty (Orphanage Faculty, Customers), Volunteers (Customers)

Assumptions:

The orphanage Eric runs lacks hands, he wants to hire some volunteers to help.

Everyone in the orphanage has access to a stable internet, and has basic knowledge about phone apps.

Use Case #11

Eric has a huge orphanage to direct, and it's lacking faculties to run the place. He's trying to hire some volunteers to help out, but he is concerned if he can manage all the volunteers who don't have much experience. Then one of Eric's colleagues introduced Eric the TeamMate app, which is an app to help management. This app comes just in time, Eric was feeling overwhelmed with work, and he can't manage the volunteers without help. He decides to create an account along with his faculty and volunteers. With the help of this app, which integrates communication, assigning tasks, rewarding system, and leaderboard all together, Eric should be able to manage the volunteers easier.

The use of TeamMate did create so much convenience, but Eric is still having a hard time managing due to how busy he is. Eric then lets one of the faculties create an account on TeamMate, who is assigned to be an administrator which allows him to also provide awards or assign tasks. Now with the help of his colleagues, Eric's workload is drastically decreased. Eric can even check the active volunteers, and make them the temporary leaders.

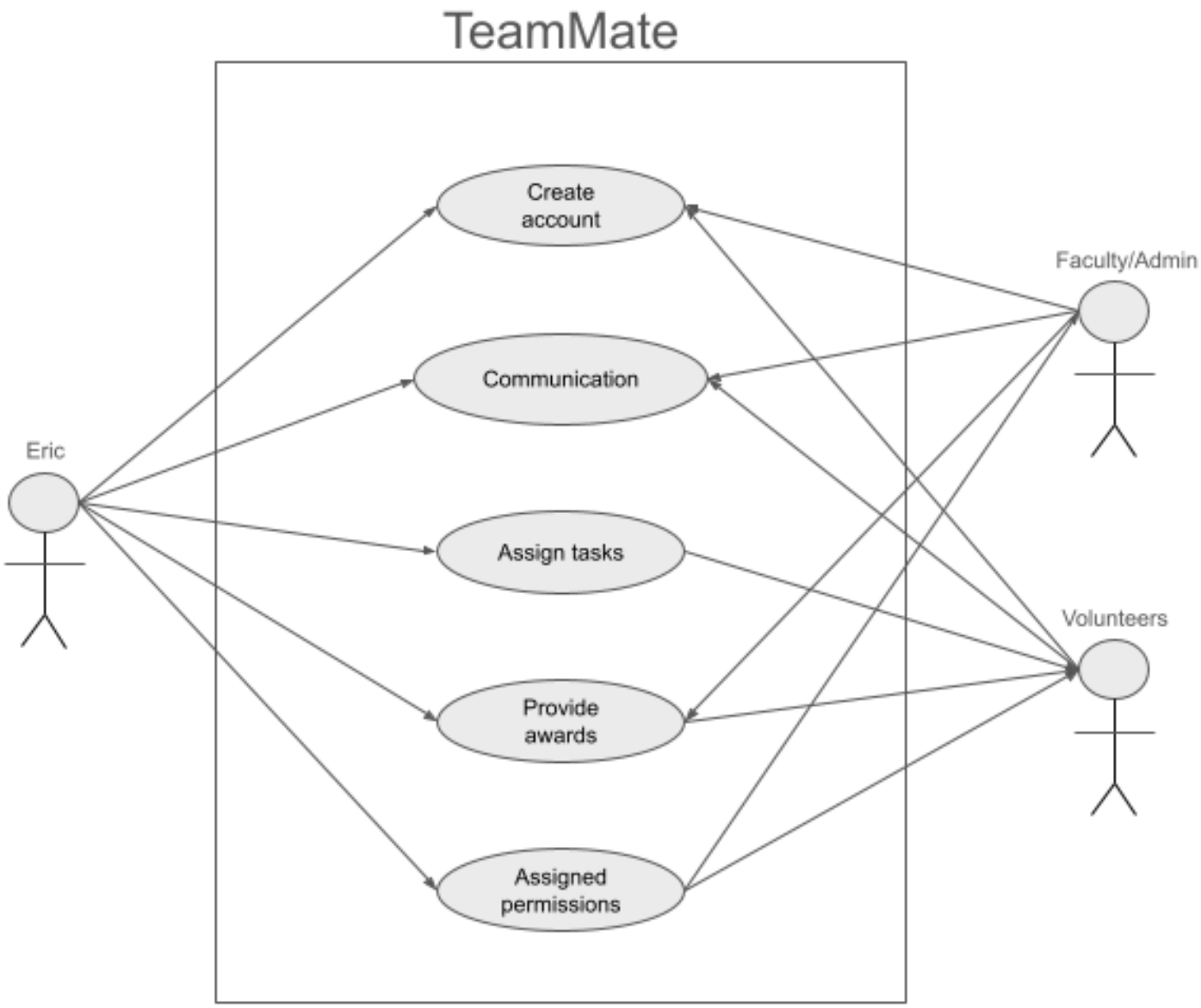
Benefits:

Increase the efficiency when managing a large group.

Can quickly see who's active over a group of people, and see who's not active.

Diagram

Use Case #11



Actors: Clianci (Yoga Center Director), Yoga students (users) TeamMate (Company)

Assumptions:

Clianci struggles to keep track of students' goals and what they need to do in order to achieve them.

Clianci has used other apps in the past to keep track of her goals but no apps seem to help her.

Students can oftenly keep track of what goal they are planning to achieve.

Use Case #12

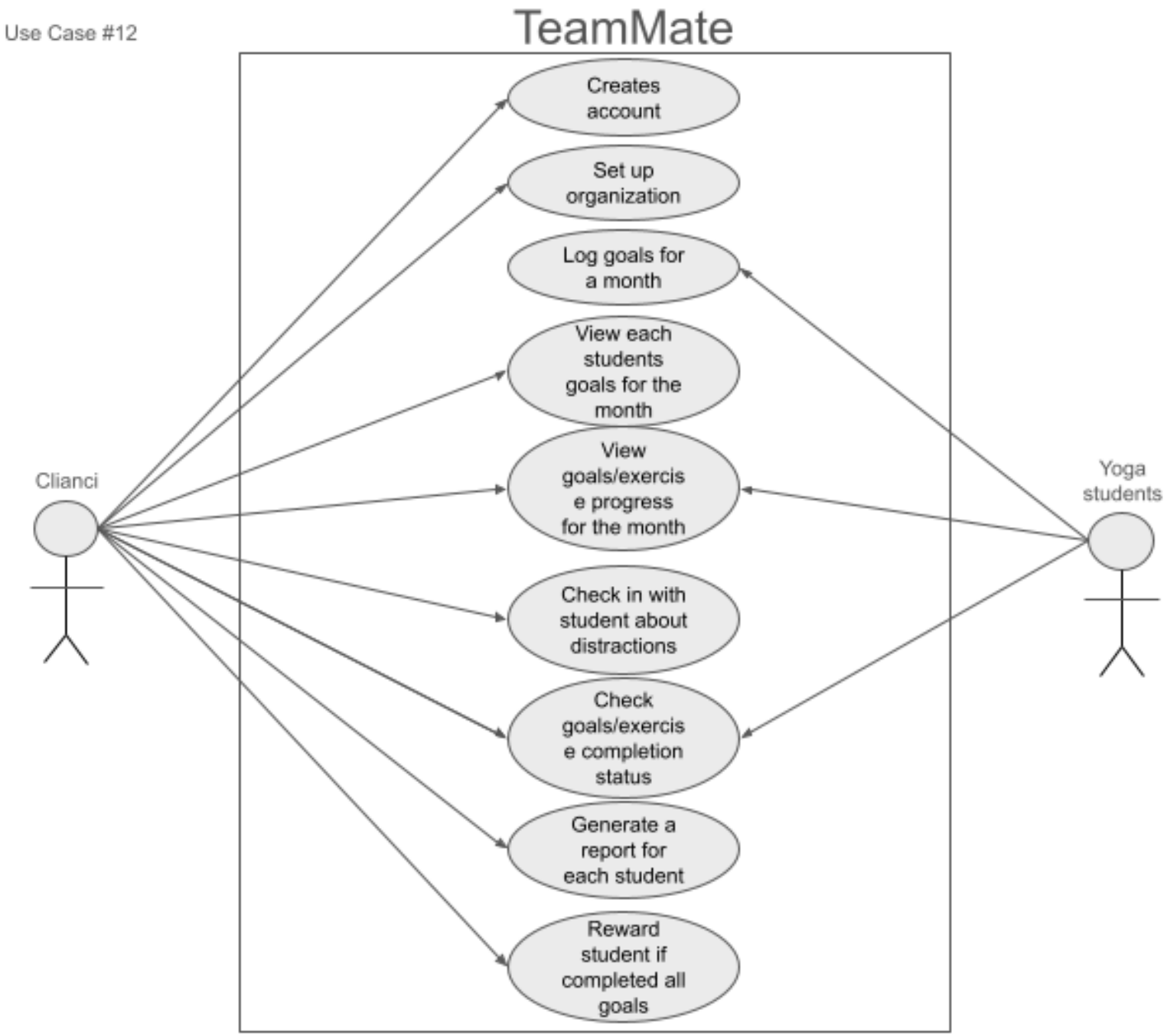
Clianci is the director of a Yoga class in Novato, she teaches Yoga classes every other day of the week, Monday, Wednesday, and Friday. Clianci wants each of her students to have new exercises/goals for her class 1 time each month. Each goal has to be achieved by the end of the month because new goals need to be created at the beginning of each month. Clianci has students that tend to have difficulty in constantly staying motivated to achieve those goals and keep working to reach where they want to reach. She decides to use the TeamMate app and create an account for her Yoga class students to be able keep track of each student's goals and achievements during each month, also to keep track of what methods her students are using to achieve those goals. She is also able to keep track of what obstacle is preventing a student from achieving that Yoga exercise/goal. For example an obstacle can be being tardy everyday, not attending class regularly, procrastination, distractions (doing other things, other then yoga in class). She will also keep track of the days a student specifically designated to work on these achievements (one by one)

Benefits:

If she uses the app to keep track of all the recent and future achievements each student of hers has, she can clearly see how much they have done during the month, it gives a sense of satisfaction to see how much her students are able to complete during a month or guides to implement changes in her class that would help the student be more engaged and complete their exercises/ achievements .

She will learn how to be more organized. All students' goals/exercises have to be completed one by one depending on what motivates them and how much time students designate each day to work towards that certain goal. As soon as they achieve the accomplishment of the goal she will reward each student with something (if the goal is reached before the month ends)

Diagram



Main Data Entities

- Account - Store user data.
    - Used to store data like username, email, password, etc to allowed users to register and login.
- Organization - Store organization data.
    - Used to store organization information, and also customization data, allowed people to rename.
- Member - Store organization member data.
    - Used to associate an account with an organization, holds organization specific information. Allowed to have one per-account per-organization.
- Events - Stores instances of events.
    - Used to store event information data, dates, times, and attendance.
- QualificationTypes - Stores types of qualifications and certifications.
    - Used to store different types of qualifications. Separated from an instance to allow easy mass-editing and issue.
- Qualifications - Stores instances of qualifications and certifications.
    - Used to store instances of qualifications, including date and time received along with any optional information about issuing party and notes.
- AwardTypes - Stores types of awards.
    - Used to store different types of awards. Separated from an instance to allow easy mass-editing and issue.
- Awards - Stores instances of issued awards.
    - Used to store instances of awards, including date and time received along with issuing party, citation, and any additional notes.
- Sections - Stores custom sections and pages for any purpose.
    - Used to store custom sections, which will be displayed on the dashboard, that invited members can access external and embedded apps on.
- Comments - Stores comments and their corresponding information.
    - Used to store comments and what page they are for, which will be displayed on their respective pages.

Functional Requirements

Account

- Users shall register for a new account.
- Accounts shall have unique emails.
- An account shall have a username.
- An account shall have a password.
- An account shall have secure passwords as per modern standards.
- Users shall log into an existing account.
- An account shall be able to upload a profile picture.
- An account shall be able to update profile information.
- An account shall be able to update the password.
- An account shall be able to update the associated email.
- An account shall create a new organization.
- Accounts shall join an existing organization.
- Accounts shall view all organizations they have joined.
- Accounts shall view all public organizations another account has joined.

Member

- Member permissions shall be members, group leaders, or admins.
- Members shall overwrite their account profile picture with a new picture.
- Members shall set a separate first and last name in each organization.
- Members shall generate event attendance reports.
- Admins shall be able to upload an image for the organization profile.
- Admins shall assign member and group leader permissions.
- Admins shall be able to issue awards to members.
- Admins reset member profile pictures if they are inappropriate.
- Admins reset member names if they are inappropriate.
- Users shall be able to leave comments and endorsements on a member’s page.
- Users shall be able to filter for members that have specific attributes.
- A member’s page shall show the last time they were logged-in.

- A member's page shall show their information for that organization, along with all attributes, qualifications, awards, etc as a dashboard.

Organization

- An organization shall have an owner.
- Organization ownership shall be able to be transferred to another member.
- An owner can assign other members to be admins.
- An organization shall have permissions that determine what members, group leaders, and admins are able to create and edit.
- Admins shall create custom sections for tailored purposes.
- Admins shall link external apps and embedded apps in custom pages.
- An organization shall have a roster page that shows all members in the organization separated by groups.
- Admins shall be able to change the order in which groups appear on the organization roster page.
- Users shall click on any member on the roster to access their member page.
- Organizations shall include an inactive or past members group that archives past members for posterity.

Awards

- Admins shall create award types.
- Admins shall update award types.
- Admins shall delete award types.
- Awards shall have an associated image.
- Awards shall have a title.
- Awards shall have a description.
- Group leaders shall nominate members for awards.
- Admins shall view nominations.
- Admins shall approve or reject nominations.
- Admins shall directly assign awards.
- Members shall view individual award citations.
- Award instances shall have a citation describing the reason for issue.
- Award instances shall have a date received.

- Award instances shall state the issuing member.
- Users shall be able to leave comments on an award instance.
- Users shall have the option to upload an associated pdf file for display with a qualification.

Qualifications

- Admins shall create qualifications types.
- Admins shall update qualifications types.
- Admins shall delete qualifications types.
- Qualifications shall have an associated image.
- Qualifications shall have a title.
- Qualifications shall have a description.
- Group leaders shall nominate members for qualifications.
- Admins shall view nominations.
- Admins shall approve or reject nominations.
- Admins shall directly assign qualifications.
- Members shall view individual qualifications.
- Qualification instances shall have a date received.
- Qualification instances shall state the issuing member.
- Users shall have the option to upload an associated pdf file for display with a qualification.

Events

- Admins shall create events.
- Group leaders shall create events.
- Events shall be public, by request, or invite only.
- Admins shall assign members to an existing event.
- Group leaders shall assign members to an existing event.
- Members shall join an event if it is public.
- Members shall request to join an event if it is by request.
- Events shall have a title.
- Events shall have a description.
- Events shall have a start datetime.

- Events shall be recurring if desired, automatically creating new events.
- Events shall be able to have pre-requisite qualifications.
- Events shall be able to be restricted to chosen groups or roles.
- Users shall have the option to opt-in for event reminders.
- Users shall be able to see an attendance record for recurring events.
- Users shall be able to leave comments on an event instance.

Custom Section

- Admins shall create custom sections.
- Custom sections shall have an editable title.
- Admins shall restrict custom sections based on member attributes.
- Custom sections shall support external application, APIs, and embedded application integrations.
- Custom sections shall be deleted by admins if desired.

Non-Functional Requirements

Performance

- The application shall respond quickly to user prompts with minimal latency.
- The application shall be able to host/handle up to 1,000 accounts without any performance problems.
- The application shall support many active-concurrent users without significant degradation in performance.

Security

- The application shall only store hashed passwords.
- The application database shall be backed up regularly.

Coding Standard

- The application shall follow consistent coding and naming conventions throughout.
- The application shall gracefully handle errors without detrimentally affecting the rest of the user experience.
- The application shall be well documented for easy understandability for other developers and admins alike.

Media Storage

- The application shall restrict profile pictures to 500x500 pixels and 5 megabytes or less.

Privacy

- The application shall have the option for organizations to be set as private, preventing users not in the organization from viewing it.
- Data shall be anonymized and pseudonymized wherever possible to minimize damage in the event of a leak.
- Only the minimum amount of data required shall be stored in order to minimize the risk of security breaches.

Scalability

- The application shall be designed to scale efficiently to handle an increasing number of users and data without performance degradation.
- The database shall scale down if the workload is low to remain cost effective and optimize resource utilization.

Reliability

- The application shall ensure high availability, with a 99.9% uptime, and incorporate robust error-handling mechanism to maintain stability.

Usability

- The application shall provide an user friendly interface, ensuring that users can easily navigate and use all features without extensive training.

Compliance

- The application shall comply with relevant industry standards and regulations, such as for data protection and privacy.

Expected Load

- The application shall respond to user queries within 500 milliseconds for requests.
- The application shall handle a minimum of 1000 transactions per second during peak hours.
- The application shall horizontally scale to support up to 10,000 concurrent users without degradation in response time.

Fault Tolerance

- The application shall recover from a server failure within 15 minutes without loss of data or interruption of service.
- Critical components of the application shall have a backup system that can take over within 5 minutes of a primary system failure.
- The application shall log all errors and provide automatic recovery for non-critical errors within 60 seconds.

Database High Level Specs

- The database shall use appropriate indexes to optimize query performance, especially for frequently accessed data.
- The database should be capable of storing up to 1 terabyte of data efficiently, with automatic data compression for storage optimization.
- All sensitive data stored in the database should be encrypted.

Browser and OS Support

- The application shall support Chrome Stable 126 (126.0.6478.153).
- The application shall support Windows 10 22H2 (19045.4651).
- The application shall support Windows 11 23H2 (22631.3737).
- The application shall support macOS Sonoma (14.5).
- The application shall support Amazon Linux (2023.4.20240611.0).

Competitive Analysis

Feature/ Company	Discord	Airtable	Google Sheets	PERSCOM	Rippling HCM
Strengths	- Learning curve is not steep. User interface is simple and clear. - Provides customizability like creating bots - Provides easy communication among the group, faster response time than emails - Provides easy management because of the features like assigning moderators or admins - Easy and fast sign up - Can create different channels for different purposes.	-A hybrid that combines spreadsheets and databases. -Able to import many different file types and other platform links. -UX and UI are ease of use, flexibility, design and visual, view system. -A wide range of built-in ready-to-use templates. -Real time Collaboration and audit trail. -Automate tasks -Use cases	- Real-time collaboration with other users - Accessible from any devices with internet -Automatically saves and stores progress to cloud -Can integrate with other Google apps such as Google Docs, Drive, etc.	- Personal profile to be able to manage various records and qualifications. - It includes tracking, calendar and event management, custom forms and data collection, various suites of communication tools, as well as the ability to integrate your personnel data into any third-party system. - Includes a roster to track information from different departments.	-Comprehensive HCM solutions -Integrates HR, payroll, and IT management -Automation of administrative tasks -Strong integration capabilities -Detailed reporting and analytics -Scalable for companies of all sizes -Strong data security
Weaknesses	- Don't have much built in features, like payroll or leaderboard - Hard to track chat history or files. - Discord mostly need to be used with other	-There is a learning curve for some advanced features -Some platform features require fees. -Time consuming to set up a database system	- Complex for most users to use advances features to automate tasks. They must be familiar with formula or have experience with scripting - Does not support a more	- The organization of each category is displayed in tabs, Not quick to find information - Complex interface, difficult to understand - The website doesn't suit	-Can be complex to set up initially -Pricing might not be suitable for smaller size teams or for individuals - Feature set focuses entirely on professional

	softwares like google docs or google calendars to make it more efficient	when you start from scratch.	formal role or team management. You can only really give others view or edit permissions.	everyone or other purposes - Does not keep a reference of the data imputed overtime	use, not friendly for volunteer, non-profit, non-professional organizations.
Pricing	Mostly Free, only some not so essential features are charged. Features such as large file sharing, HD video calls, etc.	Mostly Free, fees required for larger applications or databases or more collaborators.	- Free for basic use, but subscription based if you want Google Workspace for additional features like video conferencing with a lot of people.	- Has three different tiers with a free one week trial. Pricing increases as the user base increases. Access to powerful API, widgets and website integration, custom subdomain, and ticket and email support are locked behind higher tiers.	- Personalized and tiered pricing based on features and company size.
Social Media	- Very active on Twitter, YouTube, and other famous platforms. Lots of additional user generated content and awareness.	-They are on a number of famous sites, like Linkedin, Facebook, Instagram and so on. And they post or update their blogs regularly.	-Active on Twitter, Facebook, LinkedIn, and Youtube	Community Forums, Submitting a support ticket, giving feedback, submitting feature requests	- Active on social media, Linkedin, facebook, and Twitter - Community forums and help centers
Onboarding	- Simple and fast. As easy as creating an account and inviting people to the group	- Simple and easy. A user can sign up with username and password. And the user can sign in with google or other platform accounts.	- Simple since most people already have Google accounts. There are also a bunch of tutorials and support documentation online.	- Self-guided onboarding. Documentation included, but little support from providers.	- Comprehensive and guided onboarding with direct support and tutorials.

Feature	Discord	Airtable	Google Sheets	PERSCOM	Rippling HCM	TeamMate
Real-time Collaboration	+	+	++	+	+	+
File Sharing	++	-	-	+	+	+
Text Chats	++	-	-	-	+	-
Voice/Video Calls	++	-	-	-	-	-
User Interface	+	+	+	+	+	++
Mobile Support	++	+	+	-	+	+
Customization	+	+	++	+	-	+
Role/Group Management	+	+	+	+	+	++
Reports/Analytics	-	-	-	+	+	++
Data Recovery	-	+	+	-	+	+
Event Management	+	-	-	+	+	++
Payroll and Health Insurance	-	-	-	-	++	-
Training Scheduling and Qualification Tracking	-	-	-	+	+	++
Filtering Users based on Attributes	-	+	+	-	+	++
Attendance Tracking	-	+	+	+	+	++
External Application Integration	+	-	-	++	+	+

Summary

Each of the competitors mentioned above serves and focuses on a different niche than we do. Consequently, there are some features that TeamMate either lacks entirely or offers at a level that may seem inferior compared to other competitors. This is not a disadvantage for TeamMate; in fact, it is a strength. TeamMate is designed for a specific, yet sizable, audience that currently lacks purpose-built solutions tailored to their needs. TeamMate is designed to be used alongside other applications, including some of the listed competitors like Discord, to enhance an organization's capabilities. It remains streamlined and user-friendly, avoiding unnecessary bulk that might deter potential users from adopting TeamMate as their solution.

Checklist

DONE - Team found a time slot to meet outside of the class.

DONE - Github master chosen.

DONE - Team decided and agreed together on using the listed SW tools and deployment server.

DONE - Team ready and able to use the chosen back and front end frameworks and those who need to learn are working on learning and practicing.

DONE - Team lead ensured that all team members read the final M1 and agree/ understand it before submission.

DONE - Github organized as discussed in class (e.g. master branch, development branch, folder for milestone documents etc.).

Technology Stack

Server: AWS EC2.
Operation System: Amazon Linux AMI.
Database: MySQL 8.0.37 on an AWS RDS instance.
Web Server: Express 4.19.2.
Server-Side Language: TypeScript.

Additional Tech:
Frontend Framework: React.
Backend Framework: Node.js 22.3.0.
IDE: Visual Studio Code, MySQL Workbench.
SSL Cert: Lets Encrypt (Cert Bot).
Docker: Docker Container on the EC2 instance.

Team Contributions

- **Krishna**
  - Created Title Page and transcribed Team Discussion into Executive Summary.
  - Created two Main Use Cases.
  - Created Main Data Items and Entities.
  - Created initial Functional and Non-Functional Requirements.
  - Transcribed Competitive Analysis Summary from Team Discussion.
  - Created index and about page, setup repo as per tech stack specifications.
  - Created and set up AWS EC2 and RDS instances.
  - Installed Tech Stack on Cloud Instance and cloned Repository.
  - Writing credentials and cloud connection documentation and instructions.
-
  - Created two Main Use Cases and corresponding Diagrams.
  - Created additional Main Use Case Diagrams.
  - Researched Google Sheets as a Competitor.
  - Filled out overview table in Competitive Summary.
  - Filled out features table in Competitive Summary.
  - Created about page.
  - Appended index page.
-
  - Created two Main Use Cases and corresponding Diagrams.
  - Researched Discord as a Competitor.
  - Filled out overview table in Competitive Summary.
  - Filled out features table in Competitive Summary.
  - Created about page.
  - Appended index page.
-
  - Created two Main Use Cases and corresponding Diagrams.
  - Added additional Functional Requirements.
  - Added additional Non-Functional Requirements.
  - Researched Airtable as a Competitor.
  - Filled out overview table in Competitive Summary.
  - Filled out features table in Competitive Summary.
  - Created about page.
  - Appended index page.

-
    - Created two Main Use Cases.
    - Researched Rippling HCM as a Competitor.
    - Filled out overview table in Competitive Summary.
    - Filled out features table in Competitive Summary.
    - Created about page.
    - Appended index page.
-
    - Created two Main Use Cases.
    - Researched PERSCOM as a Competitor.
    - Filled out overview table in Competitive Summary.
    - Filled out features table in Competitive Summary.
    - Created about page.
    - Appended index page.

SW Engineering CSC648-848-05 Summer 2024

TeamMate
Personnel Management
Team 05 - Concat

Name	E-Mail	Role
Krishna		Team Lead
		Frontend Lead
		Frontend Dev
		Backend Lead
		Backend Dev
		Database Lead
		Database Dev
		Github Master

Milestone 2
July 23 2024

Version History

Revision	Date Submitted
M1V1	20 June 2024
M1V2	09 July 2024
M2V1	09 July 2024
M2V2	23 July 2024

Table of Contents

Data Definitions

- Registered Users (Anyone with a registered account)
    - Email
        - Correct format with @ and .com
    - Password
        - Modern standards with letters, numbers, special characters.
        - Stored in the database after being hashed.
    - Profile Picture
        - PNG or JPG/JPEG.
        - 400x400 as the max dimensions.
        - No greater than 2 megabytes.
    - Username
    - First Name
    - Last Name

- Organizations (Organizations created by a registered user)
    - Name
        - Plaintext title.
        - Must be unique between all organizations.
    - Owner
        - Connected to a registered user.
        - Transferable to another registered user.
    - Description

- Members (Association between a user and an organization)
    - Permissions
        - Either 0, 1, or 2 for user, group leader, and admin respectively.
    - Organization
        - Connected to an organization.
    - Group
        - Connected to a group.
    - User
        - Connected to a registered user.
    - First Name
        - Different from the Registered User’s name.
    - Last Name
        - Different from the Registered User’s name.
    - Sort Order

        - Integer that controls in which order members should appear on the frontend.
    - Last Login
        - Datetime that represents the last time this member was logged-in..
    - Active Member
        - Boolean to archive members that are no longer active for historical purposes.
- Groups (Member created category for dividing people)
    - Organization
        - Connected to an organization.
    - Sort Order
        - Integer that controls in which order groups should appear on the frontend.
    - Group Image
        - PNG or JPG/JPEG.
        - 400x400 as the max dimensions.
        - No greater than 2 megabytes.
    - Group Name

- Events (Member created event that other members can “join”)
    - Organization
        - Connected to an organization.
    - Title
    - Description
    - Date
    - Type
        - Simple string for users’ organizational purposes.
    - Prerequisites
        - Serialized string that contains group and qualification prerequisites.

- Event Reminder (Associates a Member to an Event to track attendance)
    - Event
        - Connected to an event.
    - Member
        - Connected to a member.
    - Attendance
        - Enumerated type that records if the member was present or absent.

- Qualification Type (Member-defined custom qualifications)
    - Title
    - Description
    - Image
        - PNG or JPG/JPEG.
        - 400x400 as the max dimensions.
        - No greater than 2 megabytes.

- Qualification Instance (An instance of a qualification assigned to a member)
    - Issued By
        - Connect to a member.
    - Received By
        - Connect to a member.
    - Qualification Type
        - Connect to a qualification type.
    - Nomination
        - Boolean that determines whether this instance has been issued or is just a nomination.
    - Date Given
        - Datetime.
    - Citation
        - Plaintext citation for issuance.

- Award Type (Member-defined custom awards)
    - Title
    - Description
    - Image
        - PNG or JPG/JPEG.
        - 400x400 as the max dimensions.
        - No greater than 2 megabytes.

- Award Instance (An instance of an award assigned to a member)
    - Issued By
        - Connect to a member.
    - Received By
        - Connect to a member.
    - Qualification Type
        - Connect to a qualification type.
    - Nomination
        - Boolean that determines whether this instance has been issued or is just a nomination.

    - Date Given
        - Datetime.
    - Citation
        - Plaintext citation for issuance.

- Comments (Members can leave comments on other member's pages)
    - UserID
        - Connect to a member.
    - Page Type
        - Determines whether the comment is on an award instance, qualification instance, event, or another member.
    - Page Key
        - Determines the index/ID of where the comment is.
    - Comment
        - Plaintext comment body.
    - Comment Date
        - Datetime.

- Custom Section (Customizable section usable for whatever the client needs)
    - Access Level
        - Determines whether the section is for all users, group leaders, or admins.
    - Plain Text
        - Plaintext body for the custom section.
    - Third Party
        - Open-ended field for code or other integrations.
    - Document Creator
        - Connected to a member.
    - Organization
        - Connected to an organization.
    - Title

Prioritized Functional Requirements

Priority 1

Account

- Users shall register for a new account.
- Accounts shall have unique emails.
- An account shall have a username.
- An account shall have a password.
- An account shall have secure passwords as per modern standards.
- Users shall log into an existing account.
- An account shall be able to update profile information.
- An account shall be able to update the password.
- An account shall be able to update the associated email.
- An account shall create a new organization.
- Accounts shall join an existing organization.
- Accounts shall view all organizations they have joined.
- Accounts shall view all public organizations another account has joined.

Member

- Member permissions shall be members, group leaders, or admins.
- Admins shall assign member and group leader permissions.
- Admins shall be able to issue awards to members.
- Admins reset member names if they are inappropriate.
- Admins reset member profile pictures if they are inappropriate.
- Users shall be able to filter for members that have specific attributes.
- A member's page shall show their information for that organization, along with all attributes, qualifications, awards, etc as a dashboard.

Organization

- An organization shall have an owner.
- An owner can assign other members to be admins.
- An organization shall have permissions that determine what members, group leaders, and admins are able to create and edit.

- An organization shall have a roster page that shows all members in the organization separated by groups.
- Admins shall be able to change the order in which groups appear on the organization roster page.
- Users shall click on any member on the roster to access their member page.

Awards

- Admins shall create award types.
- Admins shall update award types.
- Admins shall delete award types.
- Awards shall have a title.
- Awards shall have a description.
- Group leaders shall nominate members for awards.
- Admins shall view nominations.
- Admins shall approve or reject nominations.
- Admins shall directly assign awards.
- Members shall view individual award citations.
- Award instances shall have a citation describing the reason for issue.
- Award instances shall have a date received.
- Award instances shall state the issuing member.
- Users shall be able to leave comments on an award instance.

Qualifications

- Admins shall create qualifications types.
- Admins shall update qualifications types.
- Admins shall delete qualifications types.
- Qualifications shall have a title.
- Qualifications shall have a description.
- Group leaders shall nominate members for qualifications.
- Admins shall view nominations.
- Admins shall approve or reject nominations.
- Admins shall directly assign qualifications.
- Members shall view individual qualifications.

- Qualification instances shall have a date received.
- Qualification instances shall state the issuing member.

Events

- Admins shall create events.
- Group leaders shall create events.
- Events shall be public, by request, or invite only.
- Admins shall assign members to an existing event.
- Group leaders shall assign members to an existing event.
- Members shall join an event if it is public.
- Members shall request to join an event if it is by request.
- Events shall have a title.
- Events shall have a description.
- Events shall have a start datetime.

Priority 2

Account

- An account shall be able to upload a profile picture.

Member

- Members shall set a separate first and last name in each organization.
- Members shall generate event attendance reports.
- Admins shall be able to upload an image for the organization profile.
- A member's page shall show the last time they were logged-in.

Organization

- Organization ownership shall be able to be transferred to another member.
- Admins shall create custom sections for tailored purposes.
- Organizations shall include an inactive or past members group that archives past members for posterity.

Awards

- Awards shall have an associated image.
- Users shall have the option to upload an associated pdf file for display with an award.

Qualifications

- Qualifications shall have an associated image.
- Users shall have the option to upload an associated pdf file for display with a qualification.

Events

- Events shall be recurring if desired, automatically creating new events.
- Events shall be able to have pre-requisite qualifications.
- Events shall be able to be restricted to chosen groups or roles.
- Users shall be able to see an attendance record for recurring events.

Custom Section

- Admins shall create custom sections.
- Custom sections shall have an editable title.
- Admins shall restrict custom sections based on member attributes.
- Custom sections shall be deleted by admins if desired.

Priority 3

Member

- Users shall be able to leave comments and endorsements on a member's page.

Organization

- Admins shall link external apps and embedded apps in custom pages.

Events

- Users shall have the option to opt-in for event reminders.
- Users shall be able to leave comments on an event instance.

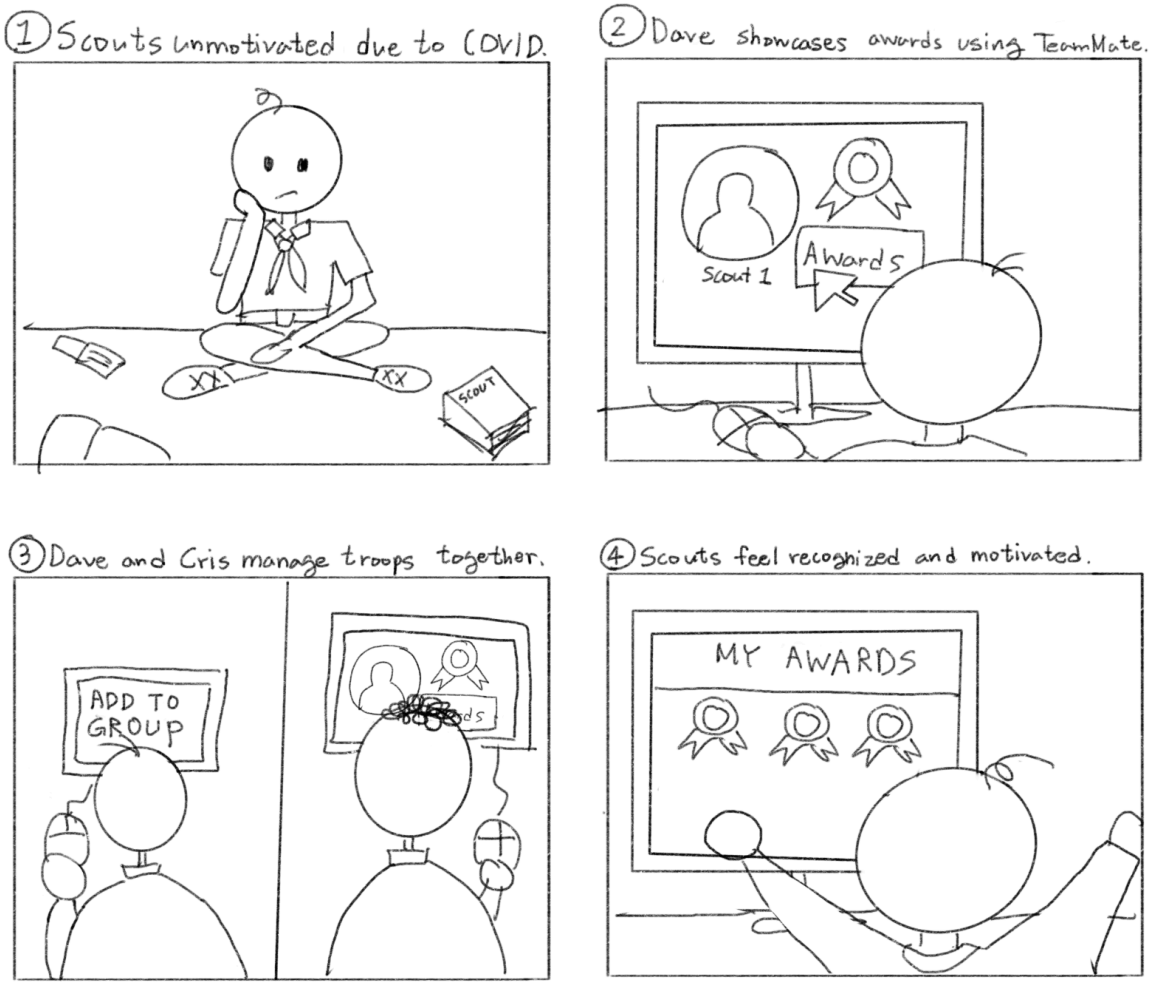
Custom Section

- Custom sections shall support external application, APIs, and embedded application integrations.

UI Mockups and Storyboards

Use Case #1

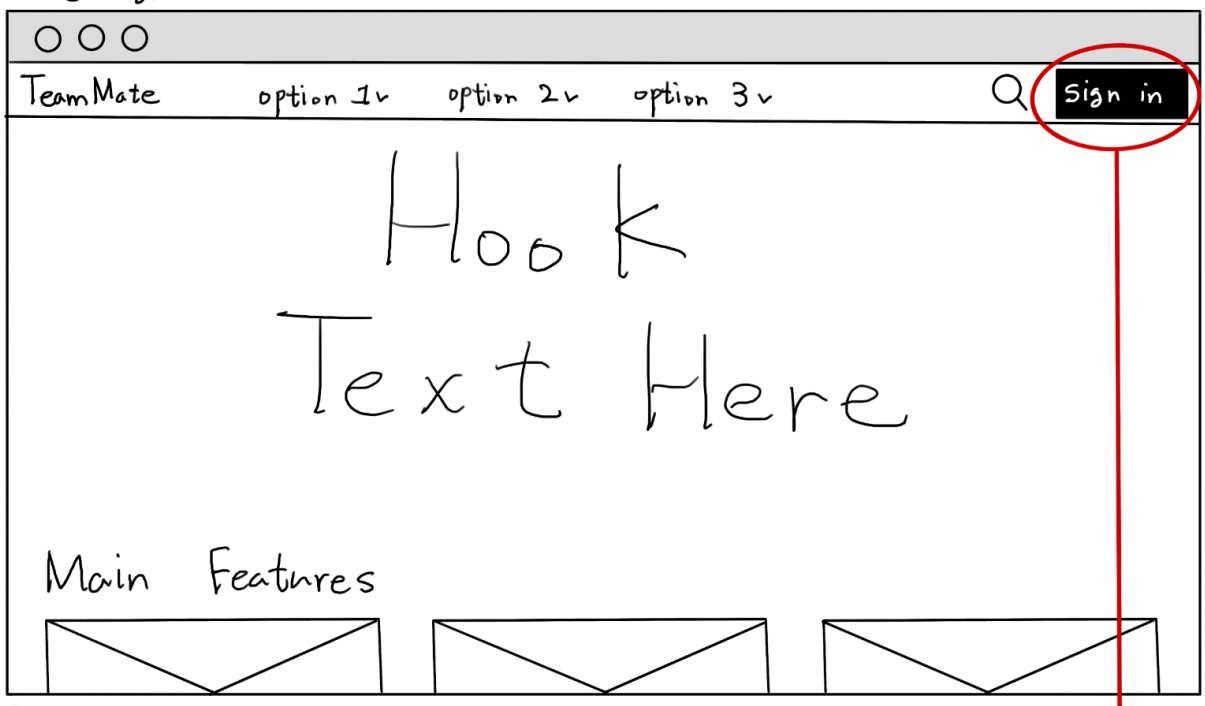
Dave, a leader for the Boy Scouts of America in Silicon Valley, faced challenges during COVID as scouts were unable to participate in outdoor activities. To maintain motivation, he created an organization on TeamMate to display scout progress and achievements online. With the help of his friend Chris, they managed the scouts' accounts, groups, and awards, making the process more efficient and ensuring scouts could still take pride in their accomplishments.



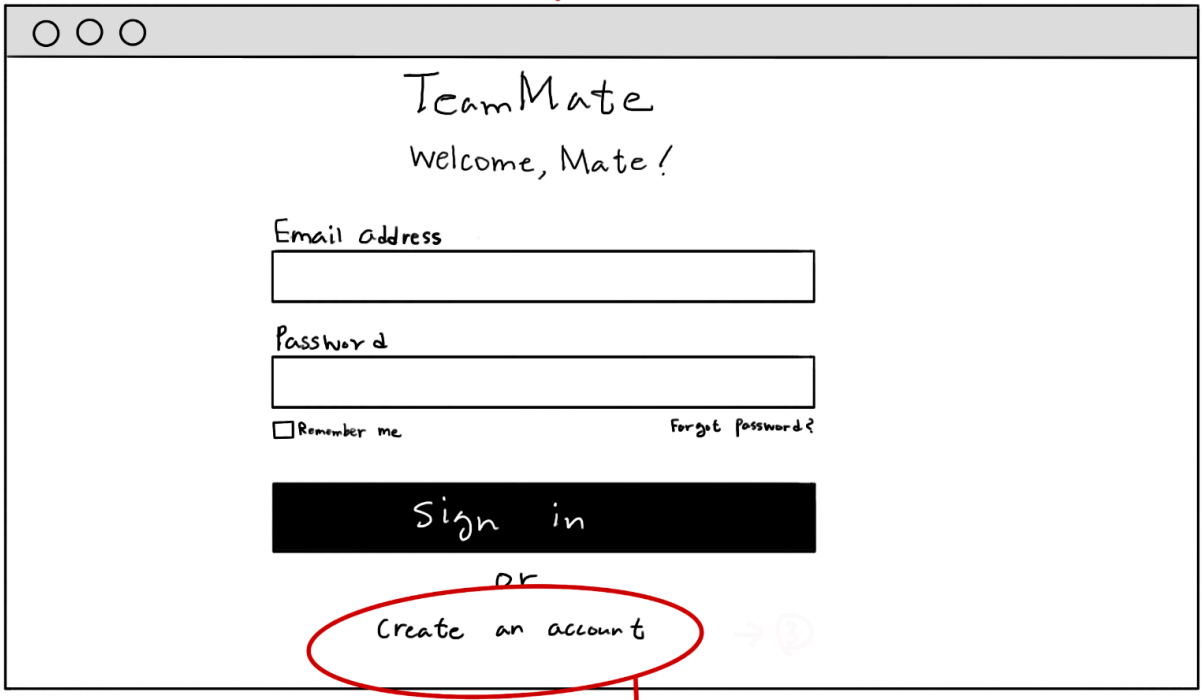
① Start - Leader's perspective

Use case # 1

Home Page

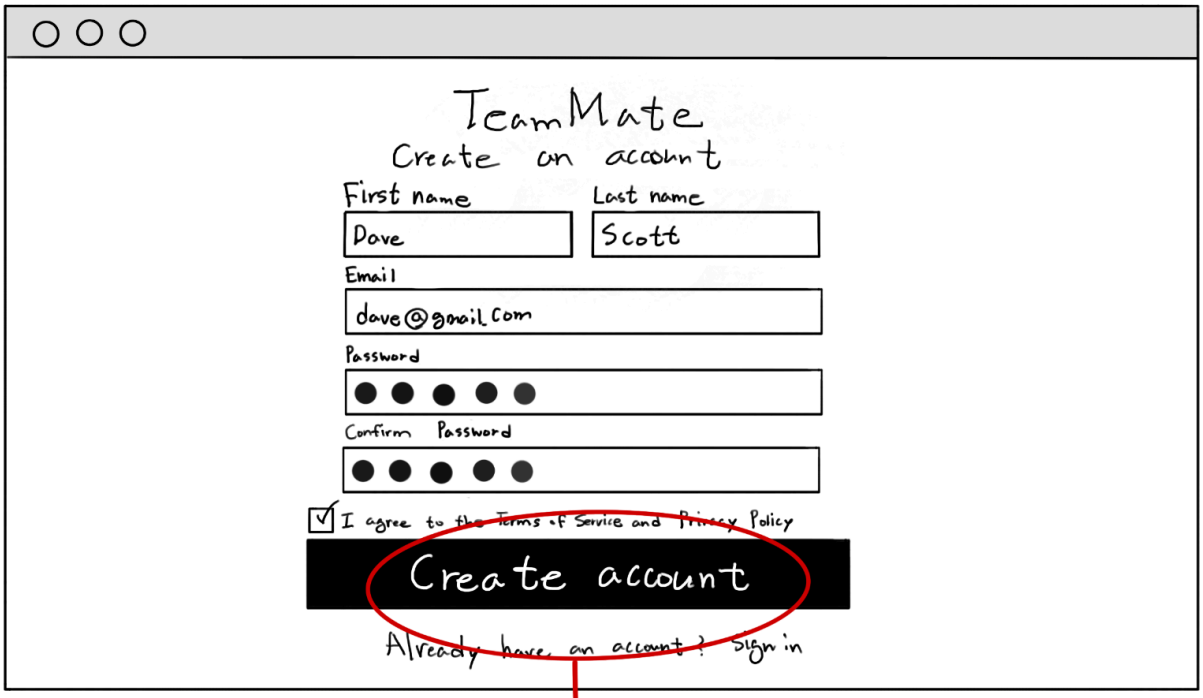


② Sign In Page



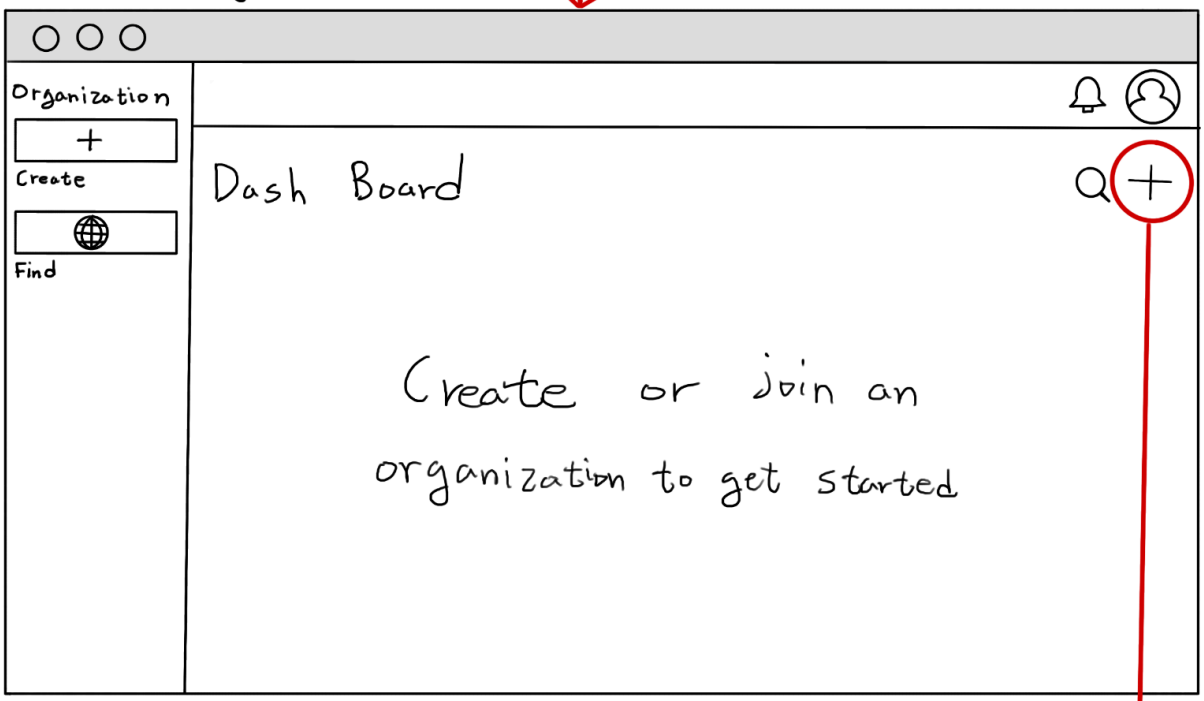
3

Create Account Page

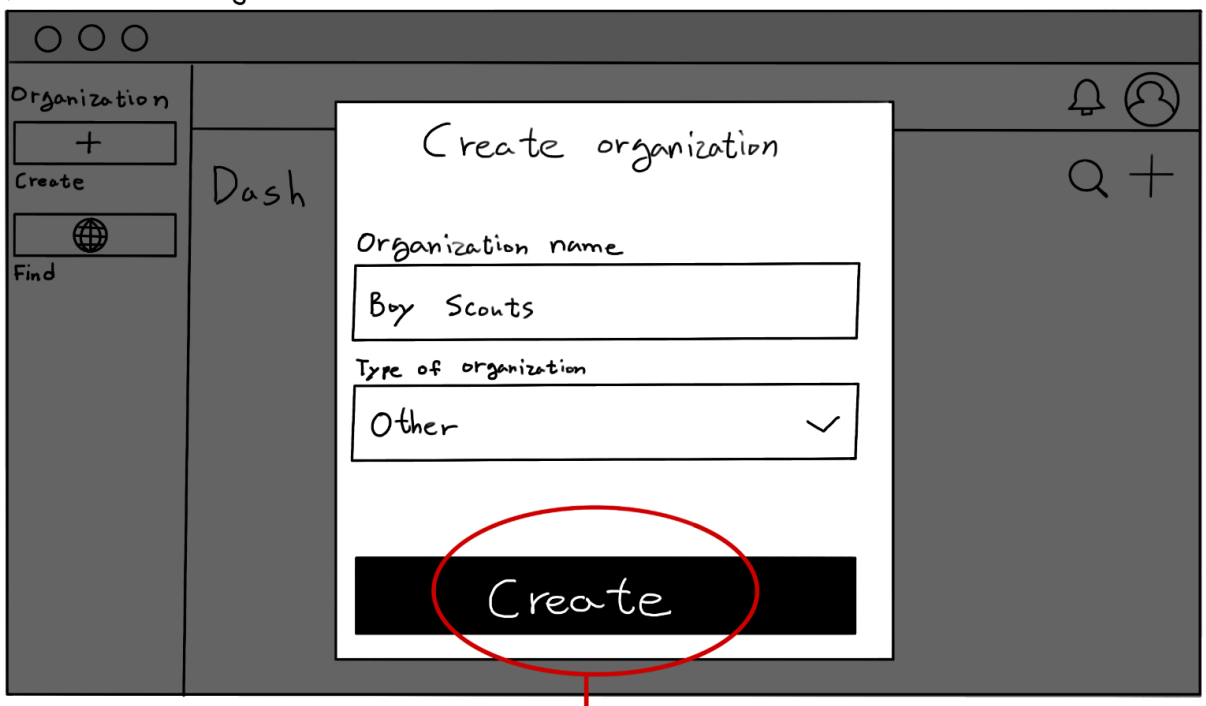


4

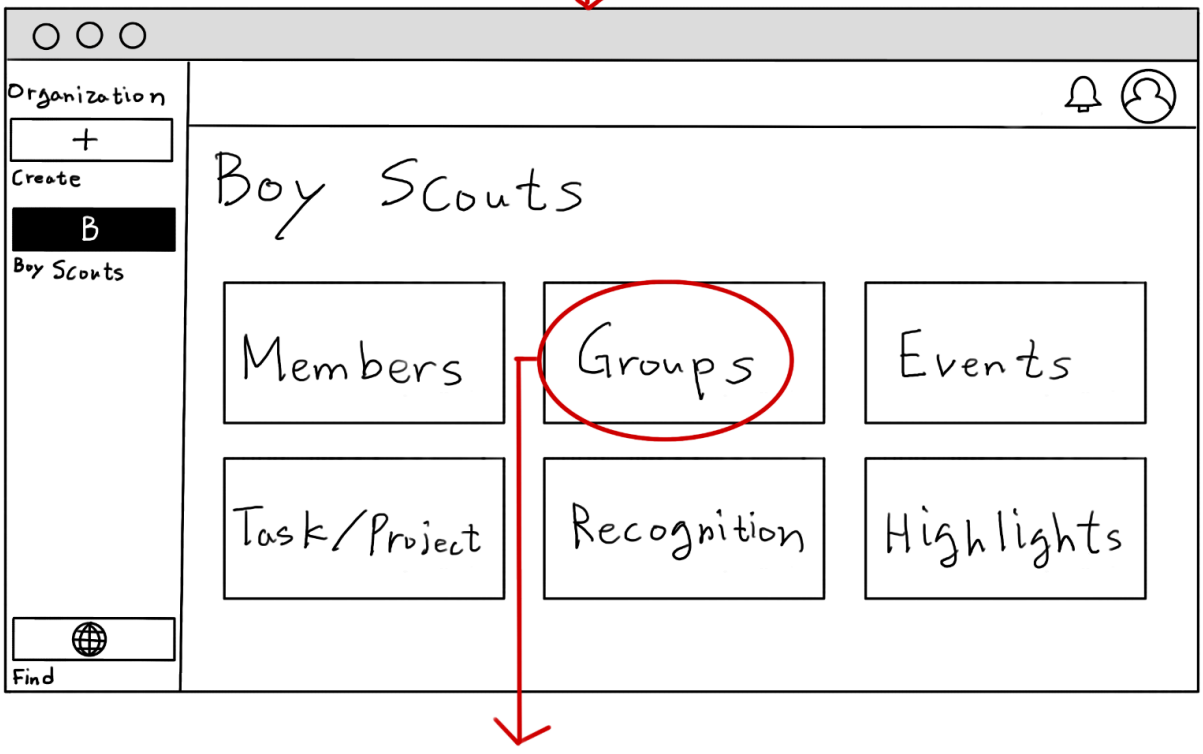
Dash Board Page



⑤
Dash Board Page

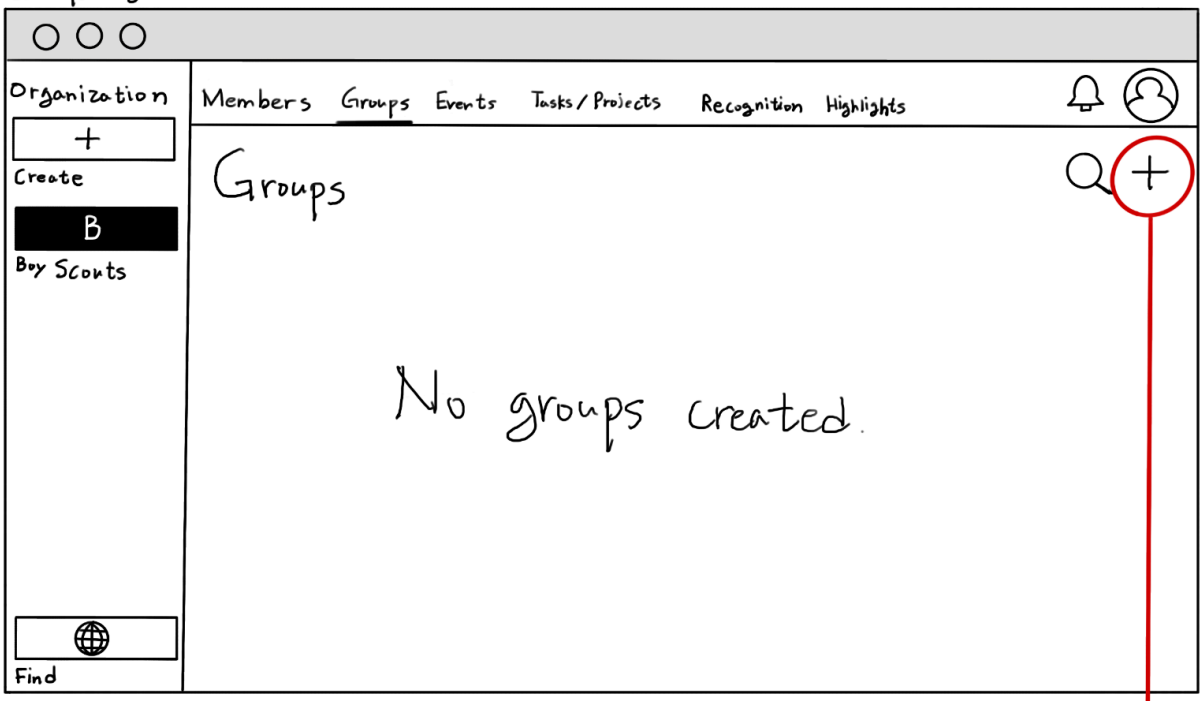


⑥
Organization Dashboard Page



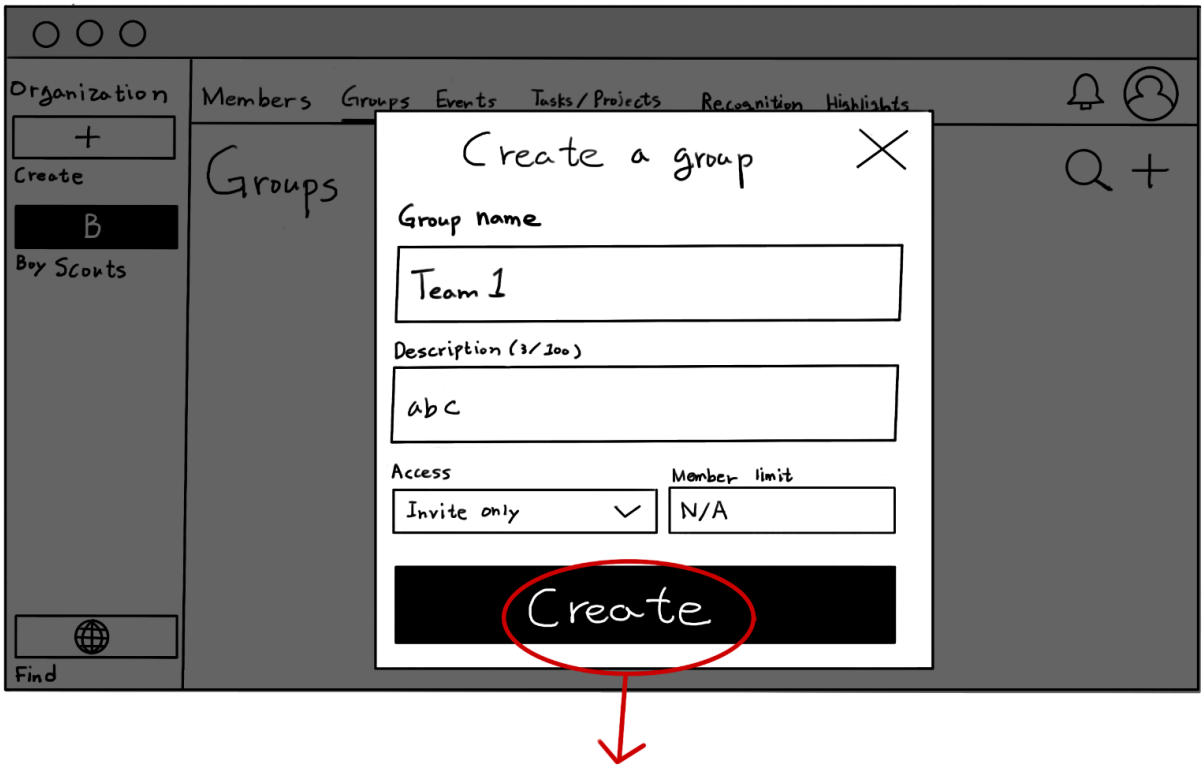
7

Groups Page



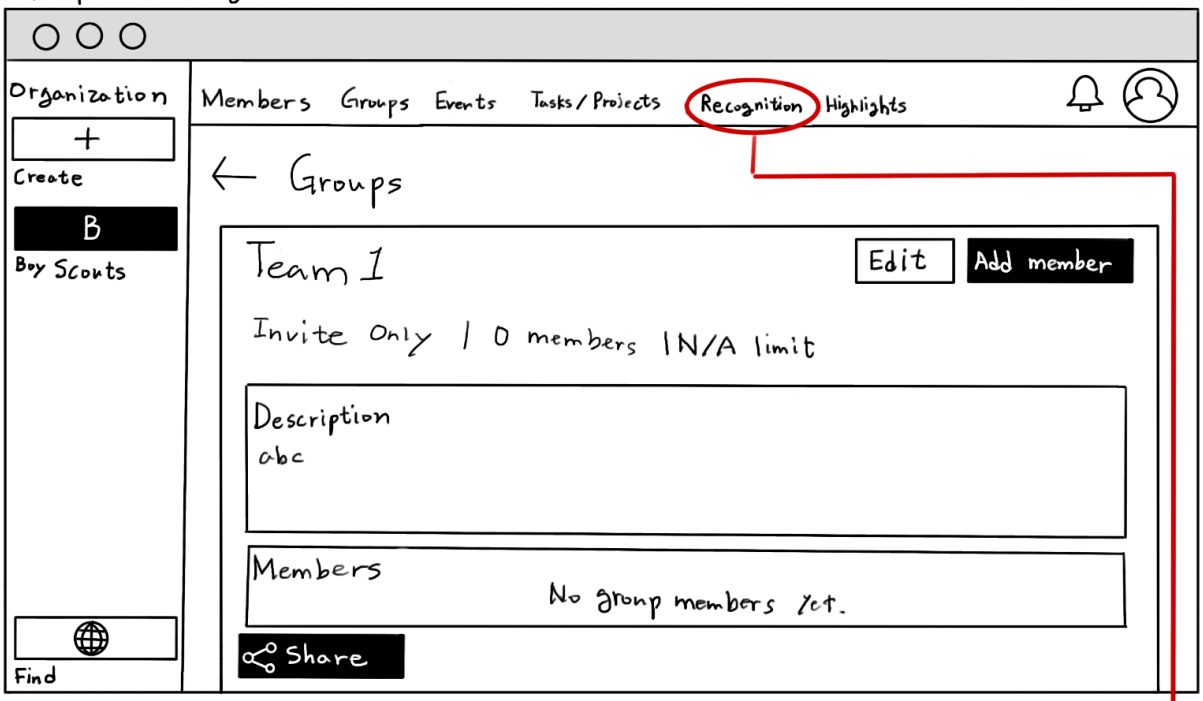
8

Group Create Page



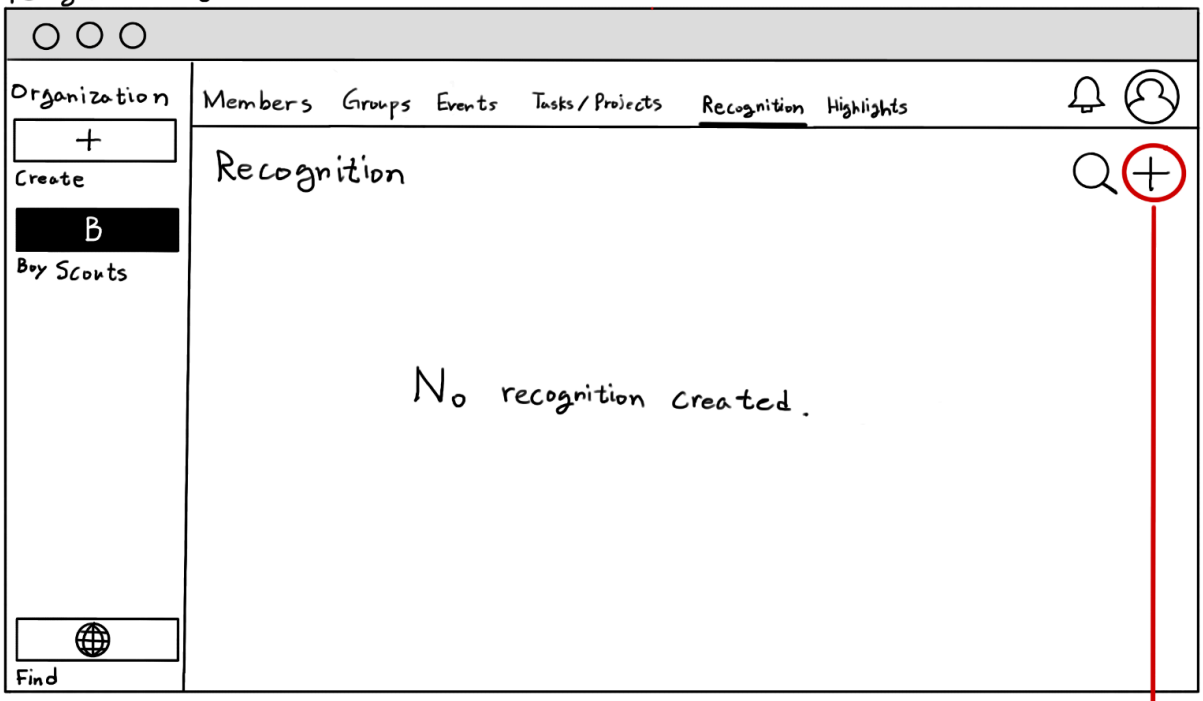
9

Group Detail Page



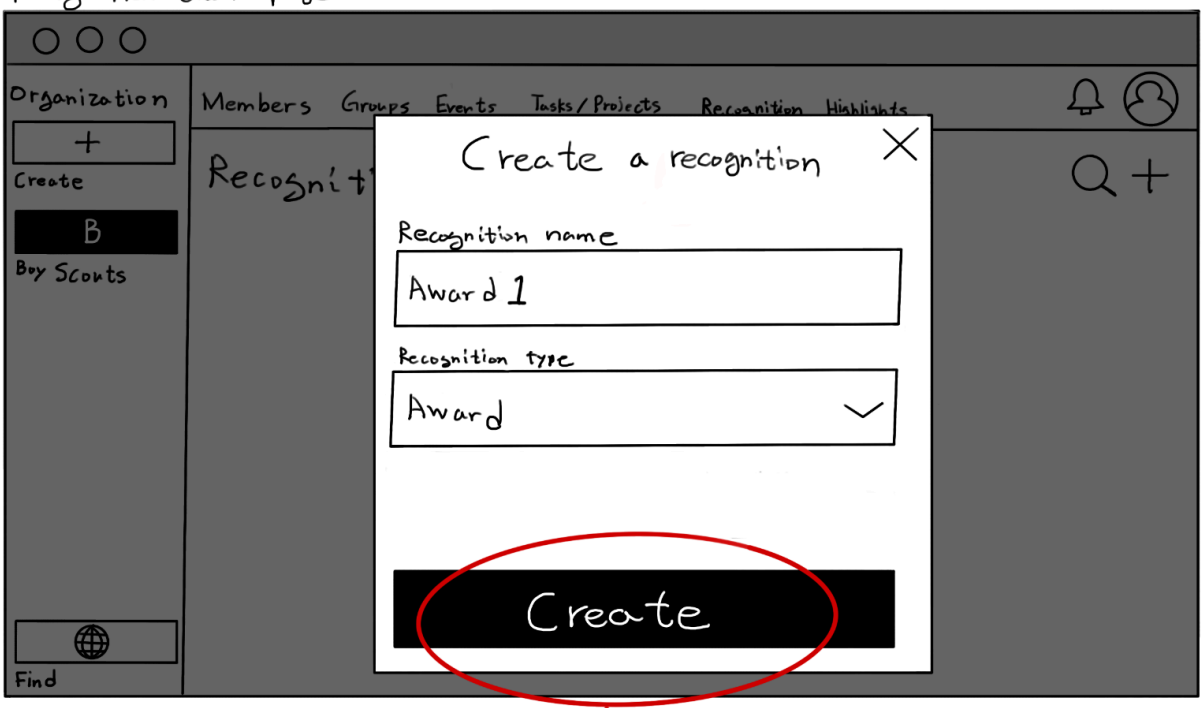
10

Recognition Page



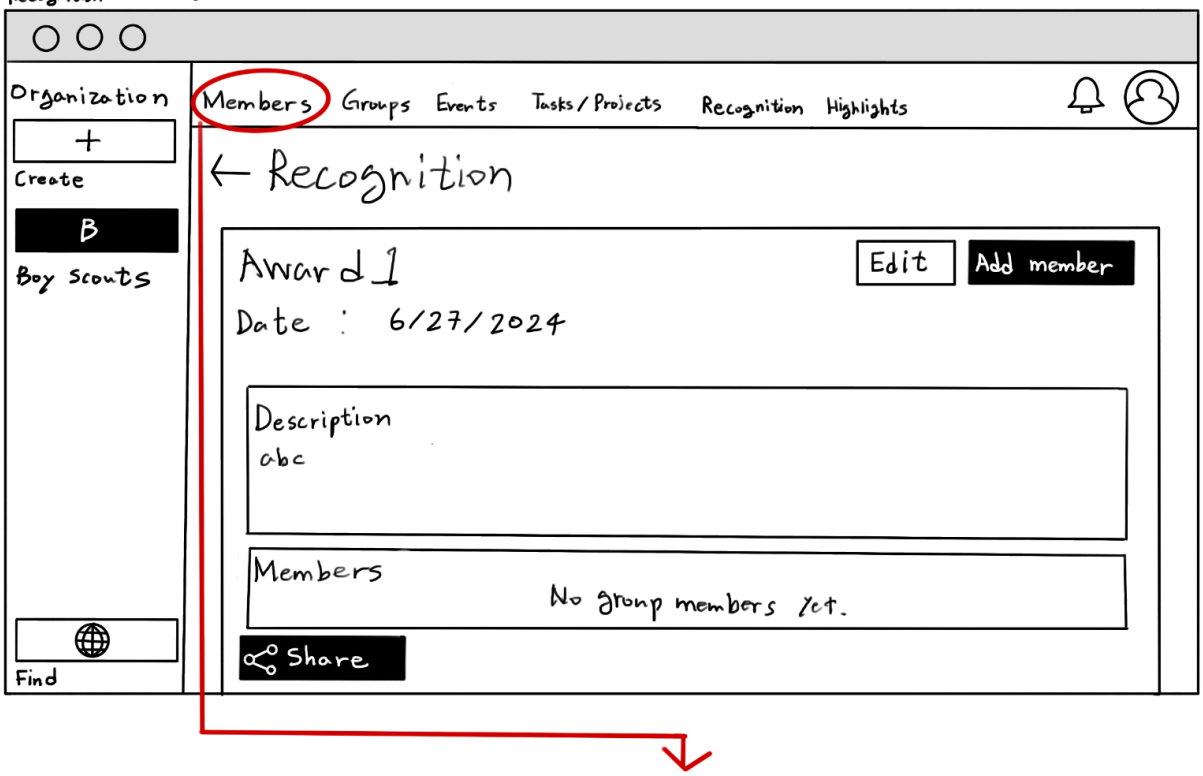
11

Recognition Create Page



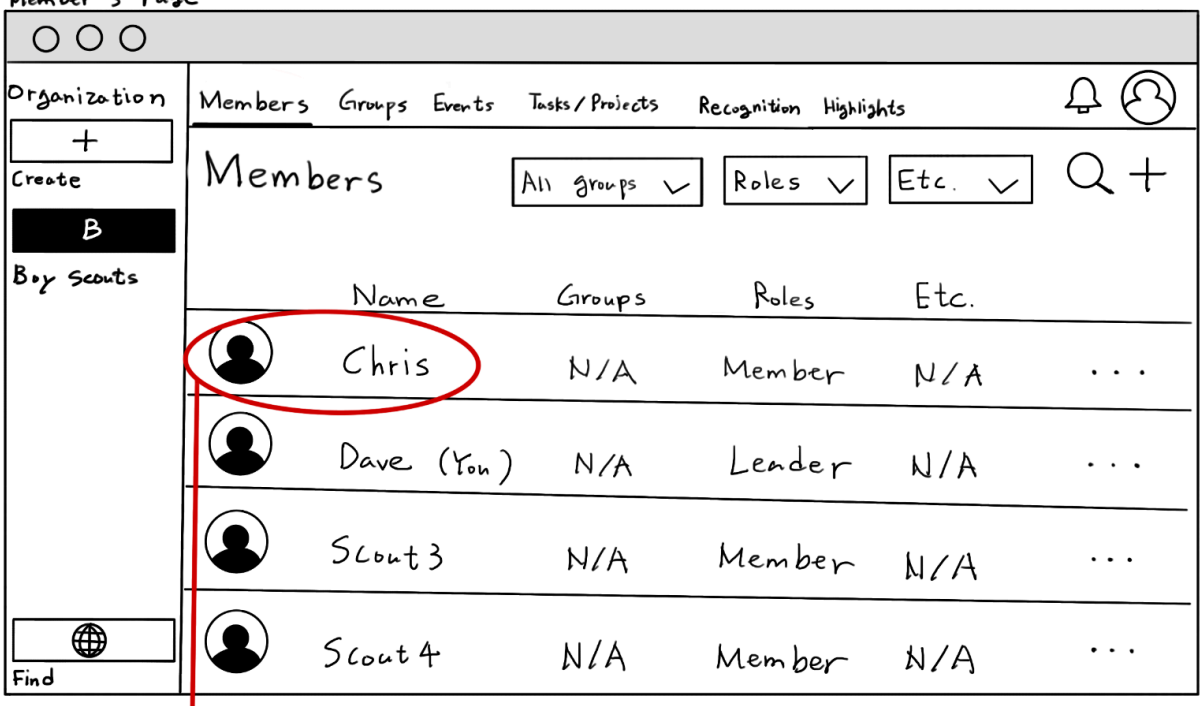
12

Recognition Detail Page



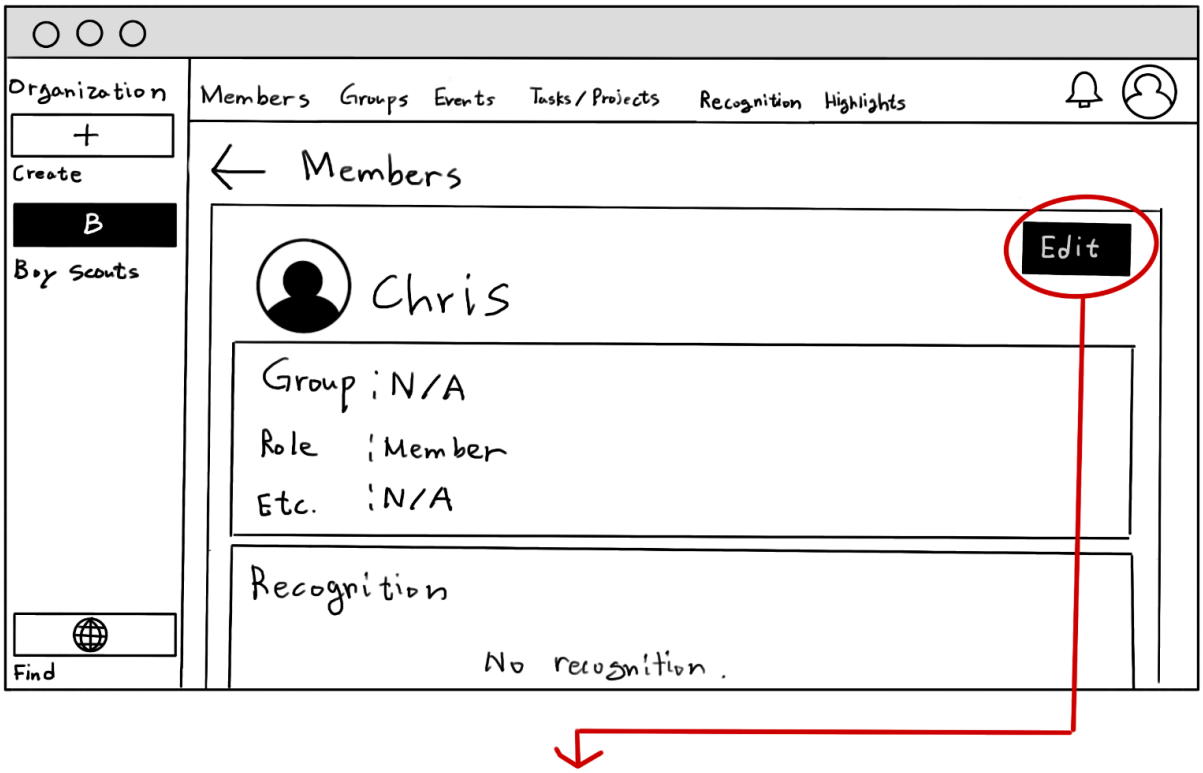
13

Member's Page



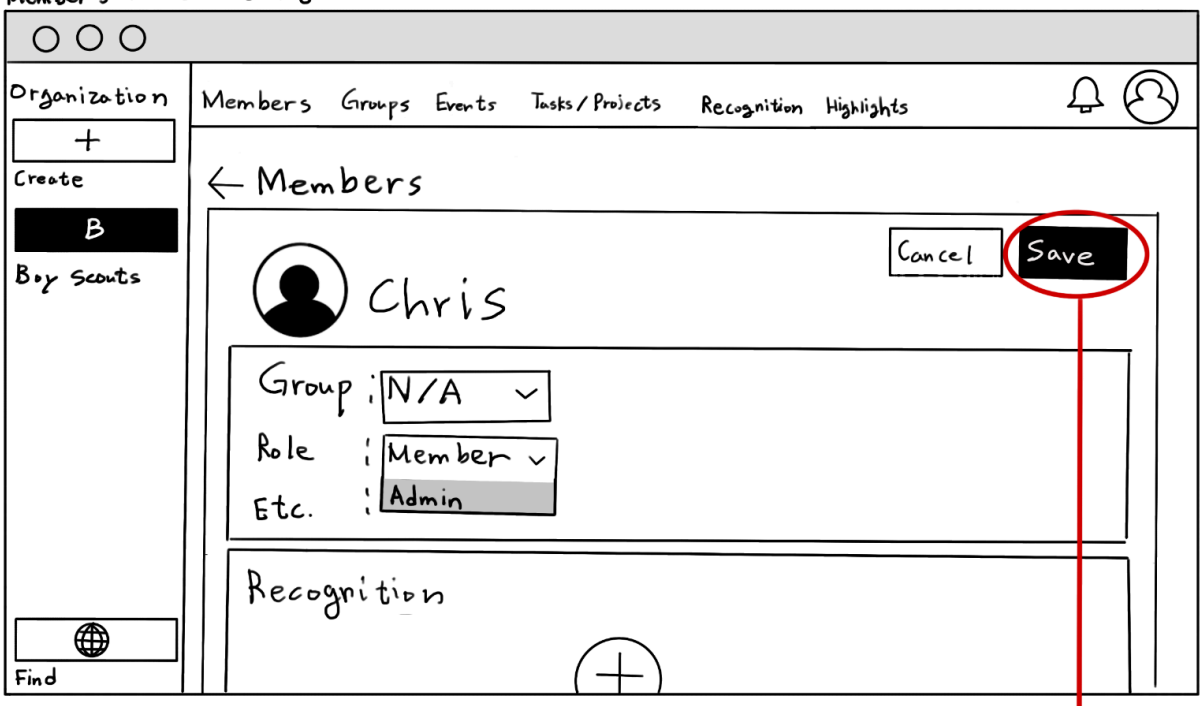
14

Member's Profile Page



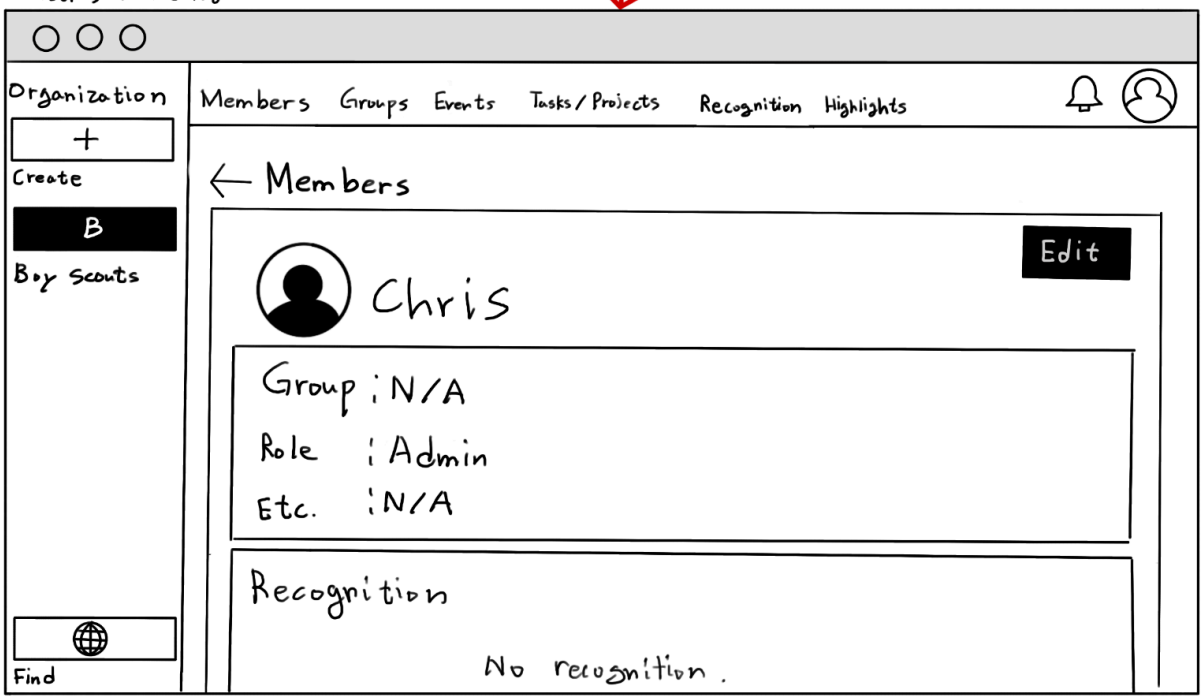
15

Member's Profile Edit Page



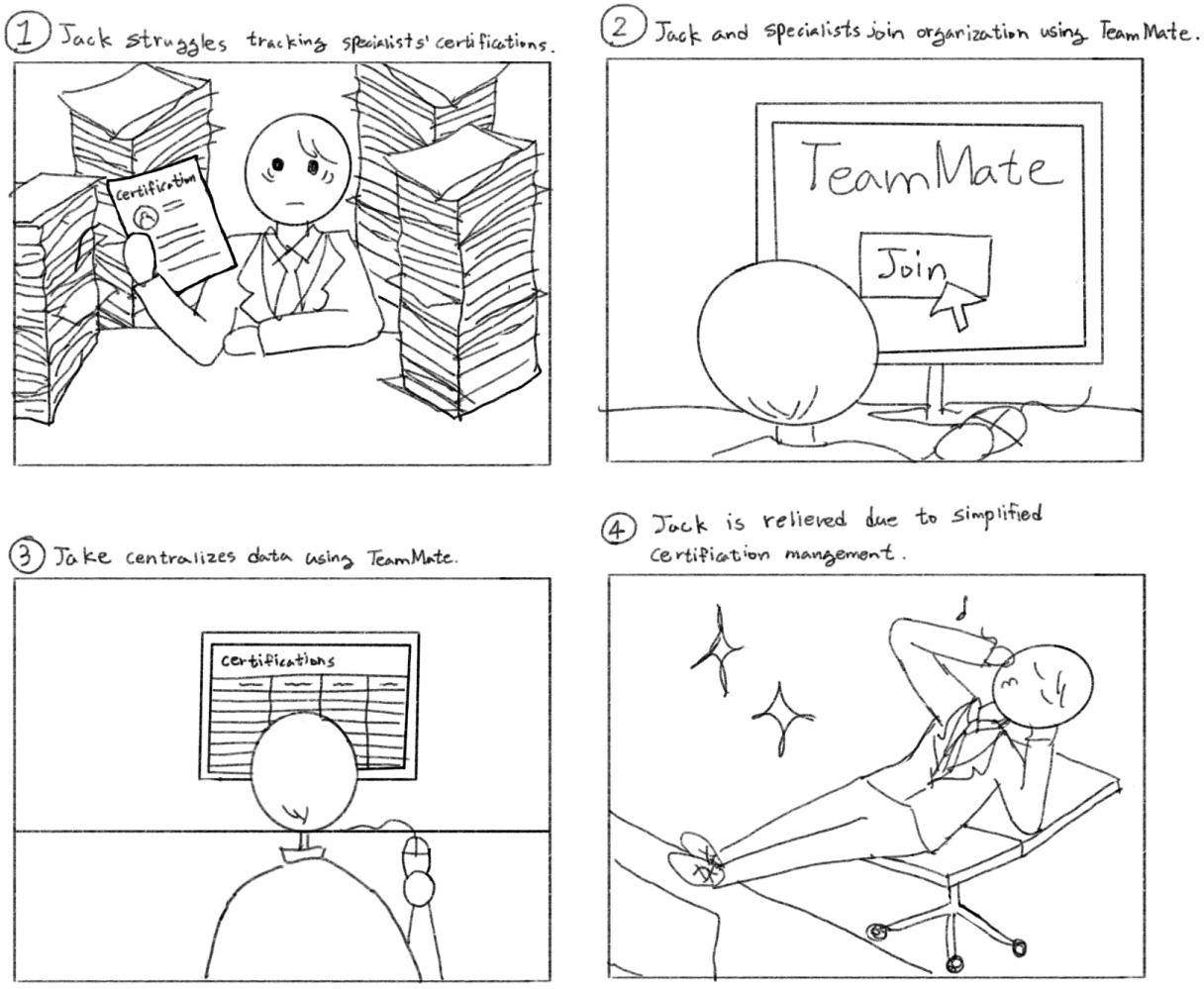
16 End

Member's Profile Page



Use Case #2

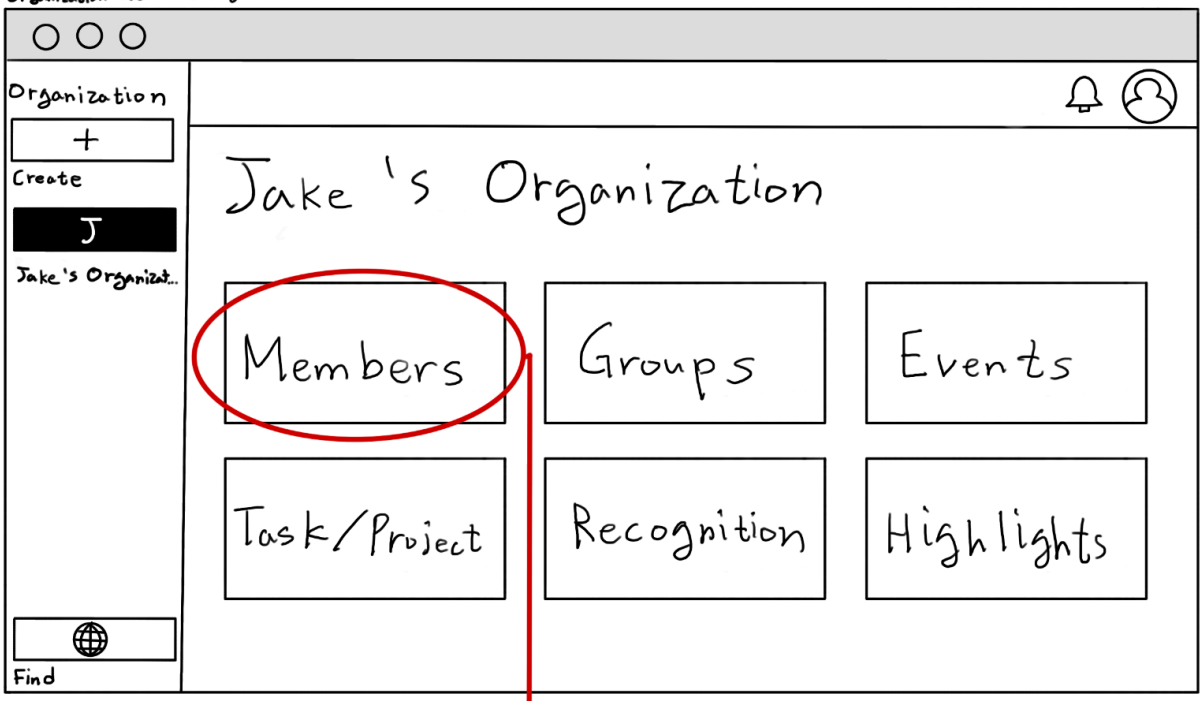
Jake, a leader of a professional group requiring numerous qualifications, found it tedious to track his subordinates' certifications across various platforms. To improve productivity, he created a TeamMate account and organization, where his specialists joined and had their qualifications imported into a centralized system. This allowed Jake to easily manage and update their qualifications, schedule training, and see attendance, simplifying the process and reducing the need to track multiple systems.



Use Case # 2

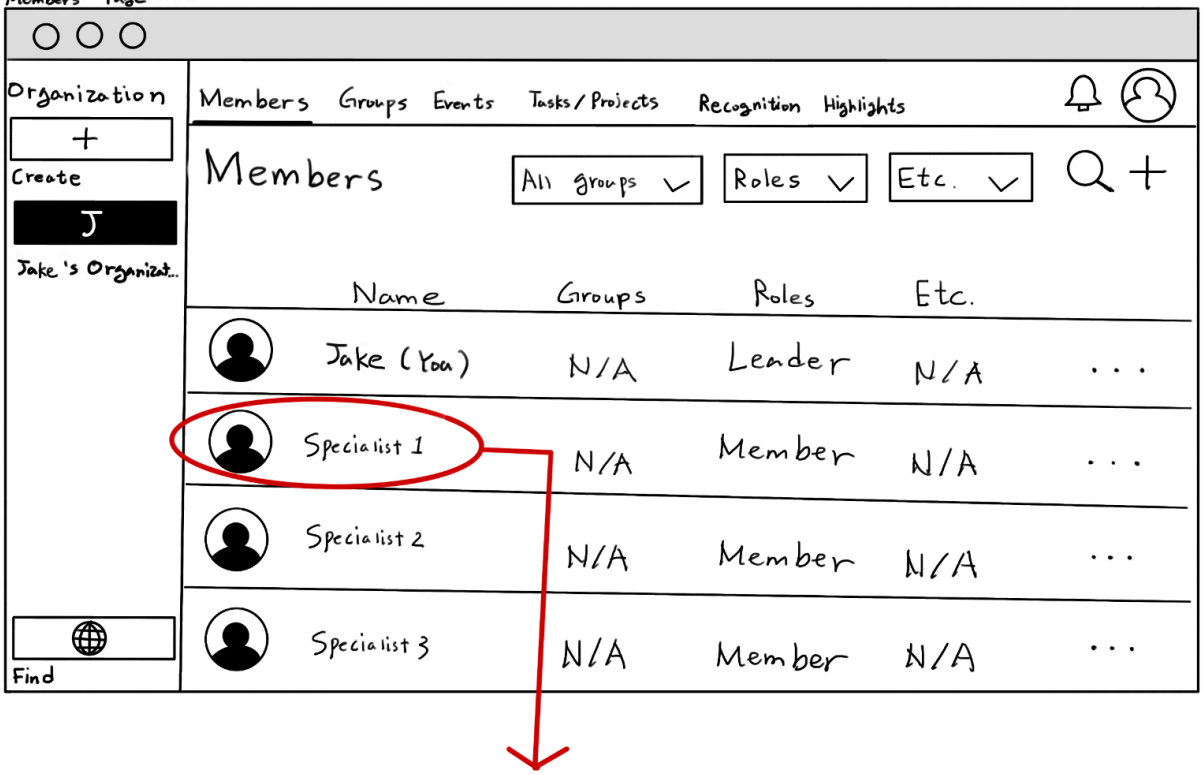
① Start - Leader Perspective

Organization Dashboard Page



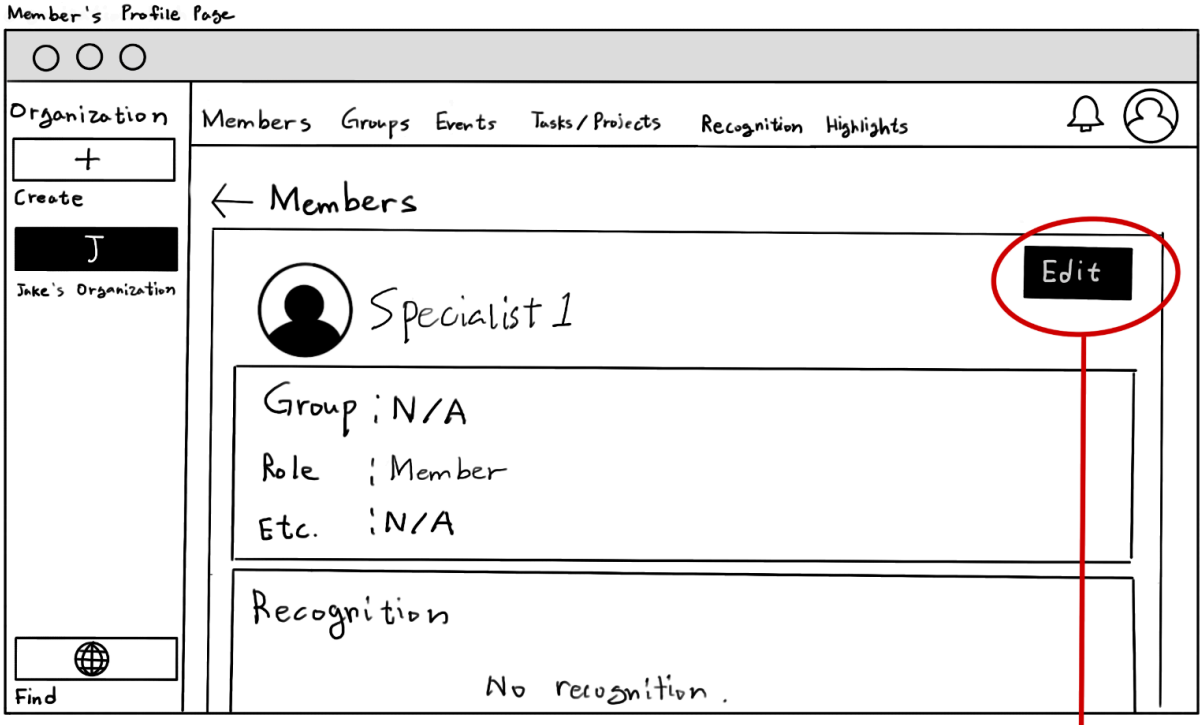
②

Members' Page



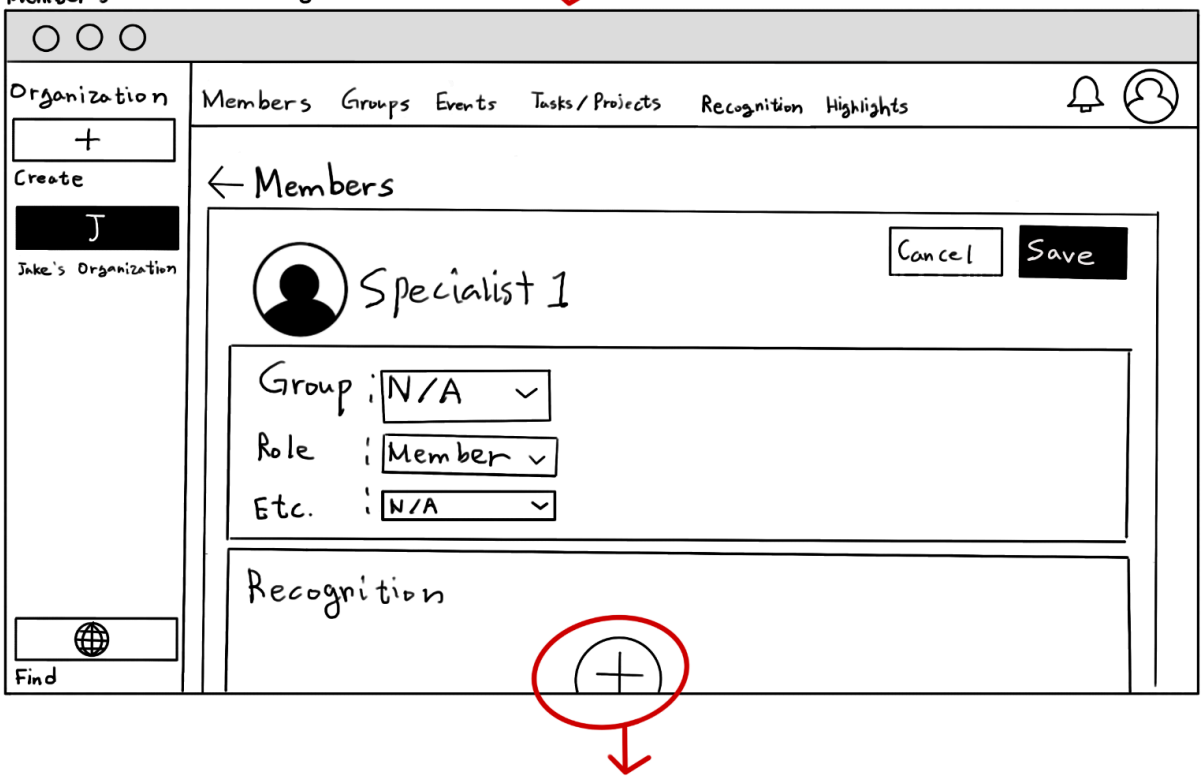
③

Member's Profile Page

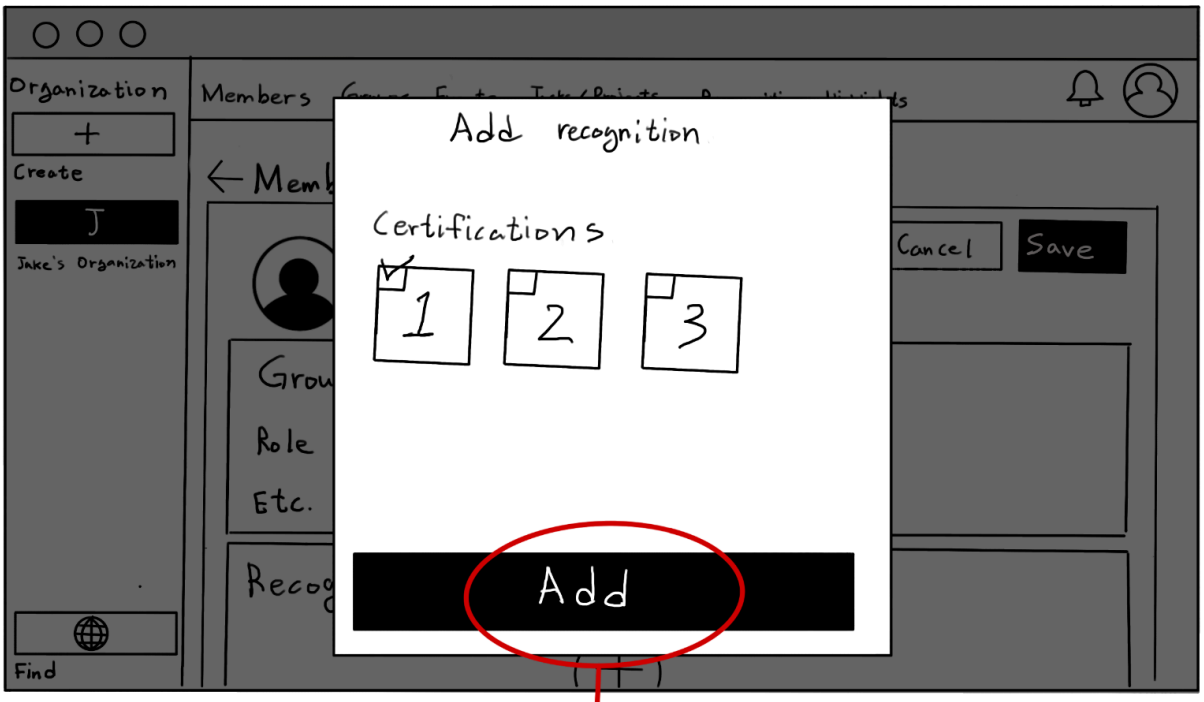


④

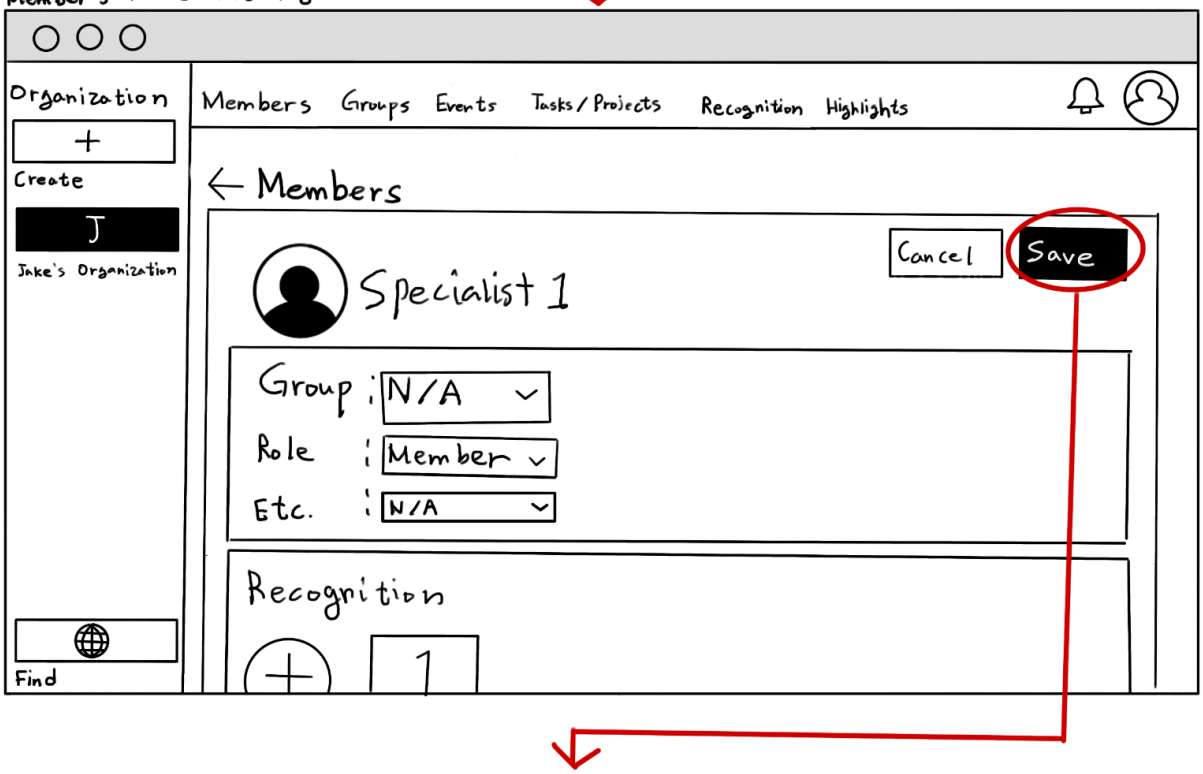
Member's Profile Edit Page



⑤
Member's Profile Edit Page

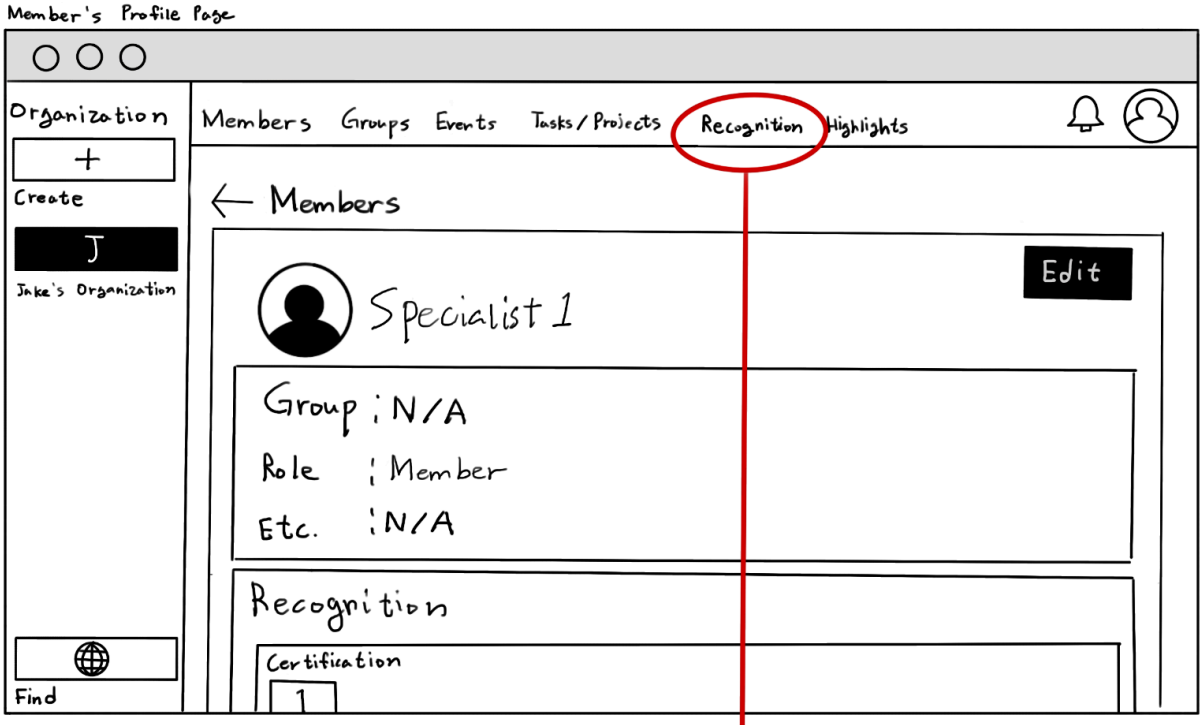


⑥
Member's Profile Edit Page



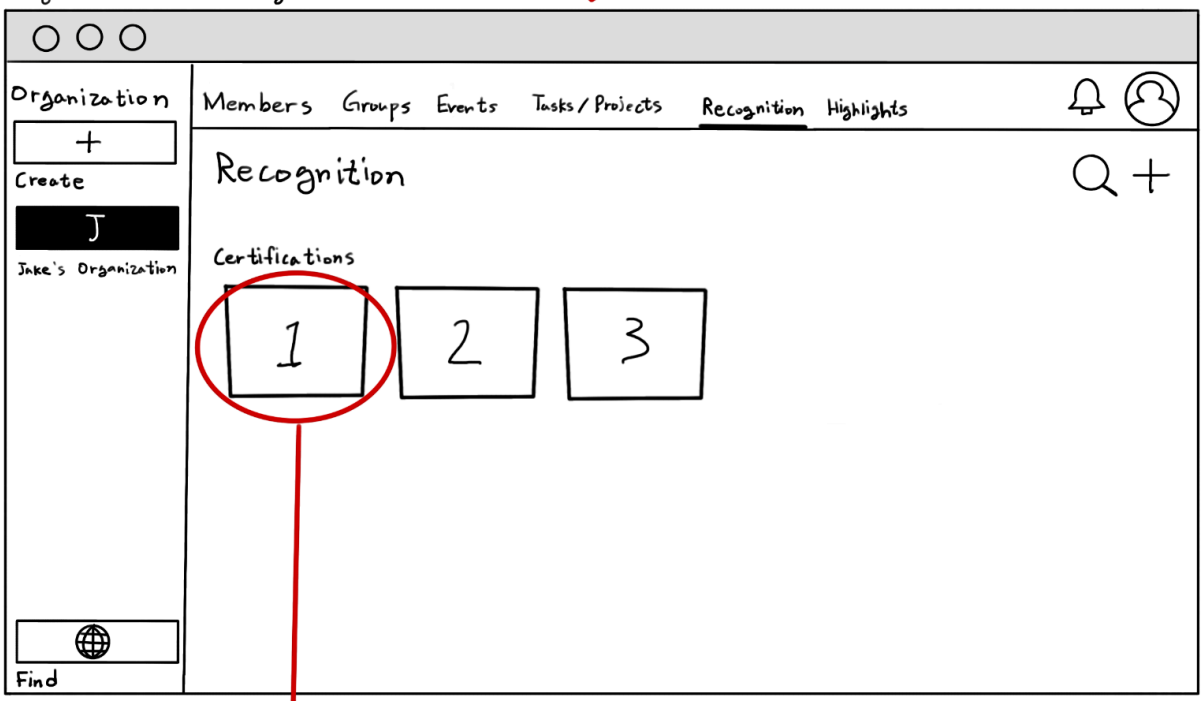
⑦

Member's Profile Page



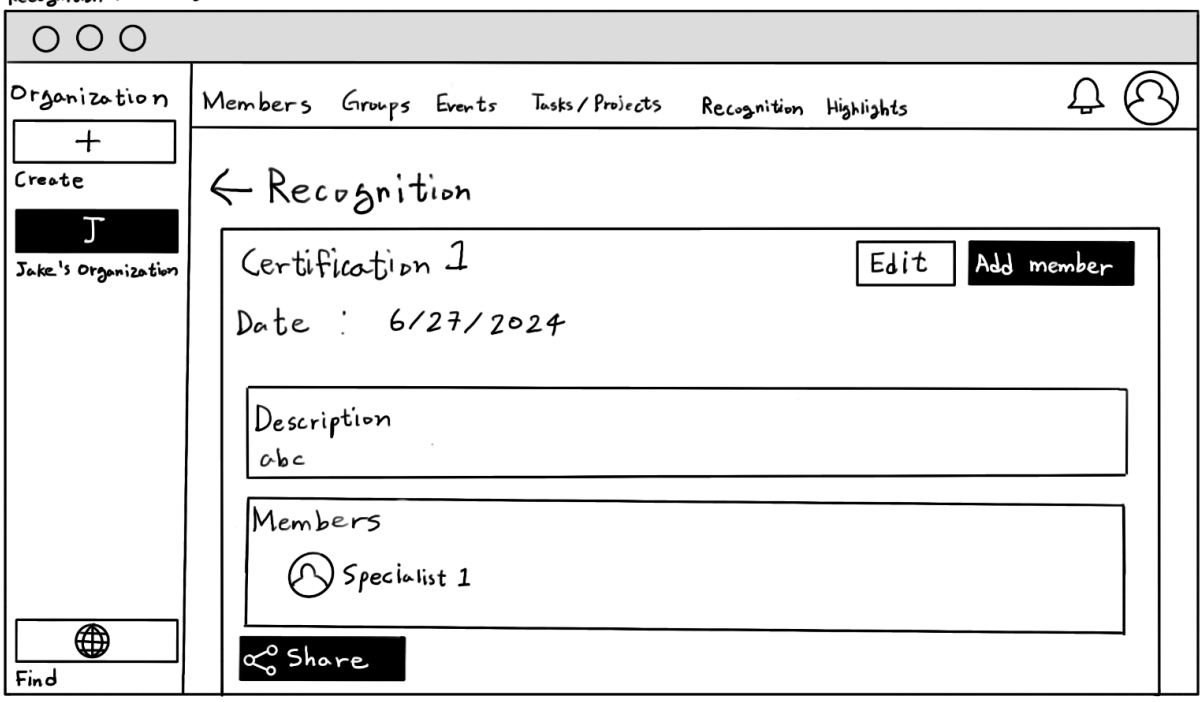
⑧

Recognition Dashboard Page



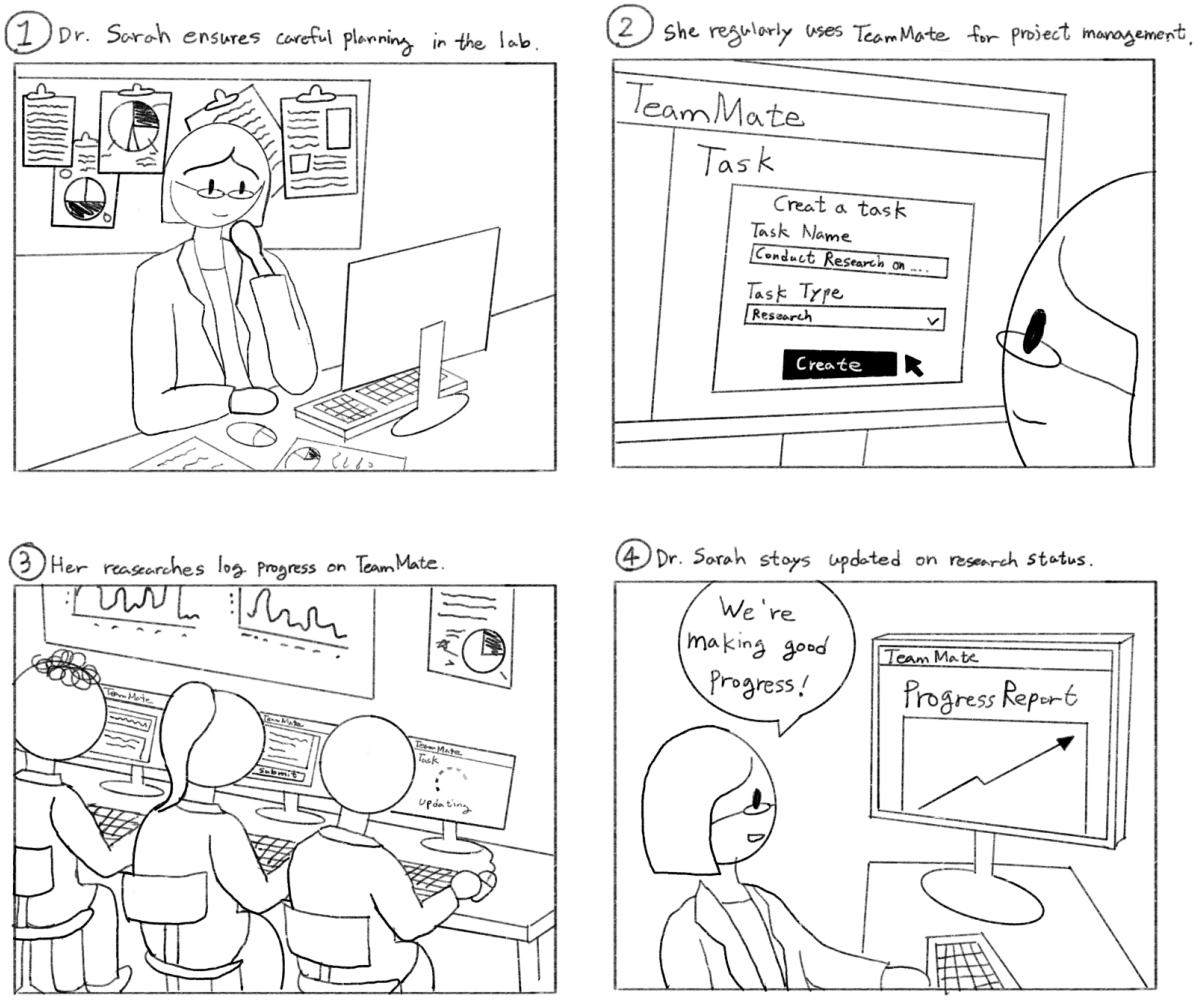
⑨End

Recognition Detail Page



Use Case #3

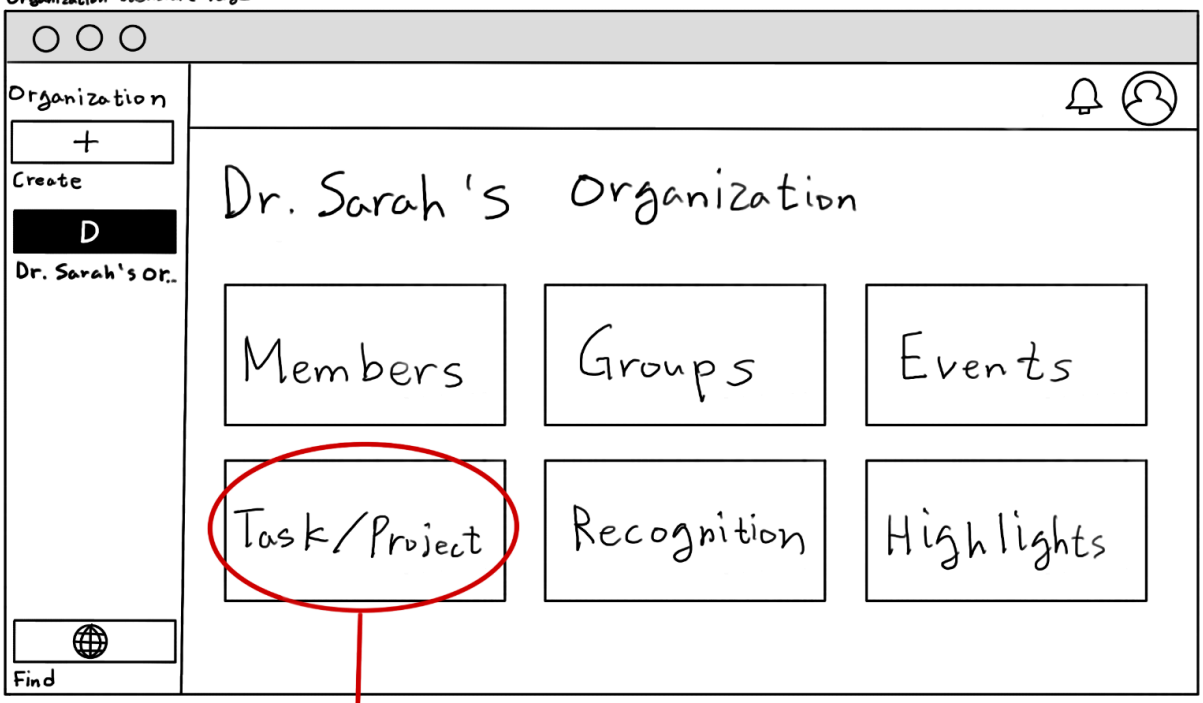
Dr. Sarah uses the TeamMate platform to manage a new research project by inputting its details, objectives, timelines, and assigned researchers. Researchers, lab assistants, and graduate students log their progress and update milestones regularly. Dr. Sarah reviews these updates, provides feedback, and adjusts timelines as needed, while the system generates a comprehensive progress report summarizing all ongoing research projects.



Use case # 3

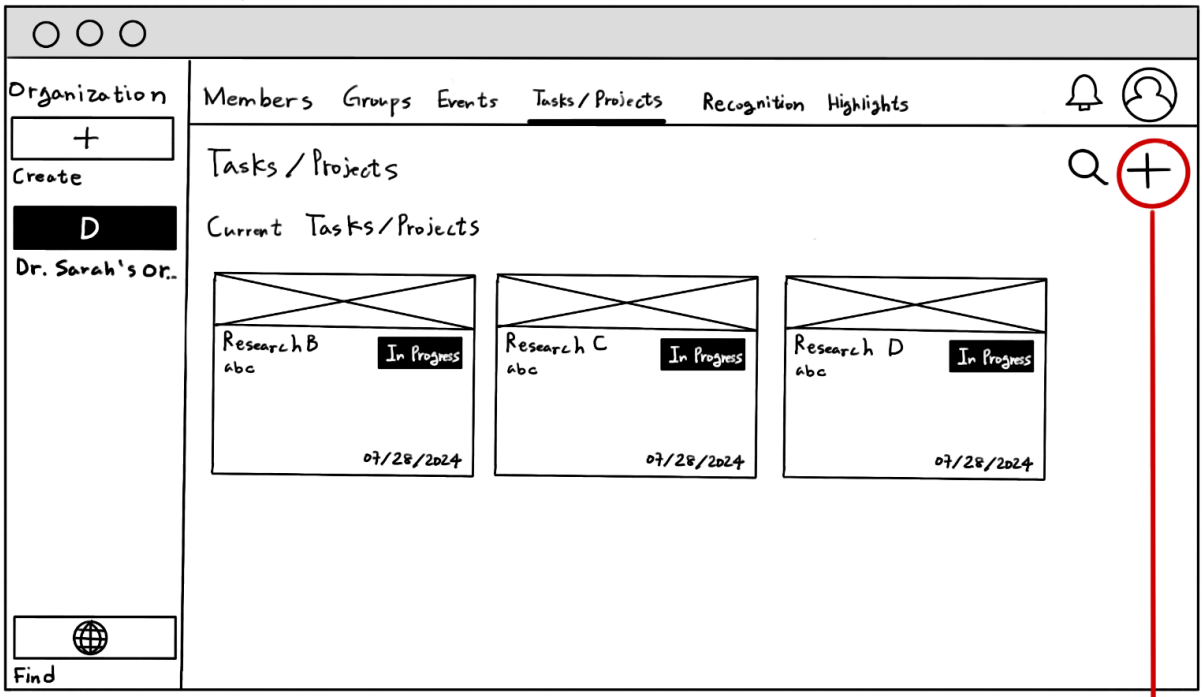
① Start- Leader Perspective

Organization Dashboard Page



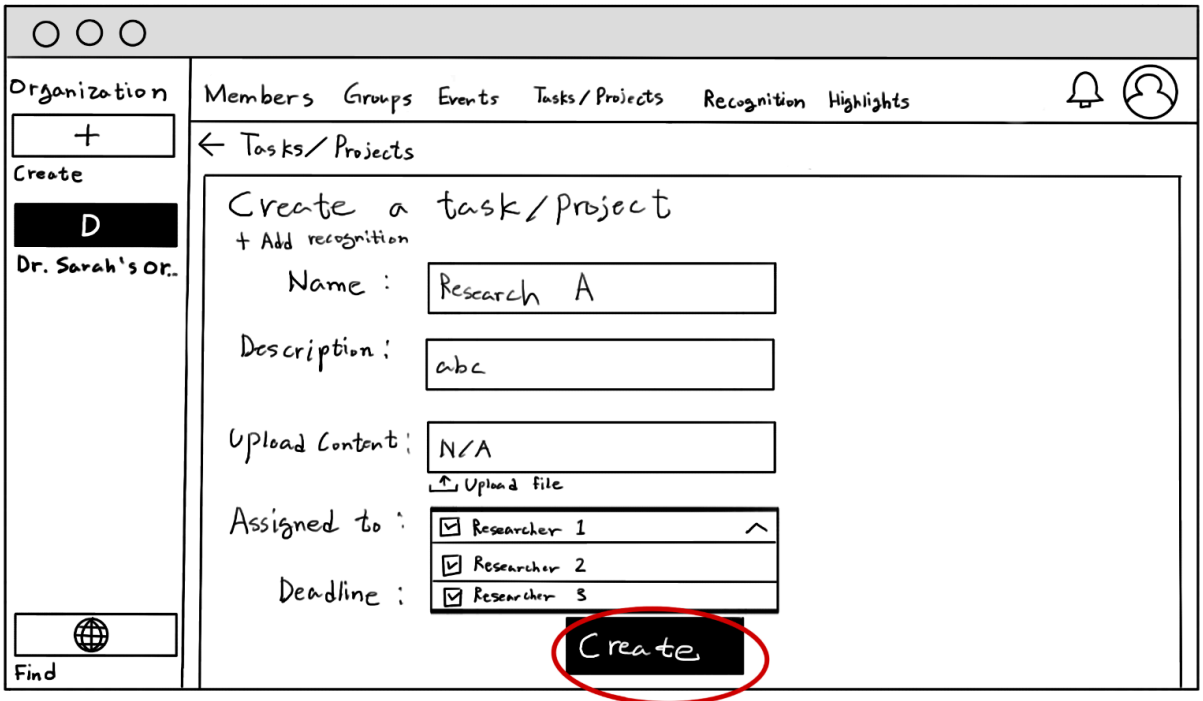
②

Tasks/Projects Page



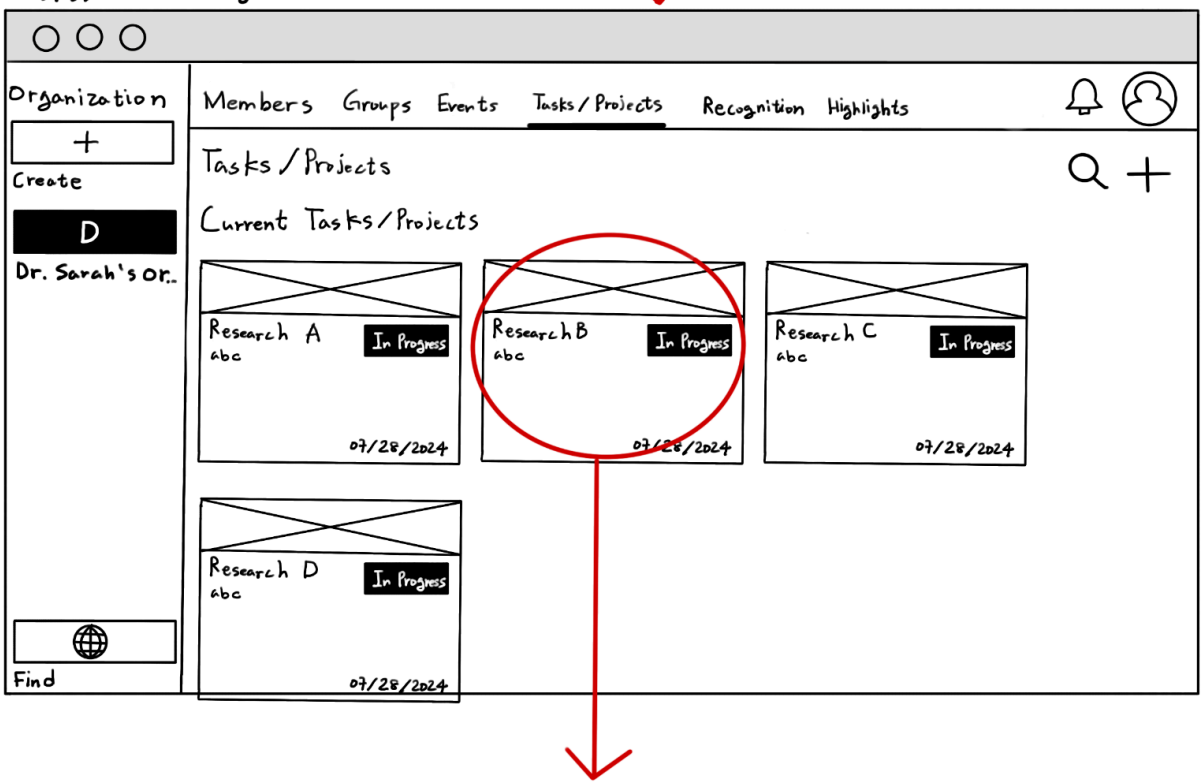
③

Task / Project Create Page



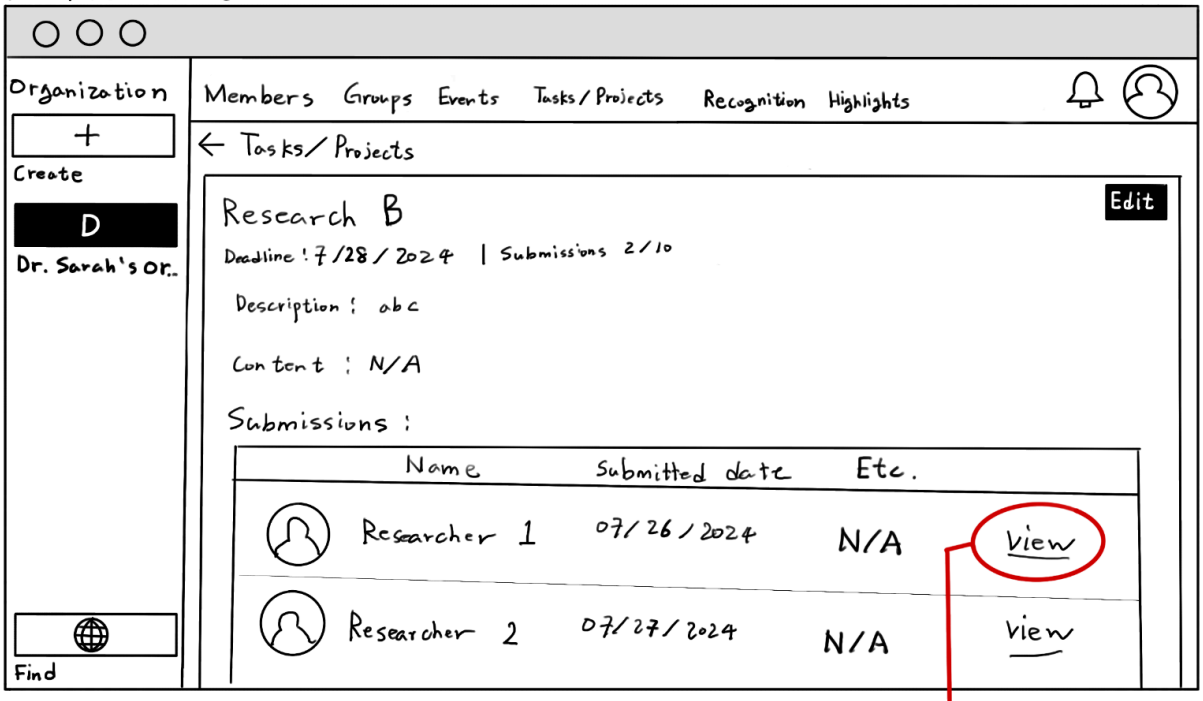
④

Tasks / Projects Page



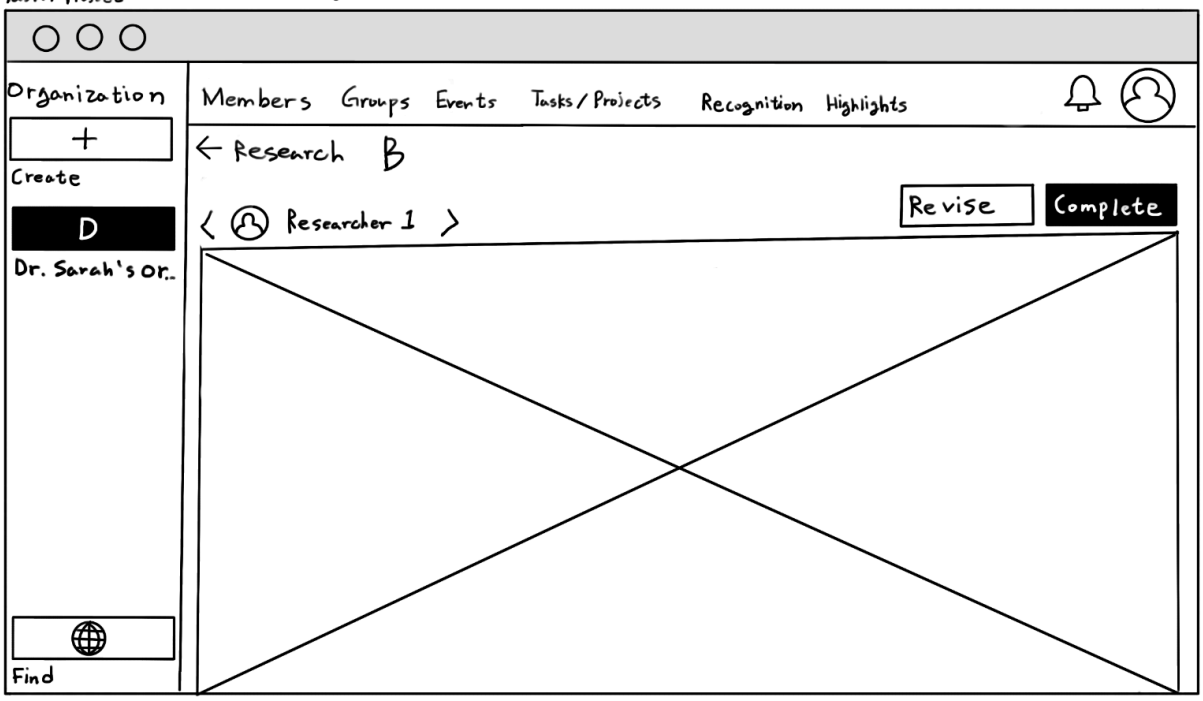
⑤

Task / Project View Page



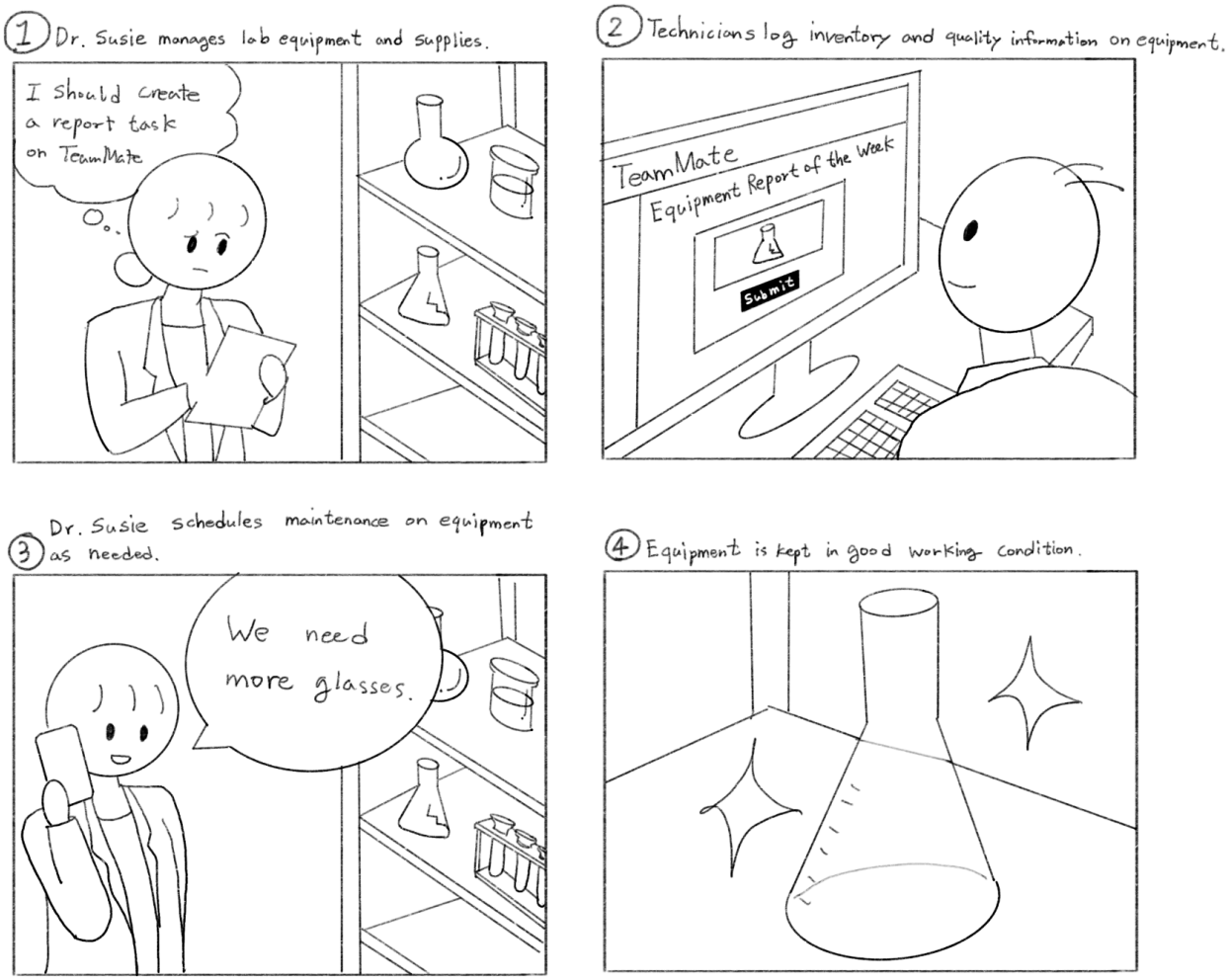
⑥ End

Task / Project Submission View Page



Use Case #4

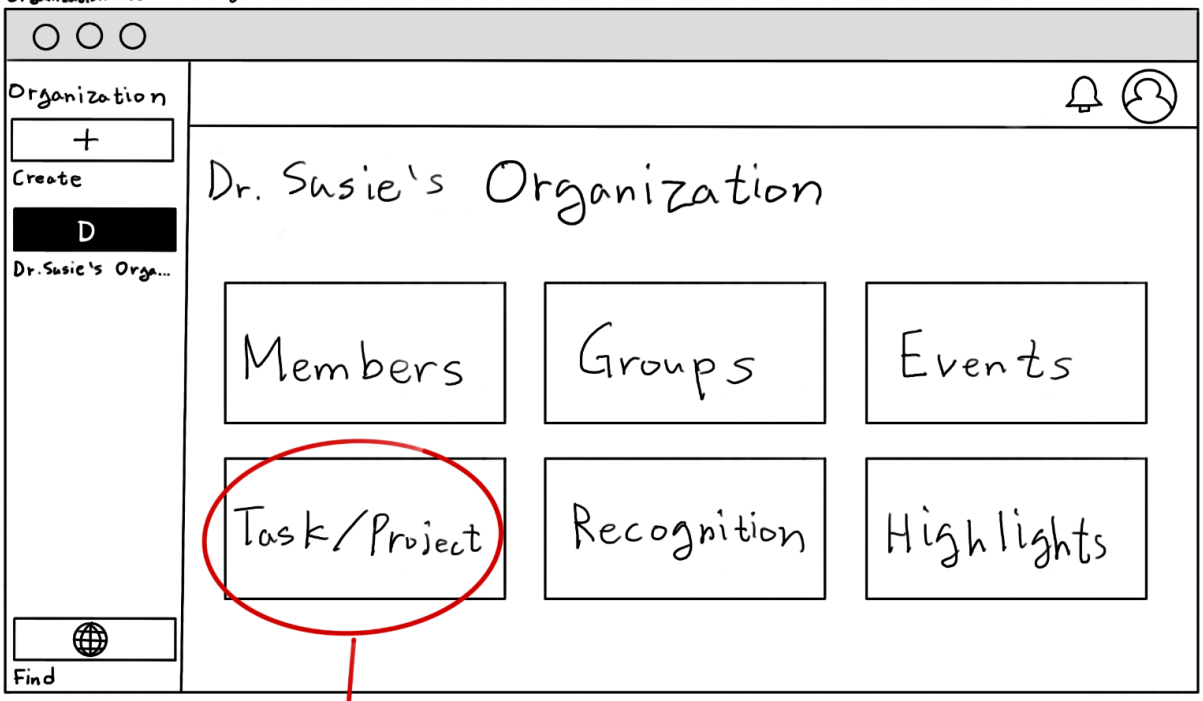
Dr. Susie uses the TeamMate platform's "Equipment and Supplies Management" section to input details about lab equipment and supplies, including quantities, locations, and maintenance schedules. Lab technicians log usage and report issues, while maintenance staff schedules and performs necessary maintenance. Dr. Susie reviews usage logs, schedules maintenance, and receives alerts when supplies are low, allowing her to place orders and update inventory, with the system regularly tracking and confirming these updates.



Use Case #4

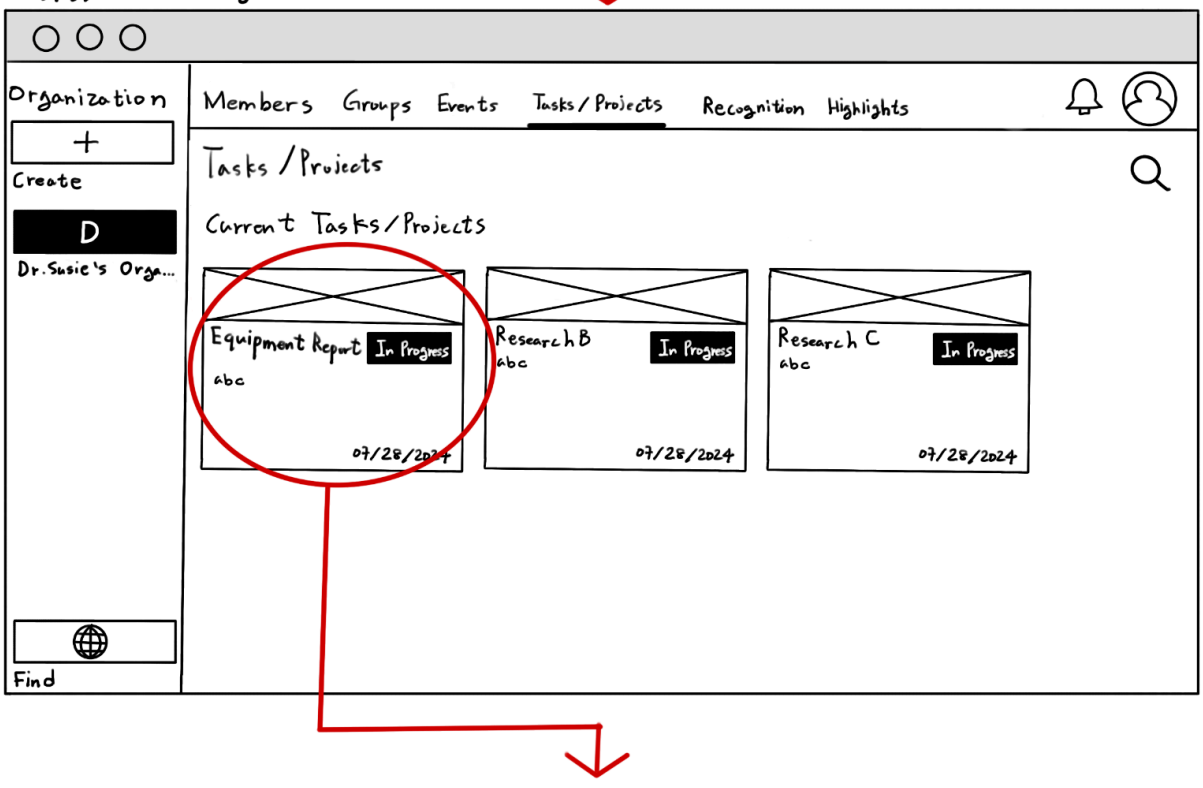
① Start – Member Perspective

Organization Dashboard Page

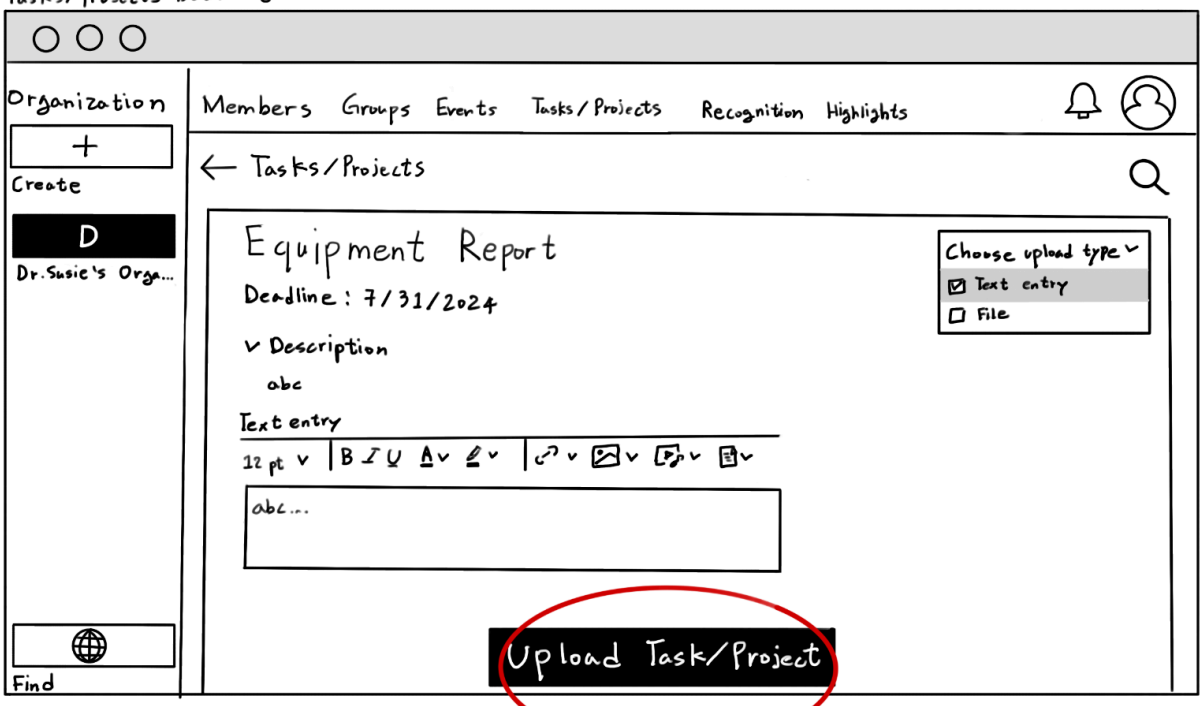


②

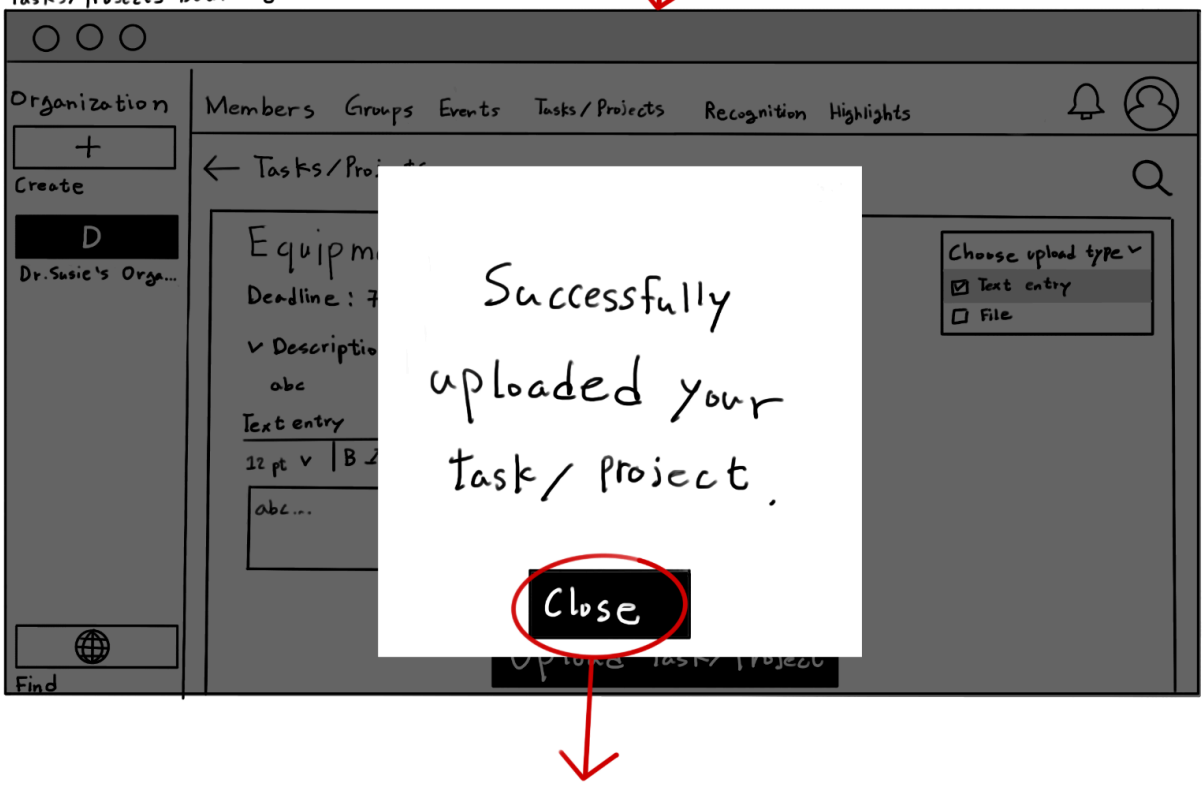
Tasks/Projects Page



③
Tasks/Projects Detail Page

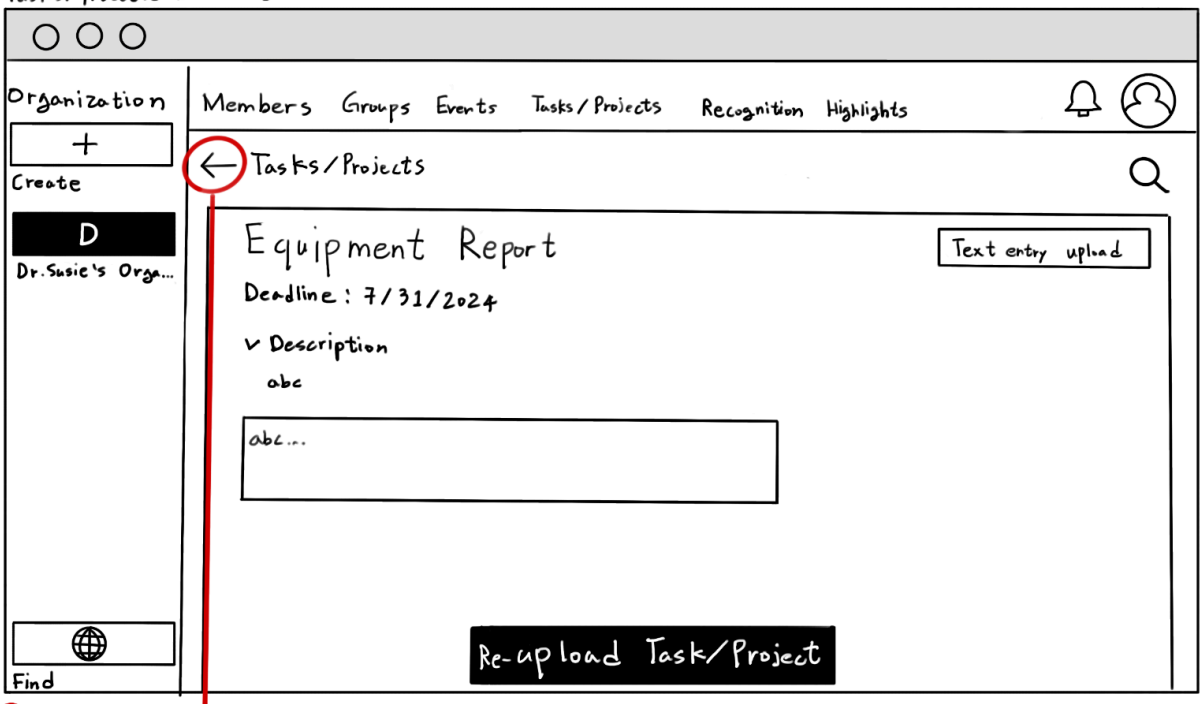


④
Tasks/Projects Detail Page



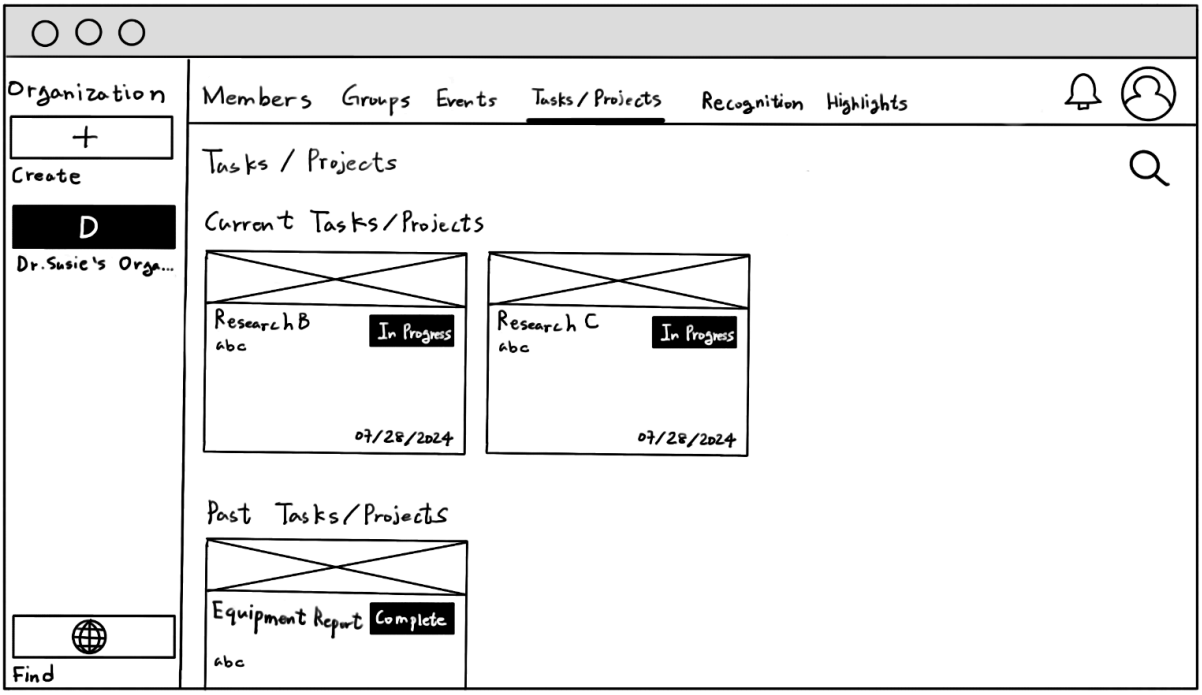
⑤

Tasks/Projects Detail Page



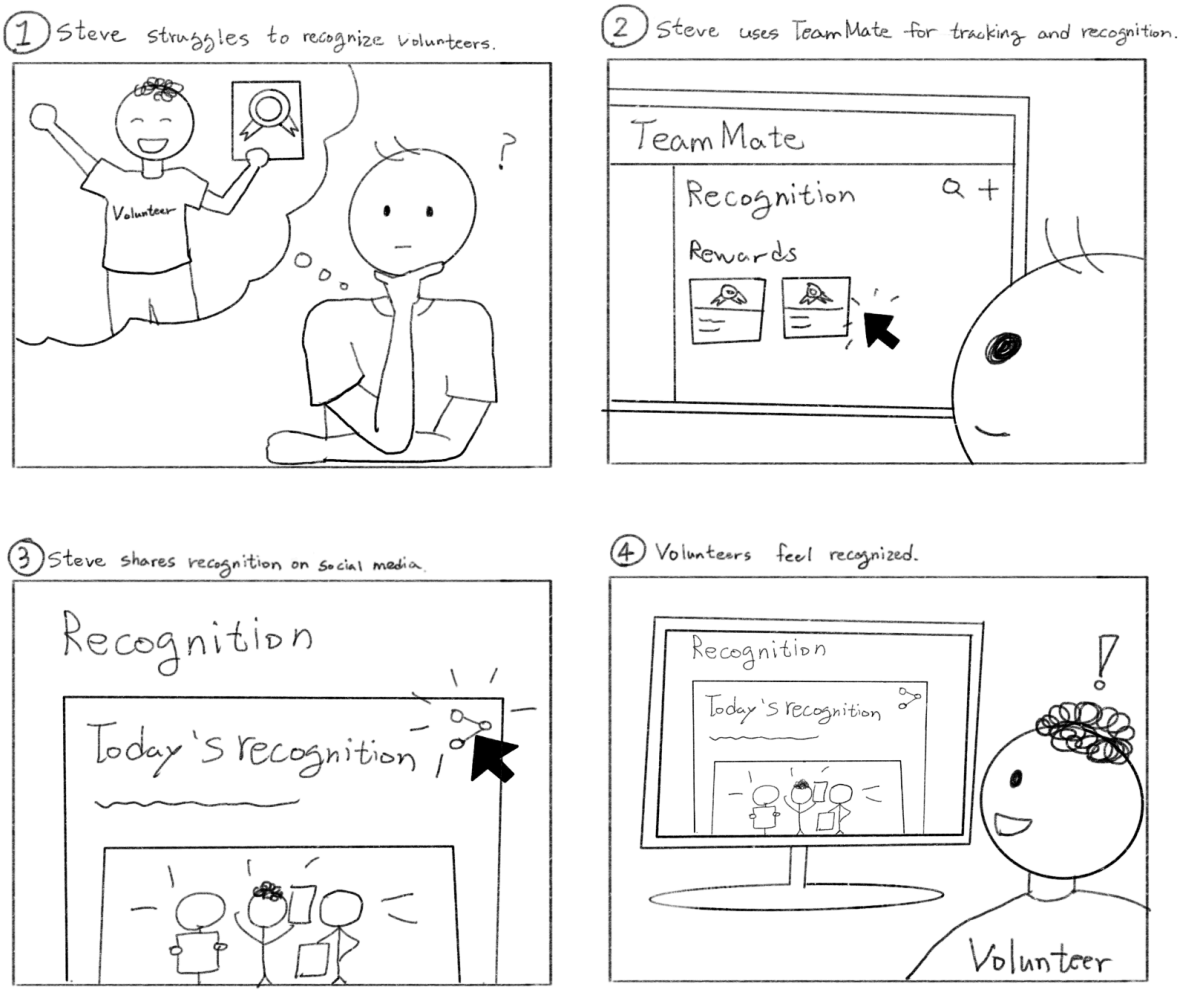
⑥ End

Tasks/Projects Page



Use Case #5

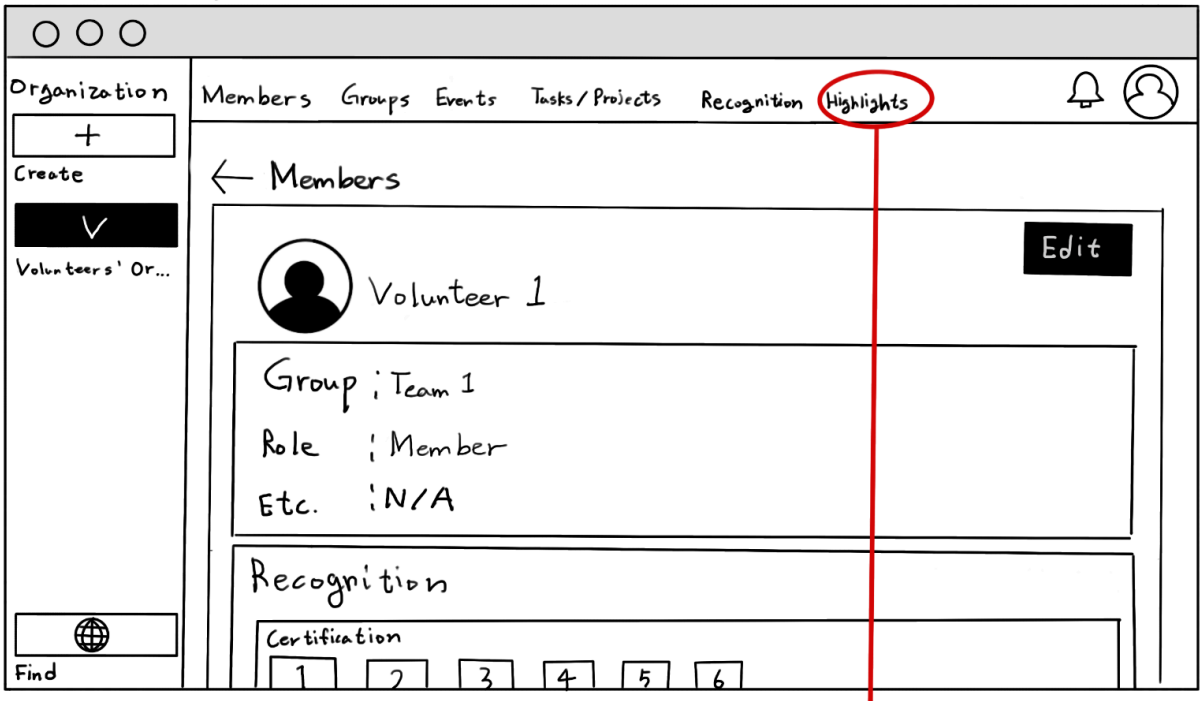
Steve, a manager at NGO XYZ, is eager to recognize and monitor volunteer activities but finds it challenging to ensure everyone is on the same pace. Discovering TeamMate, he uses the platform to track and provide a comprehensive overview of all members, ensuring no one is overlooked. He issues digital accolades visible on profiles, and TeamMate suggests awarding badges for completed tasks, which can be shared publicly on social media to recognize volunteer contributions.



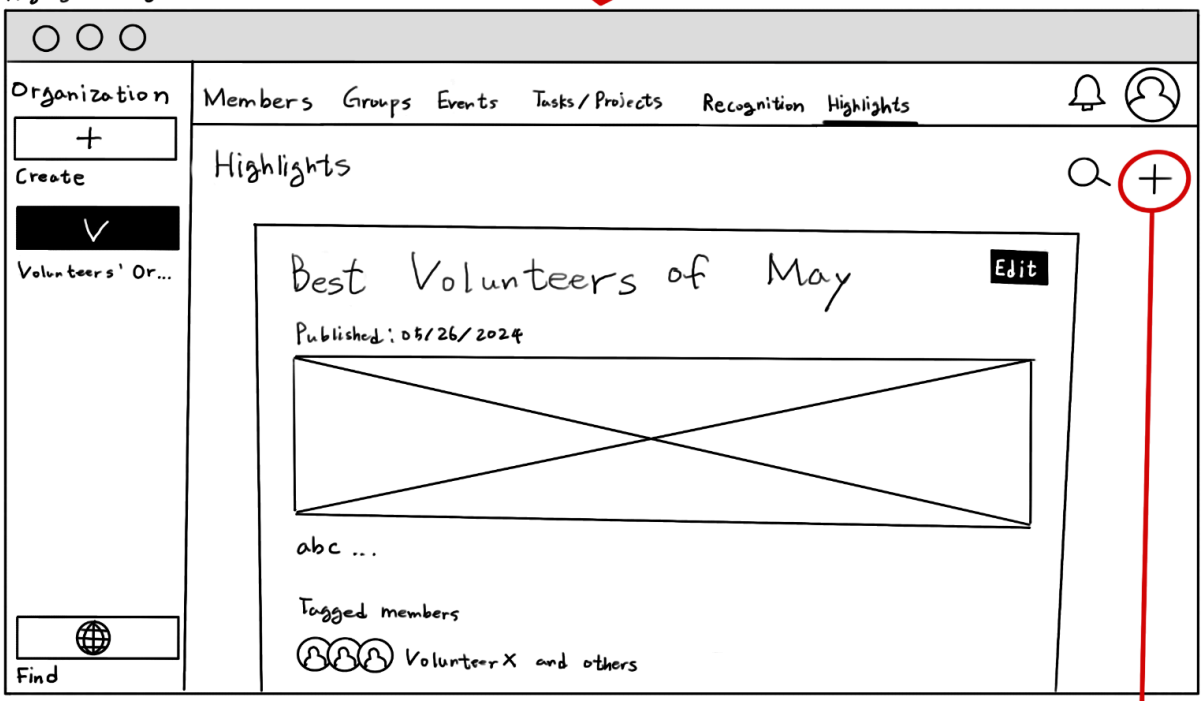
① Start - Leader Perspective

Use Case # 5

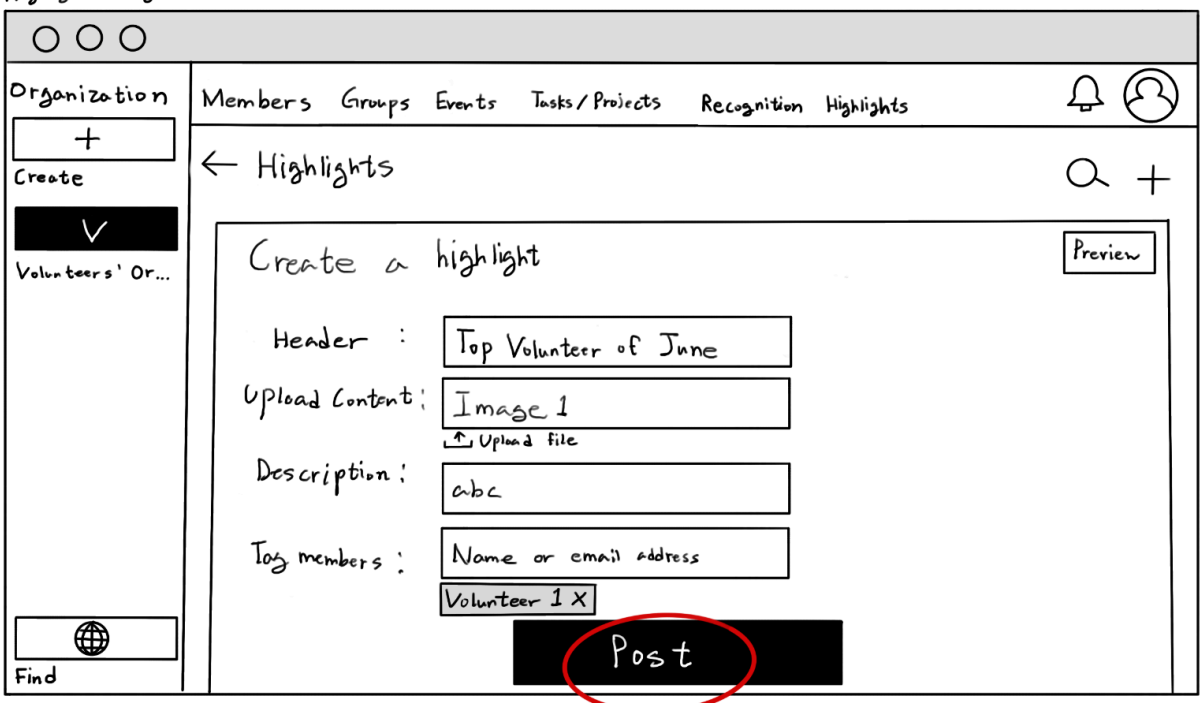
Member's Profile Page



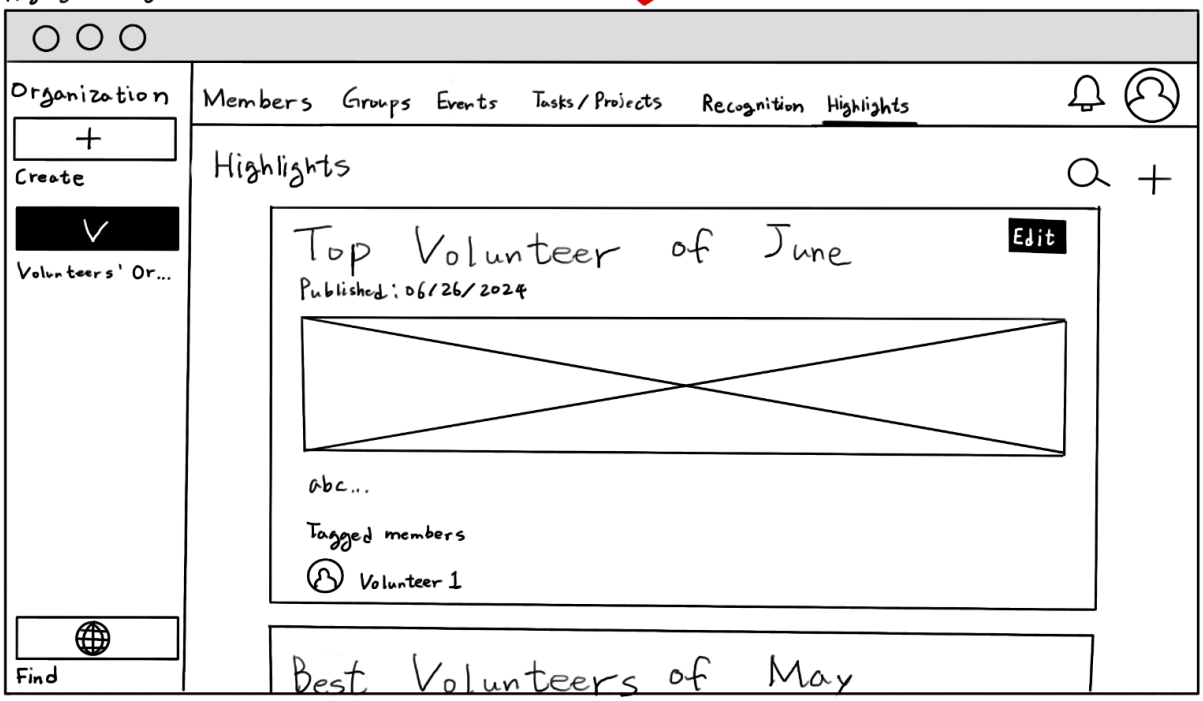
② Highlights Page



③
Highlights Page

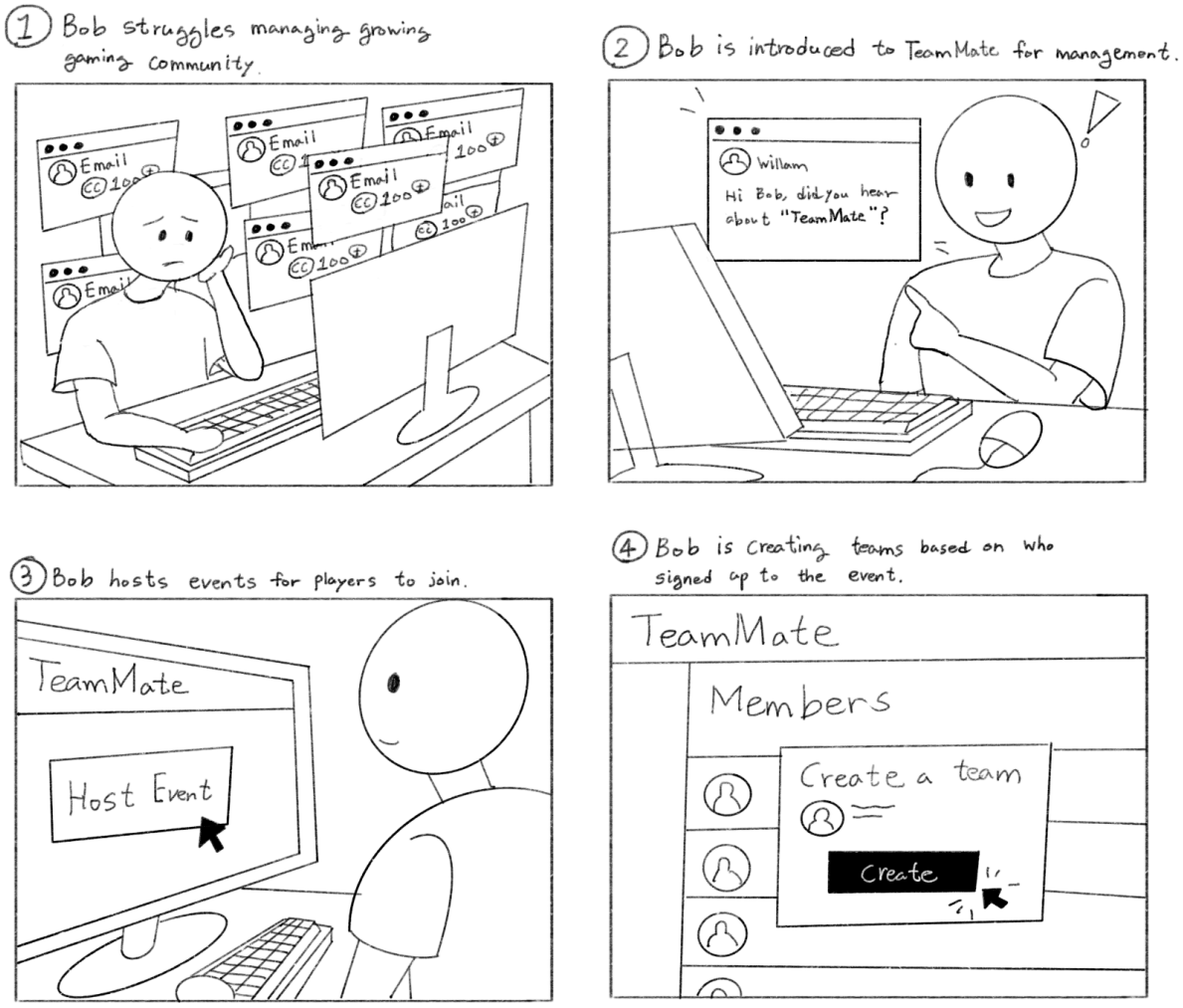


④ End
Highlights Page



Use Case #6

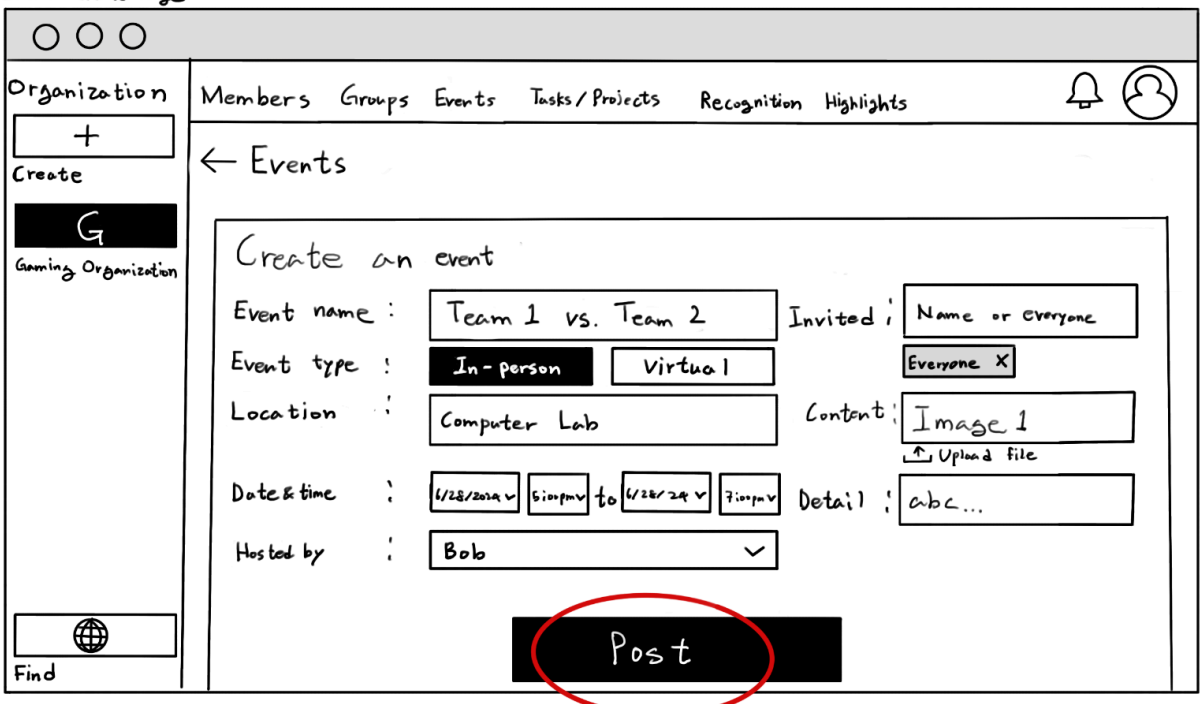
Bob, a CS graduate from SFSU, founded a gaming community called Inferno, which grew to 100 active members hosting events across the Bay Area. As the community expanded, managing events, teams, and individual achievements became challenging without raising membership fees. A member introduced him to TeamMate, an online management platform that organizes events, tracks achievements, and manages profiles. Using TeamMate, the community can efficiently handle events and tournaments, track attendance, and make announcements, ensuring smooth operations and detailed member profiles.



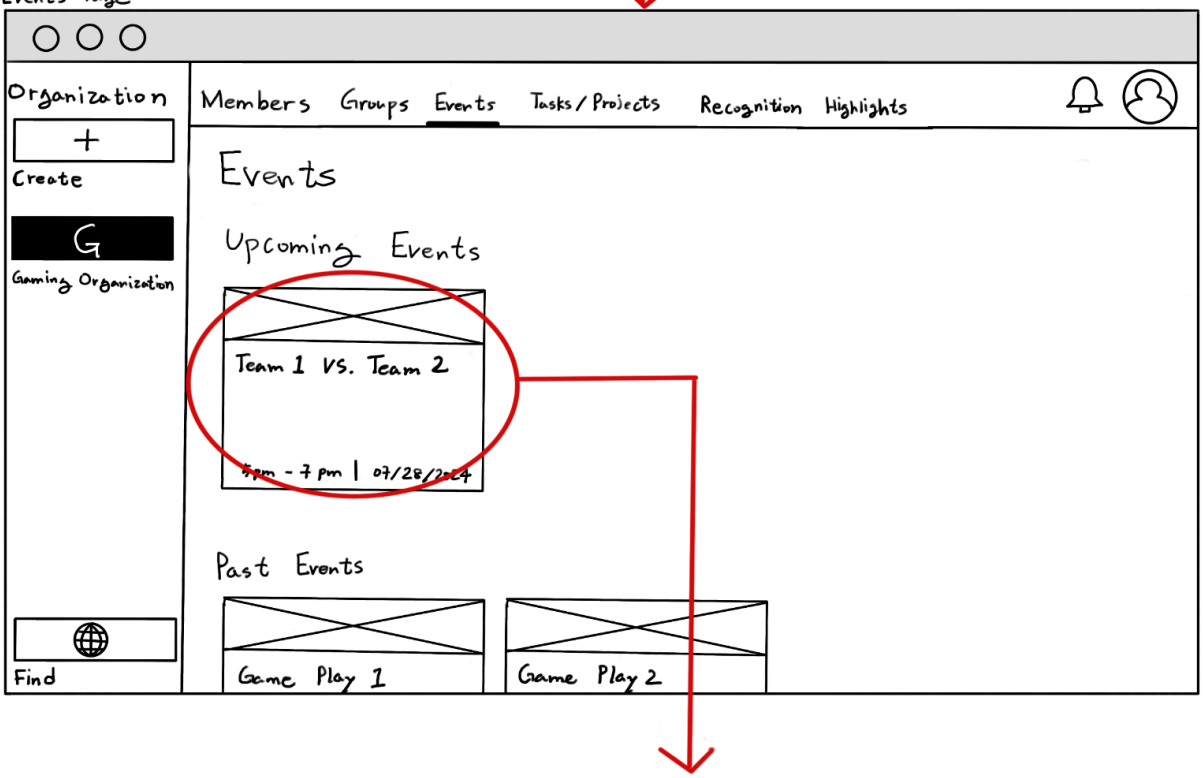
① Start - Leader Perspective

Use Case #6

Events Create Page

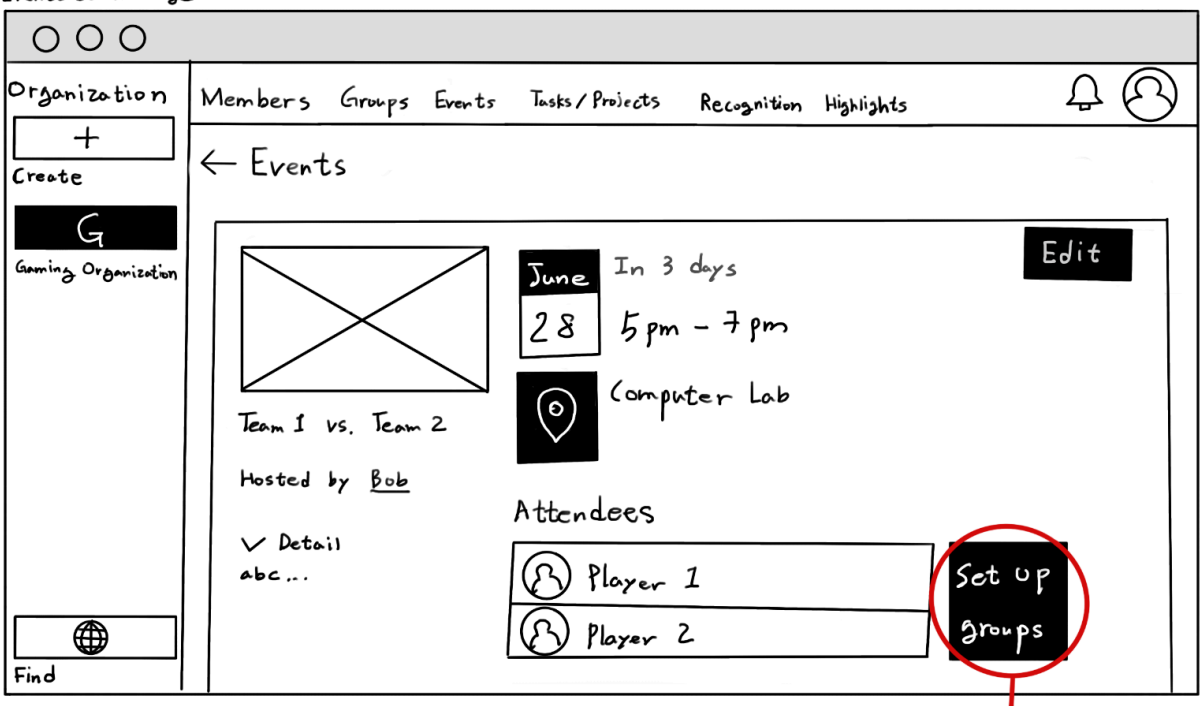


② Events Page



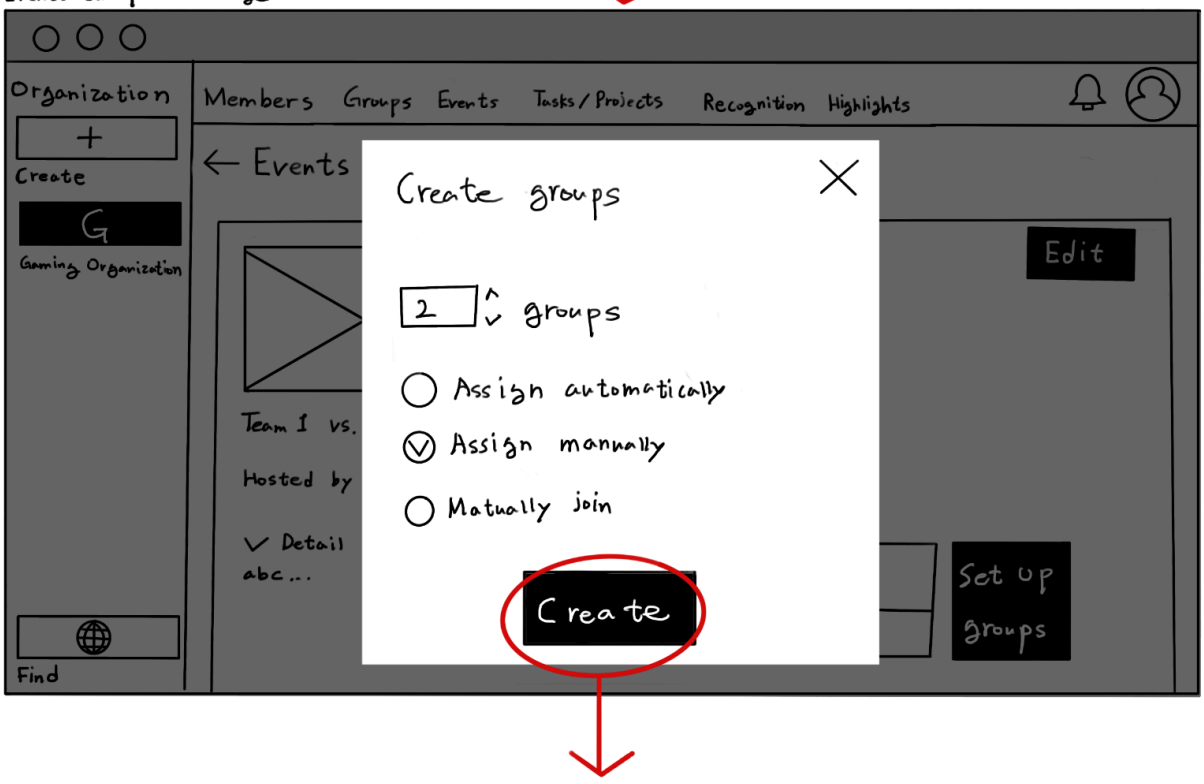
③

Events Detail Page



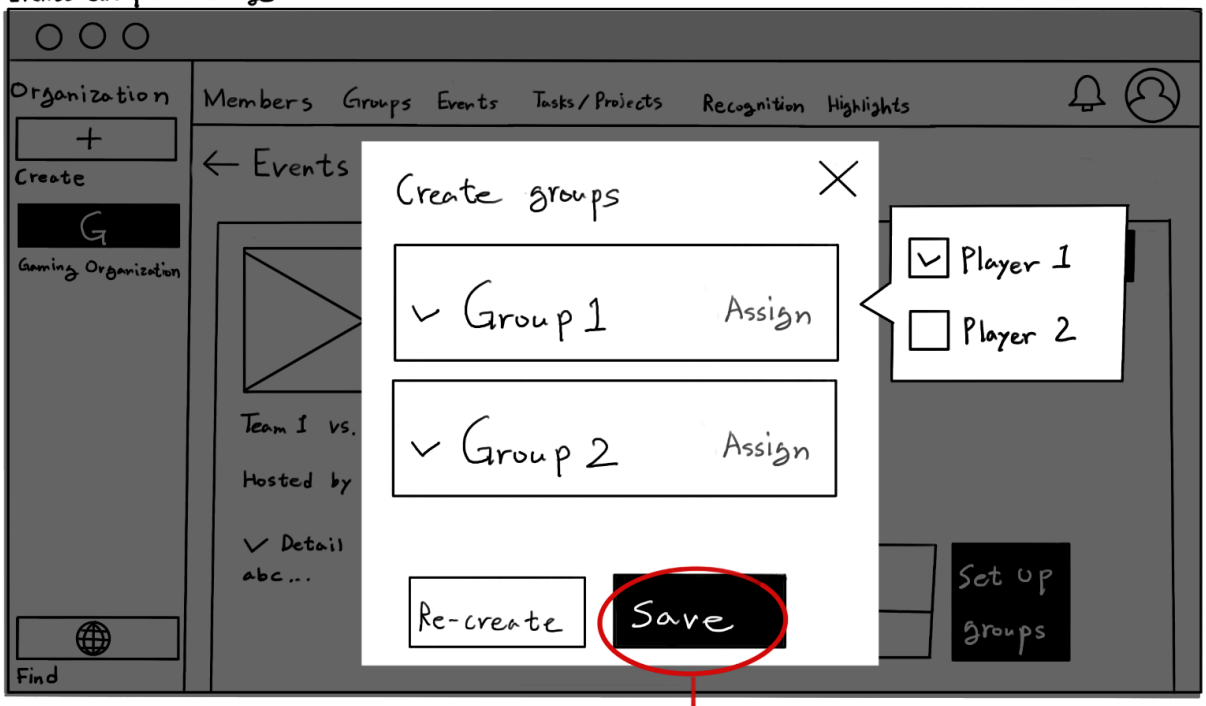
④

Events Group Create Page



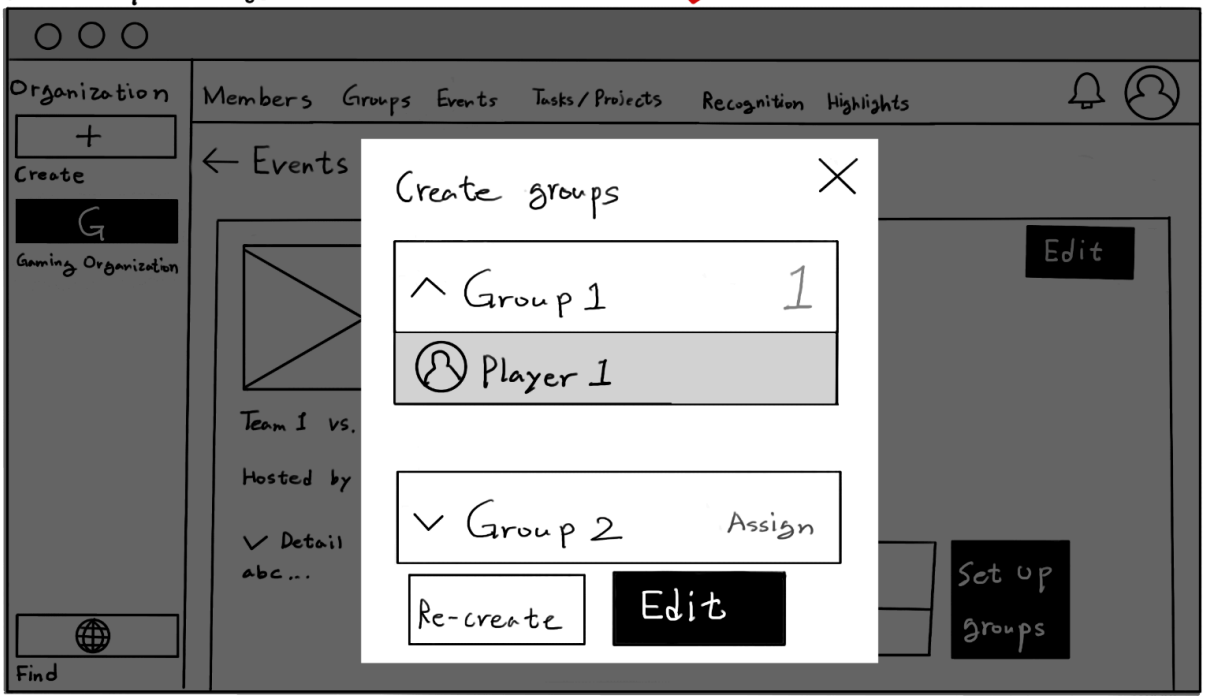
⑤

Events Group Create Page



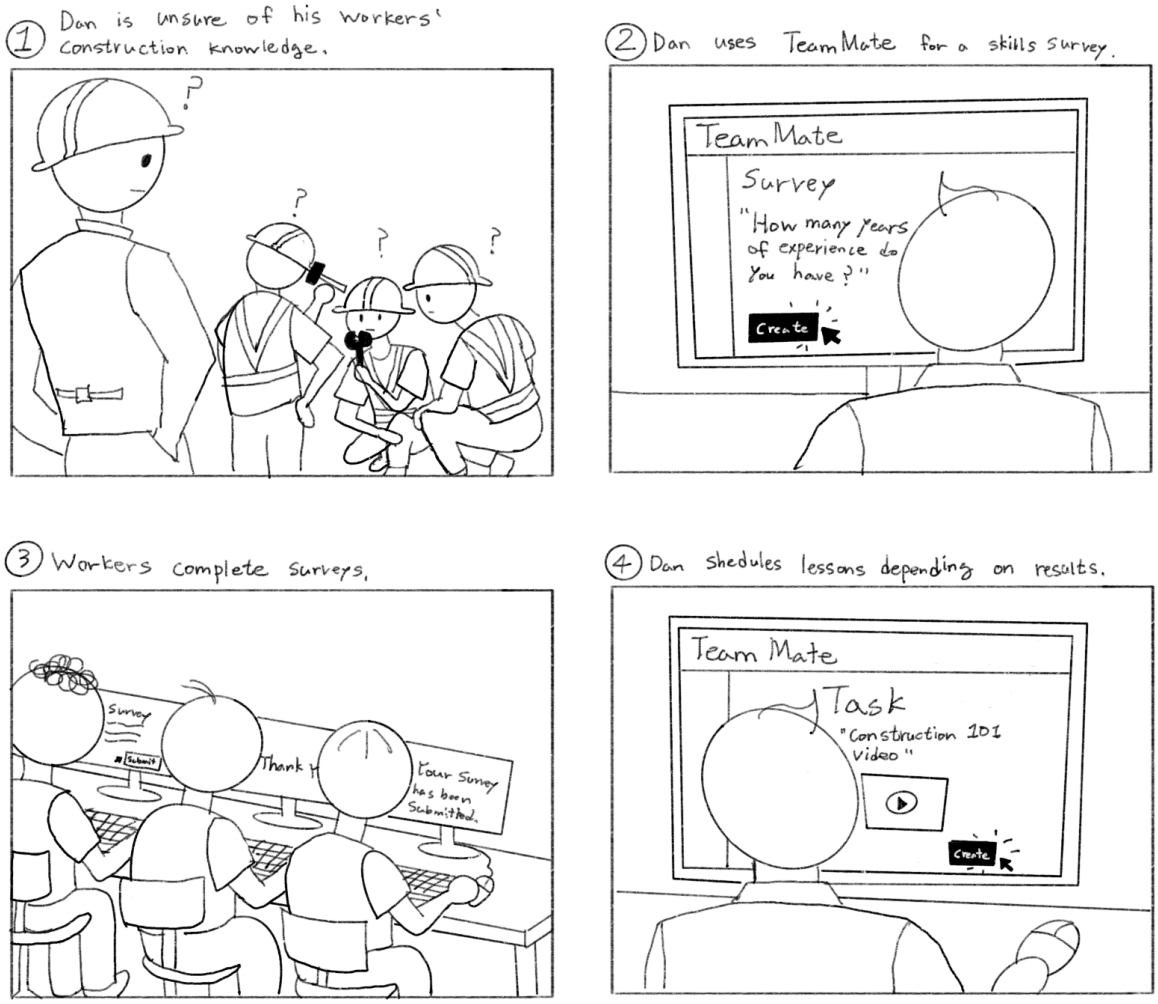
⑥ End

Events Group Create Page



Use Case #7

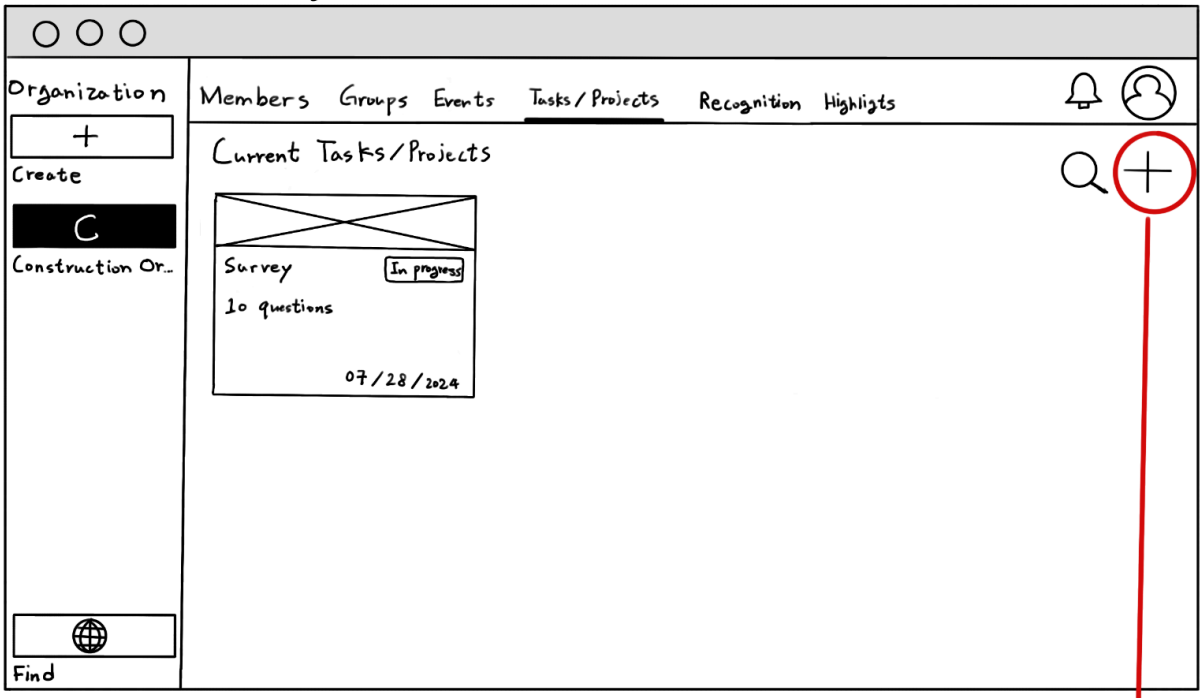
Dan, a manager at Danliland construction company, employs 20 workers and uses the TeamMate App to track their training needs and experience. Workers like Luigi, Angel, and Joshi complete a survey in the app to highlight their knowledge and past learning. The app allows Dan to manage worker accounts, monitor survey completion, and schedule meetings based on the survey results to address any learning gaps.



① Start – Leader Perspective

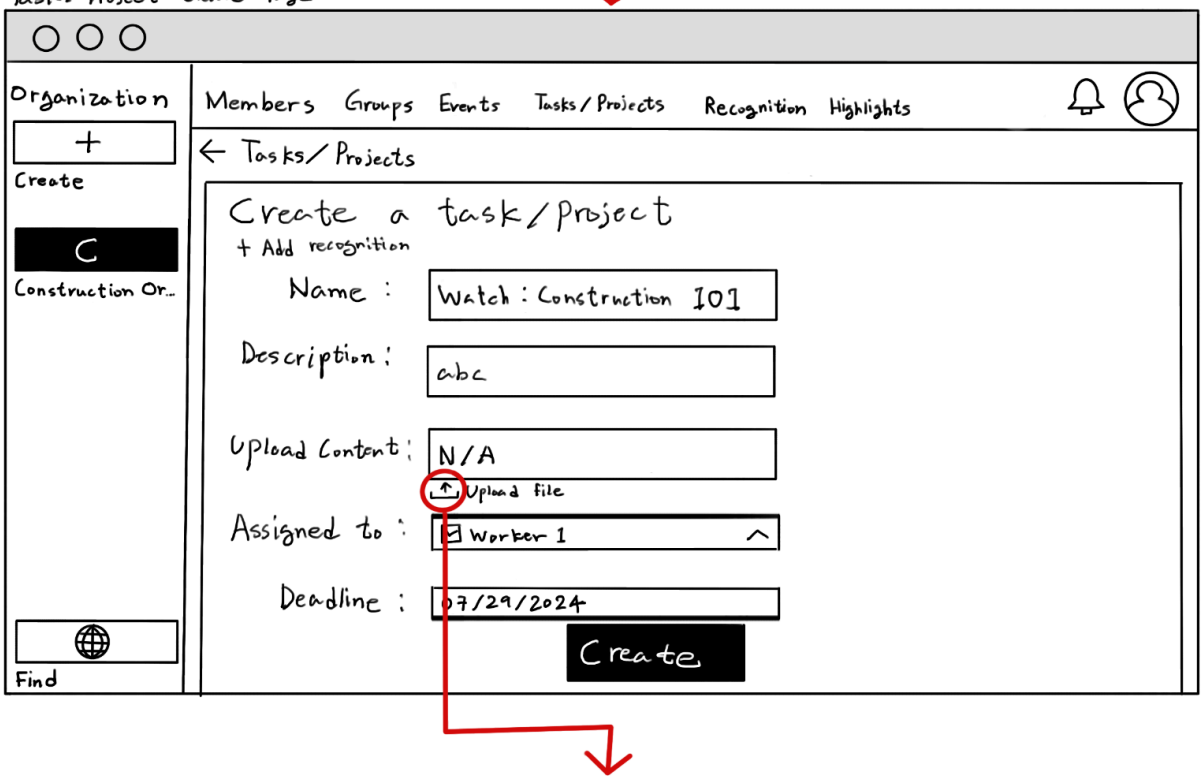
Use Case # 7

Tasks/Projects Dashboard Page



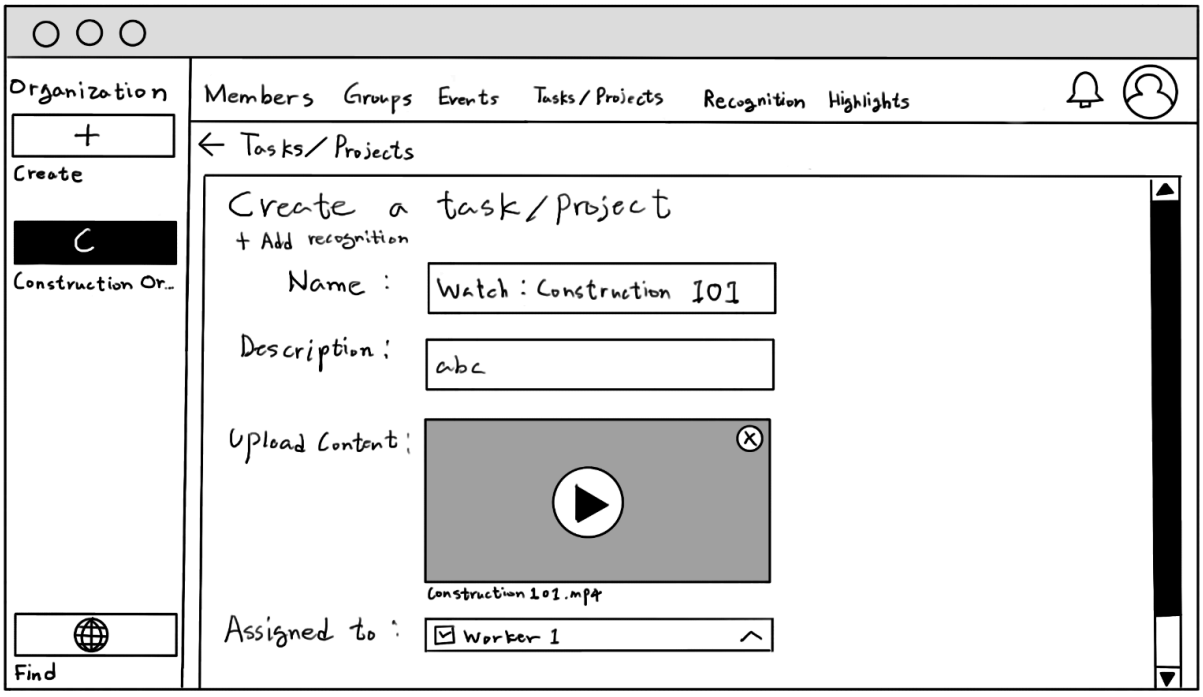
②

Task / Project Create Page



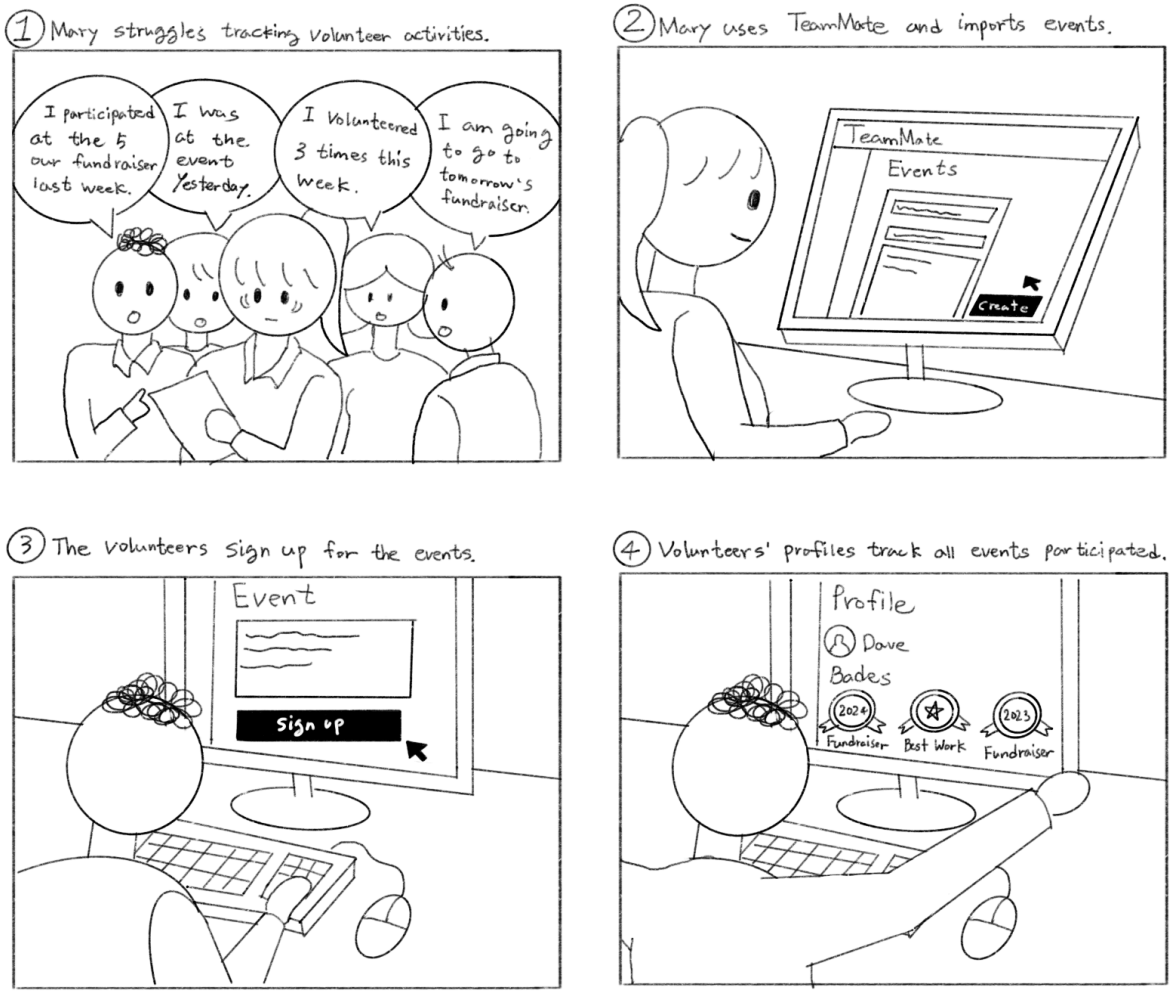
③ End

Task / Project Create Page



Use Case #8

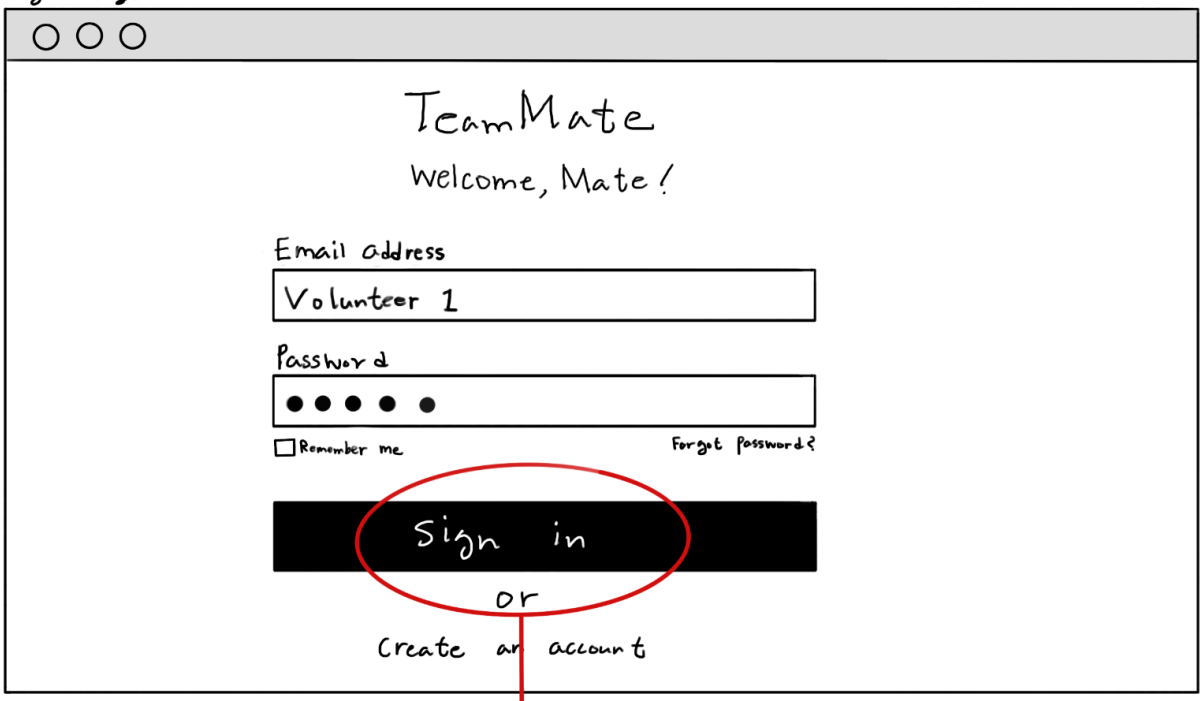
Mary, a coordinator at a local non-profit, struggled to track volunteer hours, activities, and accomplishments manually. Discovering TeamMate, she created an account, imported historical data, and set up her organization. The platform allows her to assign tasks, track hours in real-time, manage schedules, and generate reports to highlight volunteer efforts, making the process efficient and helping secure future funding.



Use Case # 8

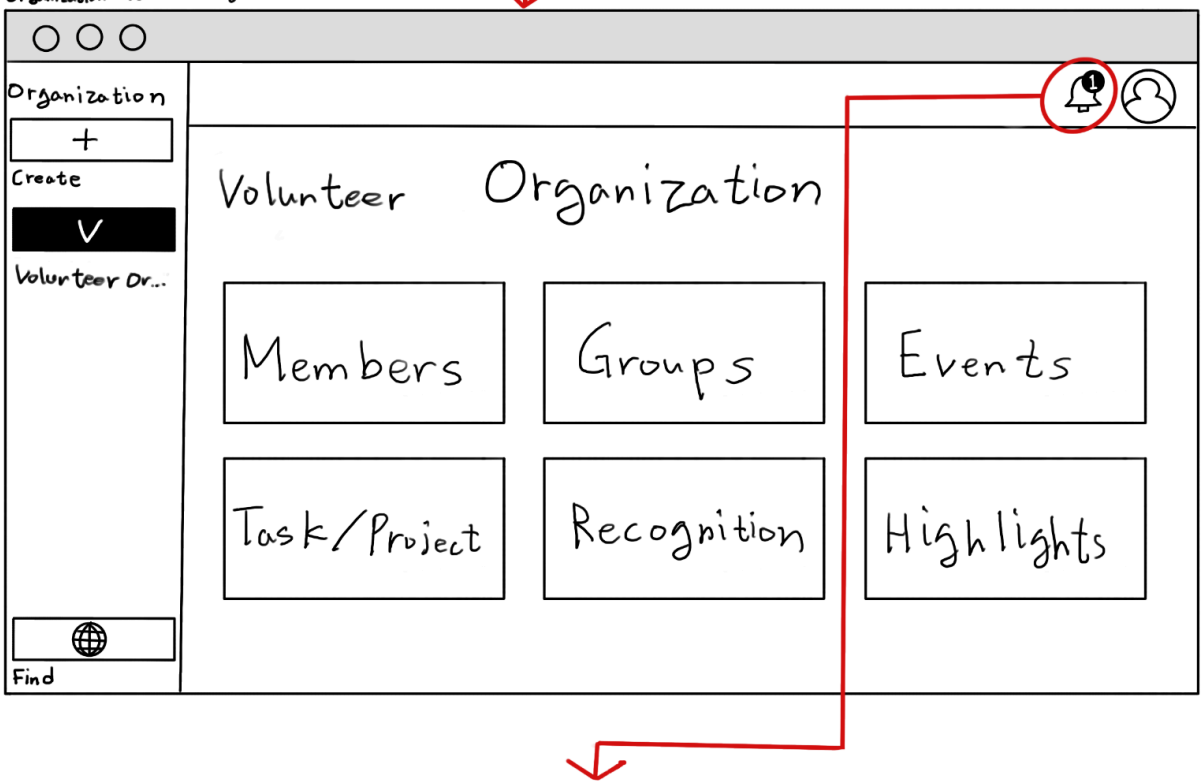
① Start – Member Perspective

Signin Page



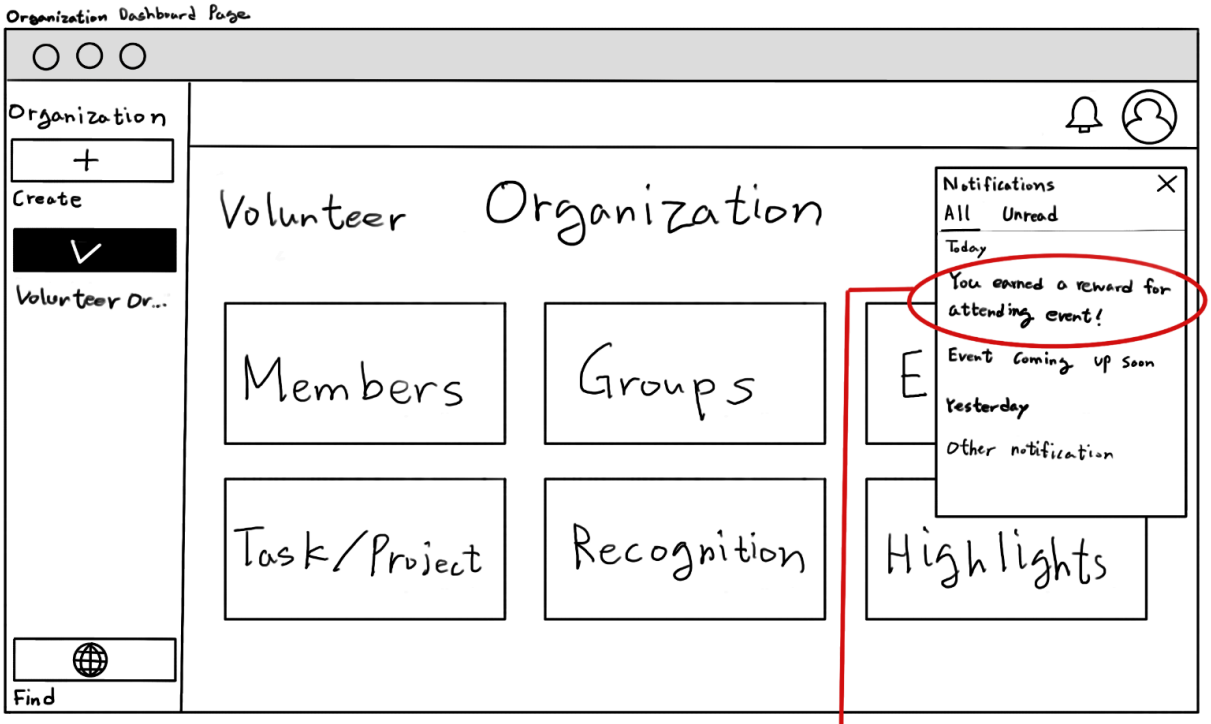
②

Organization Dashboard Page



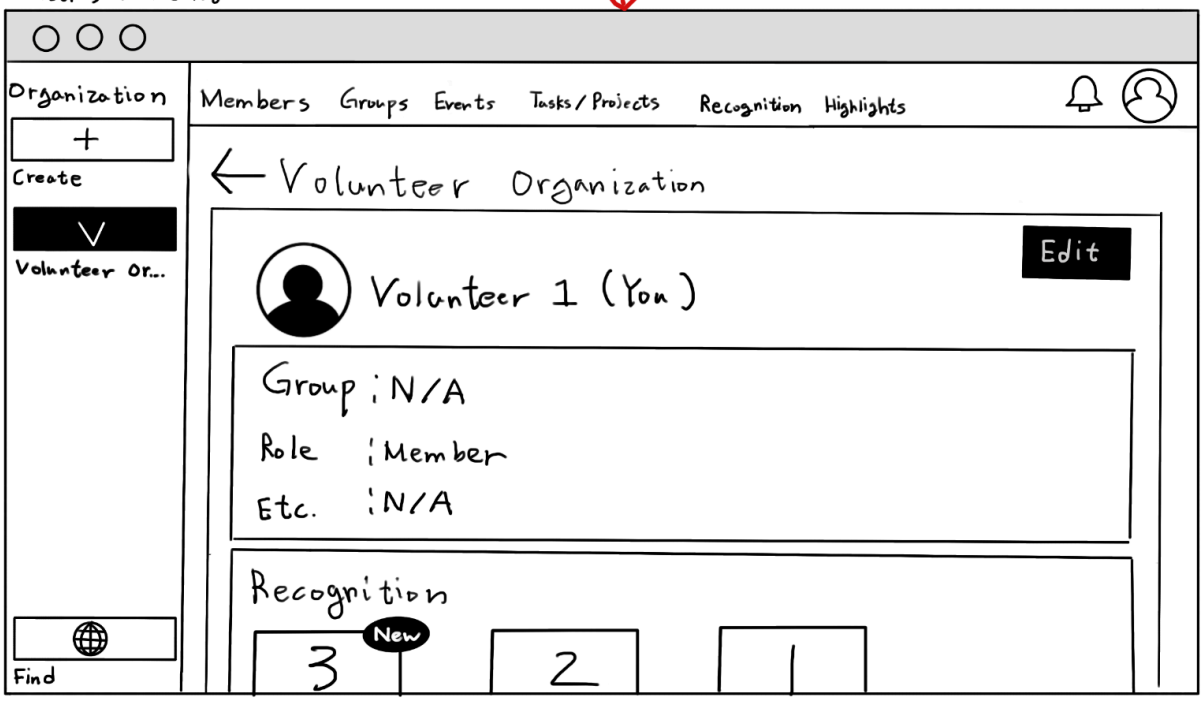
③

Organization Dashboard Page



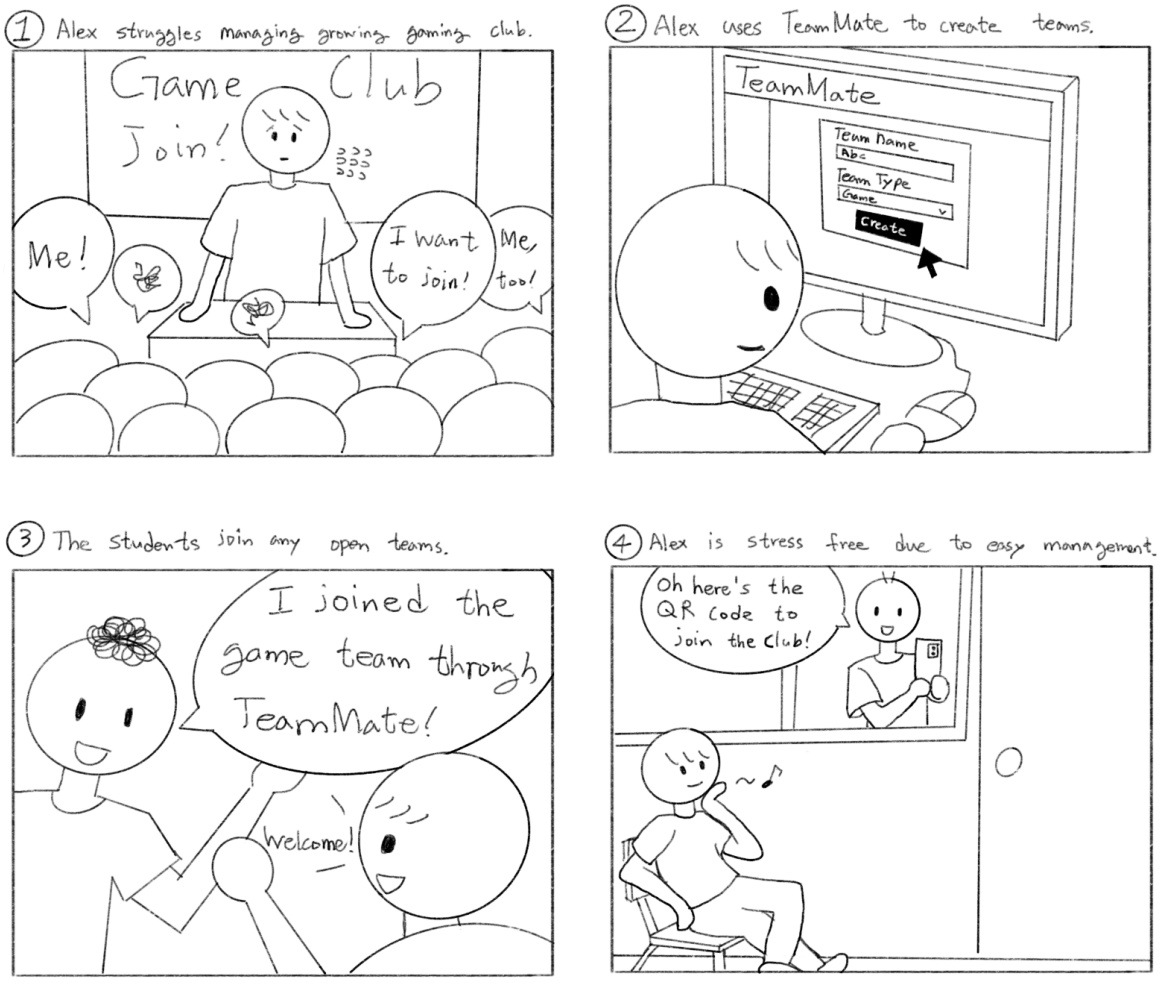
④ End

Member's Profile Page



Use Case #9

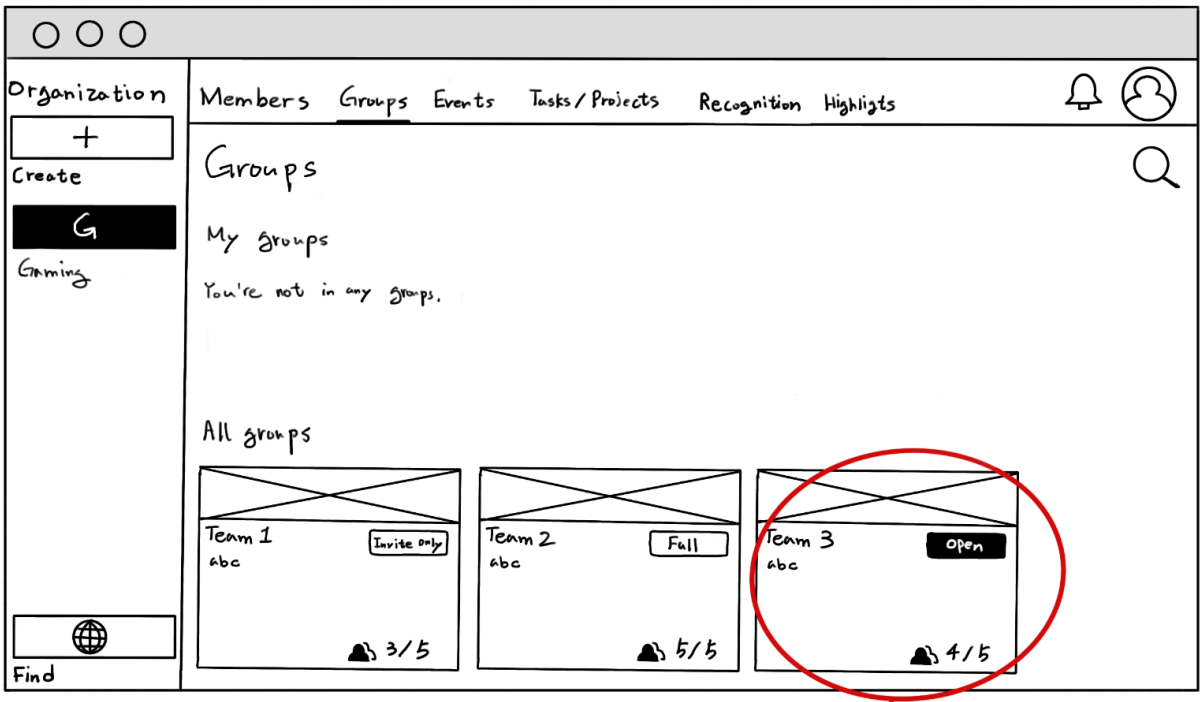
Alex, a teacher running a high school gaming club, struggled to manage the growing number of participants, organize teams, track rankings, and find new opponents. Discovering TeamMate, he created an account and set up the club's organization, enabling students to join and participate through the platform. TeamMate allows Alex to efficiently create teams, update scores, track rankings, schedule matches, and organize tournaments, while students can view their rankings, team assignments, sign up for competitions, and communicate with teammates, keeping everyone informed and motivated.



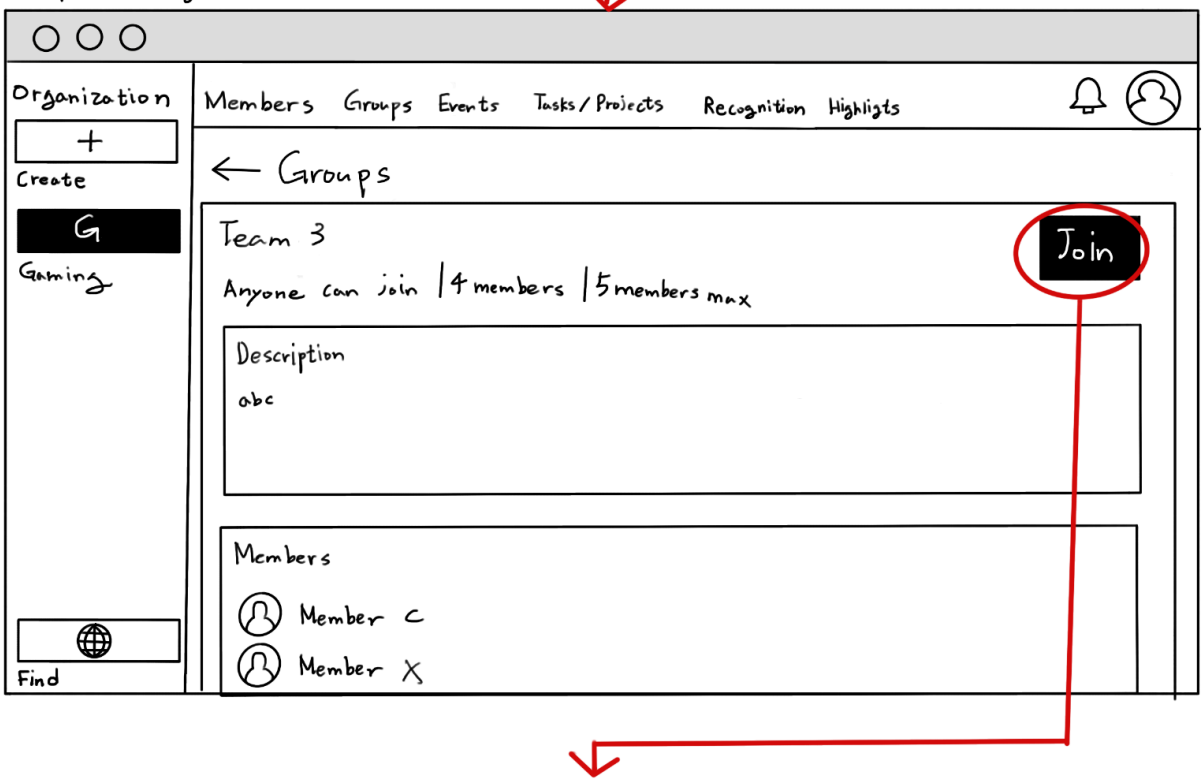
① - Member Perspective

Use case #9

Group Dashboard Page

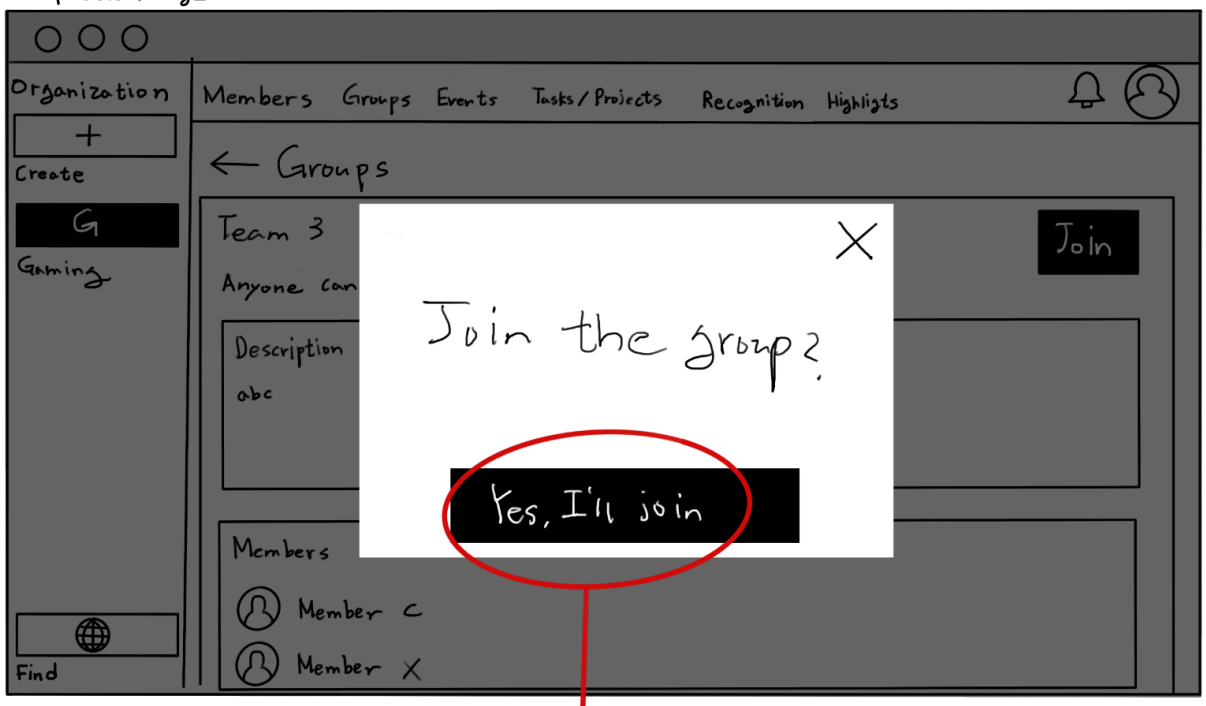


② Group Detail Page

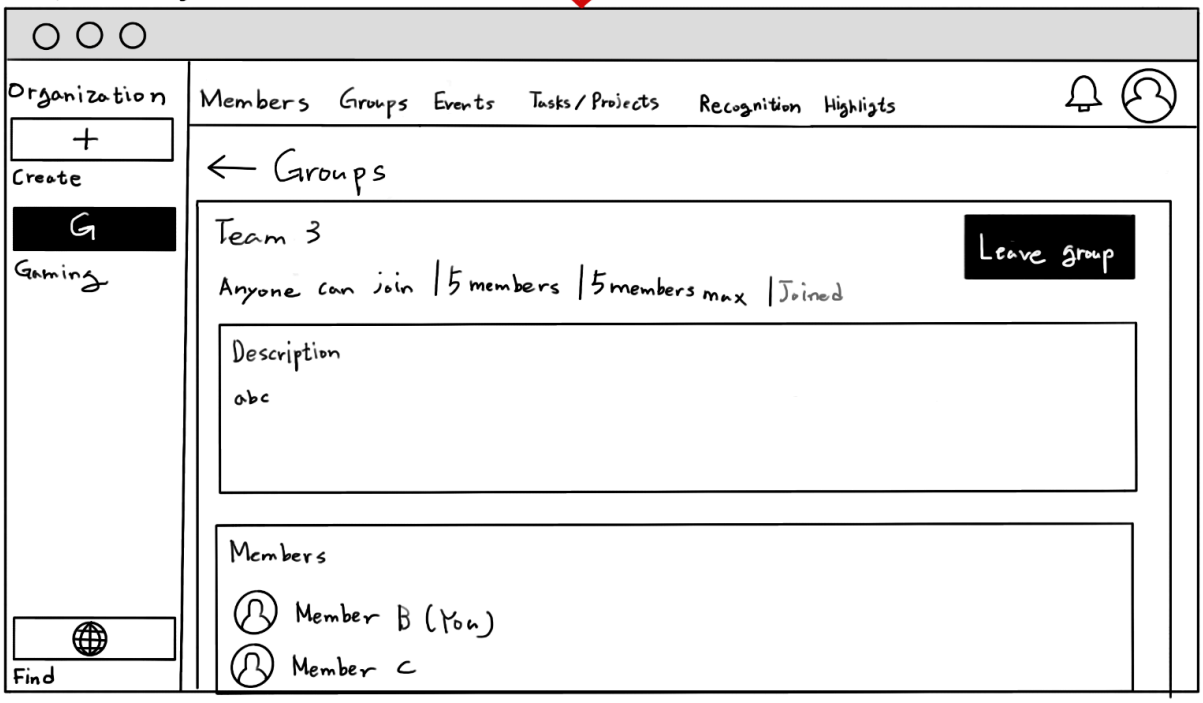


Use case #9

③
Group Detail Page

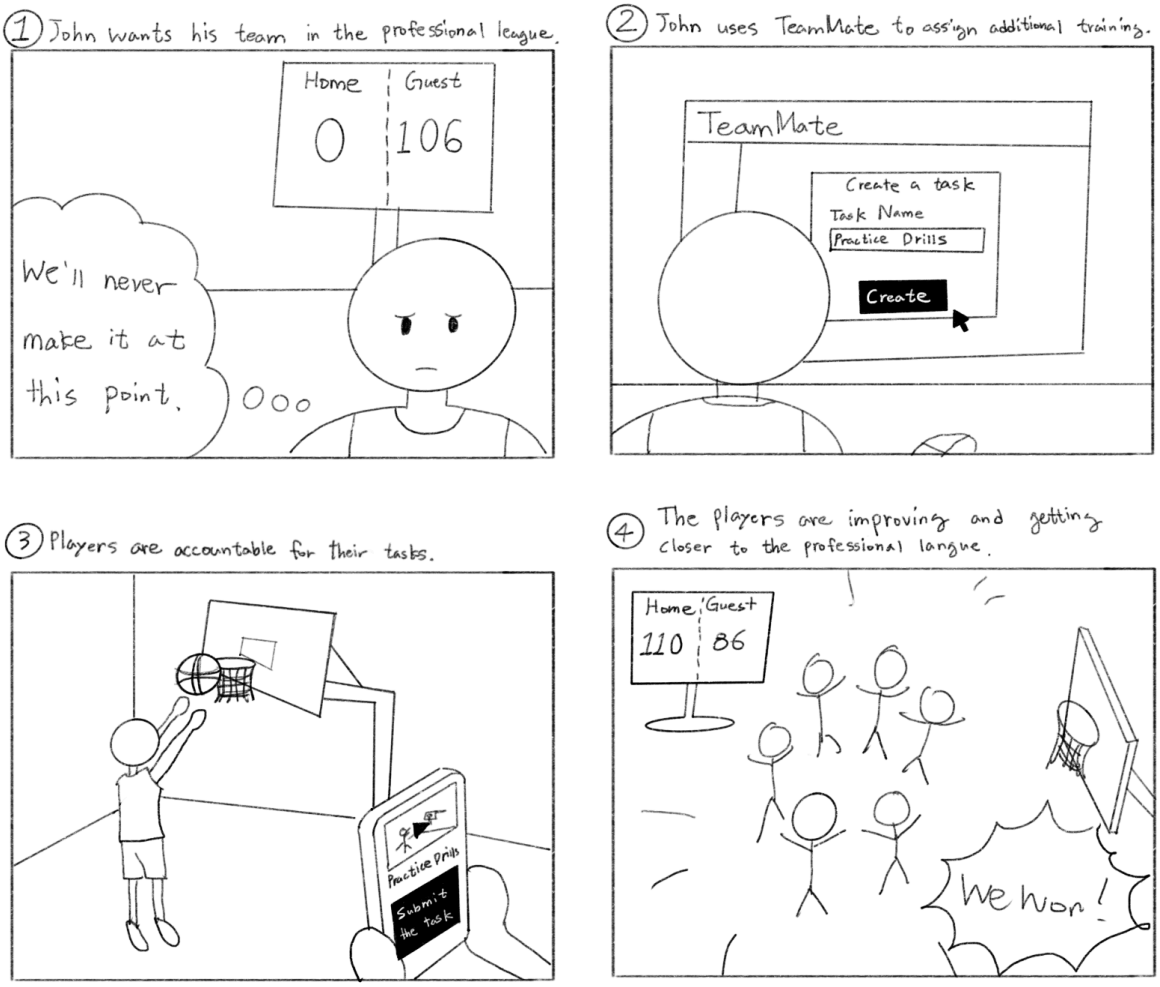


④ End
Group Detail Page



Use Case #10

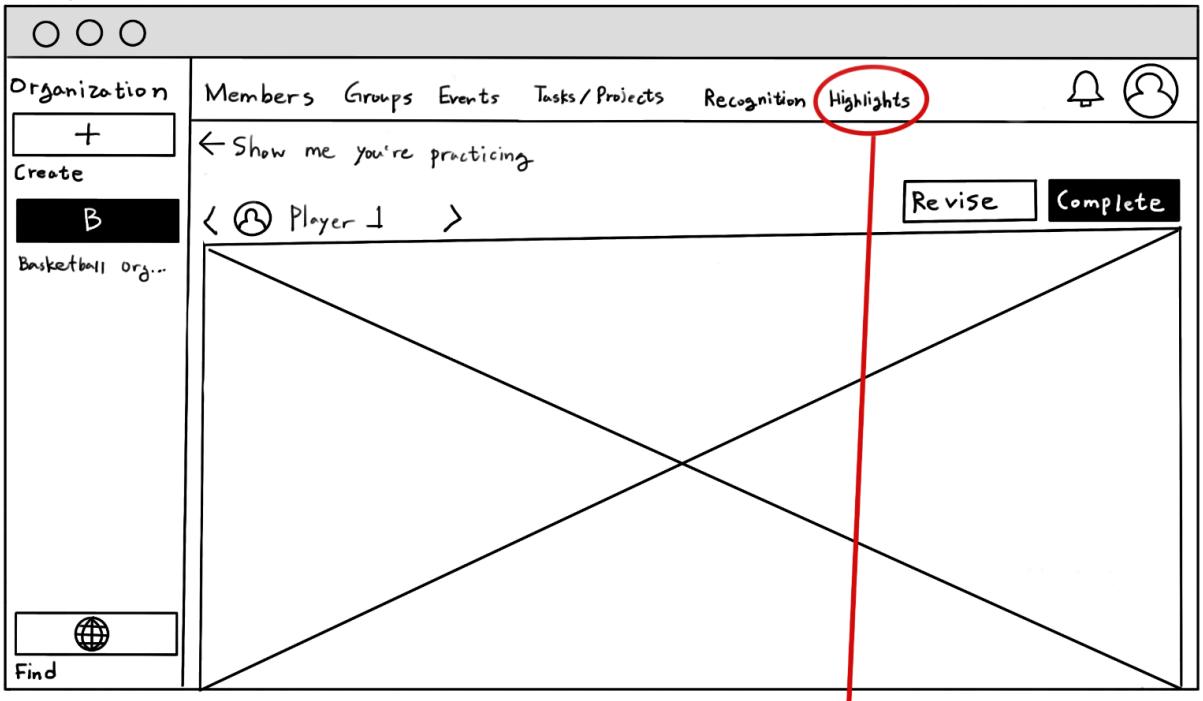
John, a full-time basketball team leader working multiple part-time jobs for funding, struggled to manage team activities, training schedules, physical conditions, and communication. After being introduced to TeamMate, he created an account and set up his organization, enabling teammates to join. The app allows John to assign tasks, track progress, make announcements, and reward active players, fostering a positive and engaged environment while simplifying team management.



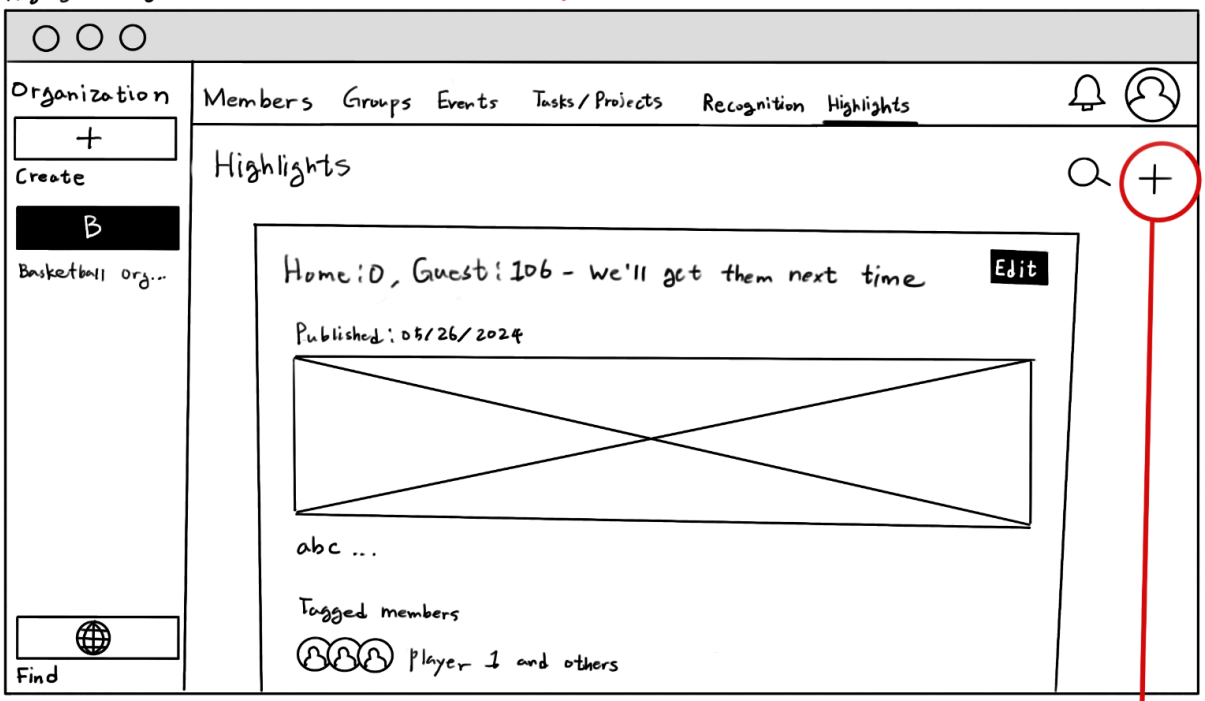
① Start – Leader Perspective

Use Case #10

Task / Project Submission View Page

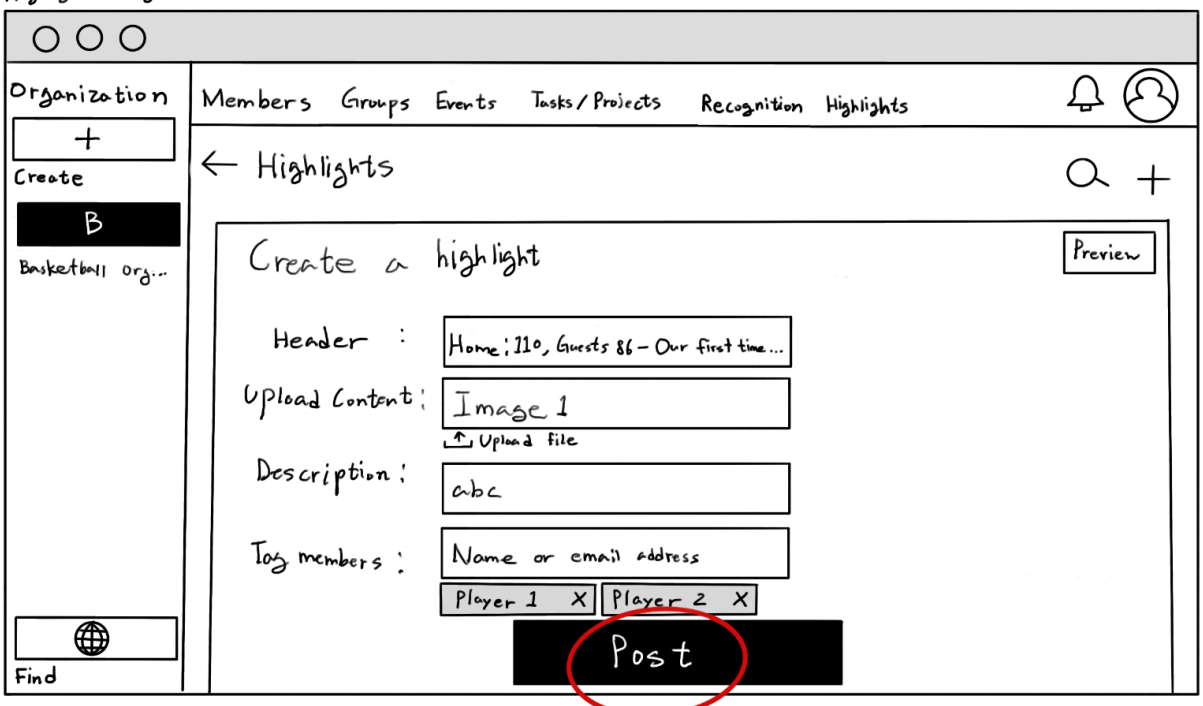


② Highlights Page

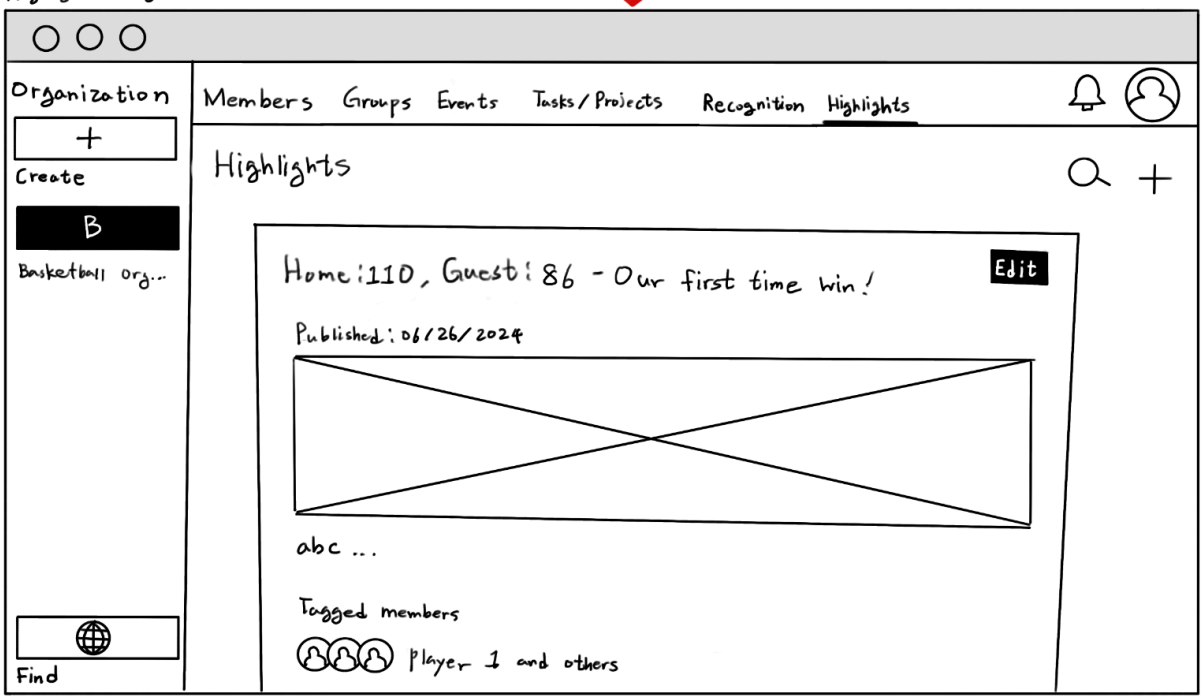


Use case #10

③ Highlights Page

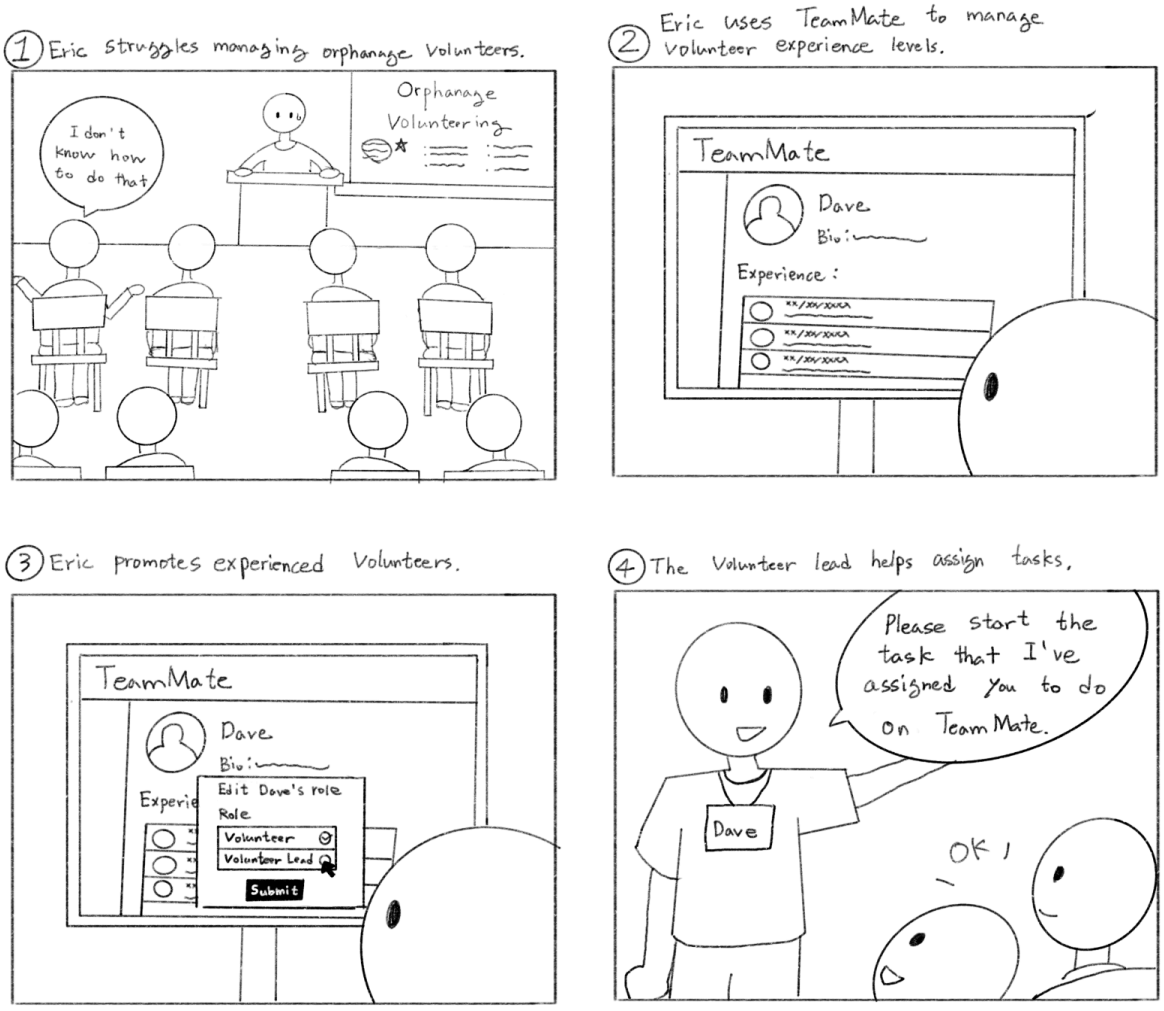


④ End
Highlights Page



Use Case #11

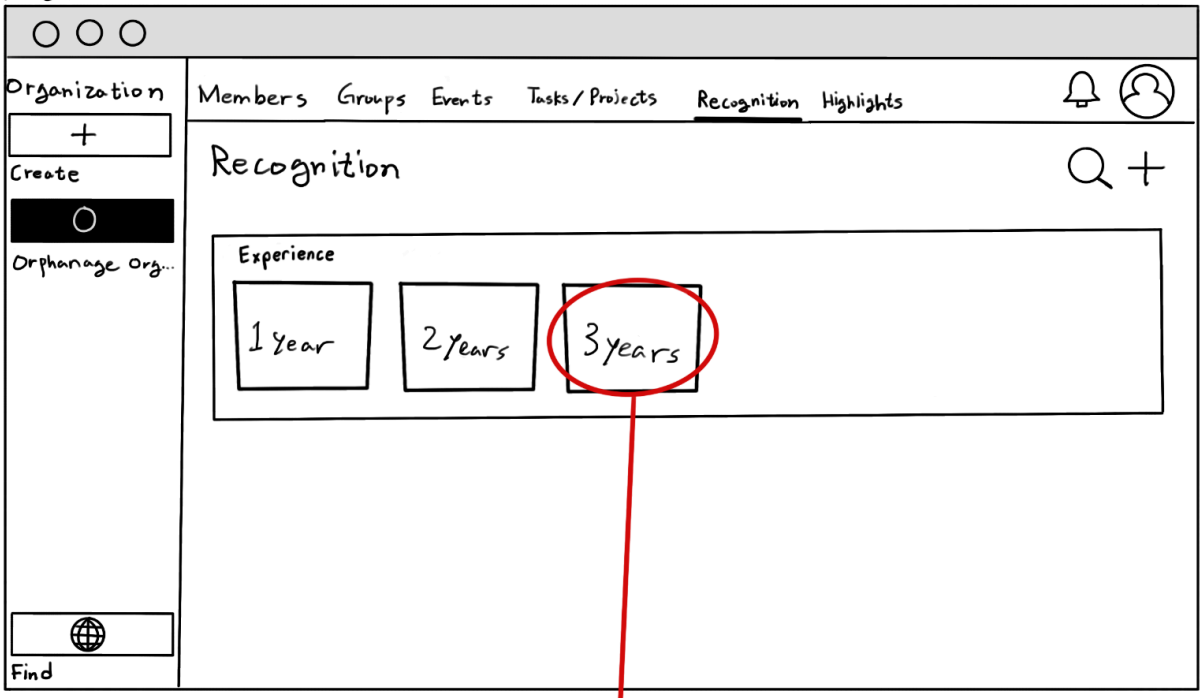
Eric, managing a large orphanage and struggling with limited staff, sought to streamline volunteer management. Introduced to the TeamMate app, he created an account and onboarded his faculty and volunteers to facilitate communication, task assignment, rewards, and a leaderboard. Despite initial convenience, Eric found managing still challenging due to his busy schedule, so he delegated administrative tasks to a trusted faculty member. This reduced Eric's workload significantly, allowing him to focus on overseeing operations and empowering active volunteers as temporary leaders.



① Start - Leader Perspective

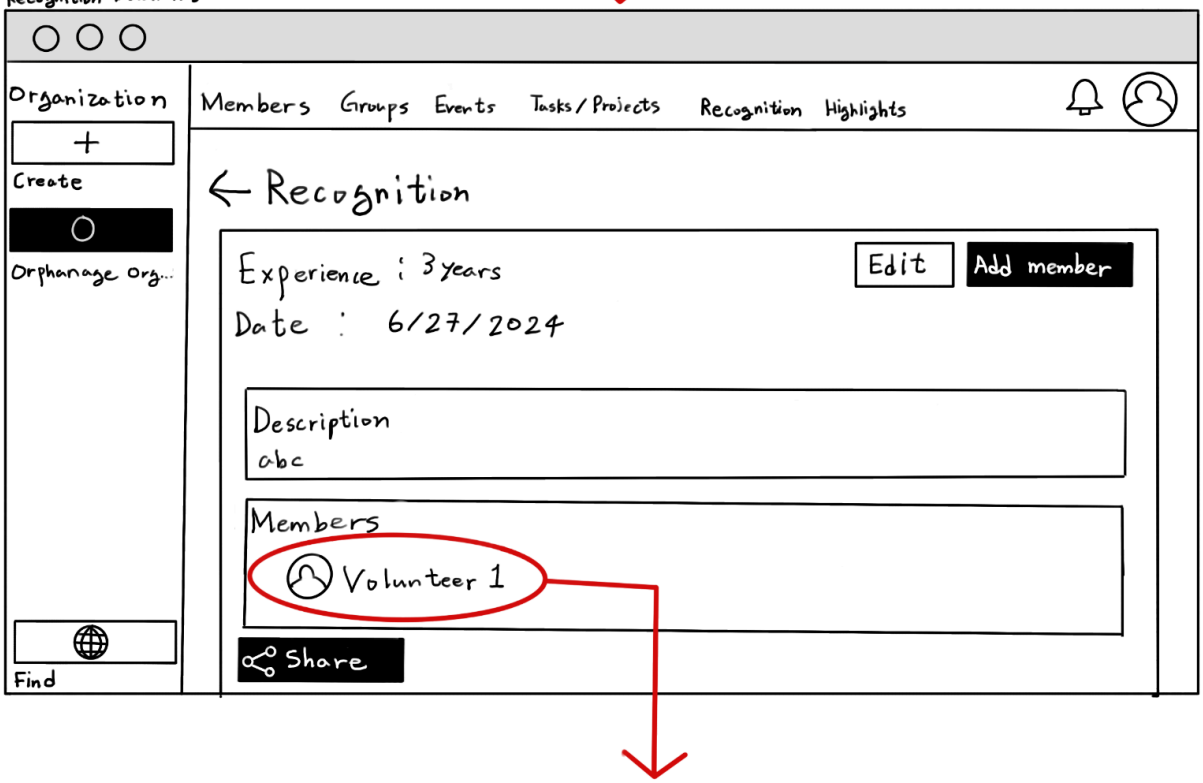
Use Case #11

Recognition Dashboard Page



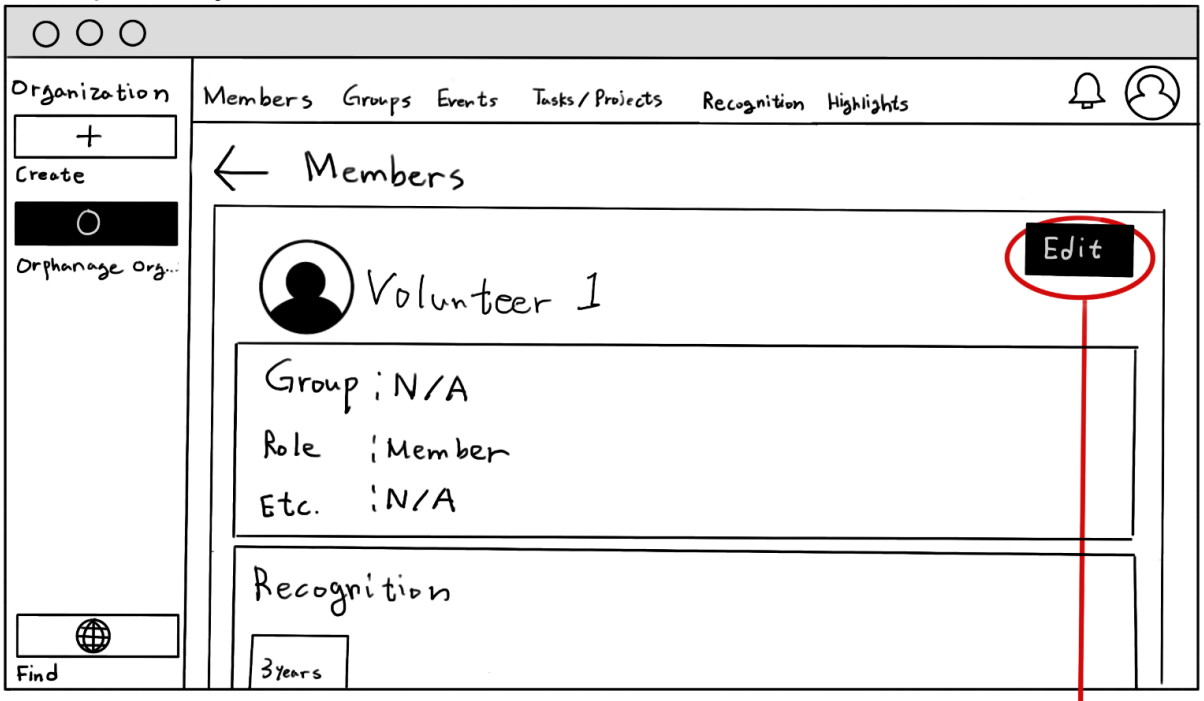
②

Recognition Detail Page



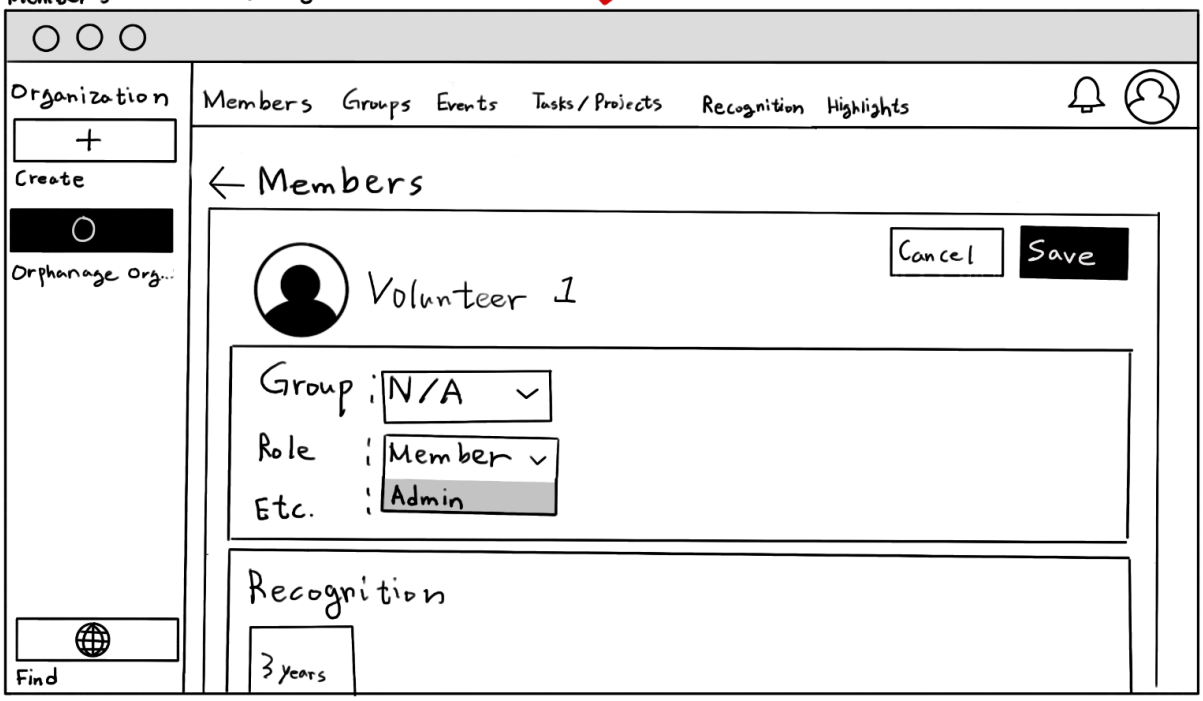
③ Use Case #11

Member's Profile Page



④ End

Member's Profile Edit Page



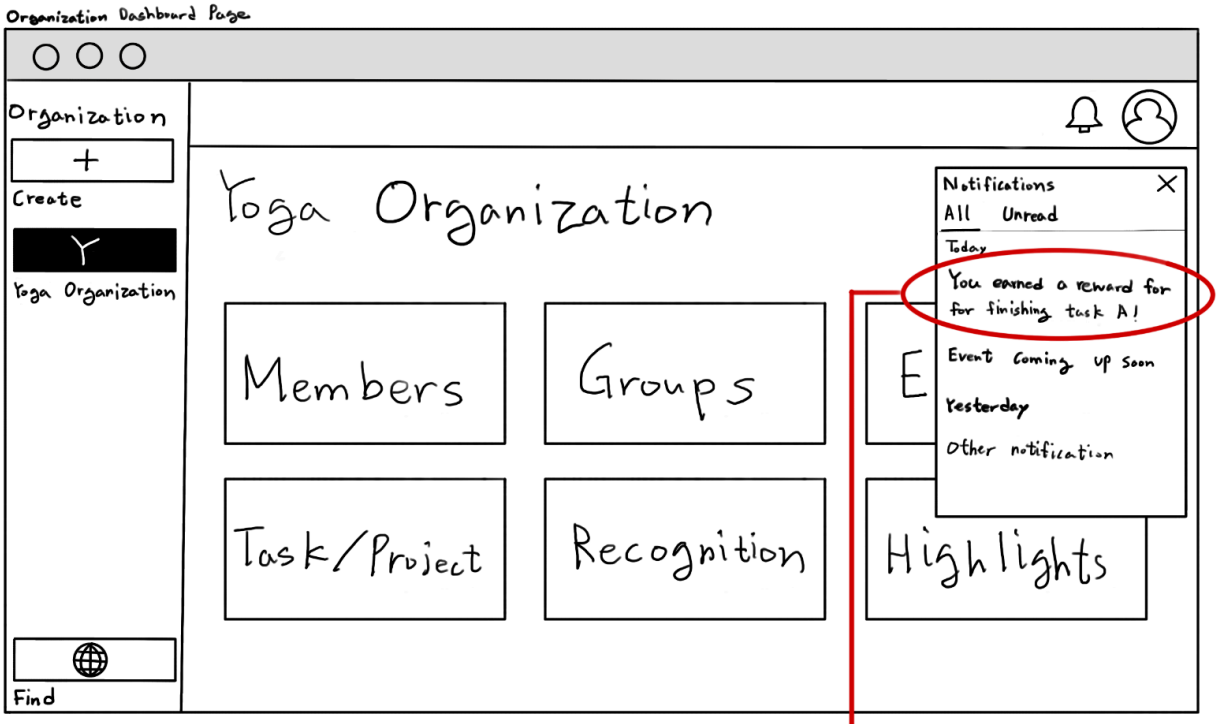
Use Case #12

Clianci, director of a Yoga class in Novato, faces challenges with student motivation and goal achievement. To address this, she uses the TeamMate app to set monthly goals for her students and track their progress and methods used. The app allows her to monitor obstacles hindering progress, such as tardiness or distractions, and track designated days for achieving goals, ensuring accountability and continuous improvement among her students.

① Start - Member Perspective

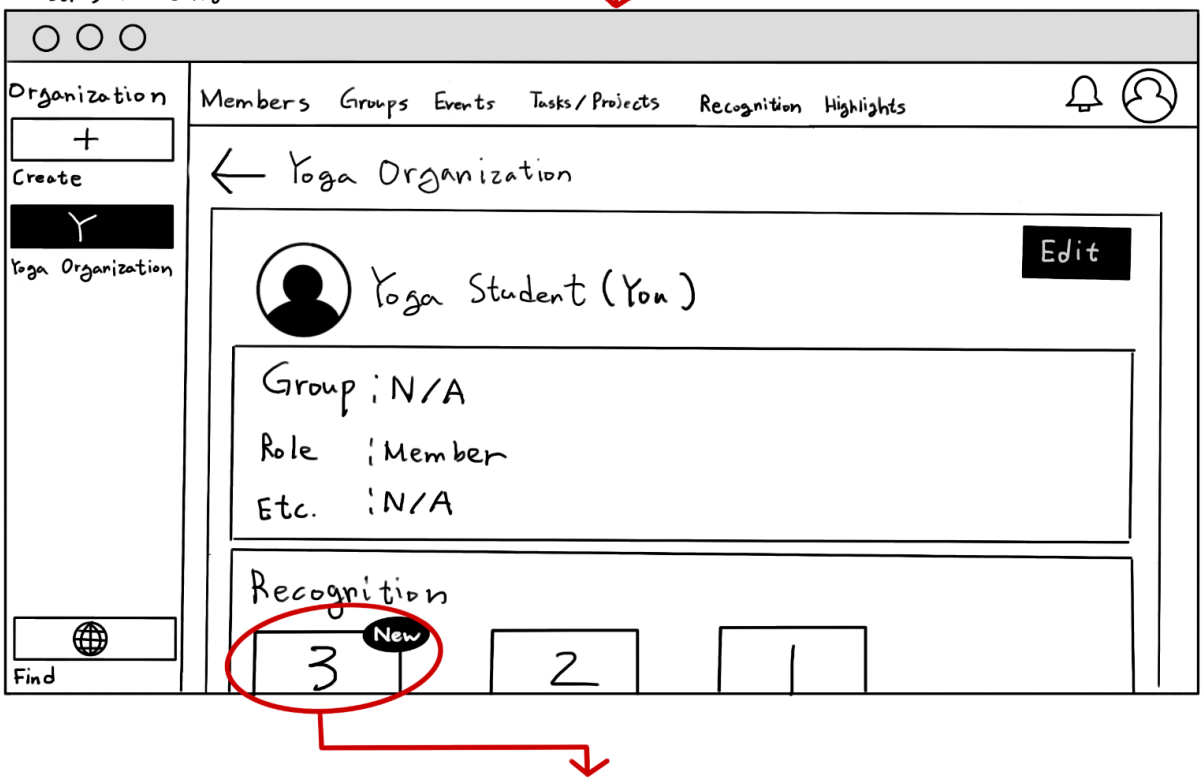
Use Case #12

Organization Dashboard Page



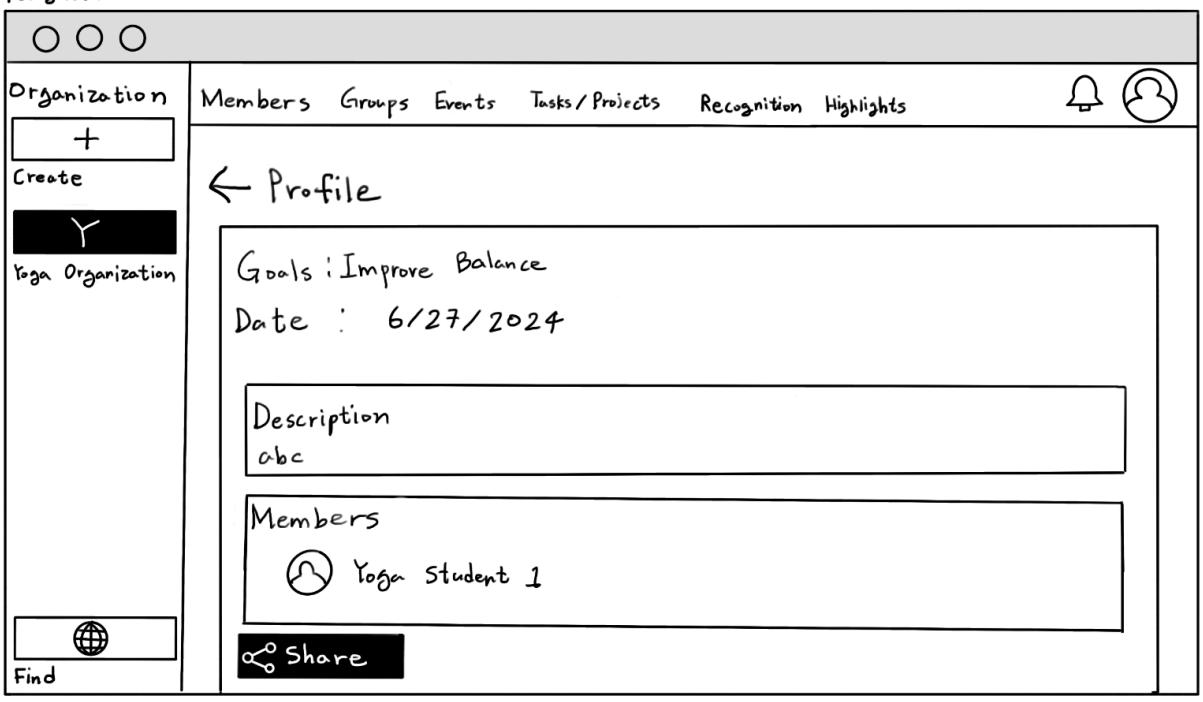
②

Member's Profile Page



③ End

Recognition Detail Page



High Level Database Architecture

DBMS Summary

For this project, we will use MySQL as the DBMS and connect it to an EC2 instance to manage and track the database. MySQL is chosen for our project because of its reliability, scalability, and wide support. It will be an excellent tool for efficiently handling database operations on a cloud-based environment like AWS EC2 we are using.

Functional Requirements

1. **Registered Users (pka: account)**
   - 1.1. Registered users shall at most make one new account.
   - 1.2. A Registered user shall have one ID.
   - 1.3. A Registered user shall have one unique email.
   - 1.4. A Registered user shall have one username.
   - 1.5. A Registered user shall have one password.
   - 1.6. A Registered user shall have at most one profile picture.
   - 1.7. A Registered user shall have one first name.
   - 1.8. A Registered user shall have one last name.
   - 1.9. A Registered user can make many comments.
   - 1.10. A registered user can make many organizations.
   - 1.11. A registered user can be one or many members.
2. **Member**
   - 2.1. A member shall have an ID.
   - 2.2. A member is a registered user.
   - 2.3. A member shall have one first name.
   - 2.4. A member shall have one last name.
   - 2.5. A member shall have one datetime stamp showing their last login.
   - 2.6. A member can receive many qualification instances.
   - 2.7. A member can receive many award instances.
   - 2.8. A member can give many qualification instances.
   - 2.9. A member can give many award instances.
   - 2.10. A member shall have an organization.

2.11. A member shall have many event reminders.

2.12. A member can have a group.

2.13. A member can create a custom section if able.

3. Organization

3.1. An organization shall have an ID.

3.2. An organization shall have a name.

3.3. An organization shall have one owner (member).

3.4. An organization will have one description.

3.5. An organization shall have many groups.

3.6. An organization shall have many events.

3.7. An organization shall have one or many members.

3.8. An organization shall have many custom sections.

3.9. An organization shall have many award types

3.10. An organization shall have many qualification types

4. Award Instance

4.1. An award instance shall have an ID.

4.2. An award instance shall have one member who issued it.

4.3. An award instance shall have one member who received it.

4.4. An award instance shall have one award type.

4.5. An award instance shall have one nomination status.

4.6. An award instance shall have the date of when it was given.

4.7. An award instance shall have a citation.

5. Award Type

5.1. An award type shall have an ID.

5.2. An award type shall have one description.

5.3. An award type shall have an associated image.

5.4. An award type shall have a title.

5.5. An award type shall have one or many award instances.

5.6. An award type will have an associated org

6. Qualification Instance

6.1. An qualification instance shall have an ID.

6.2. A qualification instance shall have one member who issued it.

6.3. A qualification instance shall have one member who received it.

6.4. A qualification instance shall have one qualification type.

6.5. A qualification instance shall have one nomination status.

6.6. A qualification instance shall have the date of when it was given.

6.7. A qualification instance shall have a citation.

7. Qualification Type

7.1. A qualification type shall have an ID.

7.2. A qualification type shall have one description.

7.3. A qualification type shall have an associated image.

7.4. A qualification type shall have a title.

7.5. A qualification type shall have one or many qualification instances.

7.6. A qualification type will have an associated org

8. Events

8.1. An event shall have an ID.

8.2. An event shall have one participating organization.

8.3. An event shall have one title.

8.4. An event shall have one description.

8.5. An event shall have one start datetime.

8.6. An event shall have an event type.

8.7. An event shall have many prerequisites.

8.8. An event shall have one or many event reminders.

9. Custom Section

9.1. A custom section will have an ID.

9.2. A custom section will have one title.

9.3. A custom section will have access levels to edit.

9.4. A custom section will have a “plaintext” section.

9.5. A custom section will have a third party section.

9.6. A custom section will have a member be the creator.

9.7. A custom section will belong to one organization.

10. Comments

10.1. A comment shall have an ID.

10.2. A comment shall have one user associated with it.

10.3. A comment shall have one page type.

10.4. A comment shall have one page key.

10.5. A comment shall have one comment description.

10.6. A comment shall have one date for when it was posted.

11. Event reminder

11.1. An event reminder shall have an ID.

11.2. An event reminder shall have one event.

11.3. An event reminder shall have one member.

11.4. An event reminder shall track if a member went to the event or not (bool).

12. Groups

12.1. A group shall have an ID.

12.2. A group shall have one organization.

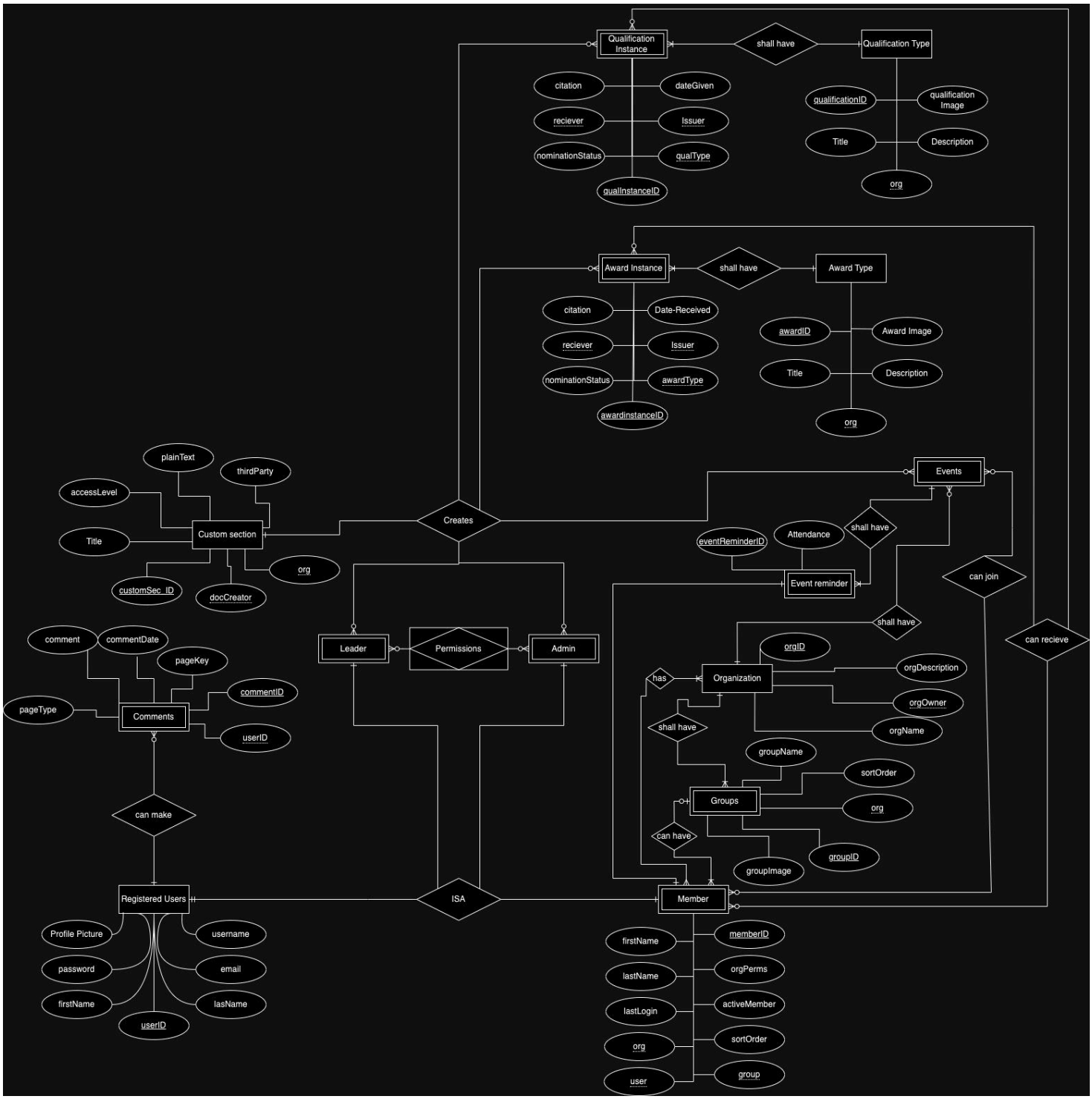
12.3. A group shall have one group name.

12.4. A group shall have a sort order number to organize the groups.

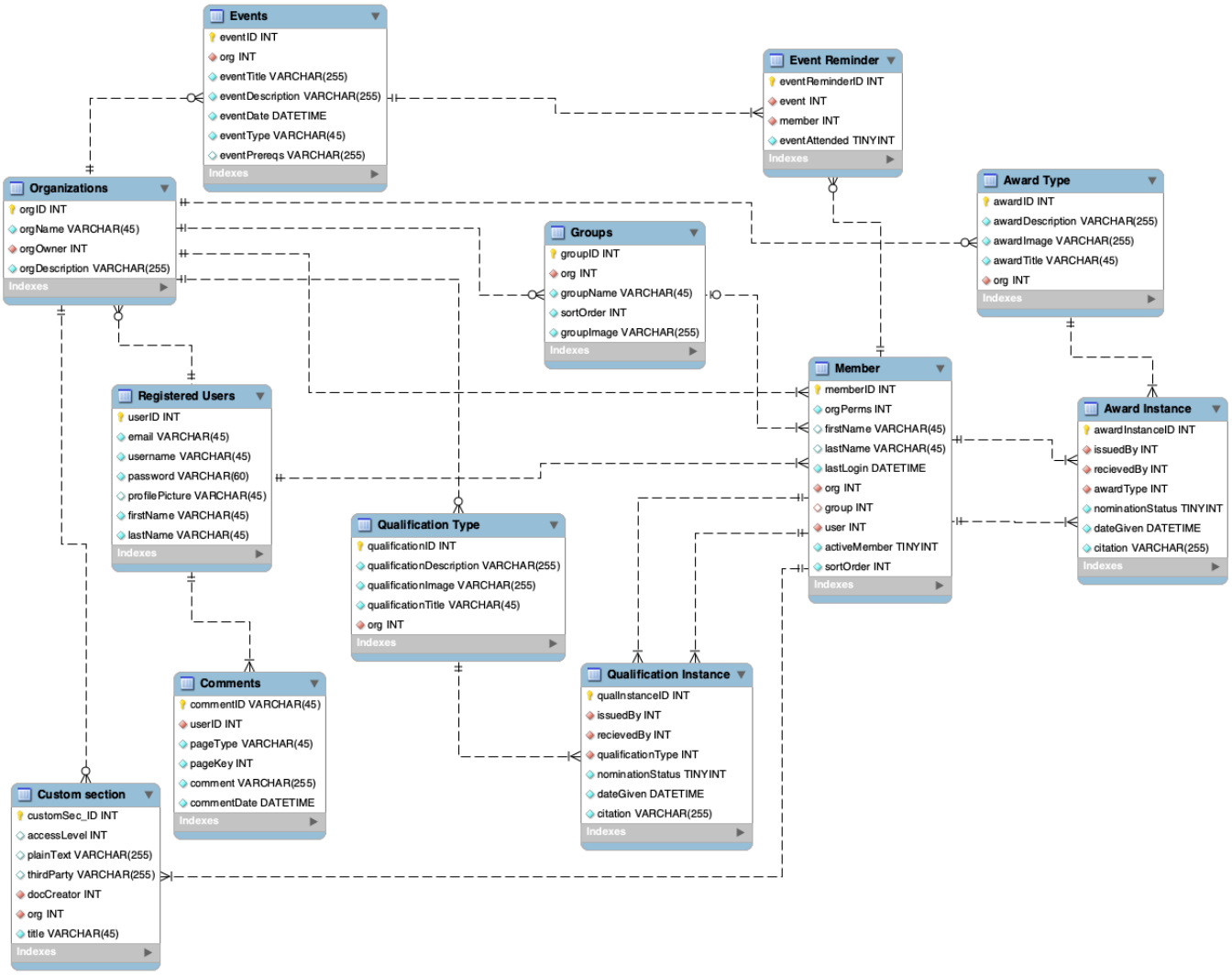
12.5. A group shall have an associated image.

12.6. A group shall have one or many members.

ERD (Entity Relationship Diagram)



EER (Enhanced Entity-Relationship)



Media Storage:

For media storage we decided to use the file systems as opposed to DB BLOBs because we as a group decided it would be easier for us to do due to our familiarity with it. We will use strings to store the file names so that we can route to it.

High Level APIs and Main Algorithms

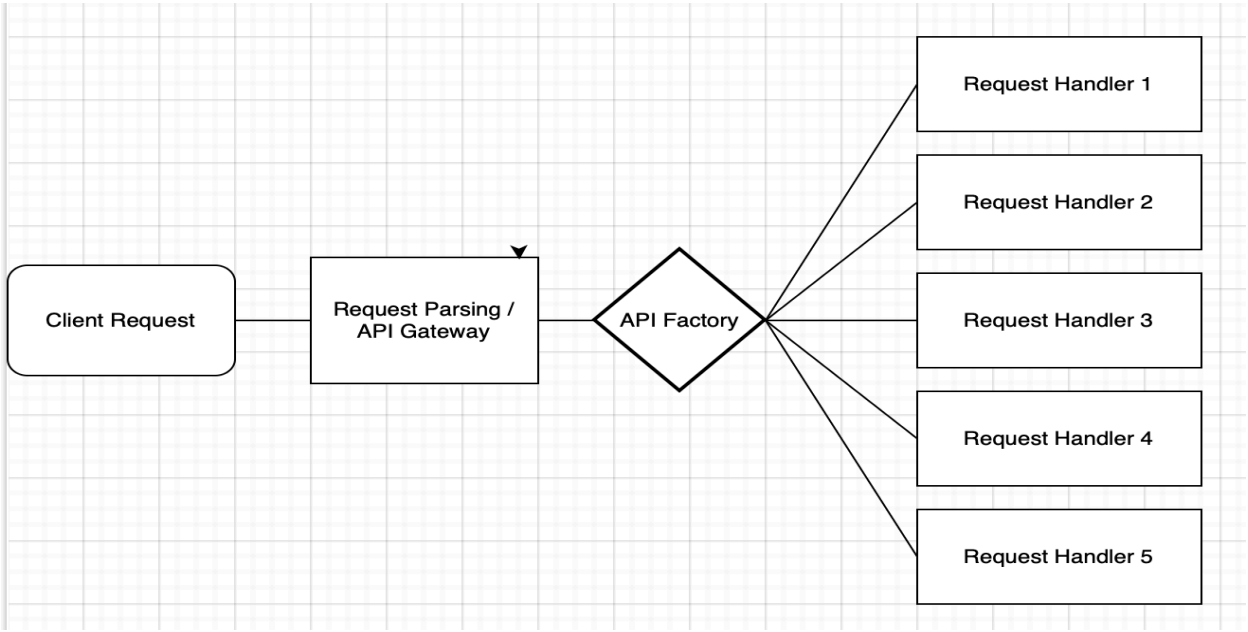
API HIGH LEVEL ARCHITECTURE

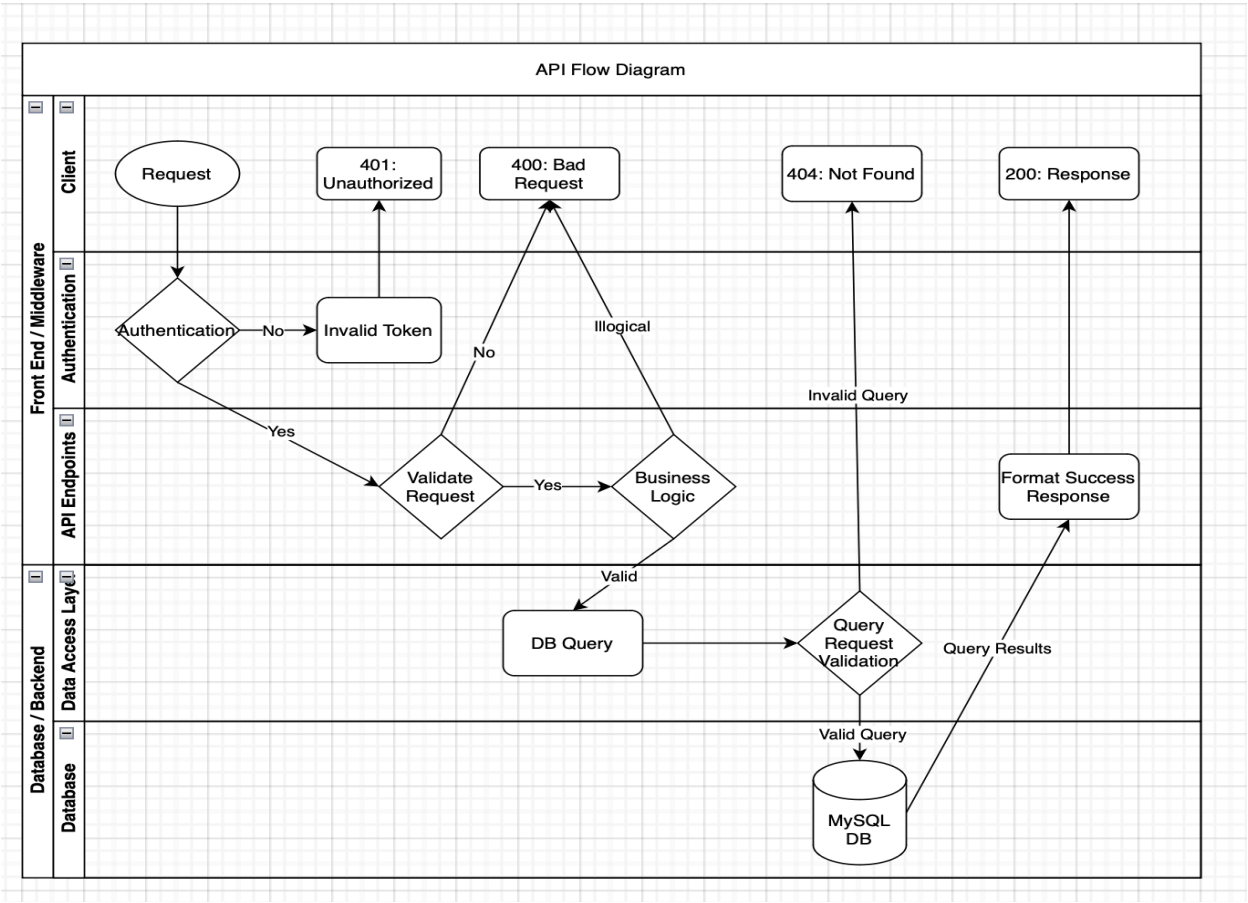
Our app, TeamMate, requires various endpoints for its features. These endpoints handle different functions such as login, create organizations, give awards, admin permissions, etc. Because of this diversity, we decided to implement a control flow that allows us to determine the API response dynamically. We need to be able to add features without interfering with the overall structure of the APIs.

This will be achieved through a monolithic architecture with a single gateway but multiple endpoints. This architecture will direct traffic from different API requests based on the type of business logic they need. By utilizing a factory pattern (to determine where the requests need to go), we can scale the number of endpoints and handlers as needed. This way, adding new features will be straightforward for developers and will not affect performance of the API.

To add another business logic/user function, we would need to add the desired route to the API factory and then from the factory to the request handler that manages all the ‘business logic’ such as database queries.

Below is a control flow of the API illustrating how it handles requests. This method ensures that we validate the request at different stages and return the correct errors based on the context. This multi-stage validation enhances security and accuracy by ensuring that the API passes through several gates before responding to the user. Steps such as authentication, request validation, business logic processing, and query validation prioritize security while ensuring accuracy. By implementing this design into our API architecture, scaling APIs with the same structure becomes easier as these rules are consistently applied.


Client Request
Request Parsing / API Gateway
API Factory
Request Handler 1
Request Handler 2
Request Handler 3
Request Handler 4
Request Handler 5


API Flow Diagram
Front End / Middleware
Database / Backend
Client
Authentication
API Endpoints
Data Access Layer
Database
Request
401: Unauthorized
400: Bad Request
404: Not Found
200: Response
Authentication
No
Invalid Token
Yes
No
Illogical
Invalid Query
Validate Request
Yes
Business Logic
Format Success Response
Valid
DB Query
Query Request Validation
Query Results
Valid Query
MySQL DB

System Design

Our approach prioritizes cost mitigation by maintaining a monolithic structure while leveraging various technologies to achieve scalability and maintainability. The design allows for the deployment of additional monolithic servers as needed, ensuring a balance between operational efficiency and scalability.

We have chosen Amazon Web Services (AWS) for its robust and stable platform, along with its ease of deployment on EC2 instances. AWS provides a comprehensive suite of services that support scalable application deployment. The application will be accessible via mobile and web applications, compatible across multiple platforms including Linux, Windows, Mac, Android, and iOS. This cross-platform compatibility will be achieved through the use of Docker containers, each configured with system-specific requirements.

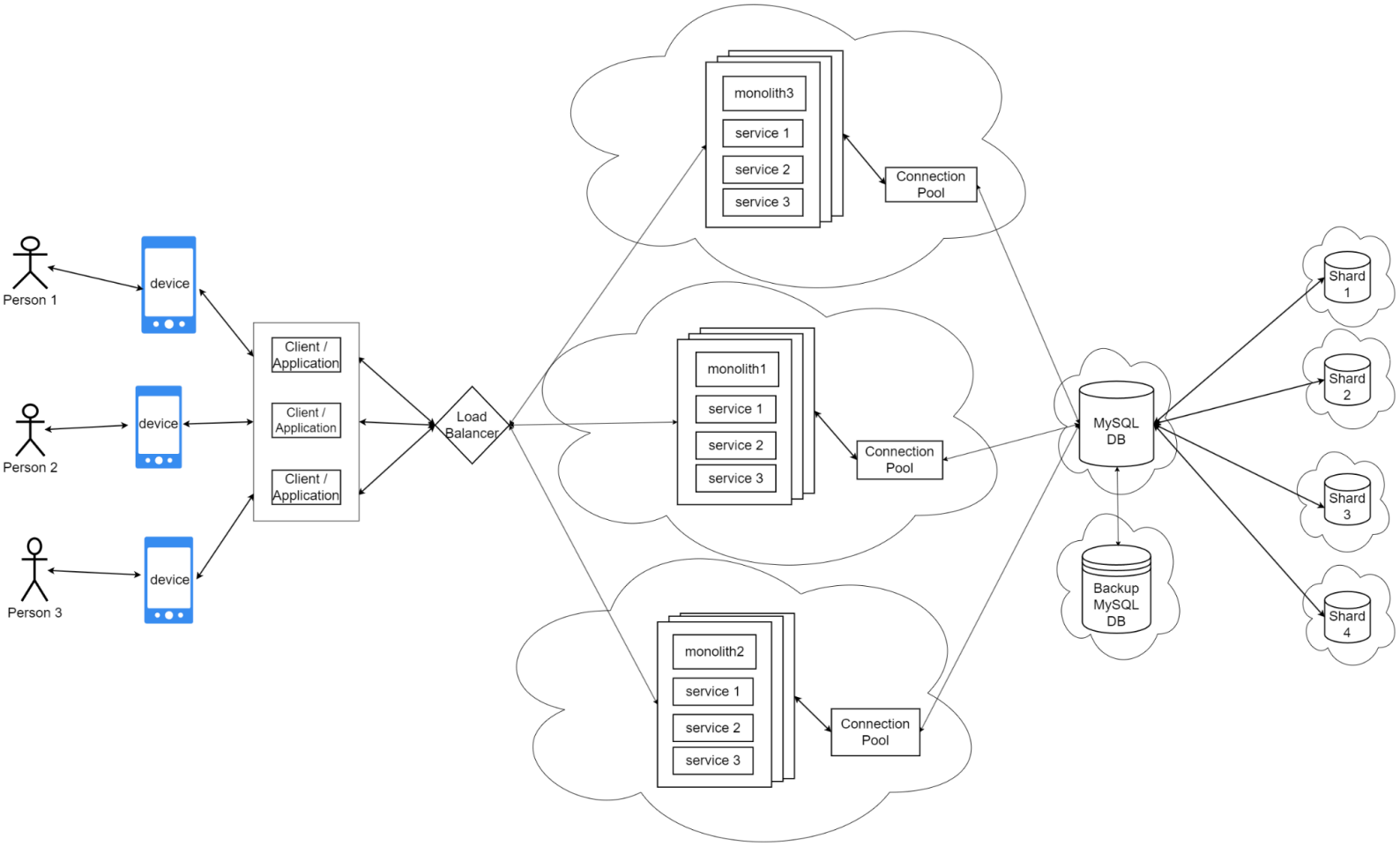
Users will interact with the platform through a React application deployed on EC2. This React front-end will communicate with a back-end server implemented using Express and TypeScript, also hosted on EC2. This setup ensures seamless interaction and maintains cross-language compatibility between the front-end and back-end. Communication between the user, application, and server will utilize TCP connections, with a message queue facilitating request and response management. This message queue decouples the application components, enabling asynchronous communication.

A message queue will precede a load balancer, distributing incoming requests among multiple servers. This ensures high availability, optimal resource utilization, reliability, and performance by preventing any single server from becoming a bottleneck. Horizontal scaling will be implemented by adding additional instances of the monolithic application on separate servers. The load balancer will manage traffic distribution across these instances, enhancing the system's ability to handle increased loads by simply adding more servers as necessary. Vertical scaling involves upgrading the resources (CPU, RAM, storage) of existing EC2 instances. AWS facilitates this by allowing easy upgrades to more powerful instance types, improving the performance of the application on individual servers when horizontal scaling is insufficient.

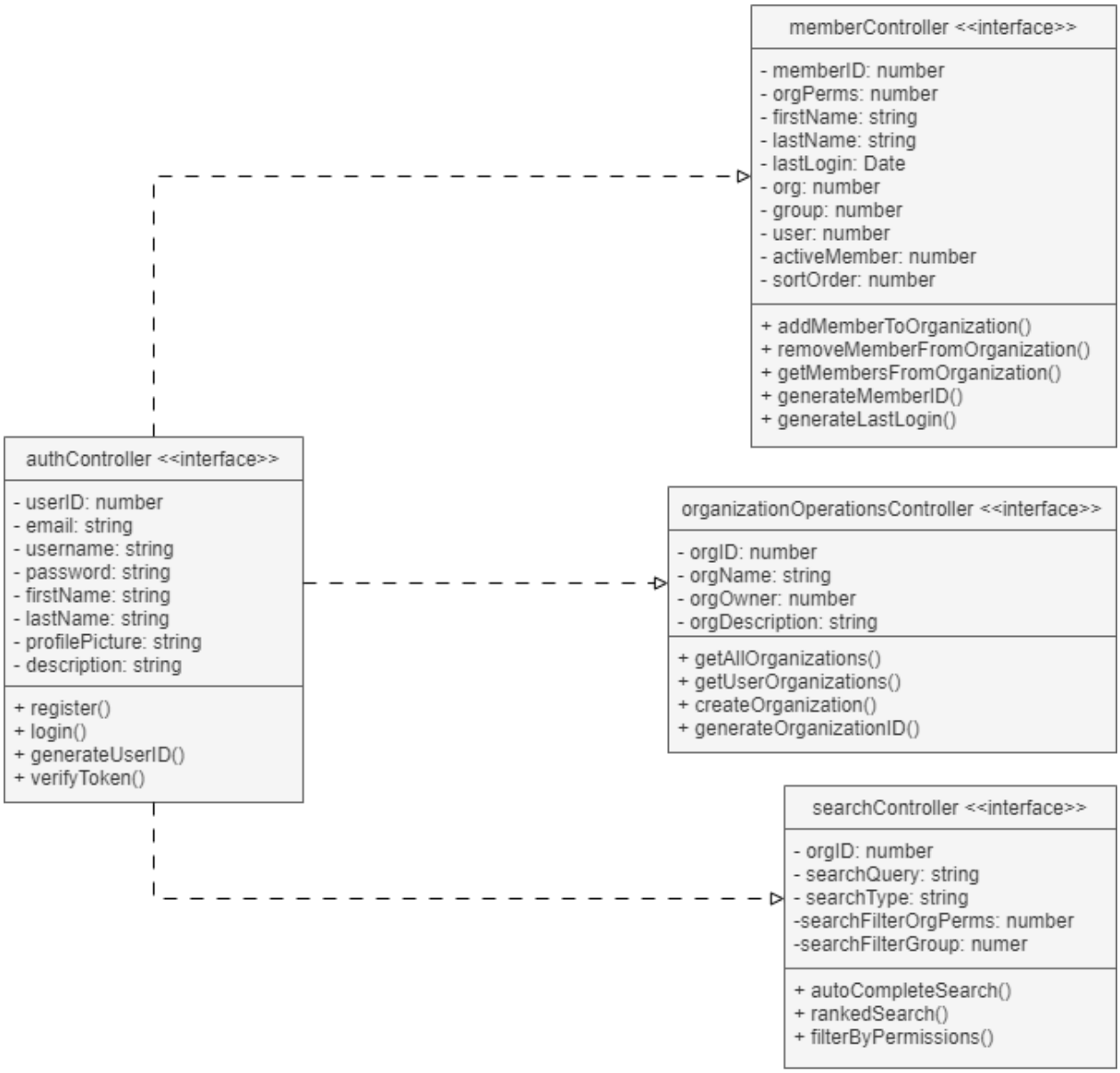
Database operations will be optimized using a connection pool, reusing connections for future database queries to reduce redundancy and manage resources efficiently. As the service grows, the database will be sharded, partitioning large datasets across multiple database nodes. This sharding strategy will increase throughput, expand storage capacity, and ensure high availability in case of a failure. A

backup database will maintain copies of critical data, updated on a regular schedule. This ensures data integrity and availability, preventing data loss in the event of a primary database failure.

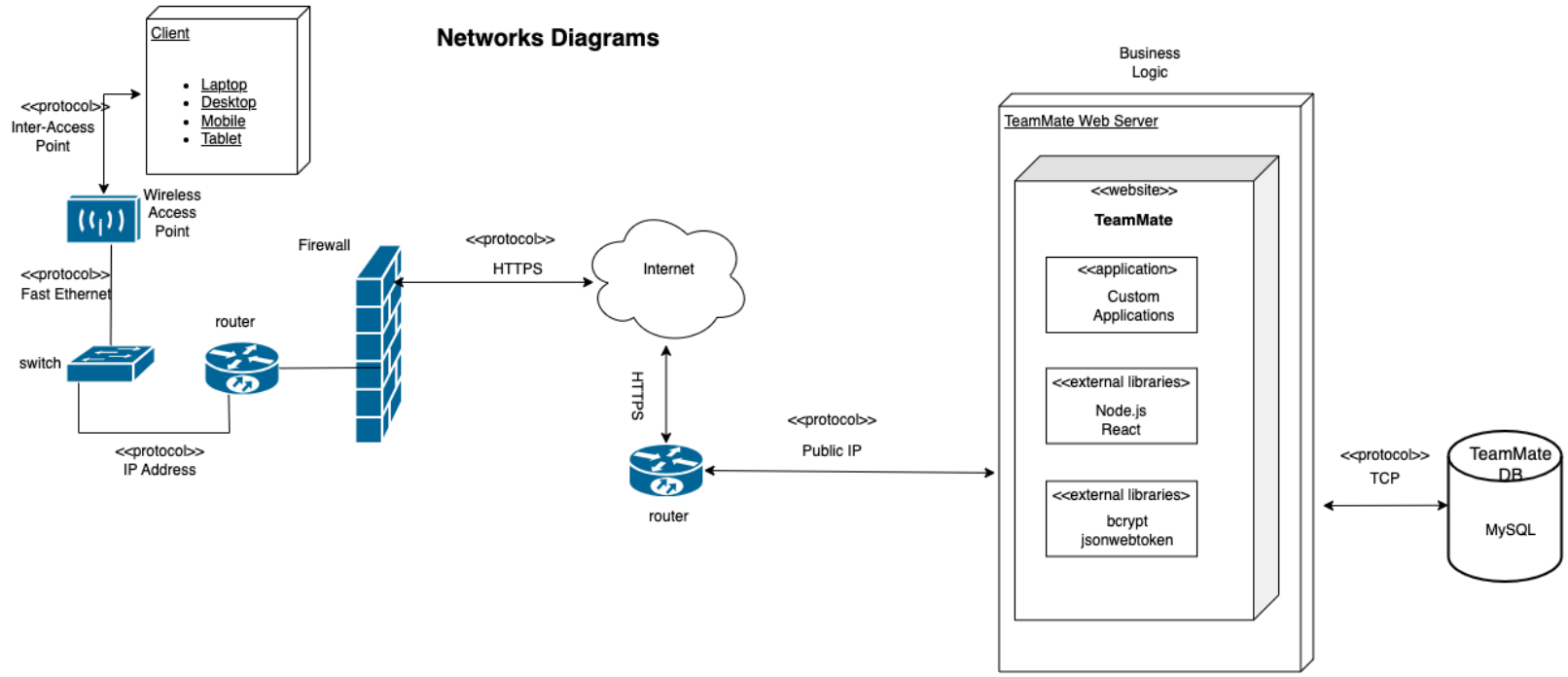
By integrating horizontal and vertical scaling methods, utilizing robust database management, and implementing message queuing for efficient communication, we can ensure that our monolithic architecture remains scalable, cost-effective, and maintainable. This design leverages AWS's capabilities to provide a reliable and scalable infrastructure that meets the needs of our users.



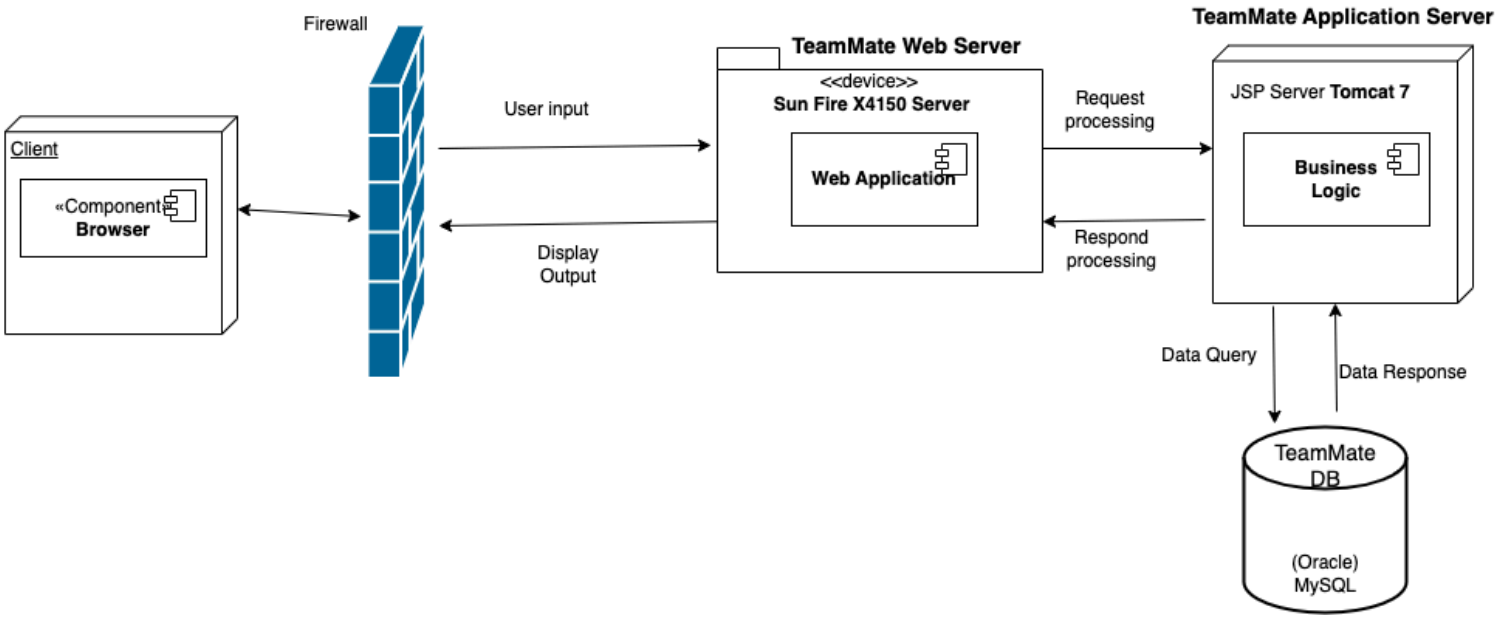
We are primarily using interfaces rather than classes. The interactions between the interfaces and the database tables are identical, and the database tables are well described in the EER (Enhanced Entity-Relationship) diagram. Below are the UML diagrams for our controllers and interfaces:



High Level Application Network and Deployment Design



Deployment Diagram



Actual Risks

Skills Risks:

A good portion of the team is lacking experience and skill in web development in-general, but also some tech specific areas like with TypeScript and React.

> *We are resolving this through having our more experienced team members mentor those who need assistance, and having the inexperienced team members do some learning on their own time to get familiar with the technologies and languages.*

Schedule Risks:

The tight deadlines due to the summer working cycle make it so we may not be able to achieve all of our goals.

> *We are managing this by prioritizing our functional requirements and focusing on essential and good-to-have features, while shelving more opportunistic features and functions until later to ensure we have a viable product first.*

Technical Risks:

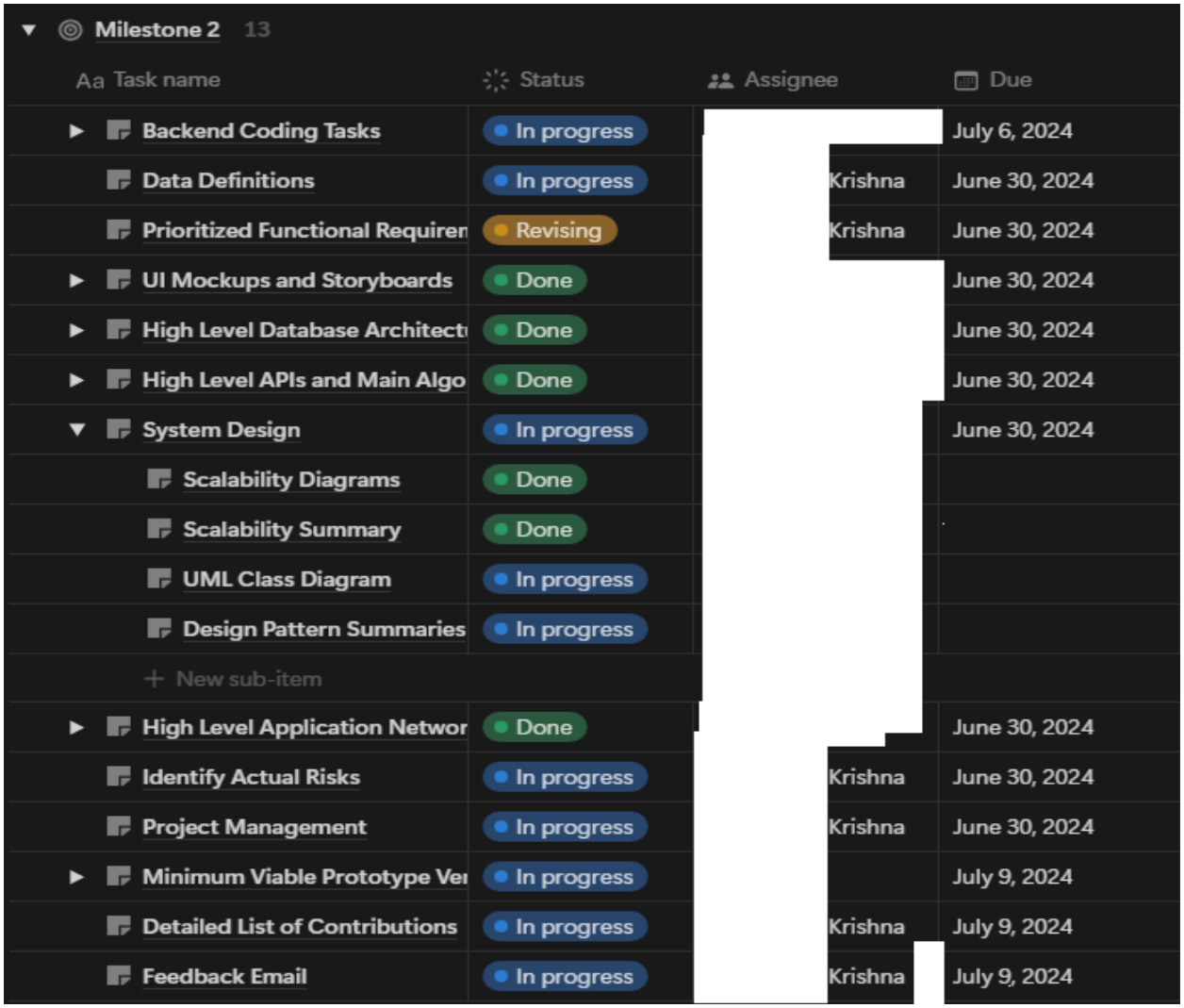
There is some issue preventing us from accessing the RDS instance from our local machines, we currently can only access the RDS instance after SSH-ing into the EC2 instance.

> *We are working on this issue and believe it is related to the AWS security groups between the EC2 and RDS instances. In the meantime however, we have resorted to learning MySQL through SSH and been able to make our prototype work through the terminal. It's not as convenient as MySQL Workbench, but it works for now.*

Project Management

For milestone 2, and going forward, the team was broken up into three sub-teams; these sub-teams were the front-end team, back-end team, and the database team. These teams were assigned tasks in a Zoom meeting, and a complimenting Notion teamspace was set up to reflect the tasks ahead. From there, teams and individuals were able to claim/assign themselves to tasks and update their progress there. The teamspace included internal deadlines to keep the team on-track.

https://www.notion.so/team/



Milestone 2 13

Task name	Status	Assignee	Due
Backend Coding Tasks	In progress		July 6, 2024
Data Definitions	In progress	Krishna	June 30, 2024
Prioritized Functional Requiren	Revising	Krishna	June 30, 2024
UI Mockups and Storyboards	Done		June 30, 2024
High Level Database Architecti	Done		June 30, 2024
High Level APIs and Main Algo	Done		June 30, 2024
System Design	In progress		June 30, 2024
Scalability Diagrams	Done		
Scalability Summary	Done		
UML Class Diagram	In progress		
Design Pattern Summaries	In progress		
+ New sub-item			
High Level Application Networ	Done		June 30, 2024
Identify Actual Risks	In progress	Krishna	June 30, 2024
Project Management	In progress	Krishna	June 30, 2024
Minimum Viable Prototype Ver	In progress		July 9, 2024
Detailed List of Contributions	In progress	Krishna	July 9, 2024
Feedback Email	In progress	Krishna	July 9, 2024

Team Contributions

- **Krishna**
  - Transcribed Prioritized Functional Requirements from team meetings.
  - Transcribed and updated Data Definitions through meetings.
  - Identified Actual Risks for the project, and transcribed it to documentation.
  - Documentation write up for the project management section.
  - Notion Teamspace creation and setup.
  - Migration and setup of project repository to EC2 and RDS instances.
  - Transcribing UML Diagrams from Backend Team's concepts and ideas.
  - Documentation formatting and consistency.
-
  - Created twelve Storyboards.
  - Created twelve Mockups.
  - Leading efforts on Frontend of the MVP.
  - Created the frontend Login page.
  - Created the frontend Registration page.
  - Frontend debugging and revisions.
  - Frontend search component.
-
  - Reorganized GitHub repository.
  - Created System Design Summaries.
  - Created Scalability Diagrams.
  - Created Scalability Summary.
  - Created Design Pattern Summaries.
  - Created Scalability Summary.
  - Created High Level API Summary and API Flow Diagrams.
  - Created Non-Trivial Algorithm Descriptions (Search).
  - Leading efforts on Backend of the MVP.
  - Backend Interfaces for Database Connection.
  - Backend Controllers for Interfaces.
  - Backend Search Component.
  - Created Login Backend.
  - Created Registration Backend.

-
    - Created Database Requirements.
    - Created writeup of DBMS definition.
    - Created Database Design Description.
    - Created Entity Relationship Diagram (ERD).
    - Created Entity Establishment Relationship (EER).
    - Forward Engineered EER.
    - Transcribed Media Storage decision from meeting.
    - Assisted Backend team.
    - Created SQL Queries for backend integration.
-
    - Contributed to ERD attributes and entities.
    - Contributed to frontend dashboard page.
    - Contributed to frontend members page.
    - Contributed to frontend styling.
    - Contributed to frontend sidebar.
    - Contributed to various miscellaneous frontend MVP features.
-
    - Created Networks Diagram.
    - Created Deployments Diagram.
    - Created Frontend Header.
    - Created Frontend Footer.
    - Created additional Frontend Headers.
    - Contributed to frontend styling.
-
    - Contributed to backend interface controllers.
    - Contributed to backend page routing.
    - Contributed to backend APIs
    - API and System Design documentation polishing.
    - Documentation formatting pass.
    - Documentation spell and grammar check pass.
-
    - Contributed to various additional miscellaneous tasks.
    - Contributed to backend APIs.
    - Contributed to backend database queries.
    - Contributed to backend interface controllers.
    - Supervising GitHub commits and branches.

SW Engineering CSC648-848-05 Summer 2024

TeamMate
Personnel Management
Team 05 - Concat

Name	E-Mail	Role
Krishna		Team Lead
		Frontend Lead
		Frontend Dev
		Backend Lead
		Backend Dev
		Database Lead
		Database Dev
		Github Master

Milestone 4
July 30 2024

Version History

Revision	Date Submitted
M1	13 June 2024
M1V1	20 June 2024
M1V2	09 July 2024
M2V1	09 July 2024
M2V2	23 July 2024
M3V1	23 July 2024
M3V2	30 July 2024
M4V1	30 July 2024

Table of Contents

Product Summary

TeamMate is a comprehensive personnel management solution designed for small to mid-sized groups, such as non-profits and community organizations. It consolidates essential management functions into a single, user-friendly platform, enabling easy registration, member tracking, event management, and award issuance.

TeamMate’s standout and superior feature is its Automated Recognition and Progress Tracking System. This system automatically tracks member participation, awards achievements, and updates progress, providing timely recognition and broad overviews of individual and group accomplishments. This holistic view is valuable for performance reviews, planning, setting future goals along with motivating the members of the organization.

Discord focuses on communication and lacks recognition and structured progress tracking, **Airtable** and **Google sheets** require extensive manual setup to achieve similar functionality, whereas **TeamMate** offers an out-of-the-box solution that simplifies administrative tasks. **PERSCOM** provides good tracking but demands more manual effort for recognition, while **Rippling HCM** caters to larger organizations with complex and costly systems, making it less suitable for smaller groups.

Committed Functions

Account

- Users shall register for a new account.
- Accounts shall have unique emails.
- An account shall have a username.
- An account shall have a password.
- An account shall have secure passwords as per modern standards.
- Users shall log into an existing account.
- An account shall create a new organization.
- Accounts shall join an existing organization.
- Accounts shall view all organizations they have joined.

Member

- Member permissions shall be members, group leaders, or admins.
- Admins shall assign member and group leader permissions.
- Admins shall be able to issue awards to members.
- A member's page shall show their information for that organization, along with all attributes, qualifications, awards, etc as a dashboard.

Organization

- An organization shall have an owner.
- An owner can assign other members to be admins.
- An organization shall have a roster page that shows all members in the organization separated by groups.
- Users shall click on any member on the roster to access their member page.

Awards

- Admins shall create award types.
- Admins shall update award types.
- Admins shall delete award types.
- Awards shall have a title.
- Awards shall have a description.
- Admins shall directly assign awards.
- Award instances shall have a date received.
- Award instances shall state the issuing member.

Qualifications

- Admins shall create qualifications types.
- Admins shall update qualifications types.
- Admins shall delete qualifications types.
- Qualifications shall have a title.
- Qualifications shall have a description.
- Admins shall directly assign qualifications.
- Members shall view individual qualifications.
- Qualification instances shall have a date received.
- Qualification instances shall state the issuing member.

Events

- Admins shall create events.
- Group leaders shall create events.
- Admins shall assign members to an existing event.
- Group leaders shall assign members to an existing event.
- Members shall be able to join an event.
- Events shall have a title.
- Events shall have a description.
- Events shall have a start datetime.

Application

http://ec2 compute-1.amazonaws.com:3000/

Usability Test Plan

Use/Test Case 1 (Organization Creation)

Objective:
Create an organization. While conducting this test, the user will also go through the registration and login process. This feature is being tested to determine if the user is intuitively able to navigate the home page and sidebar of the dashboard.

Description:
The horizontal prototype already supports these features, the user will start from the website's home page. The intended user is a first time user of the website, with no prior context as to what it is. We are measuring the time and mouse movements of the user, to see how long it takes the user to complete the test, and where all they navigate while completing the test, in order to determine if the website design makes intuitive sense to a first time user.
http://ec2 compute-1.amazonaws.com:3000/

Use/Test Case 2 (Group Creation)

Objective:
Create a group. This feature is being tested to determine if the user is intuitively able to navigate the header and corresponding pages intuitively.

Description:
The beta prototype will be supporting this feature, the user will start from the dashboard after logging in. The intended user is a new user that has just created an organization. We are measuring the time taken and mouse movements of the user to determine if they can quickly navigate to the required pages intuitively, to validate our dashboard design.
http://ec2 compute-1.amazonaws.com:3000/organizationDashboard/Testing%20Org

Use/Test Case 3 (Award Creation)

Objective:
Create an award. This feature is being tested to determine if the user is able to find the awards page and create an award.

Description:
The beta prototype will be supporting this feature, the user will start from the dashboard after logging in. The intended user is a relatively new user that has created an organization. We are measuring the time taken and mouse movements & page navigation to determine if they are able to figure out that awards are part of the recognitions tab, and can use the creation form to make an award instead of certification.
http://ec2 compute-1.amazonaws.com:3000/organizationDashboard/Testing%20Org

Use/Test Case 4 (Event Creation)

Objective:
Create an event. This feature is being tested to determine if the user is able to navigate to the events page and follow through the event creation form.

Description:
The beta prototype will be supporting this feature, the user will start from the dashboard after logging in. The intended user is a relatively new user that has created and organization. We are measuring the time taken and mouse movements to determine if they are able to intuitively navigate from the dashboard to the event page and the events creation form and complete it.
http://ec2 compute-1.amazonaws.com:3000/organizationDashboard/Testing%20Org

Use/Test Case 5 (Member Operations)

Objective:
Add a member. This feature is being tested to determine if the user can figure out how to add members to their organization and assign them attributes.

Description:
The beta prototype will be supporting this feature, the user will start from the dashboard after having created a few attributes. The intended user is a user that has explored the dashboard for some time and created various attributes. We are measuring the time taken and mouse movements to determine if they are able to intuitively navigate to the members page and invite another user to their organization.
http://ec2 compute-1.amazonaws.com:3000/organizationDashboard/Testing%20Org

Test Table

Test/Use Case	% Completed	Efficiency	Errors	Comments
Org Creation	100%	4-5 Minutes	None	Older users unable to figure out that the create bar on the sidebar is how you make an org. Inflated efficiency due to users coming up with registration info.
Group Creation	100%	<1 Minute	None	Creation complete.
Award Creation	100%	<1 Minute	None	Creation complete.
Event Creation	100%	<1 Minute	None	Creation complete.
Member Add	100%	1-2 Minutes	None	It's not clear how to find a user's ID to add into the org. Can also add duplicate members to org.

Questionnaire

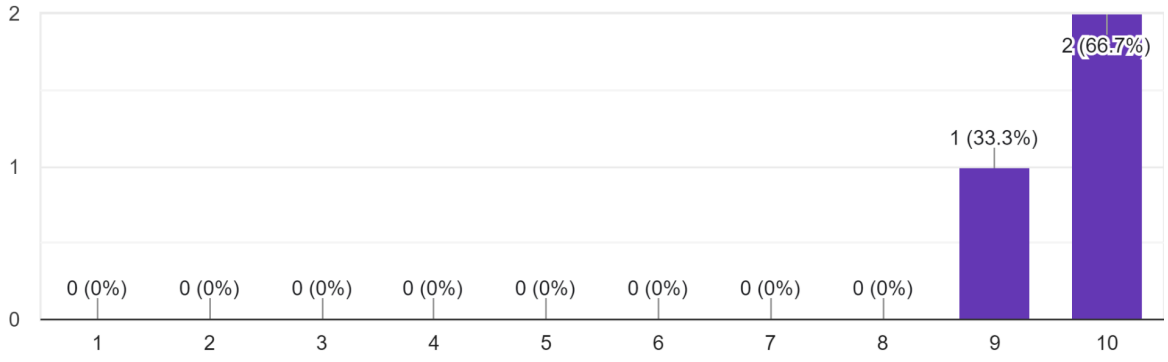
1. I found it easy to navigate from the home page to the dashboard.
2. I found it easy to create an account and login.
3. I was able to intuitively navigate the dashboard to create an organization.
4. The dashboard header was difficult to understand.
5. The group's page layout confused me.
6. I was able to intuitively navigate the dashboard to create a group.
7. I struggled to find the awards page.
8. The layout of the awards page was consistent with the other pages.
9. I was able to intuitively navigate the dashboard to create an award.
10. The event page’s layout was consistent with the rest of the application.
11. The event details page layout contained all the information I wanted it to.
12. I was able to intuitively navigate the dashboard to create an event.
13. The layout of the members page made intuitive sense.
14. Using the add button in the members page was confusing.
15. I was able to intuitively navigate the dashboard to add a member.

https://forms.gle/

Responses

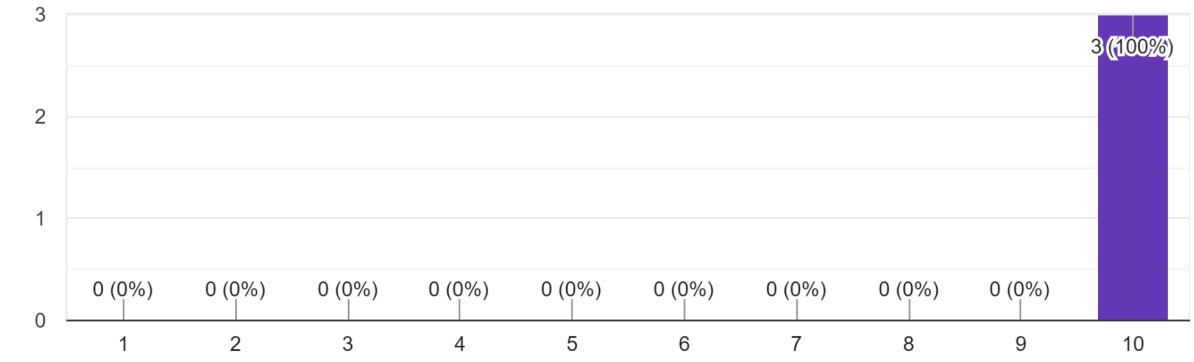
I found it easy to navigate from the home page to the dashboard.

3 responses



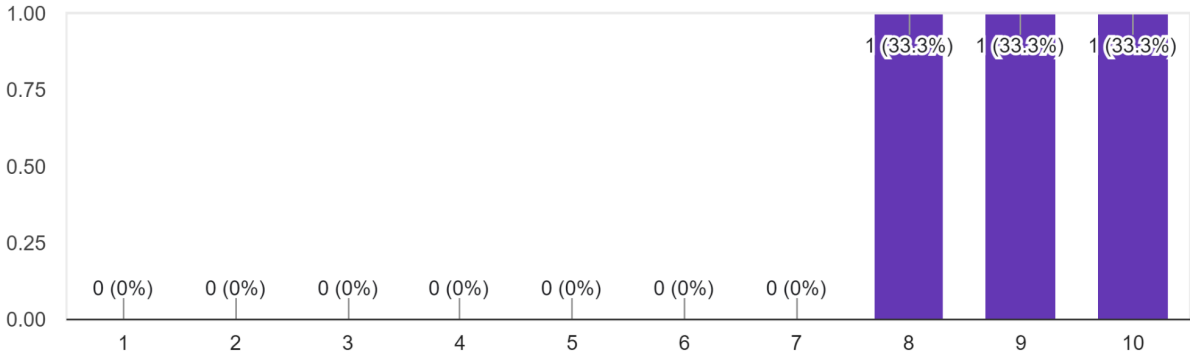
I found it easy to create an account and login.

3 responses



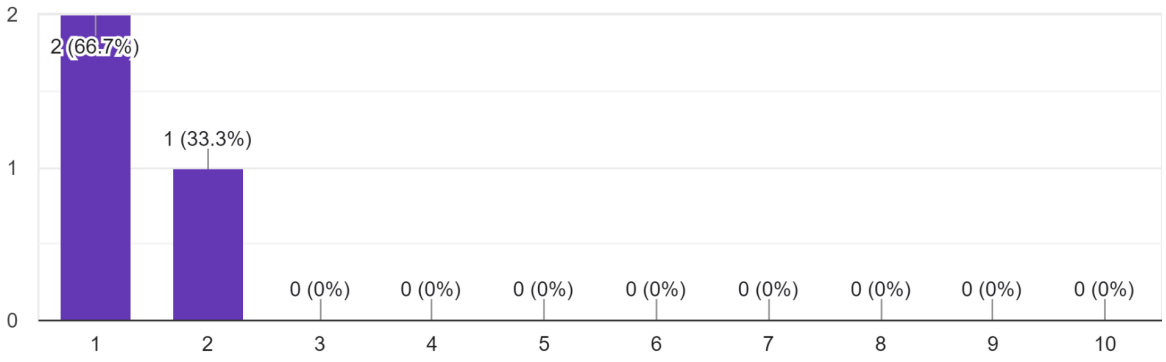
I was able to intuitively navigate the dashboard to create an organization.

3 responses



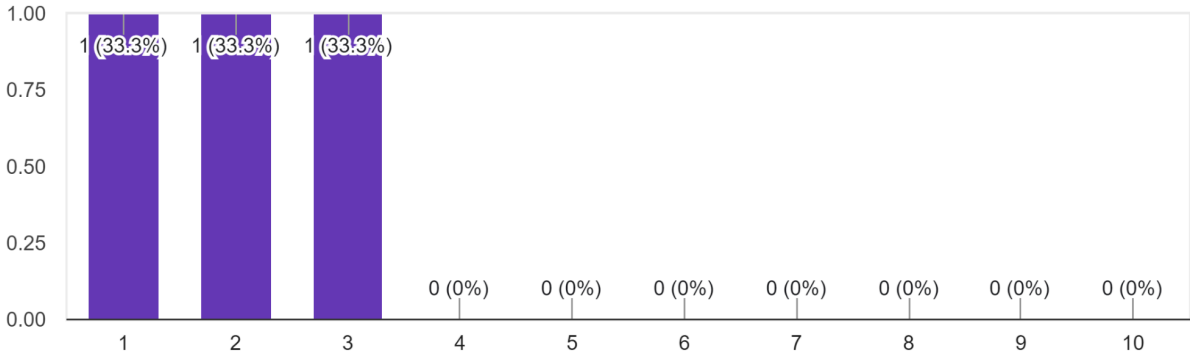
The dashboard header was difficult to understand.

3 responses



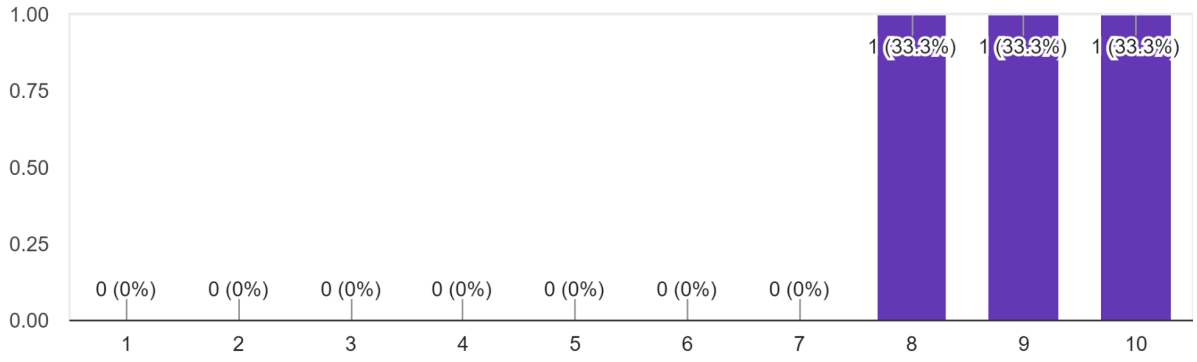
The group's page layout confused me.

3 responses



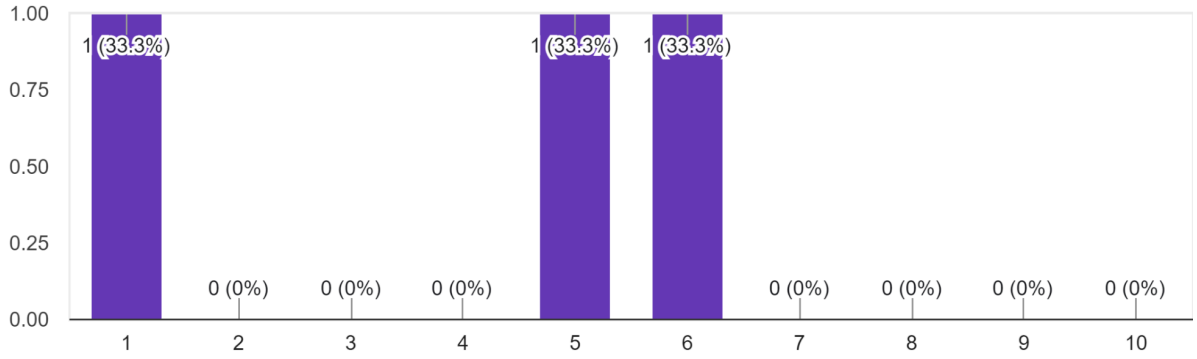
I was able to intuitively navigate the dashboard to create a group.

3 responses



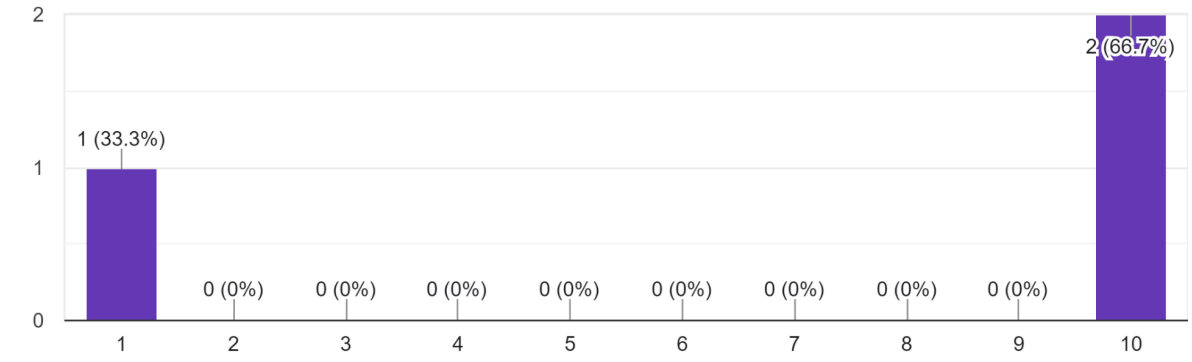
I struggled to find the awards page.

3 responses



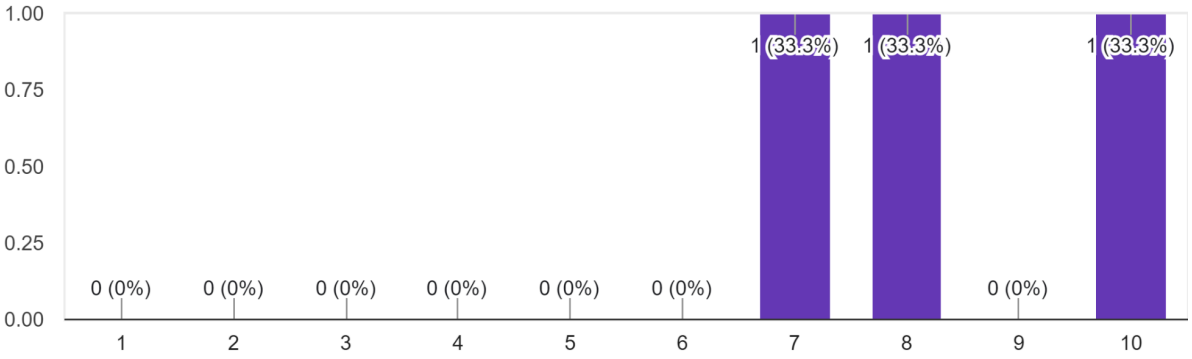
The layout of the awards page was consistent with the other pages.

3 responses



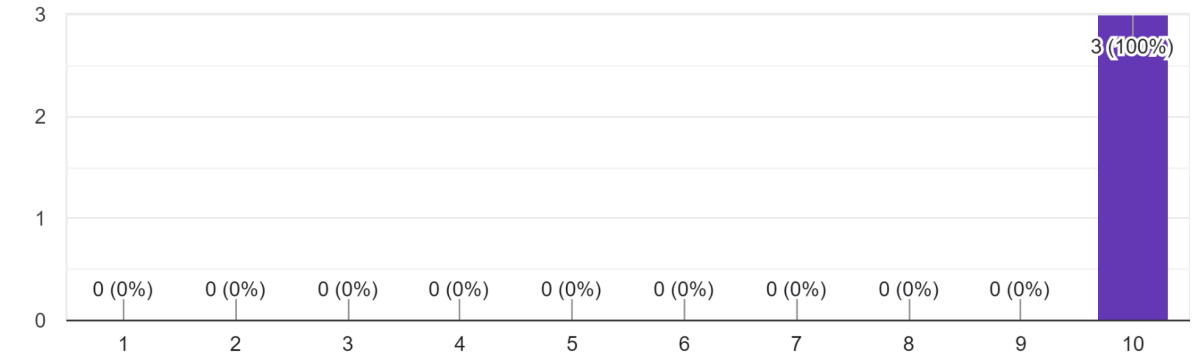
I was able to intuitively navigate the dashboard to create an award.

3 responses



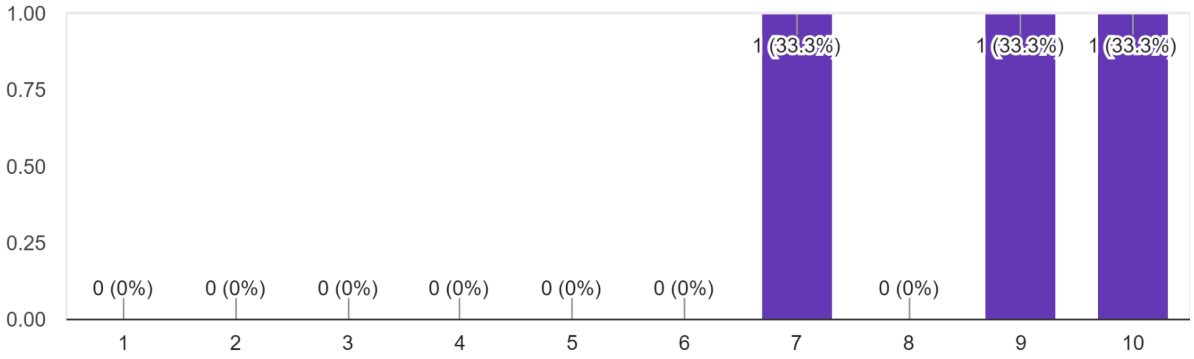
The event page's layout was consistent with the rest of the application.

3 responses



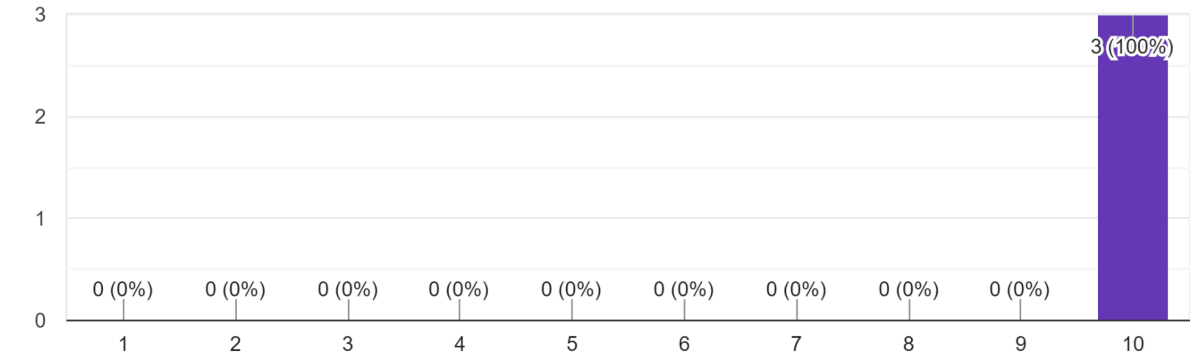
The event details page layout contained all the information I wanted it to.

3 responses



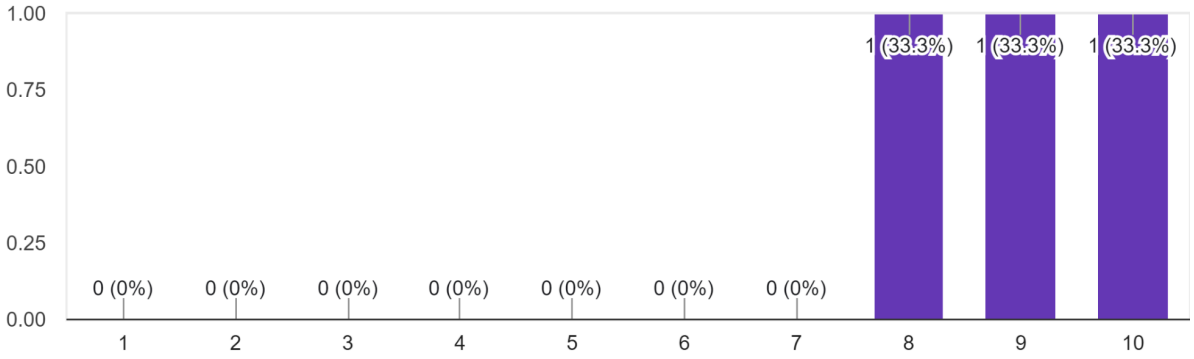
I was able to intuitively navigate the dashboard to create an event.

3 responses



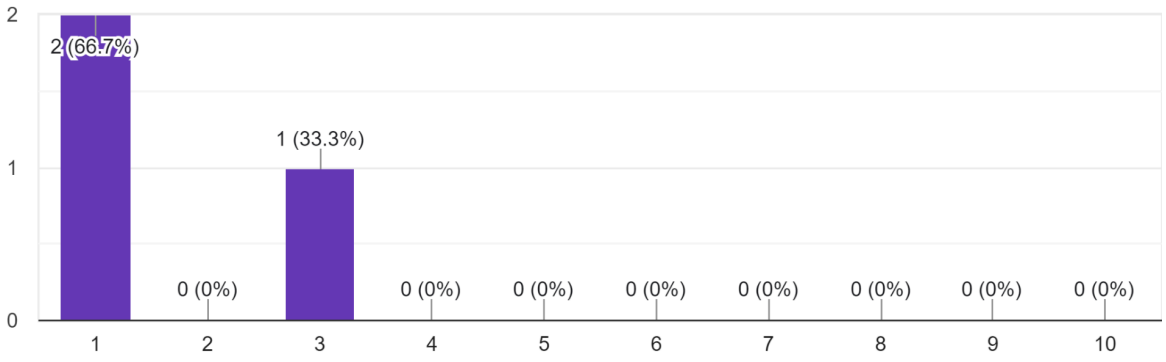
The layout of the members page made intuitive sense.

3 responses



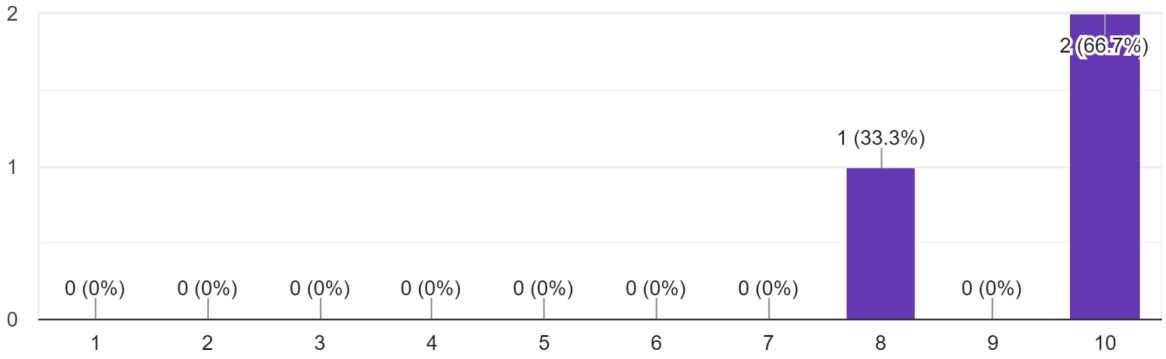
Using the add button in the members page was confusing.

3 responses



I was able to intuitively navigate the dashboard to add a member.

3 responses



QA Test Plan

QA Test 1 (Password Security)

Objective:
Create an account with a variety of passwords, valid and invalid. Ensure that password validation during registration is following modern security standards.

Plan:
The horizontal prototype already supports these features, the user will start from the website's registration page. The three tests will be to try a password that's too short, a password that is long enough, but doesn't have special characters, and a valid/secure password.
http://ec2 compute-1.amazonaws.com:3000/register

Number	Description	Test Input	Expected Output	Pass/Fail
1	Short Password	"Test"	Password must have 8 characters	Pass
2	Weak Password	"Password123"	Password must contain special characters	Pass
3	Strong Password	"Password@123"	No errors	Pass

QA Test 2 (Member Addition)

Objective:
Add three members to your organization using their Member IDs. Ensure that member addition is following good usability standards and error handling.

Plan:
The horizontal prototype already supports these features, the user will start from the website's dashboard after logging in and creating an organization. The three tests will be to add a member that doesn't exist, a member that does exist, and a member that is already in the organization..
http://ec2 compute-1.amazonaws.com:3000/Members

Number	Description	Test Input	Expected Output	Pass/Fail
1	Invalid Member	1234	Member does not exist.	Pass
2	Valid Member	2017	Member added to list.	Pass
3	Existing Member	9141	Member is already in the organization.	Fail

QA Test 3 (Search Bar)

Objective:
Search for three members from the members page. Ensure the search bar is following along with our expected load standards and requirements.

Plan:
The horizontal prototype already supports this feature, the user will start from a populated members page. The three tests will be to search for an existing member, searching for a member that doesn't exist, and a blank search bar.
http://ec2 compute-1.amazonaws.com:3000/Members

Number	Description	Test Input	Expected Output	Pass/Fail
1	Valid Search	"John Doe"	Member list with one item	Fail
2	Invalid Search	"TestUser"	Empty member list	Fail
3	Empty Search	" "	Unfiltered member list	Fail

QA Test 4 (Error Logging)

Objective:
Check if your actions are being logged in the console. Ensure the browser console is populating with appropriate messages in-line with our fault tolerance standards and requirements.

Plan:
The horizontal prototype already supports this feature, the user will start from an empty members page. The three tests will be to try to add a member that is valid, a member that does not exist, and the user themself. The browser's console will be monitored during the test to check for results.
http://ec2 compute-1.amazonaws.com:3000/Members

Number	Description	Test Input	Expected Output	Pass/Fail
1	Valid Member	2017	"Member added successfully"	Pass
2	Invalid Member	1234	"Member not found"	Pass
3	Add Self	9141	"Member already in organization"	Fail

QA Test 5 (Concurrent Users)

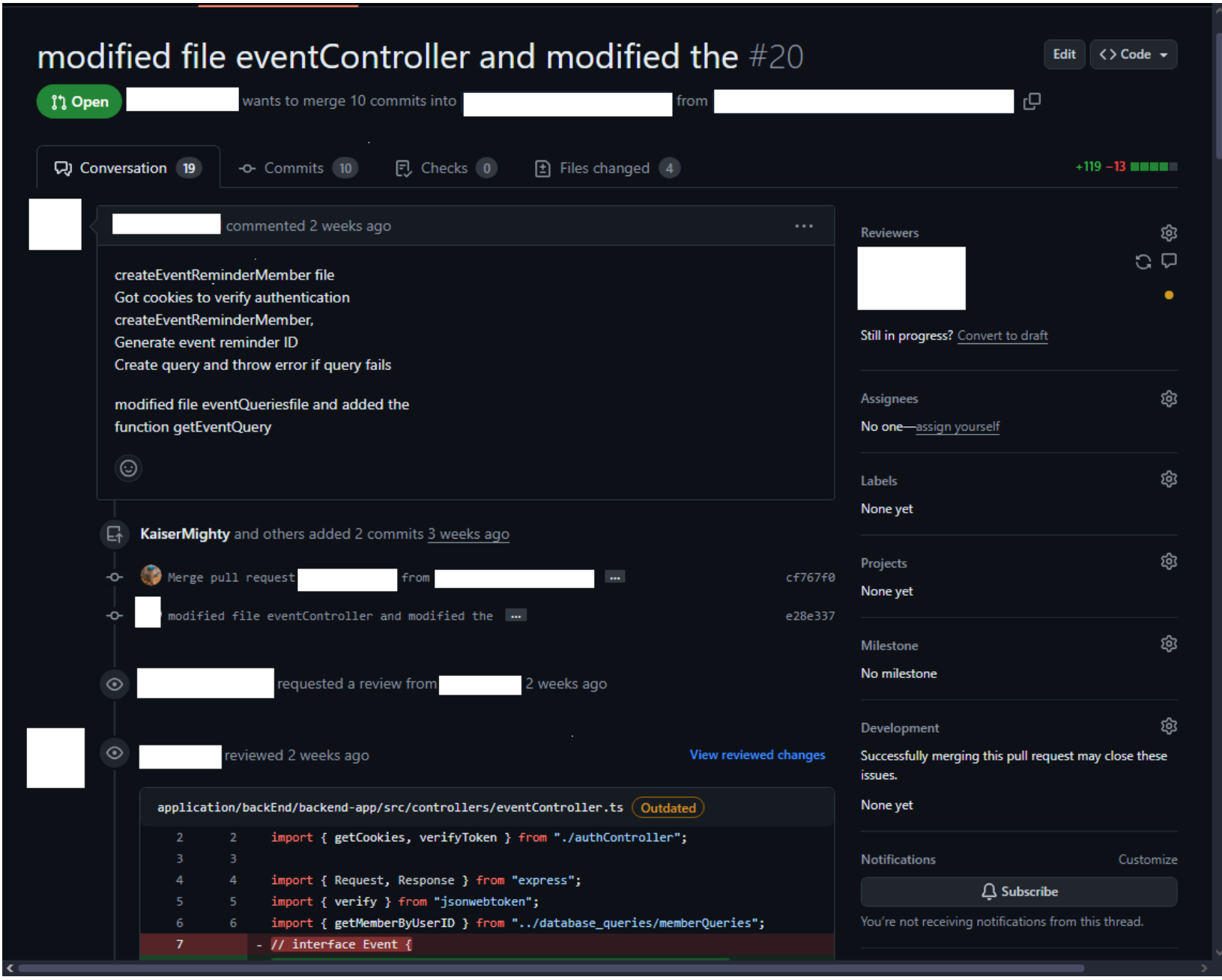
Objective:
Create an account and organization while the application is under load. Ensure the application can support many concurrent users without significant degradation in performance in-line with our performance standards and requirements.

Plan:
The horizontal prototype already supports the features we will be stress testing. We will be using Postman to simulate network traffic. The user will start from the homepage. The user will try registering, logging in, and creating an organization while the database and backend are under stress.
http://ec2 compute-1.amazonaws.com:3000/

Number	Description	Test Input	Expected Output	Pass/Fail
1	Registration	New Account	Minimal Delay	Pass
2	Login	Existing Account	Minimal Delay	Pass
3	Create Organization	New Organization	Minimal Delay	Pass

Code Review

Internal Code Review



Open **modified file eventController and modified the** #20

[redacted] wants to merge 10 commits into [redacted] from [redacted]

```
2  2   import { getCookies, verifyToken } from "./authController";
3  3
4  4   import { Request, Response } from "express";
5  5   import { verify } from "jsonwebtoken";
6  6   import { getMemberByUserID } from "../database_queries/memberQueries";
7     - // interface Event {
      7 + import { getEventQuery } from "../database_queries/eventQueries";
```

[redacted] 2 weeks ago

getEventQuery should be grouped with the other internal imports like Event, EventReminder, not with verify, request, response

Reply...

Resolve conversation

[redacted] reviewed 2 weeks ago View reviewed changes

application/backEnd/backend-app/src/controllers/eventController.ts

```
132 +         try {
133 +             let newEvent = await getEventQuery(eventID);
134 +         } catch (Error) {
135 +             return res.status(500).json({
```

[redacted] 2 weeks ago

Rather than a query error, if we cannot find the event. This should be an error that the event does not exist

Reply...

Resolve conversation

Notifications Customize

Subscribe

You're not receiving notifications from this thread.

4 participants

[redacted]

Maintainers are allowed to edit this pull request.

Lock conversation

Open **modified file eventController and modified the** #20

[redacted] wants to merge 10 commits into [redacted] from [redacted]

Resolve conversation

[redacted] reviewed 2 weeks ago View reviewed changes

randolfuy01 left a comment

Looks good so far, only a few minor issues.

[redacted] reviewed 2 weeks ago View reviewed changes

application/backEnd/backend-app/src/controllers/eventController.ts

```
136 +                 message: "Error querying event"
137 +             })
138 +         }
139 +
```

[redacted] 2 weeks ago

now that we verify that the event exists, make sure to create another query to add that eventReminder to the database.

[redacted] 2 weeks ago Author

Can you please clarify regarding the query to add eventReminder to the database:

Which database table will this record be written?
What attributes will be written to the database?

[redacted] 2 weeks ago

there should be an eventReminder, reference the table diagram inside milestone 2. If a user is creating an event reminder, presumably you would want to write all attributes to that table

Open **modified file eventController and modified the** #20

[redacted] wants to merge 10 commits into [redacted] from [redacted]

there should be an eventReminder, reference the table diagram inside milestone 2. If a user is creating an event reminder, presumably you would want to write all attributes to that table into a single row

[redacted] 2 weeks ago

there isn't an actual query for it yet, you can create one. It would be similar to createMember or createOrganization just with different fields but the syntax should be relatively similar

[redacted] 2 weeks ago Author

By 'Table Diagram' do you mean the EER (Enhanced Entity-Relationship) diagram in the Milestone 2 document?

Reply...

Resolve conversation

[redacted] n the file of eventController I modified and ... 3aef8bf

[redacted] requested a review from [redacted] last week

[redacted] commented last week Author

I modified in the eventQueries file I added insertEventReminderQuery

In the eventController I created the function generateEventReminderID() and created query to check if the event exists and
throw an error if event does not exist in the database

Open **modified file eventController and modified the** #20

[redacted] wants to merge 10 commits into [redacted] from [redacted]

[redacted] reviewed last week — View reviewed changes

application/backEnd/backend-app/src/controllers/eventController.ts

```
1  + import { Event, EventReminder, Member, Cookie} from "./models";
2  + import { getEventQuery } from "../database_queries/eventQueries";
3  + import { insertEventReminderQuery } from "../database_queries/eventQueries";
4  +
2  5    import { getCookies, verifyToken } from "./authController";
```

[redacted] last week

This needs to be grouped with the other internal imports

[redacted] last week — Author

[redacted] I am confused about which line needs to be grouped with the other internal imports?

[redacted] last week

Do u see how getCookies and verifyToken have 1 line in between them and the insertEventReminderQuery? The getCookies and verifyToken should be grouped with them.

Reply...

Resolve conversation

[redacted] reviewed last week — View reviewed changes

application/backEnd/backend-app/src/controllers/eventController.ts — Outdated — Show resolved

Open **modified file eventController and modified the** #20

[redacted] wants to merge 10 commits into [redacted] from [redacted]

ts Outdated Show resolved

[redacted] reviewed last week View reviewed changes

application/backEnd/backend-app/src/database_queries/eventQueries.ts

```
40 + // This query inserts an event reminder into the EventReminder table in the data
41 + async function insertEventReminderQuery(eventReminder: EventReminder): Promise<a
42 +     return new Promise((resolve, reject) => {
```

[redacted] last week

Have [redacted] review the database logic behind this

[redacted] last week Author

[redacted] said if you can review the logic behind this function

[redacted] last week

the query for the get events is correct

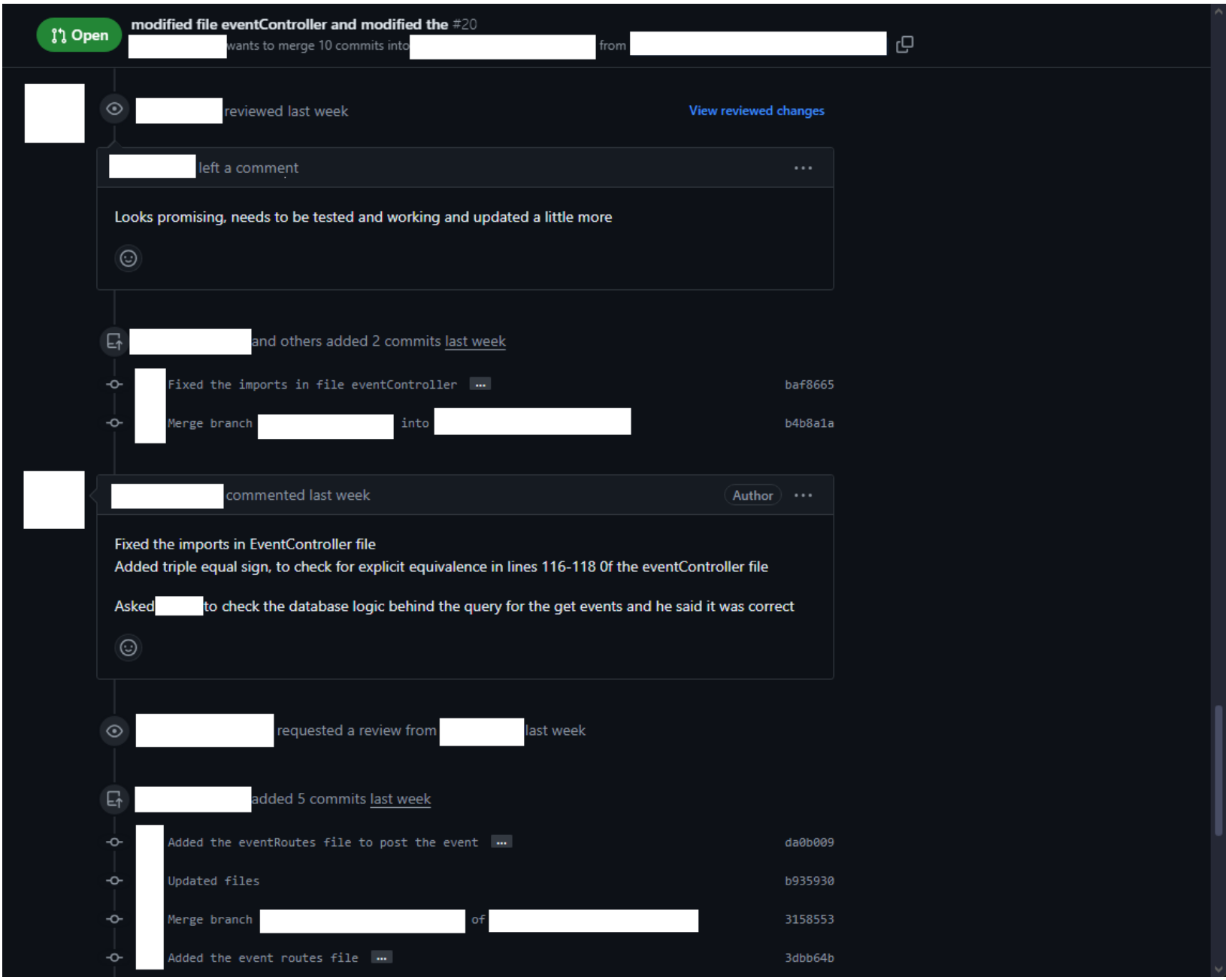
[redacted] last week

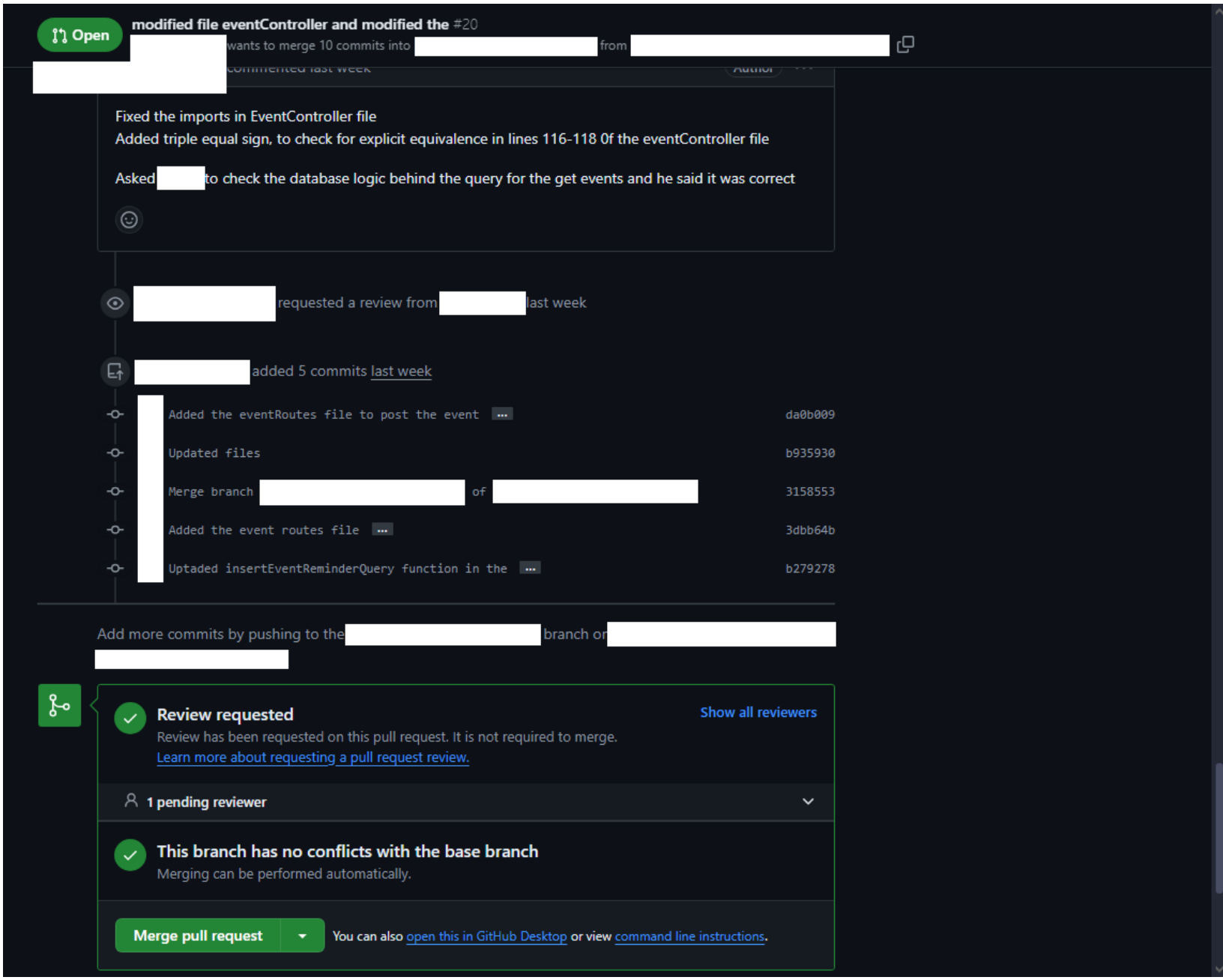
also make sure to have the db set up on your own system, you need to be capable of testing your own functions you create. its difficult for me to run it because i have 3-classes and personal responsibilites.

Reply...

Resolve conversation

[redacted] reviewed last week View reviewed changes


Open
modified file eventController and modified the #20
wants to merge 10 commits into
from
reviewed last week
View reviewed changes
left a comment
Looks promising, needs to be tested and working and updated a little more
and others added 2 commits last week
Fixed the imports in file eventController ...
baf8665
Merge branch
into
b4b8a1a
commented last week
Author
Fixed the imports in EventController file
Added triple equal sign, to check for explicit equivalence in lines 116-118 0f the eventController file
Asked
to check the database logic behind the query for the get events and he said it was correct
requested a review from
last week
added 5 commits last week
Added the eventRoutes file to post the event ...
da0b009
Updated files
b935930
Merge branch
of
3158553
Added the event routes file ...
3dbb64b


Open
modified file eventController and modified the #20
wants to merge 10 commits into
from
Fixed the imports in EventController file
Added triple equal sign, to check for explicit equivalence in lines 116-118 0f the eventController file
Asked
to check the database logic behind the query for the get events and he said it was correct
requested a review from
last week
added 5 commits last week
Added the eventRoutes file to post the event
da0b009
Updated files
b935930
Merge branch
of
3158553
Added the event routes file
3dbb64b
Uptaded insertEventReminderQuery function in the
b279278
Add more commits by pushing to the
branch or
Review requested
Review has been requested on this pull request. It is not required to merge.
Learn more about requesting a pull request review.
Show all reviewers
1 pending reviewer
This branch has no conflicts with the base branch
Merging can be performed automatically.
Merge pull request
You can also open this in GitHub Desktop or view command line instructions.

External Code Review

A

Reply Reply all Forward ···

To: Krishna

Tue 7/30/2024 11:18 AM

Hey ,

Our team has analyzed your code and left the following:

- Likes the well organization within each files
- Likes the use of using TS for better consistency
- Liked how the passwords were being hashed before storage, leading to better security
- Error handling could be improved, instead of error registering user, could specify where error is actually coming from to improve
- Integrating more comments to help testers who don't understand what's going on to still be able to acknowledge the information

Thank you for reviewing our code and we look forward to seeing your guys work at the end of the semester!

- , Team 01

From: Krishna
Sent: Tuesday, July 30, 2024 9:47 AM
To:
Subject: Re: CSC648 External Code Review

Hey ,

Our backend lead has reviewed your files and shared with me the following feedback:

- There could be specific comments that communicate what routers do what to an first-time reader.
- Authentication and security is spread out over 3 different files, this makes it harder to track.
- There could be more decoupling within the code. Everything is so interdependent on each other that making changes to the one class could affect everything else.
- The user class could be separated out, I feel that it's a little dense at the moment, that affects it's readability.
- It appears that there is a split between functional programming and object oriented programming, you could try to consolidate the styles for consistency.

We appreciate you taking the time to conduct a code review with us, and wish you good luck with the rest of the project!

Thanks,
Krishna
Team 05

From:
Sent: Monday, July 29, 2024 8:21 PM
To: Krishna
Subject: Re: CSC648 External Code Review

Hi ,

Attached is our sign up module that deals with authentication and email verification. Would be appreciated if you guys could review the email verification portion of our signup process. I will get back to you either by tonight or tomorrow morning with your code review, thank you! (code will be sent over discord as it got blocked on here)

From: Krishna
Sent: Thursday, July 25, 2024 2:20 PM
To:
Subject: CSC648 External Code Review

Hi Aaron,

I saw your message about the external code review and would like to take you up on your offer! I've attached a package that contains our user authentication logic, with the main file being authController.ts, for review. I've included three additional files that authController.ts imports for your reference as needed. Please feel free to reply to this email with a package or files of your own, and we will review them and get back to you ASAP!

Thanks,
Krishna
Team 05

Security Self-Check

In-line with modern standards, our application does not store unsecured plaintext passwords, but instead stored a secure hashed password to keep users' accounts safe.

When the backend receives the password, our controller calls a hashing function.

```
// HASH AND VERIFY HASHING PASSWORD
const hashedPassword:string = await hashPassword(registerUser.password);
const verify = verifyPassword(registerUser.password, hashedPassword);
    if (!verify) {
        return res.status(400).json({
            error: 'Password Hashing unsuccessful'
        });
    }
```

The hashing function uses bcrypt, and goes as follows:

```
import bycript from 'bcrypt';

// HASHES PASSWORD
async function hashPassword(password: string): Promise<string> {
    return await bycript.hash(password,8);
}

// VERIFIES PASSWORD
async function verifyPassword(password: string, hashPassword: string):
Promise<boolean> {
    return await bycript.compare(password, hashPassword);
}
```

This hashed password is what is passed in the MySQL query and stored in the DB:

userID	email	username	password	profilePicture	firstName	lastName
9141	testing@email.com	JohnDoe	$2b$08$UFJrQUcykSWjhQNAYzR4D./30625/63mgTZUGnHPJajNyZqGnGKUy	NULL	John	Doe
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Our search bar input data are validated to make sure we are searching the right tables in the database:

Our autocomplete function is as follows:

```
export const autoCompleteSearch = async (req: Request, res: Response) => {
    try {
        try {
            let token = req.cookies.token;
            let decoded = jwt.verify(token, JWT_SECRET_KEY);
            if (decoded === null) {
                throw new Error('Unauthorized User');
            }
        } catch (Error) {
            res.status(401).json({
                error: 'Unauthorized User'
            });
        }
        let searchQuery: string = req.body.searchQuery;
        // Query words from the database
        let members: Member[] = await
getMembersFromOrganizationQuery(req.cookies.orgID);
        let queryWords: string[] = members.map((members) =>
members.firstName + " " + members.lastName);
        const completer = new Trie();
        for (const word of queryWords) {
            completer.insert(word);
        }
        const prefix: string = searchQuery;
        const result: string[] = completer.search(prefix);
        res.status(200).json({
            queryResults: result
        });
    } catch(Error) {
        res.status(400).json({
            error: 'Error searching for topic'
        });
    }
}
```

Out ranked search function is as follows:

```
export const rankedSearch = async (req: Request, res: Response) => {
    try {
        let cookie: Cookie = await getCookies(req);
        let verified = verifyToken(cookie.token);
        if (verified === null) {
            return res.status(401).json({
                error: "Unauthorized User"
            });
        }
        let searchQuery: Search = req.body;
        let org = req.cookies.orgID;
        if (searchQuery.searchType === "members") {
            let members: Member[] = await
getMembersFromOrganizationQuery(org);
            let rankedMembers: any = rankSearched(searchQuery.searchQuery,
members.map((member) => member.firstName + " " + member.lastName));
            let sortedMembers: Member[] = rankedMembers.map((rankedMember:
{ item: string, score: number }) => {
                return members.find(member => (member.firstName + " " +
member.lastName) === rankedMember.item);
            }).filter((member: Member | undefined): member is Member =>
member !== undefined);
            return res.status(200).json({
                message: "Search for members",
                queryResult: sortedMembers
            });
        }
        // Truncated for formatting, but we have an else if for every case
    } catch(Error) {
        return res.status(400).json({
            error: 'Error searching for topic'
        });
    }
}
```

Non-Functional Specs Self-Check

Performance

- **DONE -** The application shall respond quickly to user prompts with minimal latency.
- **DONE -** The application shall be able to host/handle up to 1,000 accounts without any performance problems.
- **DONE -** The application shall support many active-concurrent users without significant degradation in performance.

Security

- **DONE -** The application shall only store hashed passwords.
- **ON TRACK -** The application database shall be backed up regularly.

Coding Standard

- **DONE -** The application shall follow consistent coding and naming conventions throughout.
- **DONE -** The application shall gracefully handle errors without detrimentally affecting the rest of the user experience.
- **ON TRACK -** The application shall be well documented for easy understandability for other developers and admins alike.

Media Storage

- **ISSUE -** The application shall restrict profile pictures to 500x500 pixels and 5 megabytes or less.
    - Images are not part of our priority 1 requirements, and as such, are not included in our prototype.

Privacy

- **DONE -** The application shall hide organizations, preventing users not in the organization from viewing it.
- **DONE -** Data shall be anonymized and pseudonymized wherever possible to minimize damage in the event of a leak.

- **DONE -** Only the minimum amount of data required shall be stored in order to minimize the risk of security breaches.

Scalability

- **DONE -** The application shall be designed to scale efficiently to handle an increasing number of users and data without performance degradation.
- **DONE -** The database shall scale down if the workload is low to remain cost effective and optimize resource utilization.

Reliability

- **DONE -** The application shall ensure high availability, with a 99.9% uptime, and incorporate robust error-handling mechanism to maintain stability.

Usability

- **DONE -** The application shall provide an user friendly interface, ensuring that users can easily navigate and use all features without extensive training.

Compliance

- **DONE -** The application shall comply with relevant industry standards and regulations, such as for data protection and privacy.

Expected Load

- **DONE -** The application shall respond to user queries within 500 milliseconds for requests.
- **DONE -** The application shall handle a minimum of 1000 transactions per second during peak hours.
- **DONE -** The application shall horizontally scale to support up to 10,000 concurrent users without degradation in response time.

Fault Tolerance

- **DONE -** The application shall recover from a server failure within 15 minutes without loss of data or interruption of service.
- **DONE -** Critical components of the application shall have a backup system that can take over within 5 minutes of a primary system failure.
- **DONE -** The application shall log all errors and provide automatic recovery for non-critical errors within 60 seconds.

Database High Level Specs

- **DONE -** The database shall use appropriate indexes to optimize query performance, especially for frequently accessed data.
- **DONE -** The database should be capable of storing up to 1 terabyte of data efficiently, with automatic data compression for storage optimization.
- **DONE -** All sensitive data stored in the database should be encrypted.

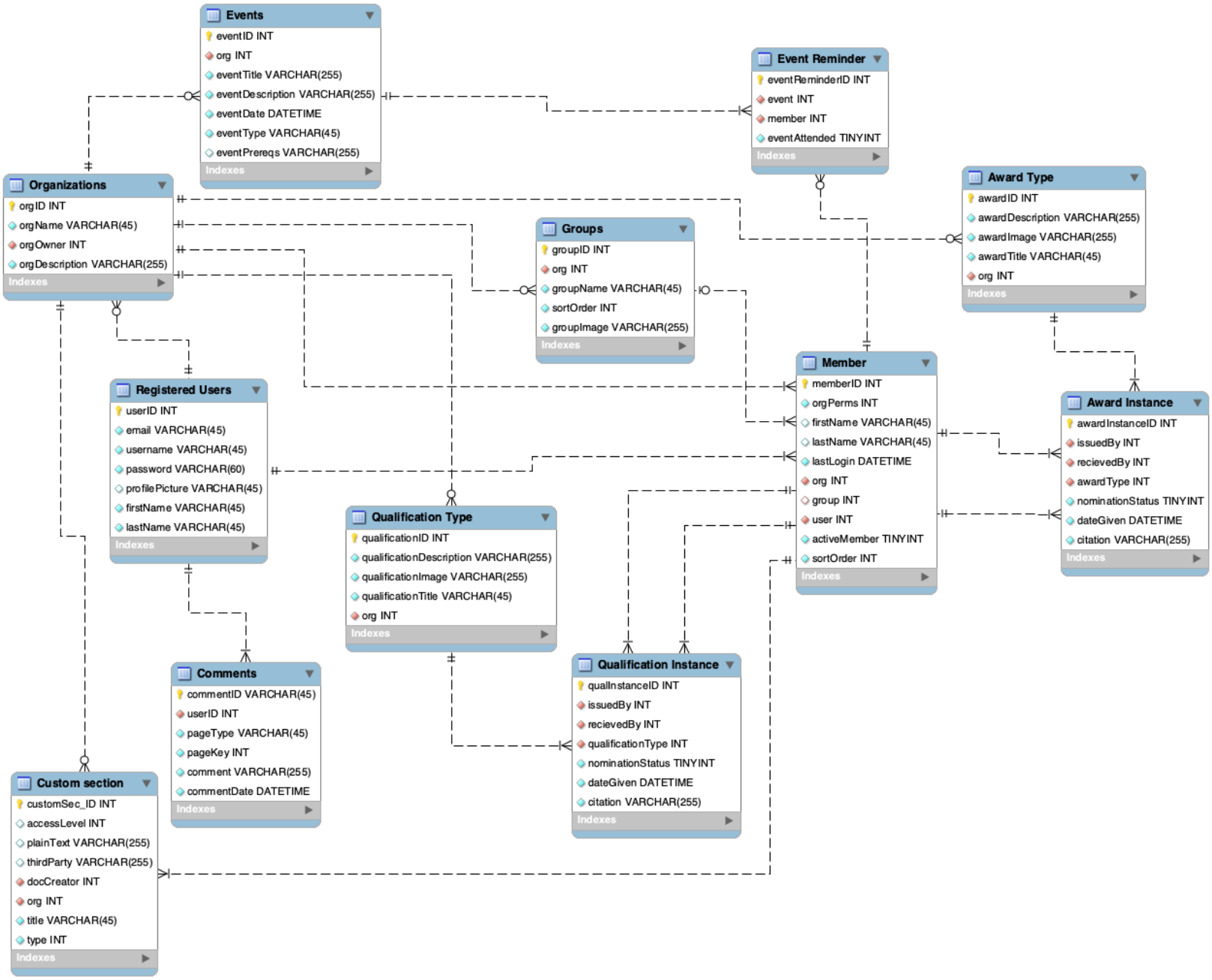
Browser and OS Support

- **DONE -** The application shall support Chrome Stable 126 (126.0.6478.153).
- **DONE -** The application shall support Windows 10 22H2 (19045.4651).
- **DONE -** The application shall support Windows 11 23H2 (22631.3737).
- **DONE -** The application shall support macOS Sonoma (14.5).
- **DONE -** The application shall support Amazon Linux (2023.4.20240611.0).

Additional Comments

Changes to the DBMS

Adding a Title (VARCHAR 45) and Type (INT) field to the Custom Section entity in order to differentiate between Projects/Tasks and Highlights.



Team Contributions

- **Krishna**
  - Revised Product Summary Section of Documentation.
  - Usability Test Plan Section of Documentation.
  - QA Test Plan Section of Documentation.
  - Security Self-Check Section of Documentation.
  - Non-Functional Requirements Self-Check Section of Documentation.
  - Group page frontend of Beta Prototype.
  - Recognitions page frontend of Beta Prototype.
-
  - Prototype revisions as per horizontal prototype feedback.
  - General restyling of all pages.
  - Oversaw other front-end developers progress on static pages.
  - Refined static pages.
  - Tasks/Projects page frontend of Beta Prototype.
  - Events page frontend of Beta Prototype.
  - Highlights page frontend of Beta Prototype.
-
  - Leading efforts on Backend of the Prototype.
  - Revised search controller of Beta Prototype.
  - Created recognition controller of Beta Prototype.
  - Created custom sections controller of Beta Prototype.
  - Revised password hashing functions.
  - Added various routing components for the backend.
  - Miscellaneous bug fixing on the backend component.
  - Completed Internal Code Review.
  - Complete External Code Review.
-
  - Revised custom sections queries of Beta Prototype.
  - Revised custom sections controller of Beta Prototype.
  - Adjusted DBMS files (EER and .sql files).
  - Revised group queries of Beta Prototype.
  - Miscellaneous bug fixing on the backend component.

-
    - Revised Case Studies page of Beta Prototype.
    - Revised Reviews page of Beta Prototype.
    - Revised Updates page of Beta Prototype.
    - Revised Server Status page of Beta Prototype.
    - Revised Features page of Beta Prototype.
    - Revised Pricing page of Beta Prototype.
    - Revised About page of Beta Prototype.
-
    - Revised Contact Us page of Beta Prototype.
    - Revised Chat Support page of Beta Prototype.
    - Revised Report a Bug page of Beta Prototype.
    - Revised Careers page of Beta Prototype.
    - Revised Culture page of Beta Prototype.
    - Revised Blog page of Beta Prototype
    - Revised Help Center page of Beta Prototype
-
    - Product Summary Description Component of Documentation.
    - Product Summary Superior Features Component of Documentation.
    - Product Summary Competitors Component of Documentation.
-
    - Product Summary Committed Functions Component of Documentation.
    - Contributed to Internal Code Review.
    - MIscellaneous contributions to backend API.

SW Engineering CSC648-848-05 Summer 2024

TeamMate
Personnel Management
Team 05 - Concat

Name	E-Mail	Role
Krishna		Team Lead
		Frontend Lead
		Frontend Dev
		Backend Lead
		Backend Dev
		Database Lead
		Database Dev
		Github Master

Milestone 4
July 30 2024

Version History

Revision	Date Submitted
M1	13 June 2024
M1V1	20 June 2024
M1V2	09 July 2024
M2V1	09 July 2024
M2V2	23 July 2024
M3V1	23 July 2024
M3V2	30 July 2024
M4V1	30 July 2024

Table of Contents

Product Summary

TeamMate is a comprehensive personnel management solution designed for small to mid-sized groups, such as non-profits and community organizations. It consolidates essential management functions into a single, user-friendly platform, enabling easy registration, member tracking, event management, and award issuance.

TeamMate’s standout and superior feature is its Automated Recognition and Progress Tracking System. This system automatically tracks member participation, awards achievements, and updates progress, providing timely recognition and broad overviews of individual and group accomplishments. This holistic view is valuable for performance reviews, planning, setting future goals along with motivating the members of the organization.

Discord focuses on communication and lacks recognition and structured progress tracking, **Airtable** and **Google sheets** require extensive manual setup to achieve similar functionality, whereas **TeamMate** offers an out-of-the-box solution that simplifies administrative tasks. **PERSCOM** provides good tracking but demands more manual effort for recognition, while **Rippling HCM** caters to larger organizations with complex and costly systems, making it less suitable for smaller groups.

Committed Functions

Account

- Users shall register for a new account.
- Accounts shall have unique emails.
- An account shall have a username.
- An account shall have a password.
- An account shall have secure passwords as per modern standards.
- Users shall log into an existing account.
- An account shall create a new organization.
- Accounts shall join an existing organization.
- Accounts shall view all organizations they have joined.

Member

- Member permissions shall be members, group leaders, or admins.
- Admins shall assign member and group leader permissions.
- Admins shall be able to issue awards to members.
- A member's page shall show their information for that organization, along with all attributes, qualifications, awards, etc as a dashboard.

Organization

- An organization shall have an owner.
- An owner can assign other members to be admins.
- An organization shall have a roster page that shows all members in the organization separated by groups.
- Users shall click on any member on the roster to access their member page.

Awards

- Admins shall create award types.
- Admins shall update award types.
- Admins shall delete award types.
- Awards shall have a title.
- Awards shall have a description.
- Admins shall directly assign awards.
- Award instances shall have a date received.
- Award instances shall state the issuing member.

Qualifications

- Admins shall create qualifications types.
- Admins shall update qualifications types.
- Admins shall delete qualifications types.
- Qualifications shall have a title.
- Qualifications shall have a description.
- Admins shall directly assign qualifications.
- Members shall view individual qualifications.
- Qualification instances shall have a date received.
- Qualification instances shall state the issuing member.

Events

- Admins shall create events.
- Group leaders shall create events.
- Admins shall assign members to an existing event.
- Group leaders shall assign members to an existing event.
- Members shall be able to join an event.
- Events shall have a title.
- Events shall have a description.
- Events shall have a start datetime.

Application

http://ec2-44-223-28-76.compute-1.amazonaws.com:3000/

Usability Test Plan

Use/Test Case 1 (Organization Creation)

Objective:
Create an organization. While conducting this test, the user will also go through the registration and login process. This feature is being tested to determine if the user is intuitively able to navigate the home page and sidebar of the dashboard.

Description:
The horizontal prototype already supports these features, the user will start from the website's home page. The intended user is a first time user of the website, with no prior context as to what it is. We are measuring the time and mouse movements of the user, to see how long it takes the user to complete the test, and where all they navigate while completing the test, in order to determine if the website design makes intuitive sense to a first time user.
http://ec2-44-223-28-76.compute-1.amazonaws.com:3000/

Use/Test Case 2 (Group Creation)

Objective:
Create a group. This feature is being tested to determine if the user is intuitively able to navigate the header and corresponding pages intuitively.

Description:
The beta prototype will be supporting this feature, the user will start from the dashboard after logging in. The intended user is a new user that has just created an organization. We are measuring the time taken and mouse movements of the user to determine if they can quickly navigate to the required pages intuitively, to validate our dashboard design.
http://ec2-44-223-28-76.compute-1.amazonaws.com:3000/organizationDashboard/Testing%20Org

Use/Test Case 3 (Award Creation)

Objective:
Create an award. This feature is being tested to determine if the user is able to find the awards page and create an award.

Description:
The beta prototype will be supporting this feature, the user will start from the dashboard after logging in. The intended user is a relatively new user that has created an organization. We are measuring the time taken and mouse movements & page navigation to determine if they are able to figure out that awards are part of the recognitions tab, and can use the creation form to make an award instead of certification.
http://ec2-44-223-28-76.compute-1.amazonaws.com:3000/organizationDashboard/Testing%20Org

Use/Test Case 4 (Event Creation)

Objective:
Create an event. This feature is being tested to determine if the user is able to navigate to the events page and follow through the event creation form.

Description:
The beta prototype will be supporting this feature, the user will start from the dashboard after logging in. The intended user is a relatively new user that has created and organization. We are measuring the time taken and mouse movements to determine if they are able to intuitively navigate from the dashboard to the event page and the events creation form and complete it.
http://ec2-44-223-28-76.compute-1.amazonaws.com:3000/organizationDashboard/Testing%20Org

Use/Test Case 5 (Member Operations)

Objective:
Add a member. This feature is being tested to determine if the user can figure out how to add members to their organization and assign them attributes.

Description:
The beta prototype will be supporting this feature, the user will start from the dashboard after having created a few attributes. The intended user is a user that has explored the dashboard for some time and created various attributes. We are measuring the time taken and mouse movements to determine if they are able to intuitively navigate to the members page and invite another user to their organization.
http://ec2-44-223-28-76.compute-1.amazonaws.com:3000/organizationDashboard/Testing%20Org

Test Table

Test/Use Case	% Completed	Efficiency	Errors	Comments
Org Creation	100%	4-5 Minutes	None	Older users unable to figure out that the create bar on the sidebar is how you make an org. Inflated efficiency due to users coming up with registration info.
Group Creation	100%	<1 Minute	None	Creation complete.
Award Creation	100%	<1 Minute	None	Creation complete.
Event Creation	100%	<1 Minute	None	Creation complete.
Member Add	100%	1-2 Minutes	None	It's not clear how to find a user's ID to add into the org. Can also add duplicate members to org.

Questionnaire

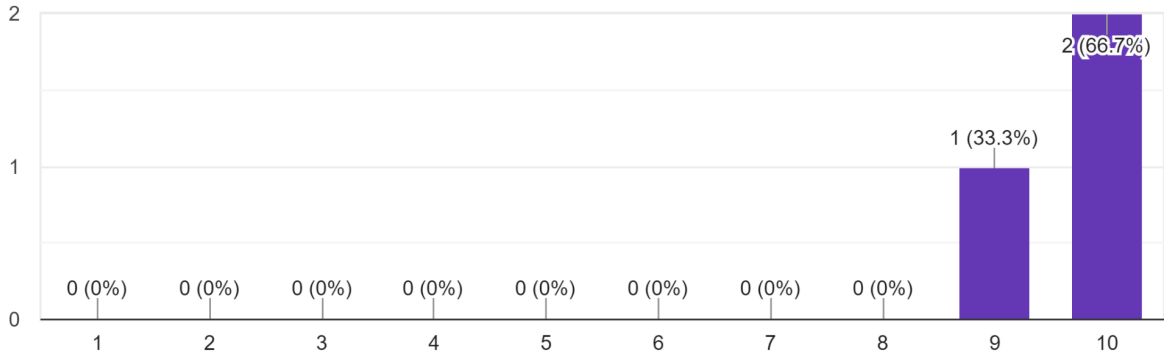
1. I found it easy to navigate from the home page to the dashboard.
2. I found it easy to create an account and login.
3. I was able to intuitively navigate the dashboard to create an organization.
4. The dashboard header was difficult to understand.
5. The group's page layout confused me.
6. I was able to intuitively navigate the dashboard to create a group.
7. I struggled to find the awards page.
8. The layout of the awards page was consistent with the other pages.
9. I was able to intuitively navigate the dashboard to create an award.
10. The event page’s layout was consistent with the rest of the application.
11. The event details page layout contained all the information I wanted it to.
12. I was able to intuitively navigate the dashboard to create an event.
13. The layout of the members page made intuitive sense.
14. Using the add button in the members page was confusing.
15. I was able to intuitively navigate the dashboard to add a member.

https://forms.gle/Xq8RFURWbQ7Rh4gp9

Responses

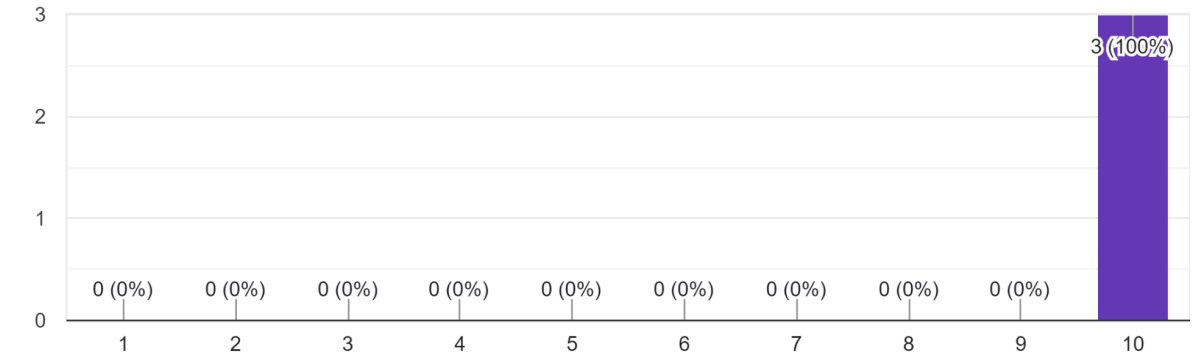
I found it easy to navigate from the home page to the dashboard.

3 responses



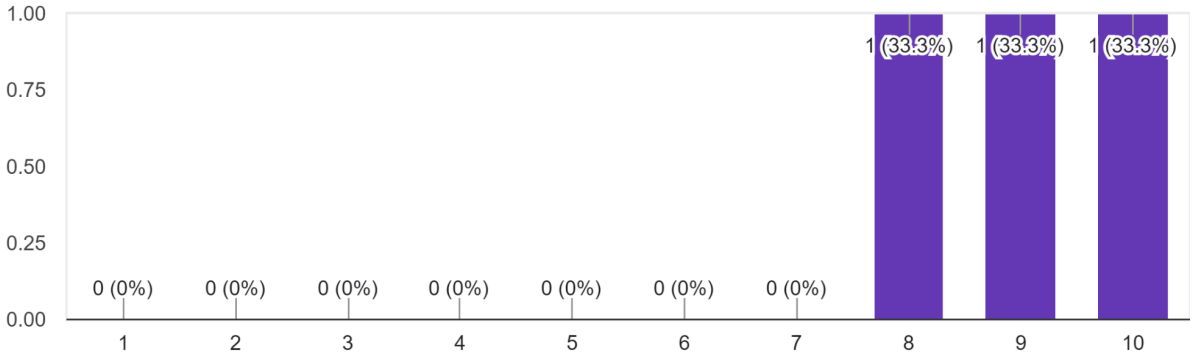
I found it easy to create an account and login.

3 responses



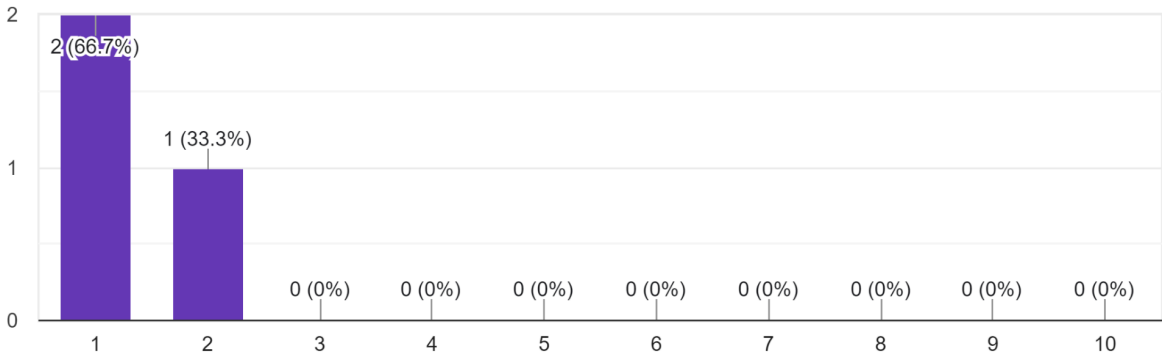
I was able to intuitively navigate the dashboard to create an organization.

3 responses



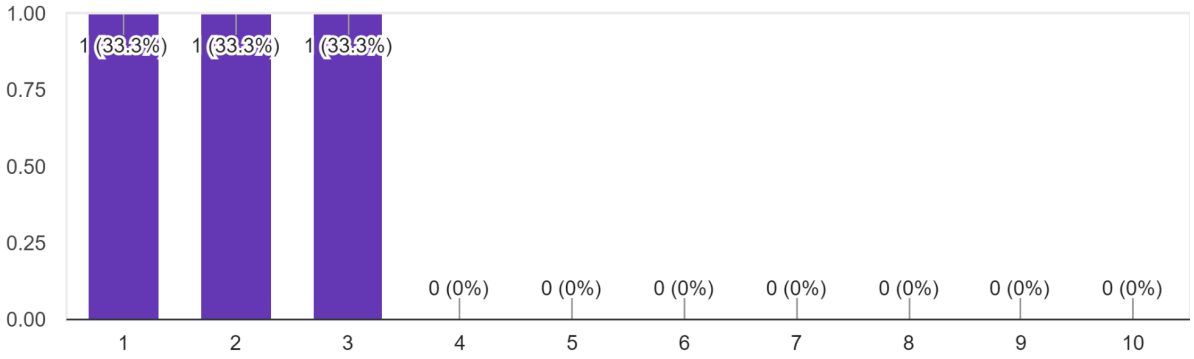
The dashboard header was difficult to understand.

3 responses



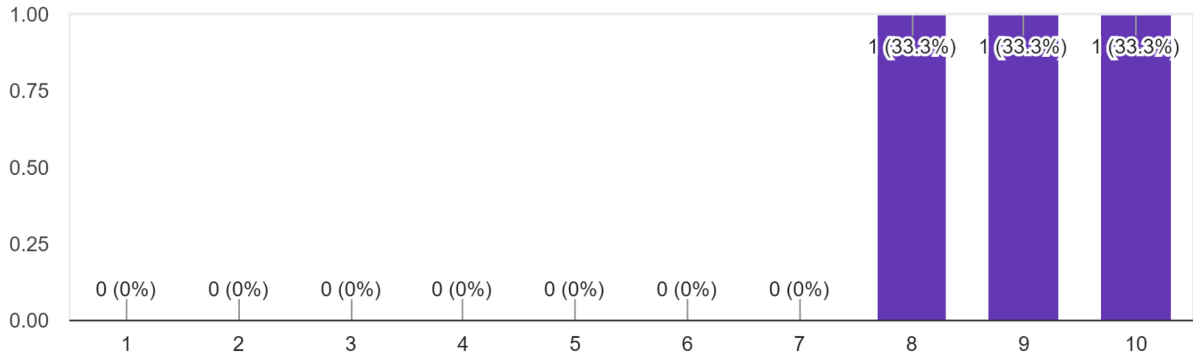
The group's page layout confused me.

3 responses



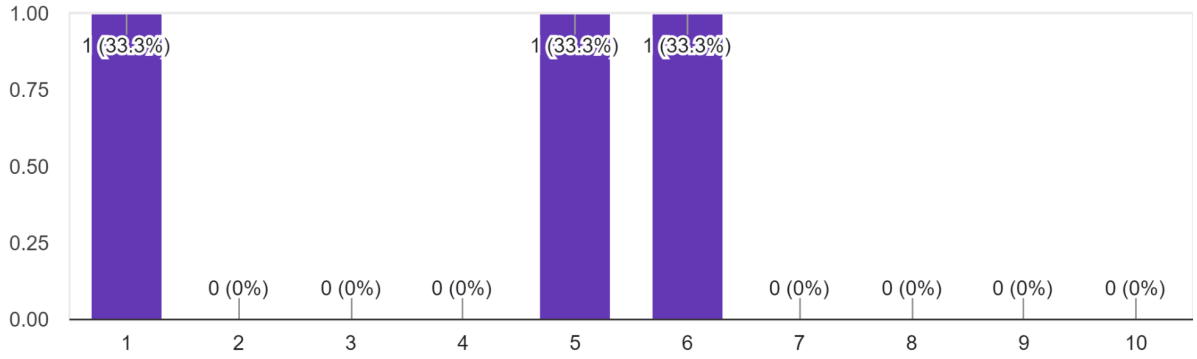
I was able to intuitively navigate the dashboard to create a group.

3 responses



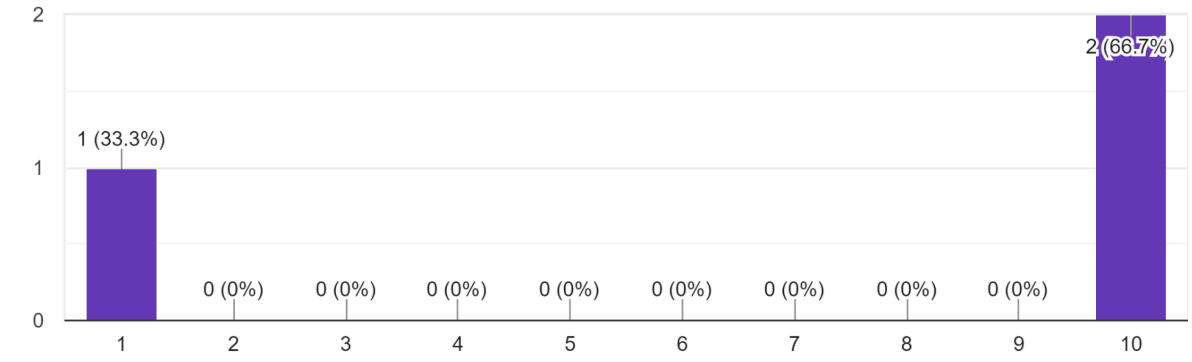
I struggled to find the awards page.

3 responses



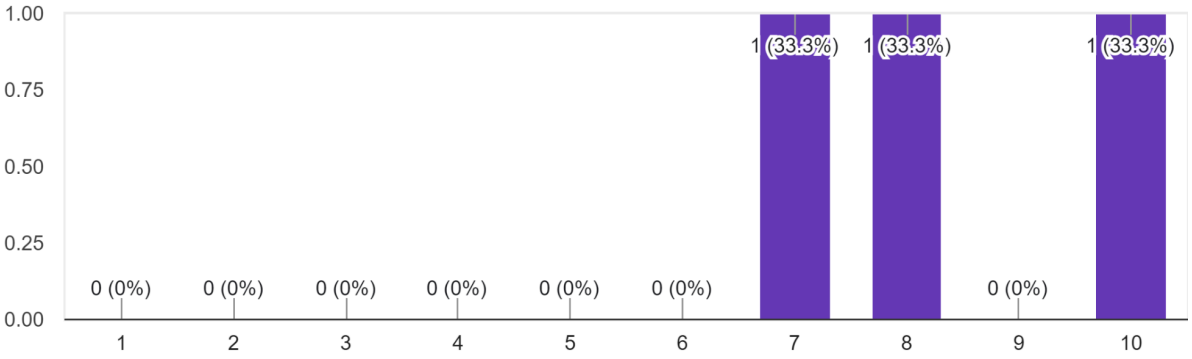
The layout of the awards page was consistent with the other pages.

3 responses



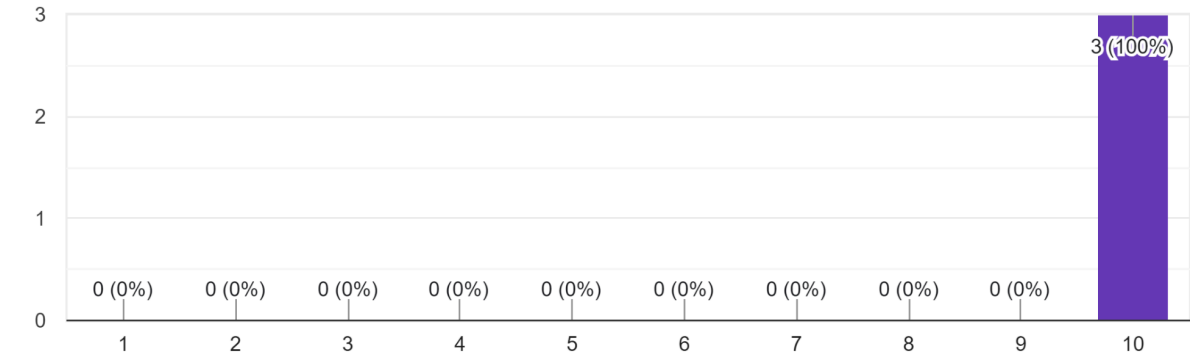
I was able to intuitively navigate the dashboard to create an award.

3 responses



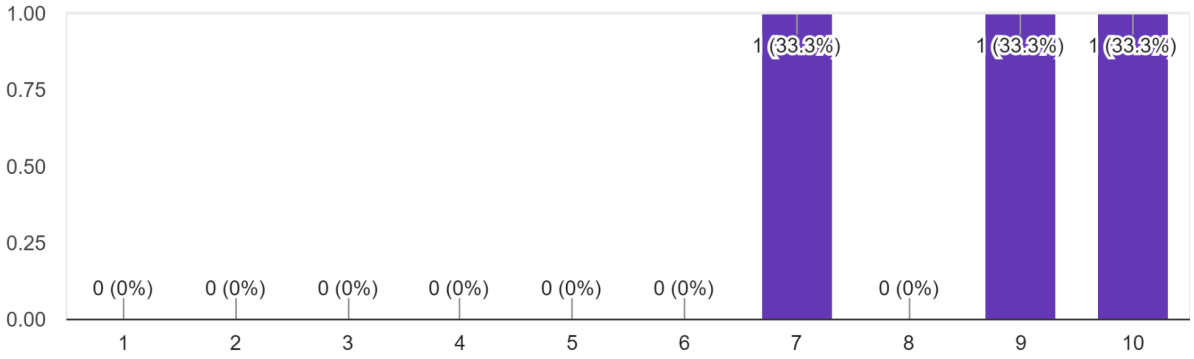
The event page's layout was consistent with the rest of the application.

3 responses



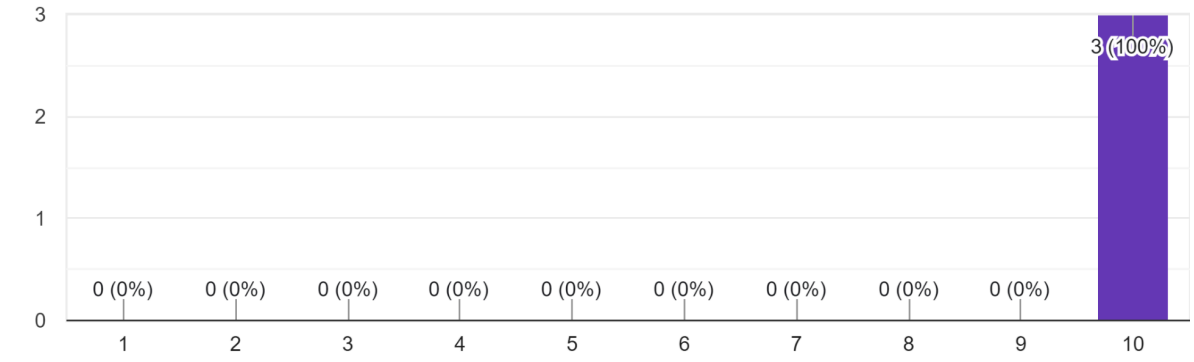
The event details page layout contained all the information I wanted it to.

3 responses



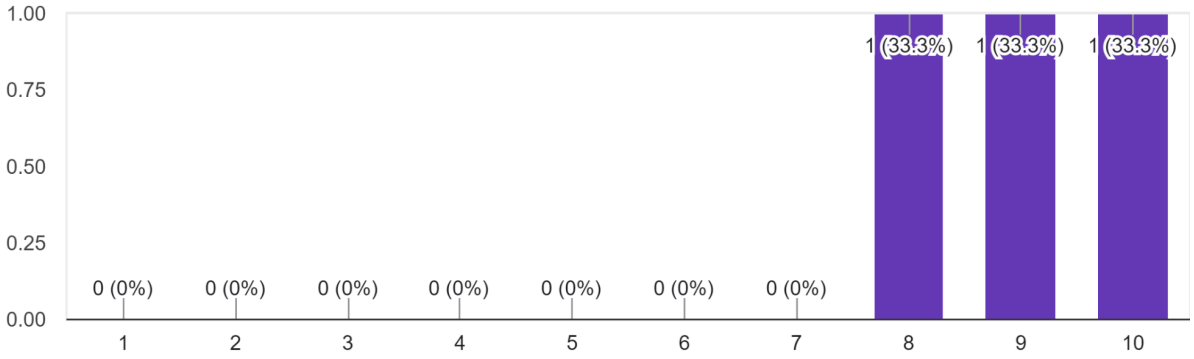
I was able to intuitively navigate the dashboard to create an event.

3 responses



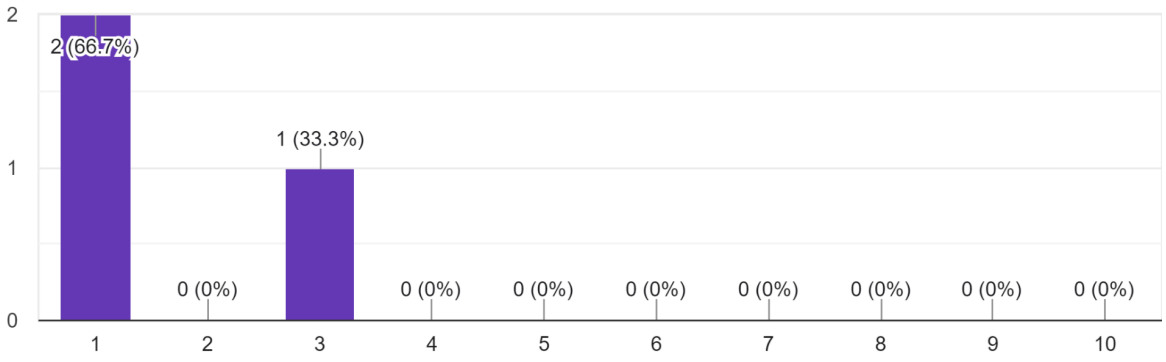
The layout of the members page made intuitive sense.

3 responses



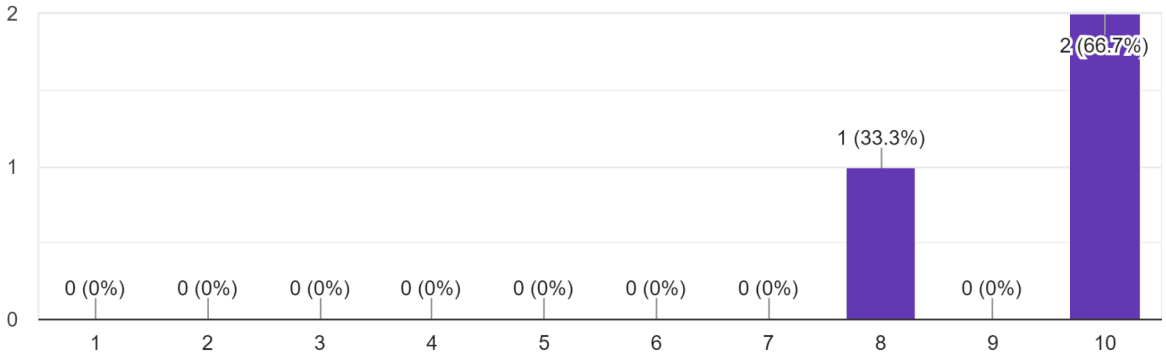
Using the add button in the members page was confusing.

3 responses



I was able to intuitively navigate the dashboard to add a member.

3 responses



QA Test Plan

QA Test 1 (Password Security)

Objective:
Create an account with a variety of passwords, valid and invalid. Ensure that password validation during registration is following modern security standards.

Plan:
The horizontal prototype already supports these features, the user will start from the website's registration page. The three tests will be to try a password that's too short, a password that is long enough, but doesn't have special characters, and a valid/secure password.
http://ec2-44-223-28-76.compute-1.amazonaws.com:3000/register

Number	Description	Test Input	Expected Output	Pass/Fail
1	Short Password	"Test"	Password must have 8 characters	Pass
2	Weak Password	"Password123"	Password must contain special characters	Pass
3	Strong Password	"Password@123"	No errors	Pass

QA Test 2 (Member Addition)

Objective:
Add three members to your organization using their Member IDs. Ensure that member addition is following good usability standards and error handling.

Plan:
The horizontal prototype already supports these features, the user will start from the website's dashboard after logging in and creating an organization. The three tests will be to add a member that doesn't exist, a member that does exist, and a member that is already in the organization..
http://ec2-44-223-28-76.compute-1.amazonaws.com:3000/Members

Number	Description	Test Input	Expected Output	Pass/Fail
1	Invalid Member	1234	Member does not exist.	Pass
2	Valid Member	2017	Member added to list.	Pass
3	Existing Member	9141	Member is already in the organization.	Fail

QA Test 3 (Search Bar)

Objective:
Search for three members from the members page. Ensure the search bar is following along with our expected load standards and requirements.

Plan:
The horizontal prototype already supports this feature, the user will start from a populated members page. The three tests will be to search for an existing member, searching for a member that doesn't exist, and a blank search bar.
http://ec2-44-223-28-76.compute-1.amazonaws.com:3000/Members

Number	Description	Test Input	Expected Output	Pass/Fail
1	Valid Search	"John Doe"	Member list with one item	Fail
2	Invalid Search	"TestUser"	Empty member list	Fail
3	Empty Search	" "	Unfiltered member list	Fail

QA Test 4 (Error Logging)

Objective:
Check if your actions are being logged in the console. Ensure the browser console is populating with appropriate messages in-line with our fault tolerance standards and requirements.

Plan:
The horizontal prototype already supports this feature, the user will start from an empty members page. The three tests will be to try to add a member that is valid, a member that does not exist, and the user themself. The browser's console will be monitored during the test to check for results.
http://ec2-44-223-28-76.compute-1.amazonaws.com:3000/Members

Number	Description	Test Input	Expected Output	Pass/Fail
1	Valid Member	2017	"Member added successfully"	Pass
2	Invalid Member	1234	"Member not found"	Pass
3	Add Self	9141	"Member already in organization"	Fail

QA Test 5 (Concurrent Users)

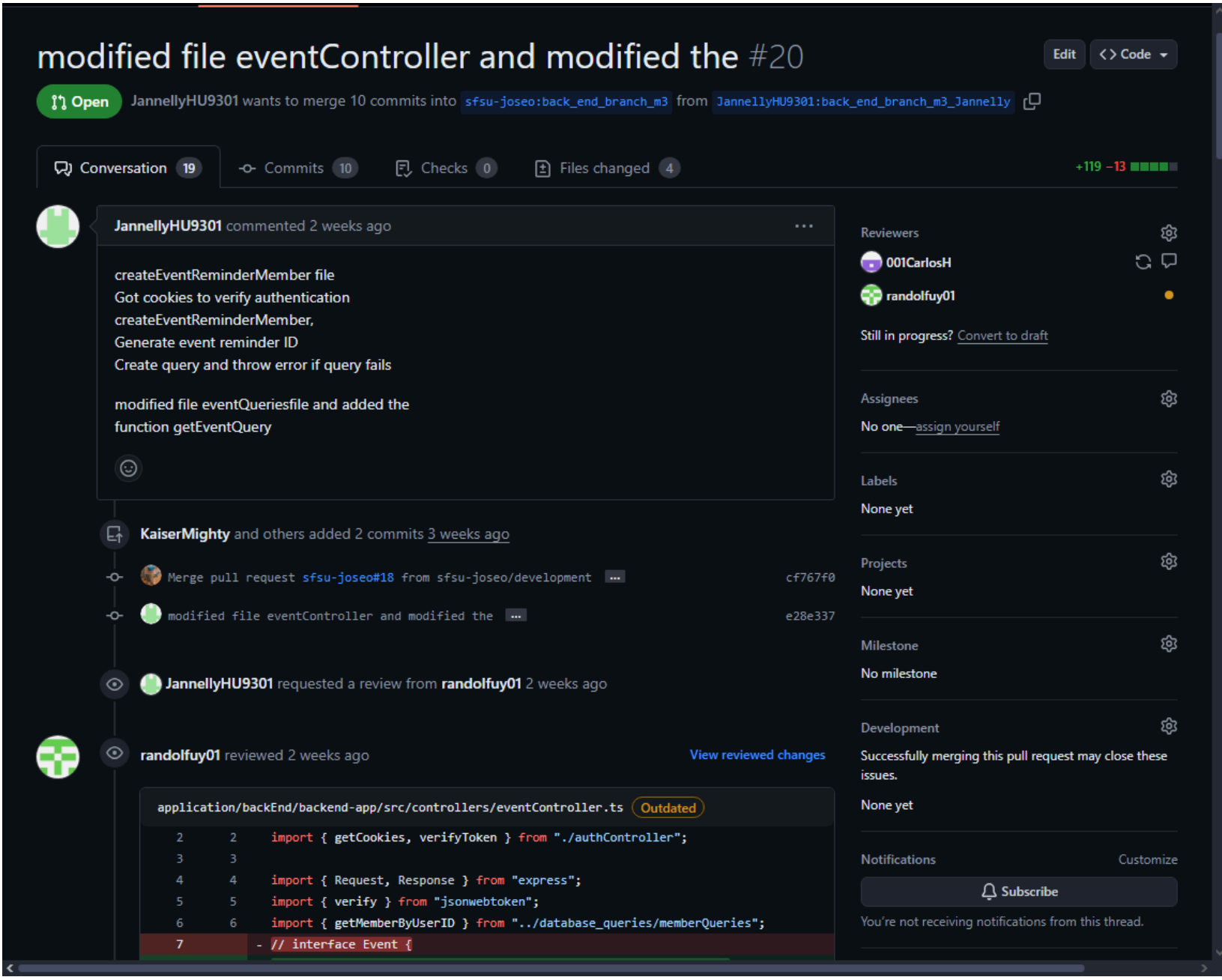
Objective:
Create an account and organization while the application is under load. Ensure the application can support many concurrent users without significant degradation in performance in-line with our performance standards and requirements.

Plan:
The horizontal prototype already supports the features we will be stress testing. We will be using Postman to simulate network traffic. The user will start from the homepage. The user will try registering, logging in, and creating an organization while the database and backend are under stress.
http://ec2-44-223-28-76.compute-1.amazonaws.com:3000/

Number	Description	Test Input	Expected Output	Pass/Fail
1	Registration	New Account	Minimal Delay	Pass
2	Login	Existing Account	Minimal Delay	Pass
3	Create Organization	New Organization	Minimal Delay	Pass

Code Review

Internal Code Review



Open **modified file eventController and modified the** #20

JannellyHU9301 wants to merge 10 commits into `sfsu-joseo:back_end_branch_m3` from `JannellyHU9301:back_end_branch_m3_Jannel…`

```
2  2  import { getCookies, verifyToken } from "./authController";
3  3
4  4  import { Request, Response } from "express";
5  5  import { verify } from "jsonwebtoken";
6  6  import { getMemberByUserID } from "../database_queries/memberQueries";
7     - // interface Event {
      7 + import { getEventQuery } from "../database_queries/eventQueries";
```

randolfuy01 2 weeks ago

getEventQuery should be grouped with the other internal imports like Event, EventReminder, not with verify, request, response

Reply...

Resolve conversation

randolfuy01 reviewed 2 weeks ago — View reviewed changes

`application/backEnd/backend-app/src/controllers/eventController.ts`

```
132 +         try {
133 +             let newEvent = await getEventQuery(eventID);
134 +         } catch (Error) {
135 +             return res.status(500).json({
```

randolfuy01 2 weeks ago

Rather than a query error, if we cannot find the event. This should be an error that the event does not exist

Reply...

Resolve conversation

Notifications — Customize

Subscribe

You're not receiving notifications from this thread.

4 participants

Maintainers are allowed to edit this pull request.

Lock conversation

Open **modified file eventController and modified the** #20

JannellyHU9301 wants to merge 10 commits into `sfsu-joseo:back_end_branch_m3` from `JannellyHU9301:back_end_branch_m3_Jannel…`

Resolve conversation

randolfuy01 reviewed 2 weeks ago — View reviewed changes

randolfuy01 left a comment

Looks good so far, only a few minor issues.

randolfuy01 reviewed 2 weeks ago — View reviewed changes

`application/backEnd/backend-app/src/controllers/eventController.ts`

```
136 +                message: "Error querying event"
137 +            })
138 +        }
139 +
```

randolfuy01 2 weeks ago

now that we verify that the event exists, make sure to create another query to add that eventReminder to the database.

JannellyHU9301 2 weeks ago — Author

Gabe,
Can you please clarify regarding the query to add eventReminder to the database:

Which database table will this record be written?
What attributes will be written to the database?

randolfuy01 2 weeks ago

there should be an eventReminder, reference the table diagram inside milestone 2. If a user is creating an event reminder, presumably you would want to write all attributes to that table

Open **modified file eventController and modified the** #20

JannellyHU9301 wants to merge 10 commits into `sfsu-joseo:back_end_branch_m3` from `JannellyHU9301:back_end_branch_m3_Jannel…`

there should be an eventReminder, reference the table diagram inside milestone 2. If a user is creating an event reminder, presumably you would want to write all attributes to that table into a single row

randolfuy01 2 weeks ago

there isn't an actual query for it yet, you can create one. It would be similar to createMember or createOrganization just with different fields but the syntax should be relatively similar

JannellyHU9301 2 weeks ago Author

By 'Table Diagram' do you mean the EER (Enhanced Entity-Relationship) diagram in the Milestone 2 document?

Reply...

Resolve conversation

`n the file of eventController I modified and ...` `3aef8bf`

JannellyHU9301 requested a review from **randolfuy01** last week

JannellyHU9301 commented last week Author

I modified in the eventQueries file I added insertEventReminderQuery

In the eventController I created the function generateEventReminderID() and created query to check if the event exists and
throw an error if event does not exist in the database

Open **modified file eventController and modified the** #20

JannellyHU9301 wants to merge 10 commits into `sfsu-joseo:back_end_branch_m3` from `JannellyHU9301:back_end_branch_m3_Jannel…`

randolfuy01 reviewed last week — View reviewed changes

`application/backEnd/backend-app/src/controllers/eventController.ts`

```
+ import { Event, EventReminder, Member, Cookie} from "./models";
+ import { getEventQuery } from "../database_queries/eventQueries";
+ import { insertEventReminderQuery } from "../database_queries/eventQueries";
+
  import { getCookies, verifyToken } from "./authController";
```

randolfuy01 last week

This needs to be grouped with the other internal imports

JannellyHU9301 last week — Author

Gabe, I am confused about which line needs to be grouped with the other internal imports?

randolfuy01 last week

Do u see how getCookies and verifyToken have 1 line in between them and the insertEventReminderQuery? The getCookies and verifyToken should be grouped with them.

Reply...

Resolve conversation

randolfuy01 reviewed last week — View reviewed changes

`application/backEnd/backend-app/src/controllers/eventController.ts` Outdated — Show resolved

Open **modified file eventController and modified the** #20

JannellyHU9301 wants to merge 10 commits into `sfsu-joseo:back_end_branch_m3` from `JannellyHU9301:back_end_branch_m3_Jannel…`

ts Outdated Show resolved

randolfuy01 reviewed last week View reviewed changes

`application/backEnd/backend-app/src/database_queries/eventQueries.ts`

```
40 + // This query inserts an event reminder into the EventReminder table in the data
41 + async function insertEventReminderQuery(eventReminder: EventReminder): Promise<a
42 +     return new Promise((resolve, reject) => {
```

randolfuy01 last week

Have Carlos review the database logic behind this

JannellyHU9301 last week Author

@001CarlosH Gabe said if you can review the logic behind this function

001CarlosH last week

the query for the get events is correct

001CarlosH last week

also make sure to have the db set up on your own system, you need to be capable of testing your own functions you create. its difficult for me to run it because i have 3-classes and personal responsibilites.

Reply...

Resolve conversation

randolfuy01 reviewed last week View reviewed changes

Open **modified file eventController and modified the** #20

JannellyHU9301 wants to merge 10 commits into `sfsu-joseo:back_end_branch_m3` from `JannellyHU9301:back_end_branch_m3_Jannel…`

randolfuy01 reviewed last week — View reviewed changes

randolfuy01 left a comment

Looks promising, needs to be tested and working and updated a little more

JannellyHU9301 and others added 2 commits last week

- `Fixed the imports in file eventController` … `baf8665`
- `Merge branch 'sfsu-joseo:master' into back_end_branch_m3_Jannelly` `b4b8a1a`

JannellyHU9301 commented last week — Author

Fixed the imports in EventController file
Added triple equal sign, to check for explicit equivalence in lines 116-118 0f the eventController file

Asked Carlos to check the database logic behind the query for the get events and he said it was correct

JannellyHU9301 requested a review from **randolfuy01** last week

JannellyHU9301 added 5 commits last week

- `Added the eventRoutes file to post the event` … `da0b009`
- `Updated files` `b935930`
- `Merge branch 'back_end_branch_m3_Jannelly' of https://github.com/Jann…` … `3158553`
- `Added the event routes file` … `3dbb64b`

Open **modified file eventController and modified the** #20

JannellyHU9301 wants to merge 10 commits into `sfsu-joseo:back_end_branch_m3` from `JannellyHU9301:back_end_branch_m3_Jannel…`

Fixed the imports in EventController file
Added triple equal sign, to check for explicit equivalence in lines 116-118 0f the eventController file

Asked Carlos to check the database logic behind the query for the get events and he said it was correct

JannellyHU9301 requested a review from **randolfuy01** last week

JannellyHU9301 added 5 commits last week

Added the eventRoutes file to post the event ...	da0b009
Updated files	b935930
Merge branch 'back_end_branch_m3_Jannelly' of https://github.com/Jann… ...	3158553
Added the event routes file ...	3dbb64b
Uptaded insertEventReminderQuery function in the ...	b279278

Add more commits by pushing to the **back_end_branch_m3_Jannelly** branch on **JannellyHU9301/csc648-848-05-sw-engineering-su24-T05_2**.

Review requested Show all reviewers

Review has been requested on this pull request. It is not required to merge.
Learn more about requesting a pull request review.

1 pending reviewer

This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request

You can also open this in GitHub Desktop or view command line instructions.

External Code Review

A **Aaron Jacob Saeteurn Rayray**
To: Tharun Krishna

Reply | Reply all | Forward

Tue 7/30/2024 11:18 AM

Hey Tharun,

Our team has analyzed your code and left the following:

- Likes the well organization within each files
- Likes the use of using TS for better consistency
- Liked how the passwords were being hashed before storage, leading to better security
- Error handling could be improved, instead of error registering user, could specify where error is actually coming from to improve
- Integrating more comments to help testers who don't understand what's going on to still be able to acknowledge the information

Thank you for reviewing our code and we look forward to seeing your guys work at the end of the semester!

- Aaron, Team 01

From: Tharun Krishna <tkrishna1@sfsu.edu>
Sent: Tuesday, July 30, 2024 9:47 AM
To: Aaron Jacob Saeteurn Rayray <arayray@sfsu.edu>
Subject: Re: CSC648 External Code Review

Hey Aaron,
Our backend lead has reviewed your files and shared with me the following feedback:

- There could be specific comments that communicate what routers do what to an first-time reader.
- Authentication and security is spread out over 3 different files, this makes it harder to track.
- There could be more decoupling within the code. Everything is so interdependent on each other that making changes to the one class could affect everything else.
- The user class could be separated out, I feel that it's a little dense at the moment, that affects it's readability.
- It appears that there is a split between functional programming and object oriented programming, you could try to consolidate the styles for consistency.

We appreciate you taking the time to conduct a code review with us, and wish you good luck with the rest of the project!

Thanks,
Krishna
Team 05

From: Aaron Jacob Saeteurn Rayray <arayray@sfsu.edu>
Sent: Monday, July 29, 2024 8:21 PM
To: Tharun Krishna <tkrishna1@sfsu.edu>
Subject: Re: CSC648 External Code Review

Hi Tharun,
Attached is our sign up module that deals with authentication and email verification. Would be appreciated if you guys could review the email verification portion of our signup process. I will get back to you either by tonight or tomorrow morning with your code review, thank you! (code will be sent over discord as it got blocked on here)

From: Tharun Krishna <tkrishna1@sfsu.edu>
Sent: Thursday, July 25, 2024 2:20 PM
To: Aaron Jacob Saeteurn Rayray <arayray@sfsu.edu>
Subject: CSC648 External Code Review

Hi Aaron,
I saw your message about the external code review and would like to take you up on your offer! I've attached a package that contains our user authentication logic, with the main file being authController.ts, for review. I've included three additional files that authController.ts imports for your reference as needed. Please feel free to reply to this email with a package or files of your own, and we will review them and get back to you ASAP!

Thanks,
Krishna
Team 05

Security Self-Check

In-line with modern standards, our application does not store unsecured plaintext passwords, but instead stored a secure hashed password to keep users' accounts safe.

When the backend receives the password, our controller calls a hashing function.

```
// HASH AND VERIFY HASHING PASSWORD
const hashedPassword:string = await hashPassword(registerUser.password);
const verify = verifyPassword(registerUser.password, hashedPassword);
    if (!verify) {
        return res.status(400).json({
            error: 'Password Hashing unsuccessful'
        });
    }
```

The hashing function uses bcrypt, and goes as follows:

```
import bycript from 'bcrypt';

// HASHES PASSWORD
async function hashPassword(password: string): Promise<string> {
    return await bycript.hash(password,8);
}

// VERIFIES PASSWORD
async function verifyPassword(password: string, hashPassword: string):
Promise<boolean> {
    return await bycript.compare(password, hashPassword);
}
```

This hashed password is what is passed in the MySQL query and stored in the DB:

userID	email	username	password	profilePicture	firstName	lastName
9141	testing@email.com	JohnDoe	$2b$08$UFJrQUcykSWjhQNAYzR4D./30625/63mgTZUGnHPJajNyZqGnGKUy	NULL	John	Doe
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Our search bar input data are validated to make sure we are searching the right tables in the database:

Our autocomplete function is as follows:

```
export const autoCompleteSearch = async (req: Request, res: Response) => {
    try {
        try {
            let token = req.cookies.token;
            let decoded = jwt.verify(token, JWT_SECRET_KEY);
            if (decoded === null) {
                throw new Error('Unauthorized User');
            }
        } catch (Error) {
            res.status(401).json({
                error: 'Unauthorized User'
            });
        }
        let searchQuery: string = req.body.searchQuery;
        // Query words from the database
        let members: Member[] = await
getMembersFromOrganizationQuery(req.cookies.orgID);
        let queryWords: string[] = members.map((members) =>
members.firstName + " " + members.lastName);
        const completer = new Trie();
        for (const word of queryWords) {
            completer.insert(word);
        }
        const prefix: string = searchQuery;
        const result: string[] = completer.search(prefix);
        res.status(200).json({
            queryResults: result
        });
    } catch(Error) {
        res.status(400).json({
            error: 'Error searching for topic'
        });
    }
}
```

Out ranked search function is as follows:

```
export const rankedSearch = async (req: Request, res: Response) => {
    try {
        let cookie: Cookie = await getCookies(req);
        let verified = verifyToken(cookie.token);
        if (verified === null) {
            return res.status(401).json({
                error: "Unauthorized User"
            });
        }
        let searchQuery: Search = req.body;
        let org = req.cookies.orgID;
        if (searchQuery.searchType === "members") {
            let members: Member[] = await
getMembersFromOrganizationQuery(org);
            let rankedMembers: any = rankSearched(searchQuery.searchQuery,
members.map((member) => member.firstName + " " + member.lastName));
            let sortedMembers: Member[] = rankedMembers.map((rankedMember:
{ item: string, score: number }) => {
                return members.find(member => (member.firstName + " " +
member.lastName) === rankedMember.item);
            }).filter((member: Member | undefined): member is Member =>
member !== undefined);
            return res.status(200).json({
                message: "Search for members",
                queryResult: sortedMembers
            });
        }
        // Truncated for formatting, but we have an else if for every case
    } catch(Error) {
        return res.status(400).json({
            error: 'Error searching for topic'
        });
    }
}
```

Non-Functional Specs Self-Check

Performance

- **DONE -** The application shall respond quickly to user prompts with minimal latency.
- **DONE -** The application shall be able to host/handle up to 1,000 accounts without any performance problems.
- **DONE -** The application shall support many active-concurrent users without significant degradation in performance.

Security

- **DONE -** The application shall only store hashed passwords.
- **ON TRACK -** The application database shall be backed up regularly.

Coding Standard

- **DONE -** The application shall follow consistent coding and naming conventions throughout.
- **DONE -** The application shall gracefully handle errors without detrimentally affecting the rest of the user experience.
- **ON TRACK -** The application shall be well documented for easy understandability for other developers and admins alike.

Media Storage

- **ISSUE -** The application shall restrict profile pictures to 500x500 pixels and 5 megabytes or less.
    - Images are not part of our priority 1 requirements, and as such, are not included in our prototype.

Privacy

- **DONE -** The application shall hide organizations, preventing users not in the organization from viewing it.
- **DONE -** Data shall be anonymized and pseudonymized wherever possible to minimize damage in the event of a leak.

- **DONE -** Only the minimum amount of data required shall be stored in order to minimize the risk of security breaches.

Scalability

- **DONE -** The application shall be designed to scale efficiently to handle an increasing number of users and data without performance degradation.
- **DONE -** The database shall scale down if the workload is low to remain cost effective and optimize resource utilization.

Reliability

- **DONE -** The application shall ensure high availability, with a 99.9% uptime, and incorporate robust error-handling mechanism to maintain stability.

Usability

- **DONE -** The application shall provide an user friendly interface, ensuring that users can easily navigate and use all features without extensive training.

Compliance

- **DONE -** The application shall comply with relevant industry standards and regulations, such as for data protection and privacy.

Expected Load

- **DONE -** The application shall respond to user queries within 500 milliseconds for requests.
- **DONE -** The application shall handle a minimum of 1000 transactions per second during peak hours.
- **DONE -** The application shall horizontally scale to support up to 10,000 concurrent users without degradation in response time.

Fault Tolerance

- **DONE -** The application shall recover from a server failure within 15 minutes without loss of data or interruption of service.
- **DONE -** Critical components of the application shall have a backup system that can take over within 5 minutes of a primary system failure.
- **DONE -** The application shall log all errors and provide automatic recovery for non-critical errors within 60 seconds.

Database High Level Specs

- **DONE -** The database shall use appropriate indexes to optimize query performance, especially for frequently accessed data.
- **DONE -** The database should be capable of storing up to 1 terabyte of data efficiently, with automatic data compression for storage optimization.
- **DONE -** All sensitive data stored in the database should be encrypted.

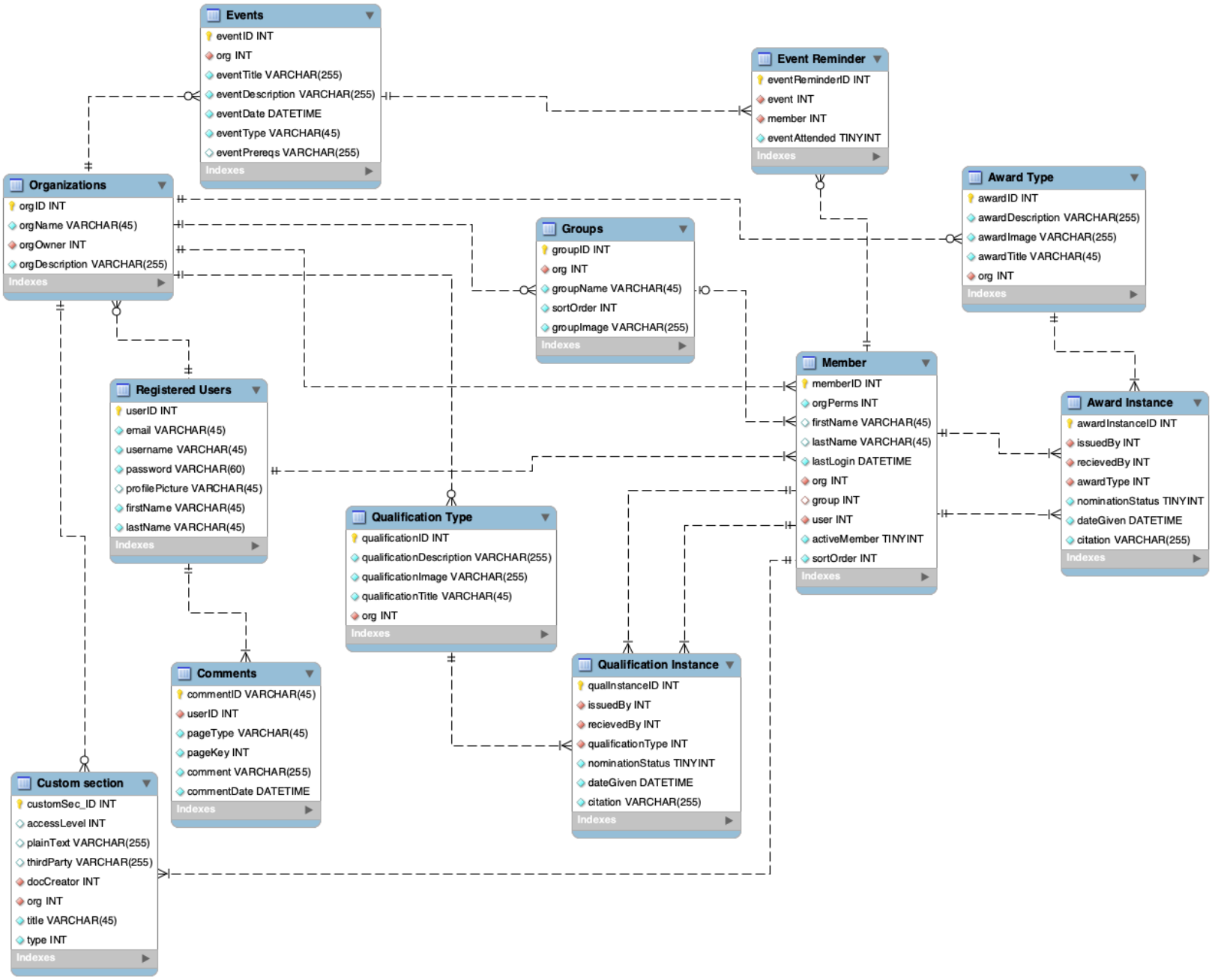
Browser and OS Support

- **DONE -** The application shall support Chrome Stable 126 (126.0.6478.153).
- **DONE -** The application shall support Windows 10 22H2 (19045.4651).
- **DONE -** The application shall support Windows 11 23H2 (22631.3737).
- **DONE -** The application shall support macOS Sonoma (14.5).
- **DONE -** The application shall support Amazon Linux (2023.4.20240611.0).

Additional Comments

Changes to the DBMS

Adding a Title (VARCHAR 45) and Type (INT) field to the Custom Section entity in order to differentiate between Projects/Tasks and Highlights.



Team Contributions

- **Krishna**
  - Revised Product Summary Section of Documentation.
  - Usability Test Plan Section of Documentation.
  - QA Test Plan Section of Documentation.
  - Security Self-Check Section of Documentation.
  - Non-Functional Requirements Self-Check Section of Documentation.
  - Group page frontend of Beta Prototype.
  - Recognitions page frontend of Beta Prototype.
-
  - Prototype revisions as per horizontal prototype feedback.
  - General restyling of all pages.
  - Oversaw other front-end developers progress on static pages.
  - Refined static pages.
  - Tasks/Projects page frontend of Beta Prototype.
  - Events page frontend of Beta Prototype.
  - Highlights page frontend of Beta Prototype.
-
  - Leading efforts on Backend of the Prototype.
  - Revised search controller of Beta Prototype.
  - Created recognition controller of Beta Prototype.
  - Created custom sections controller of Beta Prototype.
  - Revised password hashing functions.
  - Added various routing components for the backend.
  - Miscellaneous bug fixing on the backend component.
  - Completed Internal Code Review.
  - Complete External Code Review.
-
  - Revised custom sections queries of Beta Prototype.
  - Revised custom sections controller of Beta Prototype.
  - Adjusted DBMS files (EER and .sql files).
  - Revised group queries of Beta Prototype.
  - Miscellaneous bug fixing on the backend component.

-
    - Revised Case Studies page of Beta Prototype.
    - Revised Reviews page of Beta Prototype.
    - Revised Updates page of Beta Prototype.
    - Revised Server Status page of Beta Prototype.
    - Revised Features page of Beta Prototype.
    - Revised Pricing page of Beta Prototype.
    - Revised About page of Beta Prototype.
-
    - Revised Contact Us page of Beta Prototype.
    - Revised Chat Support page of Beta Prototype.
    - Revised Report a Bug page of Beta Prototype.
    - Revised Careers page of Beta Prototype.
    - Revised Culture page of Beta Prototype.
    - Revised Blog page of Beta Prototype
    - Revised Help Center page of Beta Prototype
-
    - Product Summary Description Component of Documentation.
    - Product Summary Superior Features Component of Documentation.
    - Product Summary Competitors Component of Documentation.
-
    - Product Summary Committed Functions Component of Documentation.
    - Contributed to Internal Code Review.
    - MIscellaneous contributions to backend API.

Post Analysis

During the project, our main challenges were overcoming inconsistencies in the coding aptitude of team members. Some team members were very capable, while others struggled with trivial tasks. We ended up dividing up the work in such a way that trivial tasks were handed off to those that were less capable; but as we progressed, we ran out of trivial tasks to handoff. We tried giving these members more sophisticated and impactful tasks, but a lot of the contributions made were inconsistent with what our mockups and wireframes depicted. At some point the more capable team members decided that it would just be more efficient to do most of the work ourselves, rather than having to revise everything multiple times.

A good portion of the team also lacked many of the baseline fundamental web development skills, and we struggled with team members submitting code that was more-likely-than-not generated by ChatGPT. These files needed to be redone, which reduced productivity of our more capable team members because they had to redo the work of others. It felt unfair to simply deny work to some members of the team, so I opted to give the opportunities to prove themselves with the understanding that their work would likely have to be revised anyway. A lot of time and bandwidth was expended trying to mentor team members through their development tasks, which were fairly trivial and simple in all actually. It's possible that with a smaller team, we might have been more efficient and productive, since we wouldn't have had to hold the hands of others as we moved along.

If I had to recreate the project from scratch, I would definitely include Tailwind for CSS in our tech stack. Our stylesheets had a lot of similar code that became tedious to write out. Tailwind, in conjunction with traditional CSS, would have made life a lot easier. We could have used traditional CSS and stylesheets for reused elements, and Tailwind for unique one-of-a-kind stylings to keep the stylesheet readable and small. Without tailwind, the stylesheets became very unwieldy and bloated; many times a stylesheet change intended for a certain component would end up changing something in another page entirely. Tailwind would have definitely boosted our productivity and efficiency.

Another issue was project bloat, due to the tight deadlines around summer semester, we had to move through a lot of the initial setup hastily. Less than ideal choices and foundations were created which worked at the time, but created issues during milestone 3. The free AWS EC2 instance did not have enough memory to actually compile our project. I ended up upgrading to a paid version because of the tight deadline, but if we had followed good practices early on, it's likely that we could have gotten away with just using the free AWS EC2 instance.

Team Contributions

- **Krishna**
  - Document editing and formatting for all milestones.
  - Contributed use cases, requirements, and data entities to milestone 1.
  - Created, managed, and migrated AWS cloud instances.
  - Notion creation and updating for each milestone.
  - Revised dashboard pages in-line with Figma wireframes.
  - Merging separate stylesheets for core functions into shared stylesheets.
  - Groups and recognition page backend connection.
  - Nearly all sections of the milestone 4 documentation.
-
  - Contributed use cases, all diagrams, and comp analysis to milestone 1.
  - Created all storyboards and mockups for milestone 2.
  - Created all wireframes for milestone 3.
  - Coordinated frontend team with prototypes, static pages, and revisions.
  - Created home, signing, dashboard, and other new user facing pages.
  - Created all changes related to the milestone 3 prototype feedback.
  - Events, tasks/projects, and highlights backend connection.
-
  - Created all backend/system/scalability/API design summaries and diagrams for milestone 2.
  - Restructured repo and application in-line with tech stack definition.
  - Coordinated and mentored other developers with backend interfaces.
  - Conducted internal and external code review for milestone 4.
  - Created extensive authentication and search backend components.
  - Created all other backend interfaces, controllers, and routings.
  - Enhanced application security with tokens, hashing, and cookies.
-
  - Created Database definitions, requirements, ERDs, EERs, and forward engineering for DBMS milestone 2.
  - Revised DBMS EER and forward engineering as needed for milestones 4.
  - Database to backend controller interfacing.
  - Created and revised database SQL queries for all prototypes.
  - Various bug fixes in backend controllers and models.
  - Coordinating with and mentoring other backend developers.
  - Assisted with connecting backend interfaces with the frontend.

-
    - Contributed use cases, requirements, and comp analysis to milestone 1.
    - Created all network diagrams for milestone 2.
    - Created initial header, footers, pop ups, and corresponding styling.
    - Revised front page footer to final version.
    - Contributed to the initial recognitions page.
    - Contributed to contact us, chat support, report a bug, careers, culture, blog, and help center static pages.
-
    - Contributed use cases, requirements, and comp analysis to milestone 1.
    - Contributed to database ERD diagram in milestone 2.
    - Created initial dashboard sidebar.
    - Contributed to initial dashboard, members, tasks, and events page.
    - Contributed to case studies, reviews, updates, server status, features, pricing, and about static pages.
-
    - Contributed use cases, requirements, and comp analysis to milestone 1.
    - Polishing and editing the backend section of milestone 2 documentation.
    - Contributed to the project summary section of milestone 4 documentation.
    - Contributed to user profile backend API.
-
    - Contributed use cases, requirements, and comp analysis to milestone 1.
    - Contributed to Internal Code Review for milestone 4.
    - Contributed to the project summary section of milestone 4 documentation.
    - Contributed to organization backend API.