



**School of Engineering, Computer Science, and Mathematics**

**Senior Design Project Report**

**Jamr: A Game Jam Team Collaboration/Project Management Application**

**A project submitted in partial fulfillment of the requirements for the degree of**

**BACHELOR OF SCIENCE IN COMPUTER SCIENCE**

**(Software Engineering Track)**

**By**

Logan S. Fite 1013206

Jonathan A. Myers 1021003

Daniel Z. Hill 1015908

**Supervised by**

Dr. Mohammad Faridul Haque Siddiqui

# Table of Contents

<b>Table of Contents</b>	<b>2</b>
<b>List of Figures</b>	<b>6</b>
<b>Abstract</b>	<b>8</b>
<b>1. Introduction</b>	<b>9</b>
<b>2. Background</b>	<b>10</b>
<b>3. Project Proposal/Definition</b>	<b>9</b>
3.1 Problem Statement	9
3.2 Objective	9
3.3 Approach	10
<b>4. Project Plan</b>	<b>10</b>
4.1 - Process Model	10
4.2 - Organization of the Project	11
4.3 - Management Activities	11
4.4 - Risks	11
4.5 - Methods and Techniques	12
4.6 - Work Packages	12
4.7 - Resources	13
4.8 - Schedule/Timeline	13
4.9 - Delivery	14
<b>5. Feasibility &amp; Software Requirements Specification</b>	<b>14</b>
5.1 Introduction	14
5.1.1 Purpose	14
5.1.2 Intended Audience and Reading Suggestions	14
5.1.3 Project Scope	14
5.2. Overall Description	15
5.3 System Requirements	17
5.4 Functional Requirements/System Features	18
5.4.1 List of Use Cases	18
5.4.1.1 Public User Use Cases - Web Application	18
5.4.1.2 Team Member Use Cases - Web Application	20
5.4.1.3 Team/Project Leader Use Cases - Web Application	22

5.4.1.4 Public User Use Cases - Mobile Application	23
5.4.1.5 Team Member/Leader Use Cases - Mobile Application	24
5.5 External Interface Requirements	25
5.5.1 User Interfaces	25
5.6 Other Non-Functional Requirements	28
5.6.1 Performance Requirements	28
5.6.2 Usability	28
5.6.3 Security Requirements	28
5.6.4 Software Quality Attributes	29
<b>6. System Design</b>	<b>29</b>
6.1 Introduction	29
6.2 Design Overview	29
6.2.1 Description of the Problem	29
6.2.2 Technologies Used	30
6.3 Objects	30
6.3.1 Object List and Descriptions	30
6.3.2 Object Relations	32
6.4 System Architecture	33
6.4.1 System Design	33
6.4.1.1 Implementation Approach: The Model-View-Controller Pattern	33
6.4.1.2 Middleware Authentication and Authorization	33
6.4.1.3 System Communication	34
6.4.1.4 Functional Component Design	36
6.4.2 Components/Subsystems Design	36
6.4.2.1 Web Application User Login	36
6.4.2.2 Mobile User Login	38
6.4.2.3 User Registration	39
6.4.2.4 Home Page	40
6.4.2.5 Create Project	41
6.4.2.6 Matching	42
6.4.2.7 Manage Invites	46
6.4.2.8 Manage Candidates	47
6.4.2.9 Project Dashboard	48
6.4.2.10 Project Dashboard - Calendar	49
6.4.2.11 Project Dashboard - To Do List	50
6.4.2.12 Project Dashboard - Assets/File Sharing	51

6.4.2.13 Project Dashboard - Development Log	52
6.4.2.14 Project Dashboard - Project Management	53
6.4.2.15 Project Dashboard - Whiteboard	54
6.4.2.16 Project Dashboard - Group Chat	55
6.4.2.17 Project Dashboard - Member Settings	56
6.4.2.18 User Settings	57
6.4.2.19 Profile	58
6.4.2.20 Friends	60
6.4.2.21 Messaging	62
6.4.2.22 Web Navigation Bar	64
6.4.2.23 Mobile Navigation Bar	65
6.5 Overall Application Operation	66
6.6 Entity Relationship Data Model	68
<b>7. Implementation</b>	<b>69</b>
7.1 Subsystems	69
7.1.1 User Login - Web Application	69
7.1.2 User Login - Mobile Application	70
7.1.3 User Registration	70
7.1.4 Middleware Token Authentication	71
7.1.5 Home Page	73
7.1.6 Create Project	74
7.1.7 Find a Team	75
7.1.8 Find Members	76
7.1.9 View Project Invites	77
7.1.10 Manage Candidates	78
7.1.11 Project Dashboard	79
7.1.12 Project Dashboard - Calendar	83
7.1.13 Project Dashboard - To Do List	85
7.1.14 Project Dashboard - Assets/File Sharing	87
7.1.15 Project Dashboard - Development Log	90
7.1.16 Project Dashboard - Whiteboard	92
7.1.17 Project Dashboard - Group Chat	94
7.1.18 Project Dashboard - Project Management	95
7.1.19 User Settings	98
7.1.20 User Profile	99
7.1.21 Friends	100

7.1.22 Messaging	101
7.1.23 Web Navigation Bar	102
7.1.24 Mobile Navigation Bar	103
7.2 Technologies	104
7.2.1 MongoDB	104
7.2.2 Express JS	104
7.2.3 React JS	105
7.2.4 React Native	106
7.2.5 Node JS	106
7.2.6 Axios	107
<b>8. Testing</b>	<b>108</b>
8.1 Testing Plan	108
8.2 Test Case Results	109
<b>9. Results</b>	<b>137</b>
<b>10. Summary and Future Work</b>	<b>145</b>
<b>11. References</b>	<b>146</b>
<b>12. Scrum Report</b>	<b>148</b>
12.1 User Stories	148

## List of Figures

Figure 4.1 - Iterative development model	10
Figure 5.1 - Unregistered User Use Cases for Web Application	18
Figure 5.2 - Team Member's Dashboard Use Cases for Web Application	20
Figure 5.3 - Team Member and Leader Interaction Use Cases for Web Application	22
Figure 5.4 - Registered Member Use Cases for Mobile Application	23
Figure 5.5 - Team Member and Leader Use Cases for Mobile Application	24
Figure 5.6 - Landing Page UI Concept	25
Figure 5.7 - Matching Page & Navigation Bar UI Concept	26
Figure 5.8 - Project Dashboard Page UI Concept	26
Figure 5.9 - Mobile Format Concept of Home, Matching, and Dashboard Pages	27
Figure 6.1 - Object Relations Diagram	32
Figure 6.2 - Execution flow for each individual component of the application	33
Figure 6.3 - Initial Token Design Sequence Diagram	34
Figure 6.4 - Socket Room Example	34
Figure 6.5 General Socket Connection Flow	35
Figure 6.6 - Initial Web Login Sequence Diagram	37
Figure 6.7 - Initial Mobile Login Sequence Diagram	38
Figure 6.8 - User Registration Sequence Diagram	39
Figure 6.9 - User Registration Sequence Diagram	40
Figure 6.10 - Initial Create Project Sequence Diagram	41
Figure 6.11 - User Find Team Sequence Diagram	42
Figure 6.12 - Leader Find Member Sequence Diagram	43
Figure 6.13 - User Matching Activity Diagram	44
Figure 6.14 - Project Leader Matching Activity Diagram	45
Figure 6.15 - Manage Invites Sequence Diagram	46
Figure 6.16 - Manage Candidates Sequence Diagram	47
Figure 6.17 - Initial Project Dashboard Sequence Diagram	48
Figure 6.18 Dashboard Component Diagram	49
Figure 6.19 - Initial Calendar Sequence Diagram	50
Figure 6.20 - Initial To-Do List Sequence Diagram	51
Figure 6.21 - Initial Asset Sequence Diagram	52
Figure 6.22 - Development Log Sequence Diagram	53
Figure 6.23 - Initial Project Management Sequence Diagram	54
Figure 6.24 - Initial Whiteboard Sequence Diagram	55
Figure 6.25 - Initial Group Chat Sequence Diagram	56

Figure 6.26 - Initial Member Settings Sequence Diagram	57
Figure 6.27 - Initial User Settings Sequence Diagram	58
Figure 6.28 - Initial User Profile Sequence Diagram	59
Figure 6.29 - Initial Friends Sequence Diagram	60
Figure 6.30 - Friends Activity Diagram	61
Figure 6.31 - Initial Messaging Sequence Diagram	62
Figure 6.32 - Messaging Activity Diagram	63
Figure 6.33 - Initial Web Navigation Bar Sequence Diagram	64
Figure 6.34 - Initial Mobile Navigation Bar Sequence Diagram	65
Figure 6.35 - Overall Web Application Diagram	66
Figure 6.36 - Overall Mobile Application Diagram	67
Figure 6.37 - Entity Relationship Data Model	68
Figure 7.1 - Revised Middleware Token Authentication Sequence Diagram	71
Figure 7.2 - Revised Project Dashboard Sequence Diagram	80
Figure 7.3 - Subsystem Communication Diagram for Project Dashboard	81
Figure 7.4 - Revised Calendar Sequence Diagram	83
Figure 7.5 - Revised To-Do List Sequence Diagram	85
Figure 7.6 - Revised Asset Sequence Diagram	88
Figure 7.7 - Revised Development Log Sequence Diagram	90
Figure 7.8 - Revised Whiteboard Sequence Diagram	92
Figure 7.9 - Revised Project Management Sequence Diagram (Member)	95
Figure 7.10 - Revised Project Management Sequence Diagram (Leader)	96
Figure 7.11 - Mongo Schema & Collection Object	104
Figure 7.12 - Express Routes	105
Figure 7.13 - React Page for Displaying User Profile	105
Figure 7.14 - Example of a React Native Render Method	106
Figure 7.15 - Server Code to Get Project Calendar Events	106
Figure 7.16 - Axios GET request to fetch candidate profiles for a project	107
Figure 7.17 - Axios POST request to add a new candidate to a project	107
Figure 9.1 - Landing Page	137
Figure 9.2 - User Registration Page	138
Figure 9.3 - Project Dashboard - All Components (Web)	139
Figure 9.4 - Project Dashboard - All Components (Mobile)	140
Figure 9.5 - Matching Page - Web & Mobile Views	142
Figure 9.6 - Project Management	143
Figure 9.7 - Profile Page	144

## **Abstract**

Game jam events are a common way for individuals interested in the game development industry to gain hands-on experience through participation. Programmers, artists, writers, and other creatives are often tasked to form short-term teams and undergo the full process of the game development flow from initial planning to product testing. These teams may be assigned at random or team leaders may recruit individual contributors in an effort to speed up the development process. Similar to hackathons, game jams are often time restricted and enforce some unique requirement, challenging users to work quickly. The user-lead nature of these events, however, directly results in inefficiencies caused by manual partner searching and project management decisions. Jamr proposes to solve this problem by creating a system that directly integrates with the most popular game jam hosting site itch.io, providing functionalities for users to find a project that matches their skillset or to create a project and find members capable of completing the project, along with managing the project entirely through the application through a suite of project collaboration tools.



## **1. Introduction**

This document serves to introduce the problem statement for which the project Jamr addresses, and provides extensive information about each stage of the project development from initial planning phases to showcasing a functional prototype with details on further implementation. Design and implementation considerations are carefully addressed, with hopes that the information provided in this report thoroughly describe the methods and technologies used to build a functional deliverable that offers a solution to the problem.

## **2. Background**

This project focuses on providing a combination of tools to the game jam community, who may benefit greatly upon completion of the project. Game development is one area of programming which many developers have an interest in, ourselves included, and game jam events offer exposure to the game development industry through short term contests. Participants are often encouraged to form teams of contributors who they may not know personally in order to complete their projects on time. The user-lead nature of these events and our own experiences with game jams inspired the project Jamr, an application designed to minimize the inefficiencies of manual partner searching and collaboration tool organization.

## **3. Project Proposal/Definition**

### **3.1 Problem Statement**

Itch.io is a leading platform for hosting game jams and giving participants of all skill levels the ability to gain game development experience. However, little support is given for participants to find their teams and manage their projects. Each jam is hosted by a different sponsor or community, making it difficult for users to develop a standard for efficient means of collaboration and project management. Jams are often short on time, yet forming a team can take days. Additional time is wasted in preparing the project in a way that is convenient to all contributors to manage the project.

### **3.2 Objective**

The objective of Jamr is to not only allow users who partake in itch.io game jams to easily find group members with various and specific skill sets, but also to manage the game project entirely within the application in order to finish the numerous aspects of game design in the short time frame allotted by a jam.

### 3.3 Approach

Through Jamr, Each user will have a profile accessed with itch.io's OAuth that has information about their experience, skills, and past jam scores. Users will be able to further modify their profile and create projects within the application that may be visible to other users as potential contributors. After a user requests to be a part of the posted project, the poster can sort through the applicants, assess their skills, and decide whether or not to contact them for help on the team. Upon project creation, a dedicated project dashboard accessible by all members provides the functionality to manage all aspects of the project and keep track of deadlines, eliminating the need for using a combination of external tools. Planned features include a shared calendar, a central assets module for uploading game resources such as artwork or music, task management with tags to assign and track responsibility progress, a collaborative whiteboard, project development logs, and a group text-chat for communication.

## 4. Project Plan

### 4.1 - Process Model

The process model the team has resolved to is the Iterative Model. This model demonstrates an initial, simplified implementation of the project which over time will become more complex and broader as the team progresses to the final product. Due to the uncertainty of deadlines caused by the pandemic, this structure will allow the team to focus on developing a functioning deliverable with as many features as time allows.

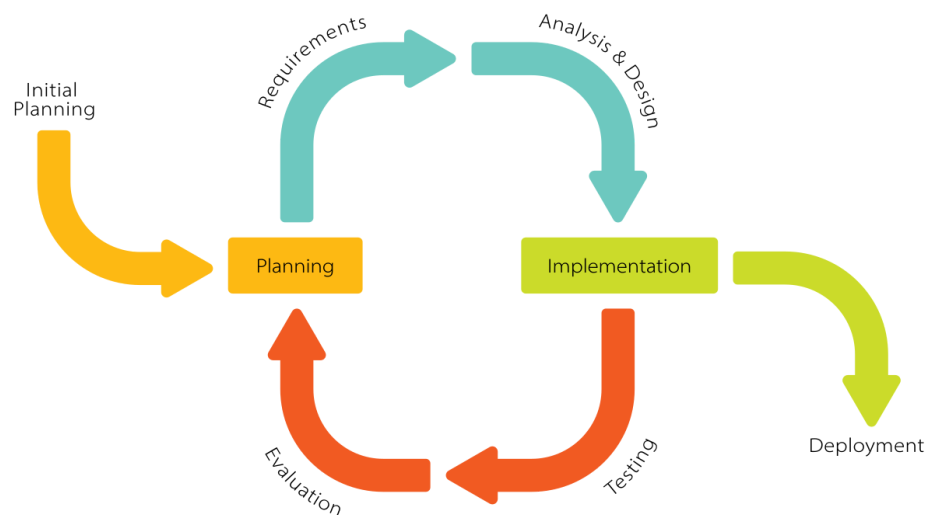


Figure 4.1 - Iterative development model

[https://en.wikipedia.org/wiki/Iterative\\_and\\_incremental\\_development#/media/File:Iterative\\_Process\\_Diagram.svg](https://en.wikipedia.org/wiki/Iterative_and_incremental_development#/media/File:Iterative_Process_Diagram.svg)

Under this model, smaller groups of feature requirements can be defined, implemented, and tested before expanding into further functionality to meet milestone objectives. Required milestones to be achieved by deliverable presentation due date include Itch.io API authorization/authentication for login and server requests, user chat capabilities, user profile customization and team finding functionality using a match/abandon system, and team formation with project management in the form of deadline timers with task and asset management.

Continuing efforts will also be made to achieve additional milestones adding advanced management functionality, modern UI/UX design, and deploying the web application. The same approach will be taken within the second semester to develop a corresponding mobile application made in React Native with a functioning deliverable deployed upon completion. Because efforts are required to implement the same features on a different architecture that must communicate with the same application, similar milestones apply. The iterative process model helps ascertain when these milestones are reached by confirming that each group of functionality modules have individually undergone the complete development process and are tested for errors when combined within the application.

## **4.2 - Organization of the Project**

In support of the iterative process model in an unfamiliar environment, an agile/SWAT team project organization will be used. This approach allows each member to divide feature implementation responsibilities while still gaining experience in all areas of developing within the MERN stack. Each member will be responsible for implementing their own assigned tasks and acting as a manager of the project to help ensure completion of the project.

## **4.3 - Management Activities**

Regular project progress reports are required to guide the constant development of the project and to push for consideration of priorities. Progress reports include feasibility analysis and requirements specification, design and implementation documentation, and complete testing.

## **4.4 - Risks**

There are many risks that could arise during the development of the web application such as integration with 3rd party systems (itch.io API), data migration/population, underestimation of the time and resources used for development, new features that may be brought up during production without reasonable time to implement them, user data security, and poor team cohesion. All of these contribute to the risks that need to be accounted for and diminished during the development of the web application.

## 4.5 - Methods and Techniques

The application will be developed within two repositories (Web and Mobile Application) managed by GitHub for version control using ESLint to enforce modern JavaScript writing standards. This keeps code between components written by each team member consistent, easing the cost of maintenance and testing. Model definitions and database entity relation diagrams will also be produced for reference during implementation to ensure efficient design and increase the rate of development.

During implementation, the primary means of ensuring efficiency and correctness is dependent upon the programmer. At each stage of development, the team will test individual components as part of the life cycle method for unit testing along with their combinations within the app to ensure they integrate within the application. Testing through the use of tools like Insomnia will help identify unexpected component errors with user inputs and module interactions, asserting that required content is always rendered quickly and responsively on different resolutions, and ensuring API success.

## 4.6 - Work Packages

The project is composed of the following work packages:

- Design: This includes class/model definitions, ER diagrams, and UI element prototypes
- Test Plan: Divided into frontend and backend aspects
  - Frontend: Input validation, authentication, responsiveness, interaction testing
  - Backend: Request authentication & validation, API responsiveness, security
- Code: Subdivided into two main categories, each with their own subtasks
  - Team finding/collaboration ability
    - Two-way matching matching functionality for existing projects to find members and individuals to find projects or other collaborators
  - Customizable user profile
    - User's capacity
      - Skill set in specific language, environment, artistic form
      - Previous jam contribution showcase
    - User's availability
    - User biography for other relevant information
  - Customizable project profile
    - Provides idea descriptor, genre, environment, language(s), etc
  - Built-in messaging
  - Filterable match preferences

- Project Management Tools
  - Team/project formation through matching with other developers or artists or by direct user invitation
  - Project dashboard
    - Task management
    - File sharing
    - Group messaging
    - Calendar/deadline timer
    - Development Logs
    - Collaborative Whiteboard

## 4.7 - Resources

The resources used for creating the logic and user interface for the team management web app are MongoDB, Express JS, React JS, React Native, Node JS, and the itch.io API. Tools like Insomnia will be used to assist in API testing. Version control will be handled through GitHub. Additionally, ESLint will be used by all team members to enforce the same coding standards for Javascript.

The mobile application will be written in React Native and relies on emulator software such as Android Studio and XCode for efficient development.

## 4.8 - Schedule/Timeline

Semester I:

The time the team is given during the first semester will be designated towards firstly configuring the needed requirements for the development of the web application. With a simple design for functionality the team will begin implementing the design to construct a feasible prototype to demonstrate a working adaptation of the final product. The team strives to complete as many functional components as time permits, with a focus on providing basic functionality to the specified functional requirements in chapter 5 before completing advanced functionalities. As a result, the deliverable for this semester will provide Itch.io users the functionality to register user profiles, form teams of collaborators, and manage projects through a task organizer and other accessibility modules.

Semester II:

The team will continue to develop any missing features from the first round of development from Semester I. Upon final functionality completion, the team will work on developing a strong user interface that is responsive and resizable, along with additional non-functional requirements. The

team expects to have the web application ready for deployment for users to access. In addition, the team collectively understands the benefit involved with the knowledge of modern frameworks. React Native is used to develop mobile applications for both Android and iOS as an alternative to React, which the team has no experience with. A mobile version of the web application that will be available on all devices will be the focus of the rest of the second semester. Giving users mobile accessibility develops further opportunities for participants.

## **4.9 - Delivery**

When the web application is complete to the team's desired standards, it will be deployed online allowing itch.io users to begin collaboration and task management within their game jam projects by accessing the web application or downloading the app to their mobile device. Android and iOS builds of the native application will also be released for internal, invite-only testing due to the complexity of publicly releasing mobile applications.

# **5. Feasibility & Software Requirements Specification**

## **5.1 Introduction**

### **5.1.1 Purpose**

The purpose of this chapter is to present a detailed description of the web app Jamr. It will explain the purpose, features, interfaces, uses, and constraints of the application.

### **5.1.2 Intended Audience and Reading Suggestions**

Game Jam participants who wish to find a team to collaborate with for short term game development, particularly those familiar with the Itch.io platform. This document also helps developers understand the approach and tools required to create the application at a high level.

### **5.1.3 Project Scope**

Jamr is an application that allows users to find and collaborate with a team on a time constrained game development project. Users can create a project for other users to apply to, based on what areas of development are needed. They can then work together through the dashboard of the application to schedule important tasks and share media. Users may also communicate with each other through a built-in messaging system in order to ease collaboration efforts.

## **5.2. Overall Description**

### **5.2.1 Product Perspective**

Jamr is made for individuals interested in gaining experience in the game development industry, regardless of skill level. Direct connection into the Itch.io game jam community helps new users get started in an unfamiliar environment and connect with other driven individuals to lead and learn game development projects.

### **5.2.3 User Classes and Characteristics**

- Novice or proficient game developers looking to gain experience from working with others
- Artists, musicians, and writers that want to test their skills and speed by creating assets to be used in game development

#### **Public User**

Public users of the application include those who wish to use the application to participate in game jam development, manage their projects, or connect with other users in the game development community. Public users is an abstraction for all accounts falling under an associated itch.io account, with additional roles and tags attributed to uniquely identify and verify the roles and actions of each user. For instance, a public user may also be any of the following:

- Team member
- Team leader
- Contributor/Generic Member

A public user is verified into the application once they have completed an OAuth login process through itch.io and completed an account registration setup form. These users will then have the functionality to access their personal homepage, manage invitations, private chat with other users, search for projects, create projects, and personalize their public profile.

#### **Team Member**

Team members are public users who have been accepted into a project via search matching or direct invitation from a team leader. Members will have insight access into the project(s) they have been accepted into, and authorization to perform certain actions that modify the state of a project. These actions include creating and editing task events, uploading resources, utilizing team-wide group chats, and accessing a collaborative white board. Members of a project may also give a 5-star based rating to their teammates to affect their reputation to improve matching results for future projects.

**Team/Project Leader**

The project leader user is a team member with additional privileges who administers the project by overseeing the life of the project from creation to completion. Leaders manage project members and coordination of the project by utilizing the same tools as members along with additional restricted tools to manage team information such as toggling recruitment mode to search for new members, removing current members, and modifying project information and deadlines.

**5.2.4 Operating Environment**

- Web browser
- iOS
- Android

**5.2.5 Design and Implementation Constraints**

Jamr is constructed by the MERN stack, utilizing React as a front-end framework. Functional modules are written as individual components wrapped in their own containers returning dynamic HTML elements that support high levels of dynamic interaction. Because React utilizes JSX, a Java-Script extension to HTML, to render page elements, some constraints for this approach are any framework-dependent disadvantages such as an increased learning curve or specific limitations of the framework.

Additional limitations come from mobile development through React Native. This framework uses native alternatives to JSX/HTML tags and CSS, further increasing the learning curve for the project. Because Jamr is not being publicly released as an official product, the mobile applications (iOS & Android builds) will not be released on the App store and may only be installed directly through invite only access to the latest build package. This limitation will require users to be in direct contact with the developers in order to keep the app updated by reinstalling the application upon each update.

**5.2.6 Assumptions and Dependencies**

Jamr is meant to work alongside game jams, and itch.io is the largest host of these jams. They allow their users to post their finished products on their site for others to play on. Jamr uses their API to allow users to login with their existing itch.io profile, so the app depends on users having a profile with itch.io.



### 5.2.7 User Documentation

A complete user manual will be delivered with the project to help users get started with using the Jamr application on web and mobile.

## 5.3 System Requirements

- Software
  - MongoDB, Express JS, React JS, and Node JS for the full stack framework
  - Axios for sending requests to server API
  - Itch.io API for initializing User model & OAuth Login
  - Amazon S3 Cloud Hosting
  - GitHub for source control
  - Visual Studio Code for writing/running the application
  - Heroku PaaS for deploying the web application
  - Google Play Console for distributing internal Android testing
  - TestFlight for distributing iOS internal testing
  - Socket.io Javascript library for enabling real-time connection and communication between application users
  - React Native framework to develop the mobile application
  - Android Studio's and XCode's device emulator to run development builds of the native application and archive builds for internal testing
- Database
  - MongoDB Atlas for storing and serving application data
  - AWS S3 for file sharing and image hosting

## 5.4 Functional Requirements/System Features

### 5.4.1 List of Use Cases

#### 5.4.1.1 Public User Use Cases - Web Application

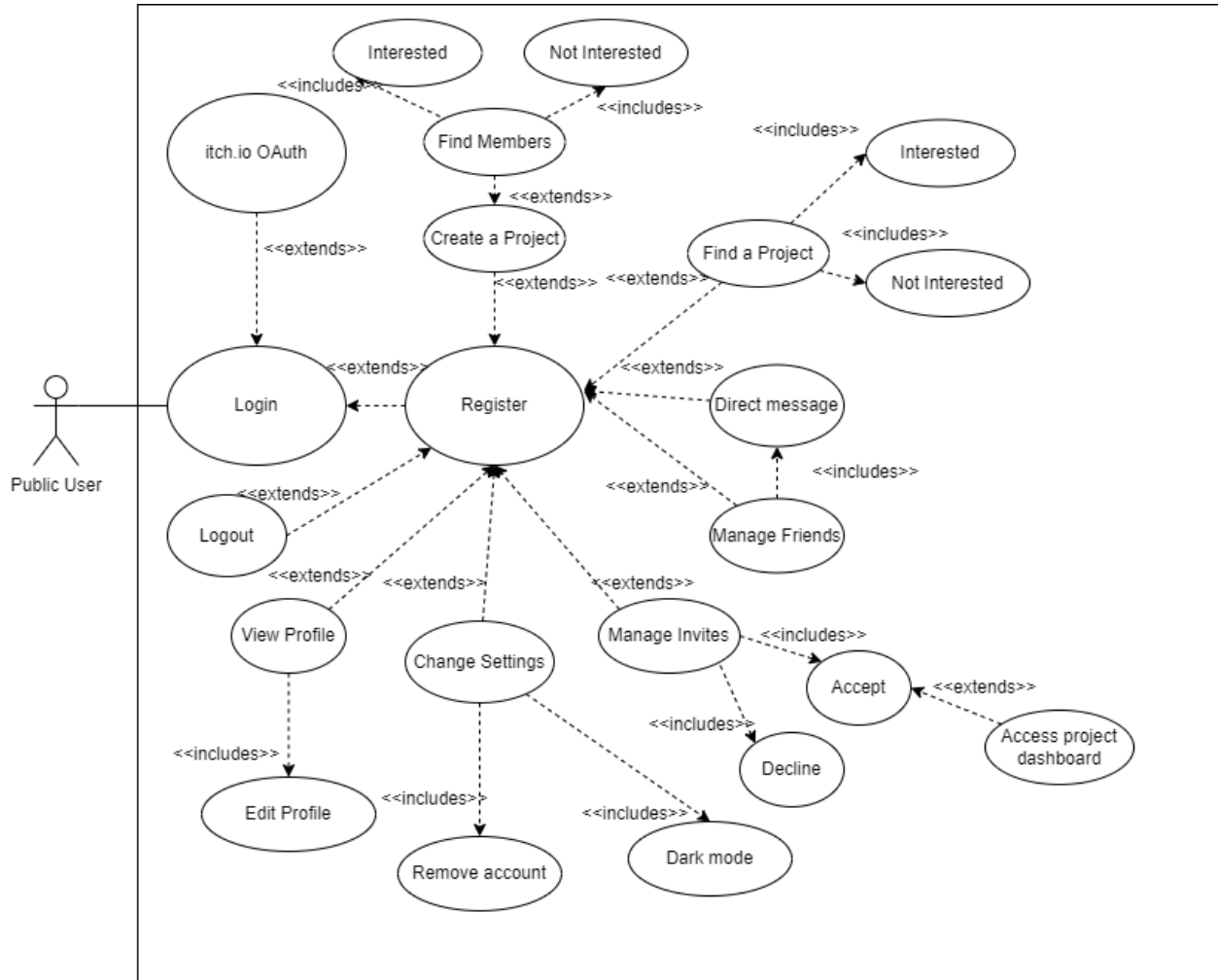


Figure 5.1 - Unregistered User Use Cases for Web Application

**5.4.1.1.1 Login/Register Profile:** Public users will initially only be allowed to register a profile, and then modify that profile before being granted access to any other feature.

**5.4.1.1.2 View Profile:** Users will be able to access the public profile of other Jamr users. From this page, users may view public information about the other user and choose to send them a friend request or remove the user from their friends list if they are already connected.

**5.4.1.1.3 Edit Profile:** Users may edit their avatar, description, and tags. They may also add their itch.io projects as showcases to their profile which will also be displayed in the matching view.

**5.4.1.1.4 Find a Project:** Public users with a registered profile will be able to search for a team and skip or apply to the projects that are presented to them. They are presented with projects that are recruiting members with a matching skillset to guarantee the user will be capable of contributing.

**5.4.1.1.5 Manage Invites:** Users may view a list of their current project invites and open a popup modal to read all information about the project and choose to accept or decline the invitation.

**5.4.1.1.6 Create a Project:** Public users with a profile can create new projects that will be presented to other users that are trying to find a project. They will decide what game jam the project is a part of and write a description about what is needed and what they want to get done. The project creator will assign “looking for” tags which help display the project only to candidates whose skills match those needed to complete the project.

**5.4.1.1.7 Direct Messaging:** All public users can add other users ID’s and begin a private messaging session. Users may only message their friends.

**5.4.1.1.8 Manage Friends:** Public users may search for other users and send them a friend request. They may also manage existing requests and choose to accept or deny any incoming friend requests. Accepting a friend request creates a mutual contact that enables direct messaging capability.

**5.4.1.1.9 Settings:** Users can delete their profile and edit application preferences such as light and dark mode for accessibility.

**5.4.1.1.10 Logout:** Users may log out of the application, clearing any active sessions.

#### 5.4.1.2 Team Member Use Cases - Web Application



Figure 5.2 - Team Member's Dashboard Use Cases for Web Application

**5.4.1.2.1 To Do List:** Each team member can create a new list item. If they created the item, lead the project, or are assigned to a task, they may remove, edit, add priority, or reassign the item. Members may also mark pending tasks as completed.

**5.4.1.2.2 Share Files:** Each team member will be able to upload files and download the files that others have uploaded. If the user uploaded the file or is a project leader, they may also delete the file.

**5.4.1.2.3 Development log:** Every time a member adds something to the project, or if a task is completed, a development log will be made showing details of the development log.

**5.4.1.2.4 Team chat:** Each team member will be able to contribute to their teams chat, where messages are delivered to every member in real-time.

**5.4.1.2.5 Calendar:** Every team member can add, edit, or remove calendar events. The project leader may edit/remove events as well as the event creator.

**5.4.1.2.6 Whiteboard:** Every member has access to a live collaborative whiteboard component where they are able to draw and erase lines on the canvas simultaneously. Upon every marker stroke, the dashboard thumbnail will update to reflect changes. Users may also clear the canvas.

**5.4.1.2.7 Member Settings:** Team members may give a rating to other project teammates, including the project leader. Ratings are based on a 5-star scale and immediately impact the receiving user's overall reputation within Jamr. Users may only give one rating per member, but can edit the score they have assigned. Team members have the ability to leave their project.

### 5.4.1.3 Team/Project Leader Use Cases - Web Application

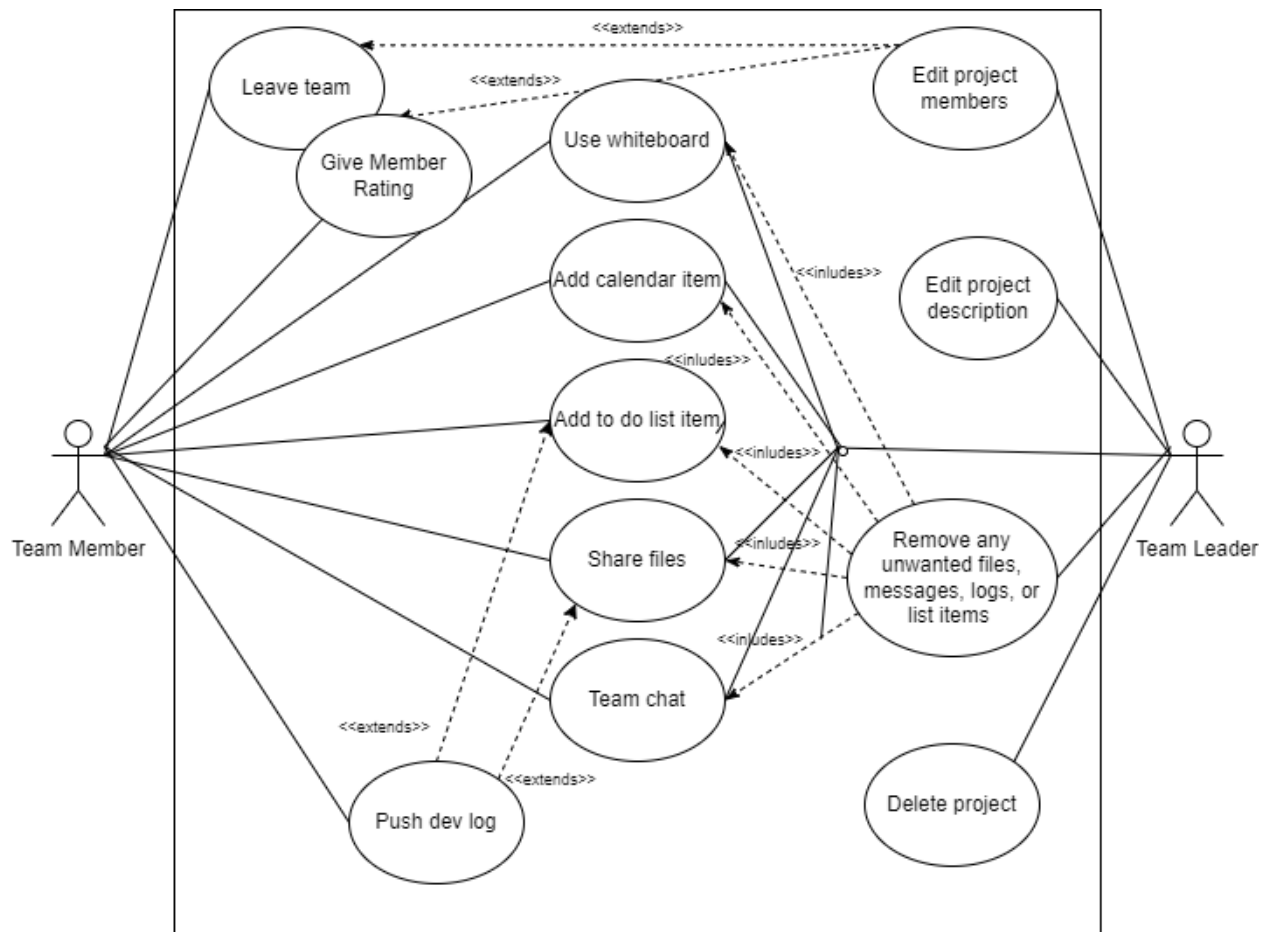


Figure 5.3 - Team Member and Leader Interaction Use Cases for Web Application

**5.4.1.3.1 Manage Project:** The project leader can edit the project description, deadline, and title. Leaders may also delete the project entirely. Users who access the manage project setting will be given the option to leave the project.

**5.4.1.3.2 Manage team members:** The project leader can remove unwanted members from the team and invite/accept new members to the team.

**5.4.1.3.3 Rate Team Members:** Leaders may also give ratings to the members of their project.

**5.4.1.3.4 Find Members:** Project leaders may use the matching functionality to find potential members for their projects by marking interested or uninterested.

#### 5.4.1.4 Public User Use Cases - Mobile Application

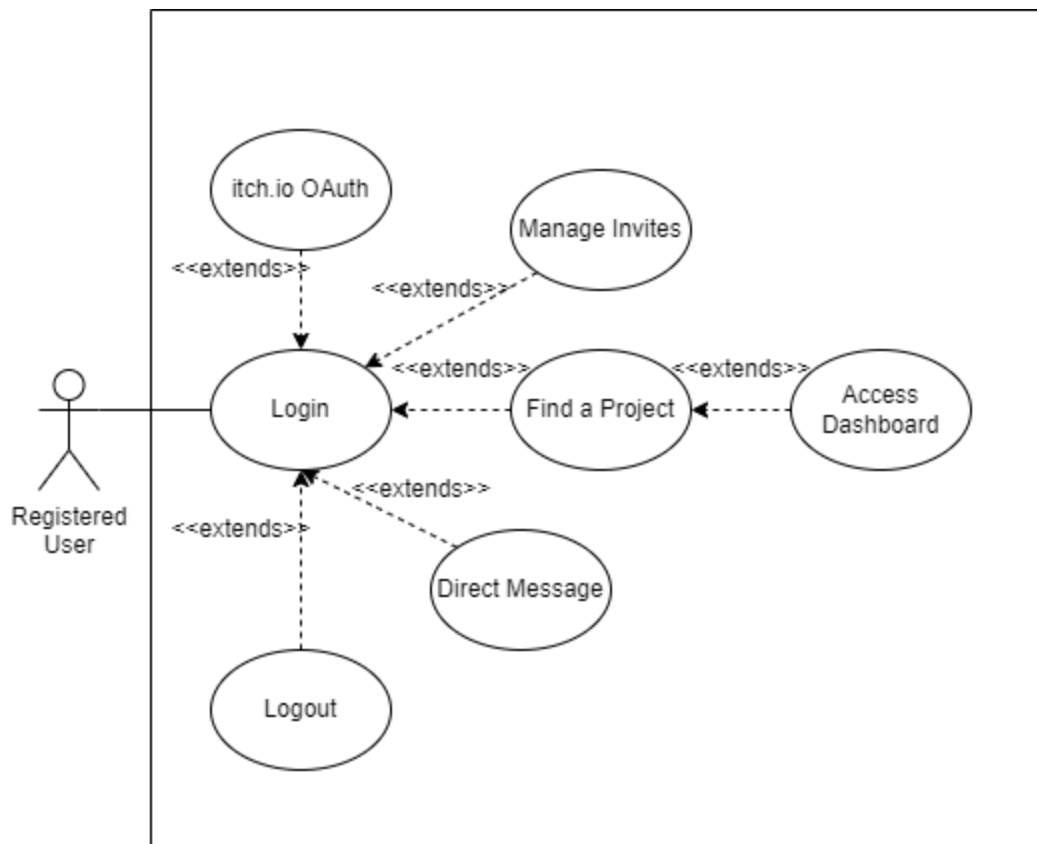


Figure 5.4 - Registered Member Use Cases for Mobile Application

**5.4.1.4.1 Login:** Public users can create an account on Jamr using their existing Itch.io account through the use of Itch.io's OAuth system. They will generate a secure web token to identify their session and keep them logged in.

**5.4.1.4.2 Logout:** Public users can log out of their account on Jamr, wiping their session tokens and preventing another person from logging into their account without authenticating on Itch.io.

**5.4.1.4.3 Find a Project:** Public users can view available active projects and choose whether they are interested, sending a request to the project leader, or uninterested.

**5.4.1.4.4 Manage Invites:** Public users can access a list of project invites and choose to accept/reject them.

**5.4.1.4.5 Direct Messaging:** All public mobile users can access their direct messages. They can send and receive messages to any existing connections.

#### 5.4.1.5 Team Member/Leader Use Cases - Mobile Application

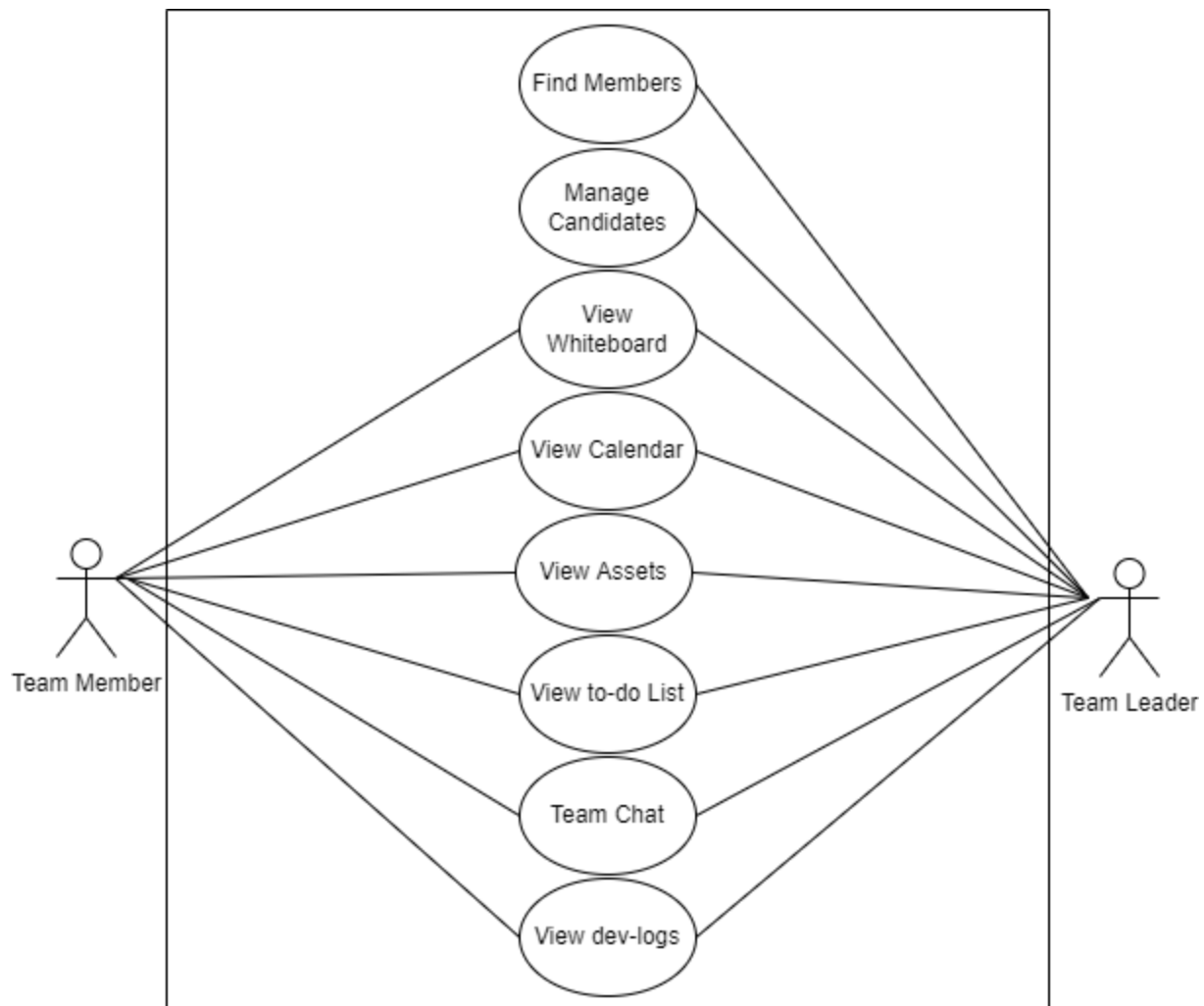


Figure 5.5 - Team Member and Leader Use Cases for Mobile Application

**5.4.1.5.1 Team Dashboard:** Team members and leaders are able to view their project dashboard to keep up with current progress and make sure they are staying on track.

**5.4.1.5.2 Team chat:** Team members and leaders on the mobile application are able to participate in their team's group chat. Messages will be delivered in real time to keep all of the team on the same page.

**5.4.1.5.3 Manage Candidates:** The team leader is able to manage the candidates list for their project. They can view information about each candidate, and choose to send them an invite or decline their candidate status.



**5.4.1.5.4 Find Members** They may also invite users to the project from the “Find Members” button on the homepage.

## 5.5 External Interface Requirements

### 5.5.1 User Interfaces

The application should be easy to navigate and components should be kept minimal for efficiency. A constantly accessible navigation bar ensures that the user is never stuck or lost in different places of the application. A modular layout of boxed-in and separated components keeps the user tuned into the individual task that they need to work on, rather than having all of the modules and elements flowing into one another. Additional design considerations are mentioned in section 5.5.3.

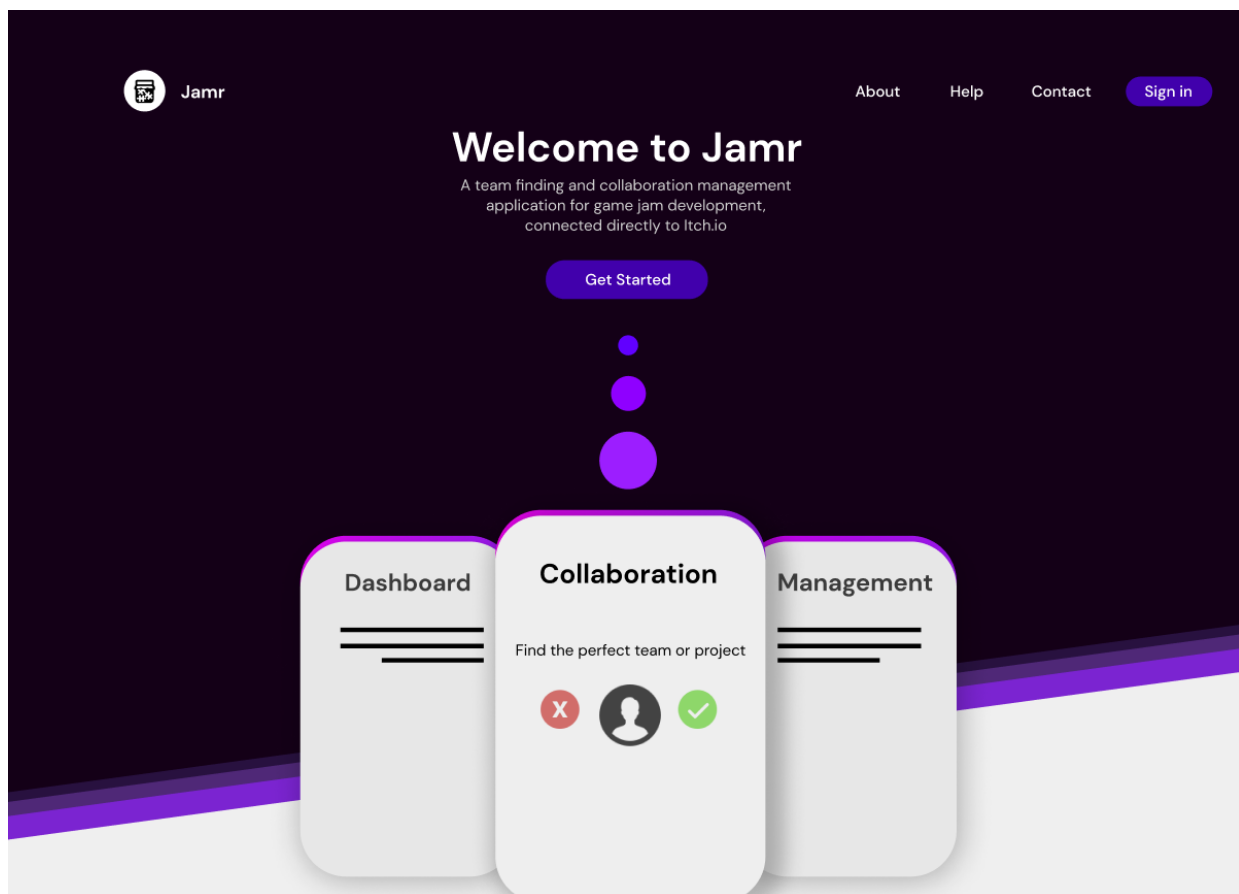


Figure 5.6 - Landing Page UI Concept

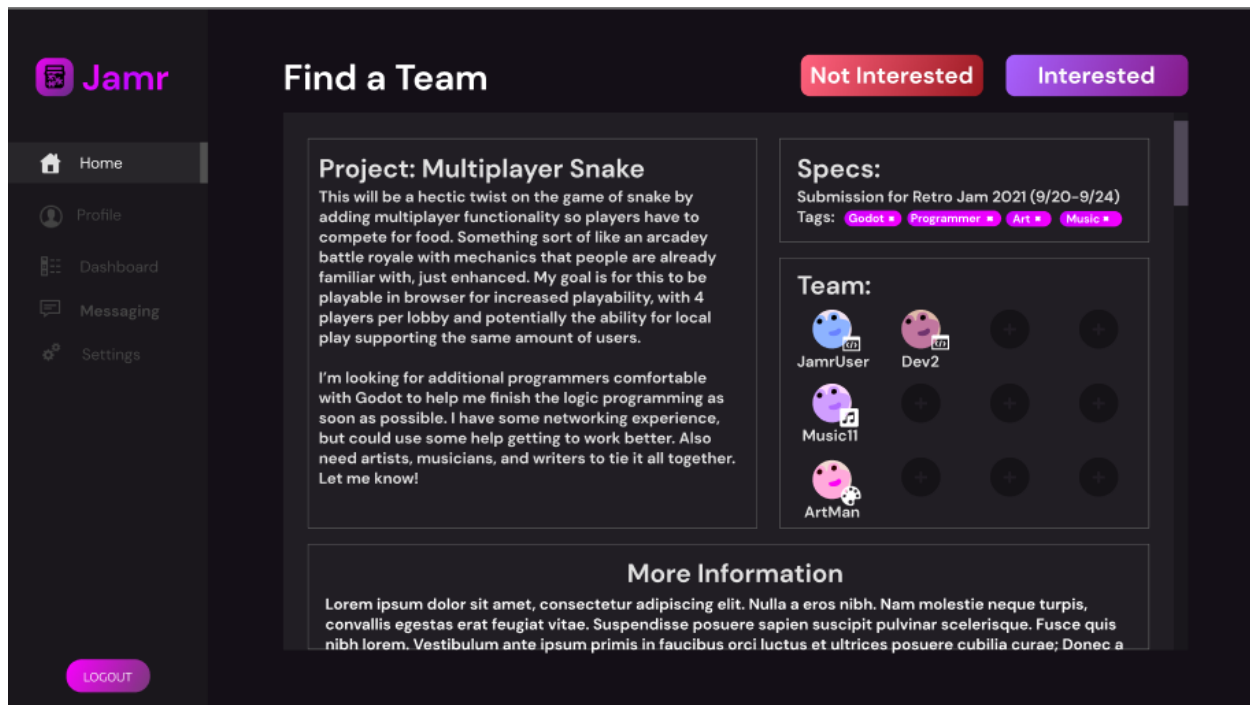


Figure 5.7 - Matching Page &amp; Navigation Bar UI Concept

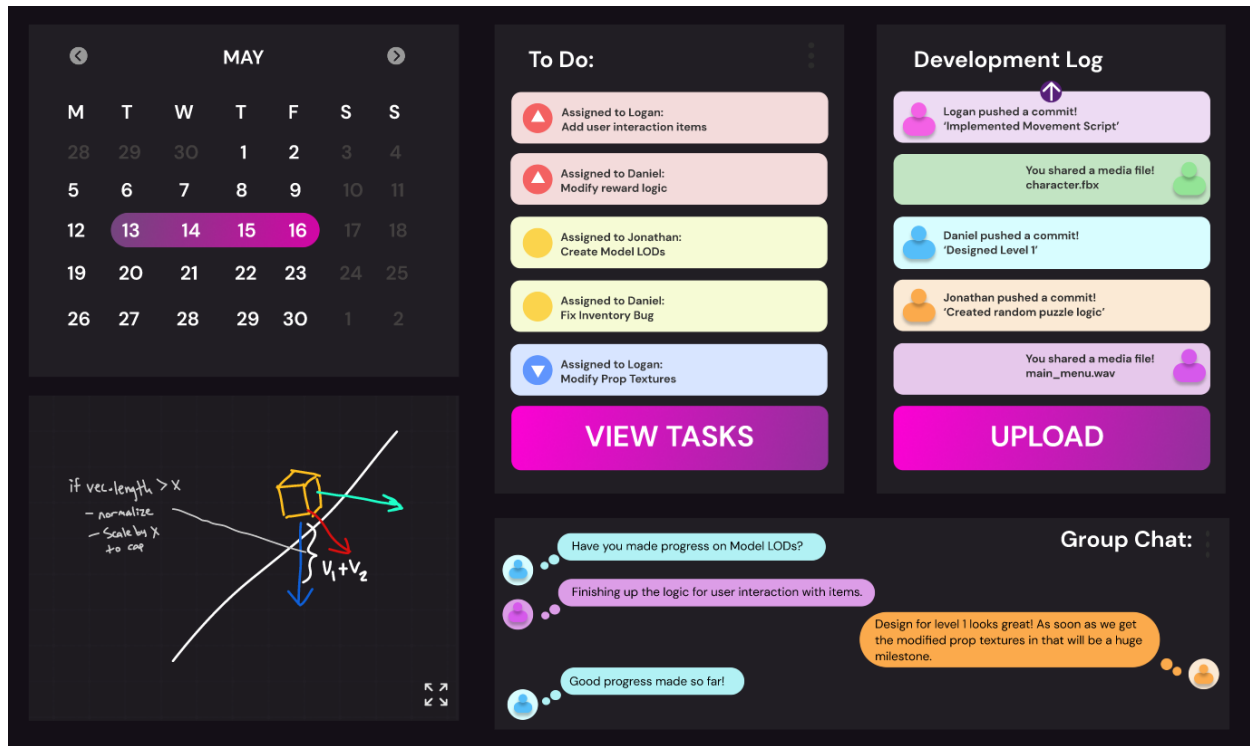


Figure 5.8 - Project Dashboard Page UI Concept

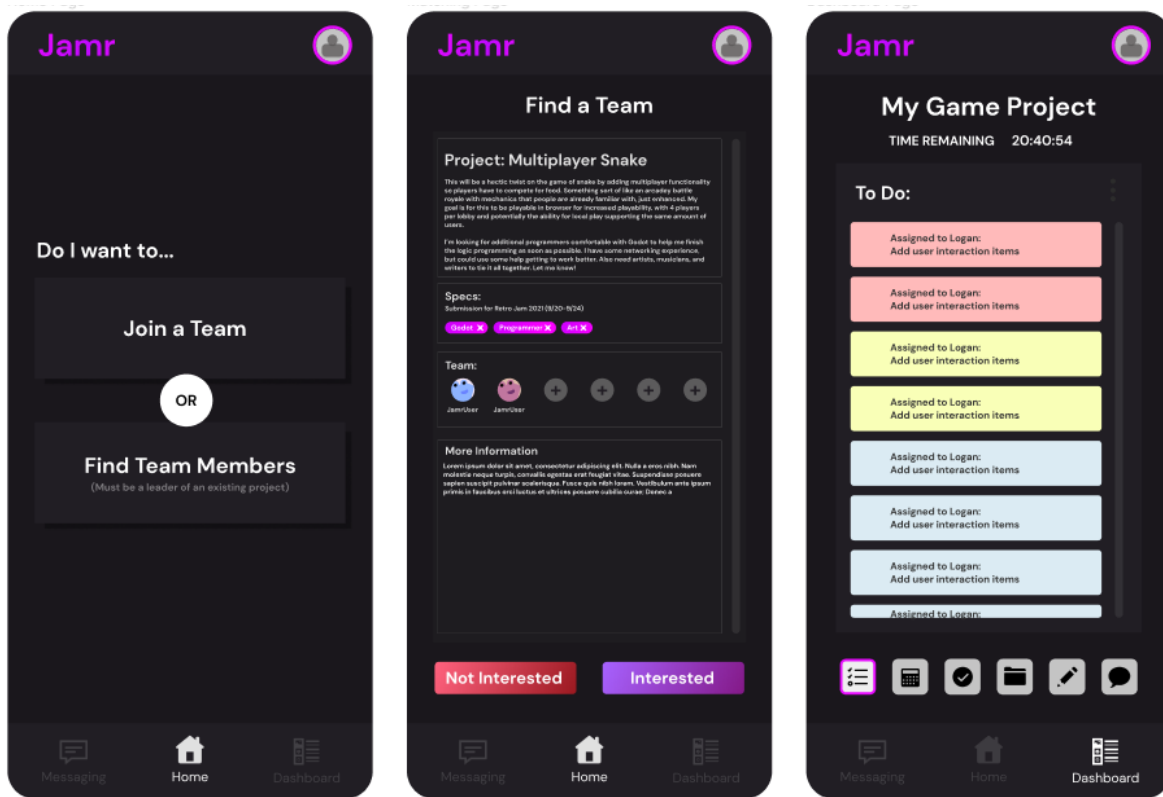


Figure 5.9 - Mobile Format Concept of Home, Matching, and Dashboard Pages

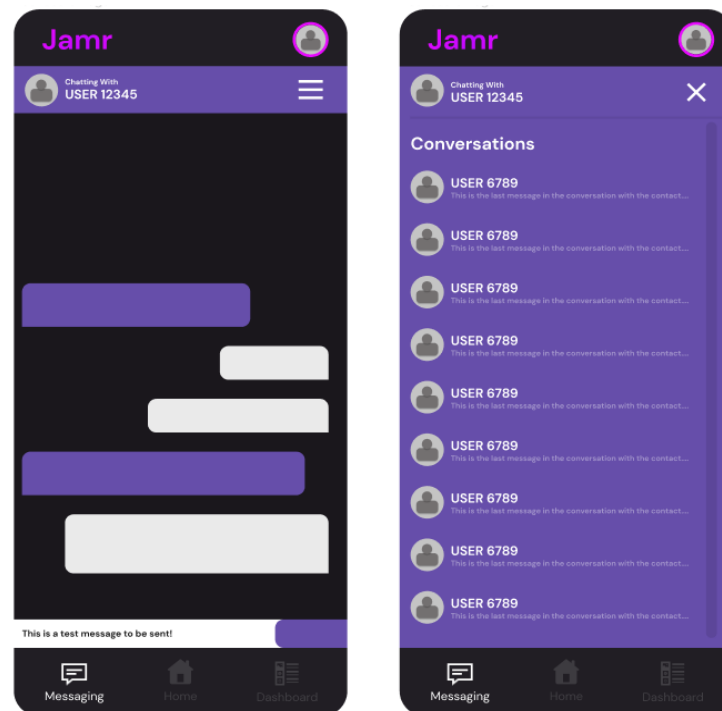


Figure 5.10 - Mobile Format Concept of Messaging Page

## **5.6 Other Non-Functional Requirements**

### **5.6.1 Performance Requirements**

The application should handle server-side events within reasonable time to ensure efficiency and promote productivity. Updates to views via button clicks or other interaction types should be near-instant for responsiveness. Depending on the task, successful CRUD (Create, Read, Update, Delete) operations should complete and re-render within seconds. Components that update remotely through sockets are expected to render changes quickly with a reasonable delay no greater than 15-30 seconds.

### **5.6.2 Usability**

The goal for the user interface design is to provide something simple and clean. The UI is kept completely uncluttered in order to help the productivity of our users. Most of the elements are a muted color scheme with minimal accents, so that the user can stay focused on what needs to be done. Vibrant colors that are not a part of the default color scheme will be used for the productivity elements, such as the to-do list, calendar, development logs, and asset management, so that the team members attention is focused on these elements most of the time.

### **5.6.3 Security Requirements**

User and project data is stored and served by MongoDB Atlas, a secure cloud based hosting solution provided directly from MongoDB Inc. Jamr utilizes the preconfigured security standards from Atlas to ensure data is properly handled and that sensitive access information such as connection strings remain confidential. Utilization of Amazon's S3 cloud hosting requires secure handling of access permissions, which is achieved through using a designated frontend and backend access account to perform specific actions. Activities that grant permission, such as uploading a file, are signed by AWS with specific upload parameters and are limited to a 60-second expiration endpoint to minimize security risks.

The application will also utilize modern and secure session and authentication handling by using JWTs (JSON Web Tokens) assigned to each user account to verify all API requests through a middleware verification function.

Socket functions for real-time updates and communication between users are limited in application and utilize the same verification approach to maintain security standards. All connections to the Socket server are configured with additional authorization credentials so that the connecting user's identity may be confirmed before establishing a connection and their identity can be used to limit the actions they may perform. The socket server extensively manages communication signals through the use of Rooms, a channel structure provided by Socket.io. When a user connects to the real-time communication server, the application separates

and subscribes them to a channel that only contains the users who are permitted to communicate with each other. See additional information and connection activity flows in the System Communication Section 6.5.1.3.

#### **5.6.4 Software Quality Attributes**

The web application is designed responsively so that content is easily accessible from any browser or device. Functional modules may appear differently on screens with limited resolutions, but all functionality will be available.

Accessibility is also taken into consideration and supported via dark and light mode themes selectable by the user.

The mobile application will closely mimic the overall design and feel of the web application. Additional buttons or navigation elements may be provided for assistance. The mobile app may also utilize additional native capabilities such as gesture controls for an increased user experience. In cases where multiple elements are present on web (such as modal popups for components like Manage Invites), the mobile application will instead split these elements into separate screens and use a nested navigator to better display the content.

## **6. System Design**

### **6.1 Introduction**

The purpose of the system design chapter is to provide more insight into the overall design choices of the application. This section includes specific implementation approaches along with data/model descriptions and diagrams to explain the relationships between them. Demonstrating how users interact with the system and the actions the server takes to reflect these changes serves to help developers understand the project from an implementation perspective.

### **6.2 Design Overview**

#### **6.2.1 Description of the Problem**

Currently game jams are posted and people interested in working together have to scroll through hundreds of pages on a forum in order to find the right teammates that they feel can accomplish what needs to be done for the final product of the game. These game jams are continuous with sometimes not a lot of time to finish the desired product. With such little time it we want to minimize that time by giving participants a way to collaborate quicker and more efficiently. Programmers, artists, sound designers, etc. spend so much time scrolling through forums to find

the team that works for them. This project intends to make finding and managing a team simple and effective.

### **6.2.2 Technologies Used**

The application will be developed within two separate repositories managed by GitHub for version control using ESLint to enforce modern JavaScript writing standards. This keeps code between components written by each team member consistent, easing the cost of maintenance and testing. The MERN stack (MongoDB, Express, React/React Native, NodeJS) will be used to store and serve user data with the assistance of itch.io's API, and allow users to connect with each other through the web and mobile applications. Additional libraries such as Axios are essential to the successful development of the project. User content will be uploaded and hosted using Amazon's S3 (Simple Storage Service) so that they can customize their profiles and share assets with their development team. Utilizing Socket.IO, real-time communication is enabled and supports chat and live collaboration modules between users. The web application is to be deployed online using Heroku.

## **6.3 Objects**

### **6.3.1 Object List and Descriptions**

#### **6.3.1.1 User**

User objects are the primary object of the application. They are responsible for all interactions within the application, including creation of new projects and member collaboration and communication. Users are composed of public and private attributes that support personalization and support for enhanced filtering for finding suitable projects, along with attributes to handle sessions and request authentication.

#### **6.3.1.2 Project**

Project objects are responsible for holding the individual attributes of projects created such as the project's details, time left for the project and the members contributing to the project. Projects are the parent object for which tasks, development logs, files, and events, and whiteboard are compositions of.

#### **6.3.1.3 Task**

The task object is responsible for showing users in a group a list of tasks with different priorities to ensure an efficient flow of productivity. Team leaders will be given the ability to make changes to this list and assign tasks to other users.

#### **6.3.1.4 Development Log**

The development log object is responsible for giving users the ability to keep track of the progress made during the development of the project. It will allow users to track what milestones have been met by the team.

#### **6.3.1.5 File**

The file object is responsible for the sharing of files between users. Users can either upload a file to the project or download a file submitted by another user.

#### **6.3.1.6 Event**

The event object is a calendar event creatable and editable by all team members within a project. These objects populate the calendar on the project dashboard.

#### **6.3.1.7 Message**

The message object will primarily be used for the communication between users. Users will be able to send private messages to each other or group wide messages to all members of the group. Both message types are represented by this object, with different message type and recipient configurations to determine which component will be using them.

#### **6.3.1.8 Whiteboard**

The whiteboard object is created upon the start of a new project. It stores only a key to the project it is a composition of, a thumbnail image of the updated whiteboard state, and a list of line objects drawn on the board.

### 6.3.2 Object Relations

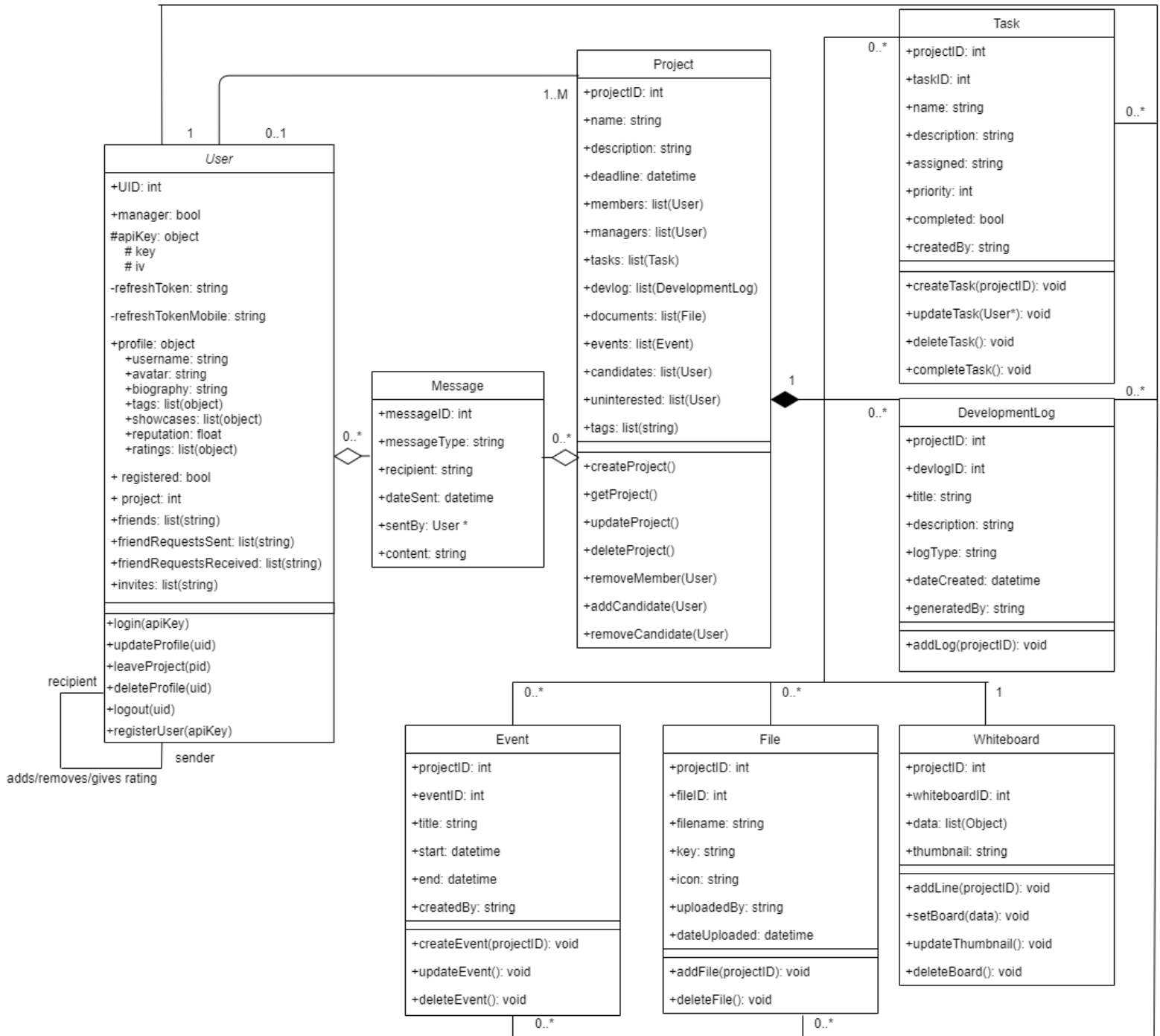


Figure 6.1 - Object Relations Diagram



## 6.4 System Architecture

### 6.4.1 System Design

#### 6.4.1.1 Implementation Approach: The Model-View-Controller Pattern

The application and its functional models are dependent upon the MVC architectural pattern for rapid prototype and development. Each interface and use case is supported by a corresponding React component broken down into a supporting model, a simplified user-interface, and a controller allowing dynamic user interaction abstracted to execute backend instructions which then modify and update the models. In combination with the MERN stack used to develop the application and the advantages it brings, this modularity greatly decreases implementation efforts by encouraging code reusability.

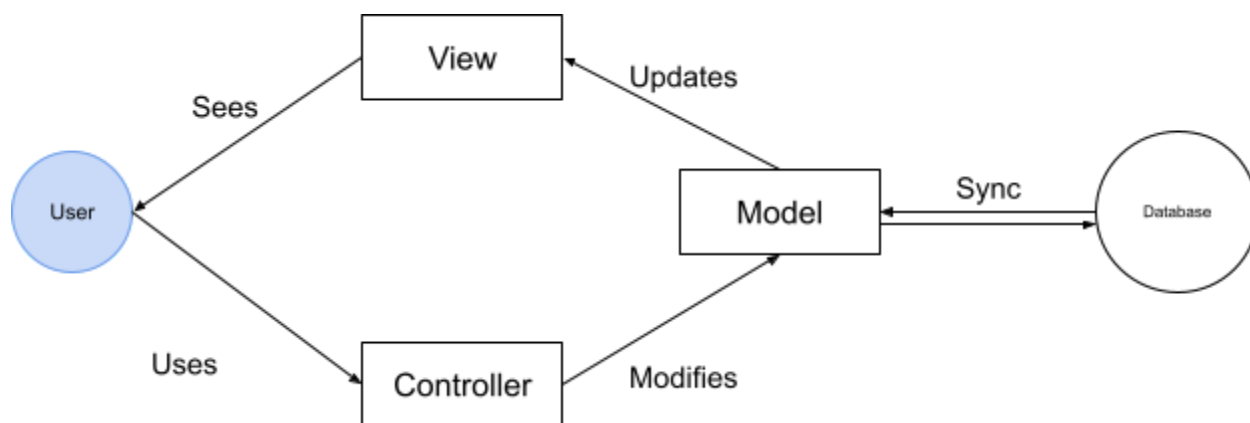


Figure 6.2 - Execution flow for each individual component of the application

#### 6.4.1.2 Middleware Authentication and Authorization

Middleware functions existing between the backend API controller and the server are used to verify all actions and requests of a user for authorization and security. Upon registration and subsequent logins, a user is generated a secure access and refresh JSON Web Token. Web and Mobile tokens are created and stored separately to avoid permission issues between devices. For a web application user, the tokens are stored in the browser's local storage. Tokens for mobile users are stored within the local application data.

When users make requests to visit application pages or use functional modules, the server first verifies the validity of these tokens. The access tokens confirm the identity and role of the user to permit or deny requests and handle login sessions so that users do not need to log in to the website upon every visit. To reflect modern security standards, the access tokens are short lived (currently 10 minutes) and need to be regenerated using a valid refresh token once expired. The control flow for validating and assigning security tokens is as follows:

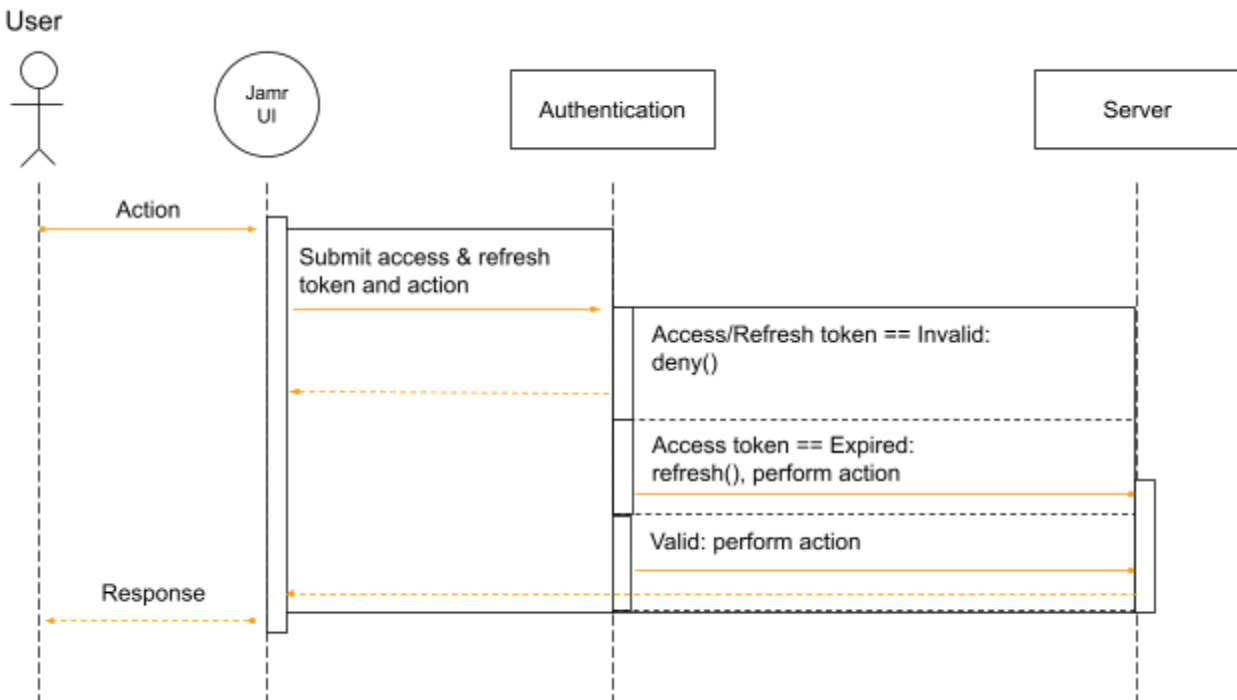


Figure 6.3 - Initial Token Design Sequence Diagram

#### 6.4.1.3 System Communication

Certain pages in Jamr support real-time communication between users. A Socket.IO server is used to support this capability and is initialized when the main Jamr application starts. Users may connect or disconnect from the socket server at any time following an interaction sequence that verifies their identity and places them into broadcast rooms specific to the purpose of their connection. The following diagram from Socket.IO generalizes the concept:

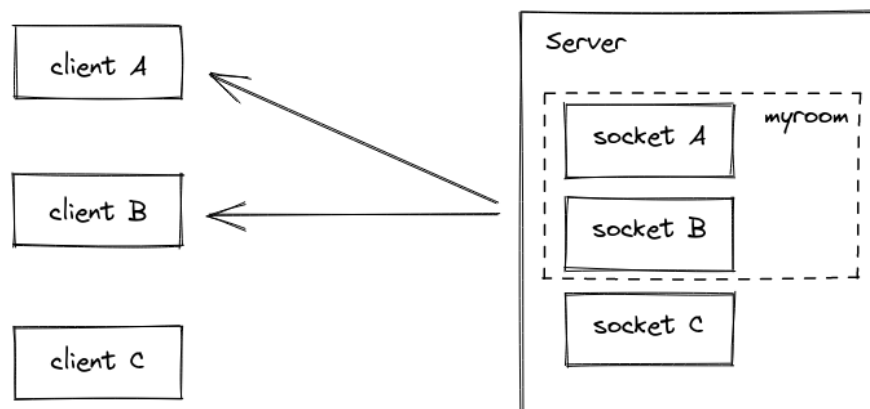


Figure 6.4 - Socket Room Example

<https://socket.io/images/rooms.png> - from <https://socket.io/docs/v3/rooms/>

Jamr uses this concept to group users into appropriate rooms upon connection to the socket server. When visiting the Project Dashboard page, members automatically send a connection handshake configured with their Project ID and are placed in a corresponding room for that ID. As a result, the user may only listen and broadcast changes to members of the same project.

Private Messaging is designed using the same idea, but with one difference. Instead of grouping two users into a private room, the application places chatters into their own room. When a message is sent, the socket server emits the message only to the recipient's room. This design allows real-time communication without the need for complex room management.

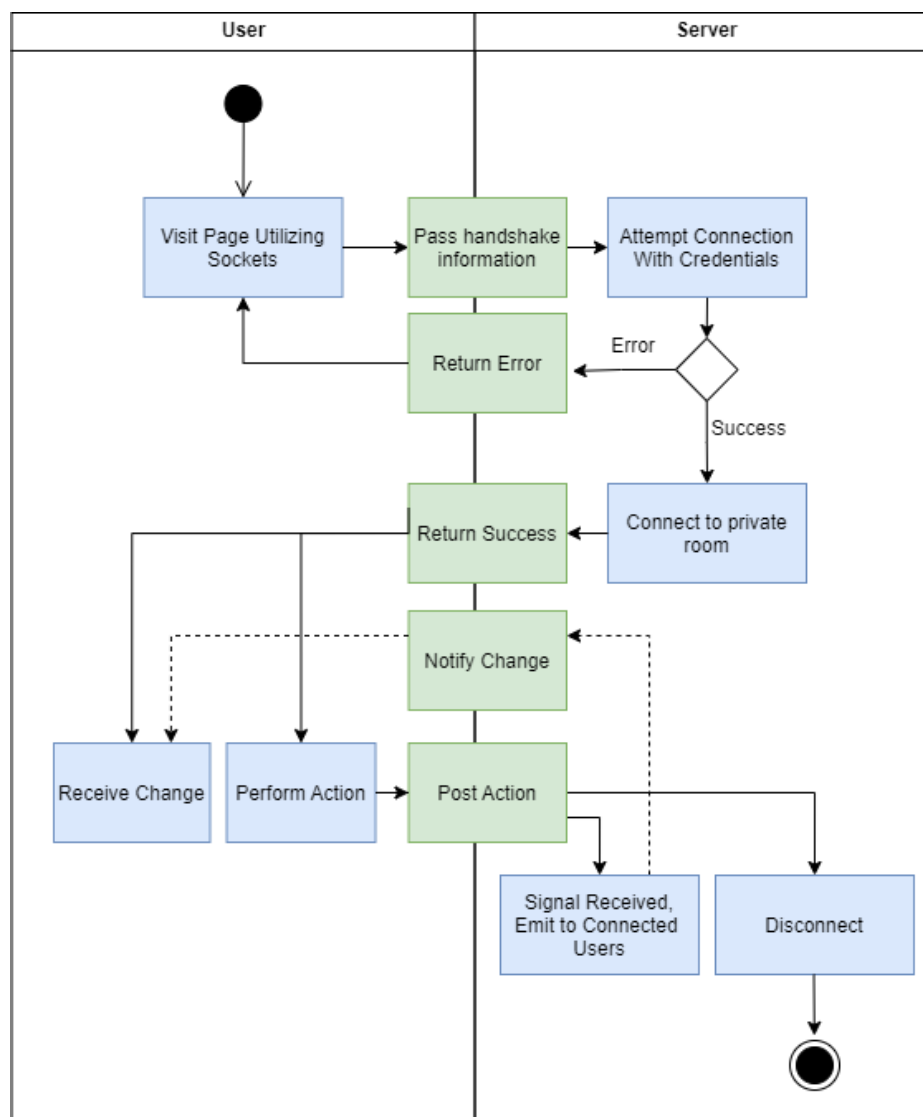


Figure 6.5 General Socket Connection Flow

Additionally, socket signals are emitted only after successful API calls. This means that a user may only communicate to other users after performing a validated action, increasing the level of security and accuracy of the messages.

#### **6.4.1.4 Functional Component Design**

Components describe the individual functional modules built under the MVC pattern with the object and relation definitions above in mind. These components act as building blocks for the application pages and provide the functionality for the use cases defined in section 6.3. Below are descriptions of the components vital to the functionality of the application and the objects, methods, and relationships they use to perform tasks within the application.

### **6.4.2 Components/Subsystems Design**

#### **6.4.2.1 Web Application User Login**

<b>Description</b>	The login component is responsible for redirecting users to an itch.io OAuth application sign in and capturing the generated permission token upon return to the Jamr application. Web tokens are also created during this process and are automatically saved to the user's browser upon successful login.
<b>Objects Used</b>	User
<b>Use Cases Supported</b>	Login, Register Profile

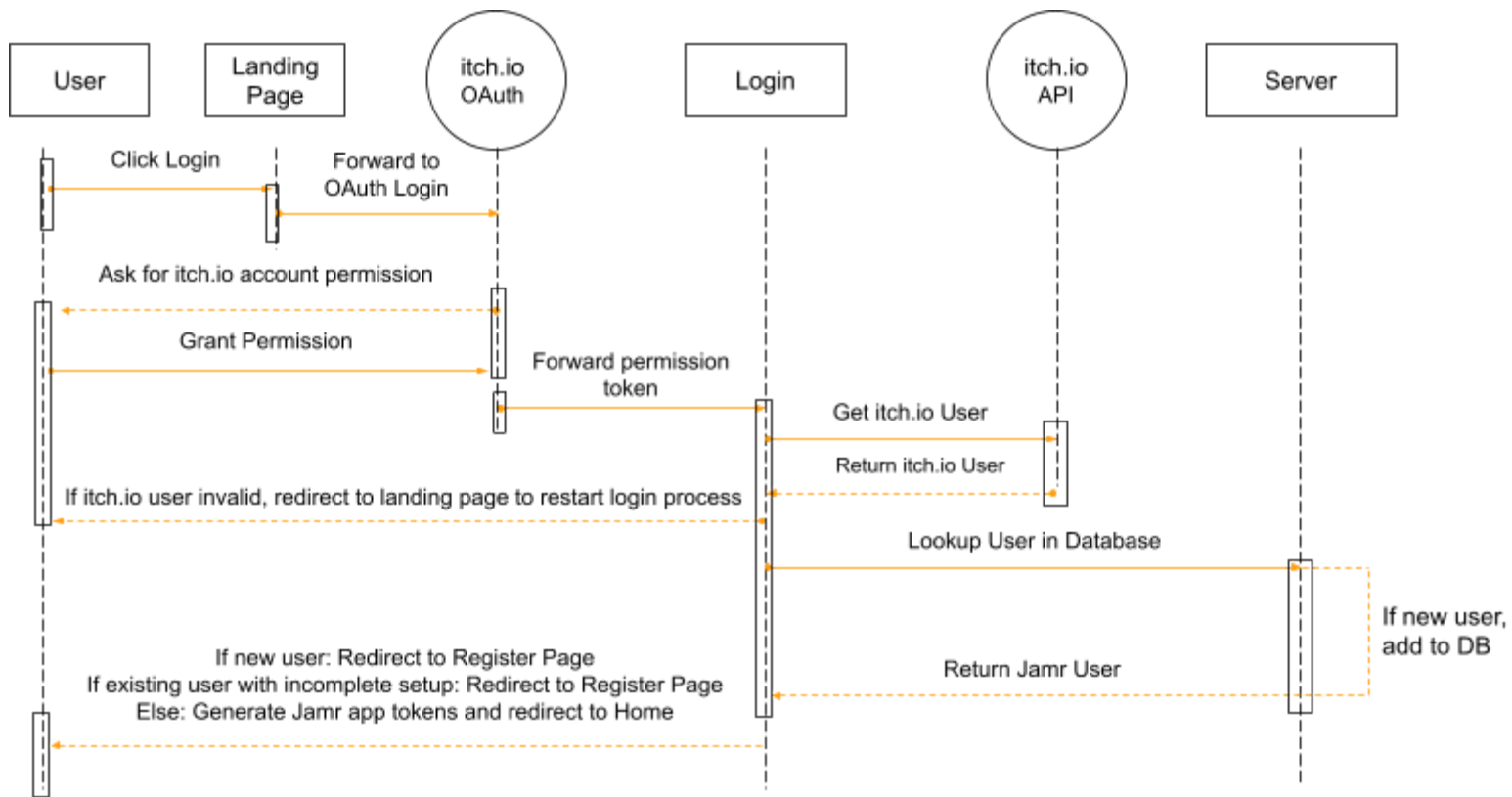


Figure 6.6 - Initial Web Login Sequence Diagram

### 6.4.2.2 Mobile User Login

<b>Description</b>	The mobile login component confirms that the user signing in has previously registered a Jamr account. If the user has previously registered, the application generates new mobile authentication tokens and automatically saves them to the user's device before redirecting to the homepage. If the user is not already registered, an information page is displayed instead.
<b>Objects Used</b>	User
<b>Use Cases Supported</b>	Login

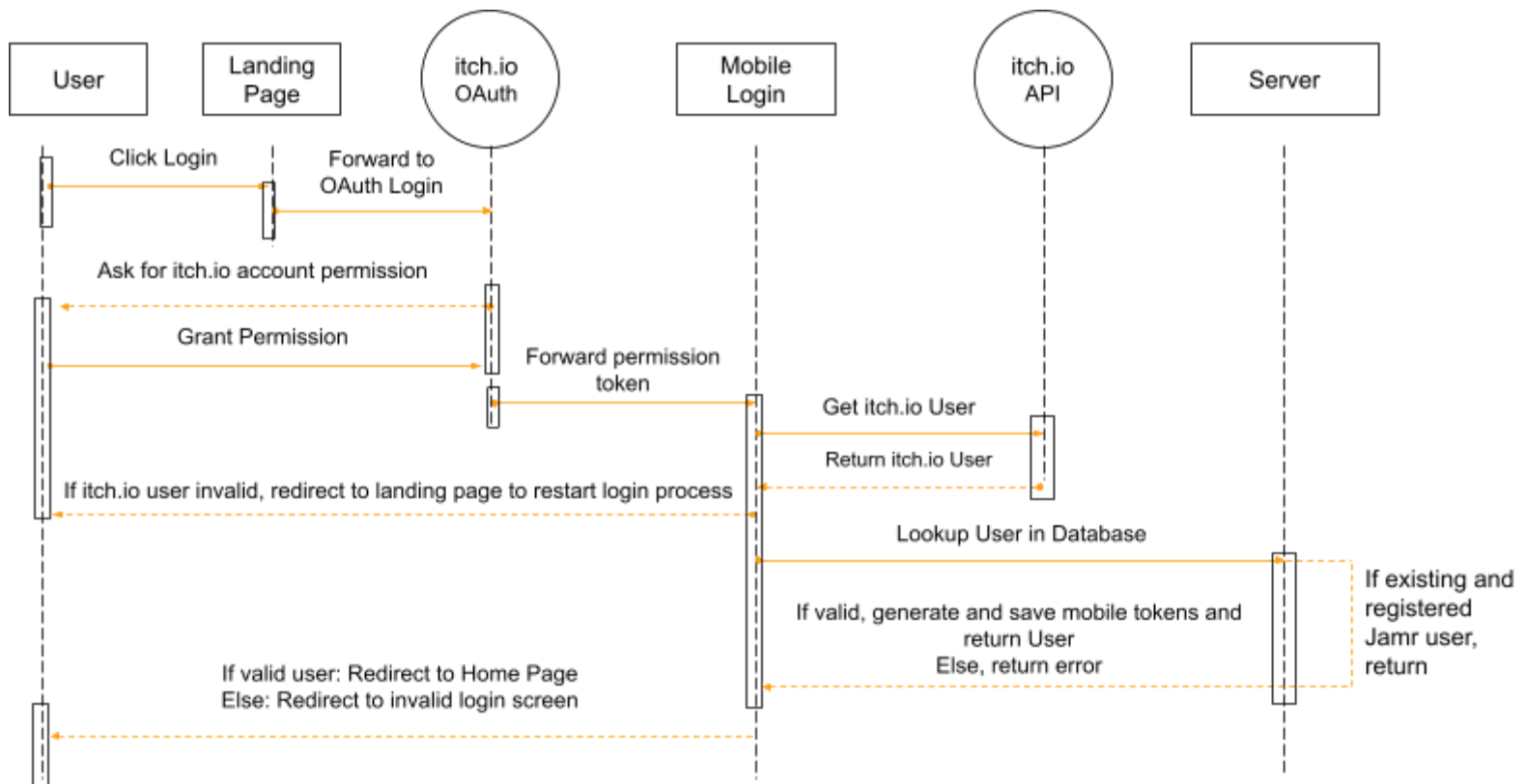


Figure 6.7 - Initial Mobile Login Sequence Diagram

### 6.4.2.3 User Registration

<b>Component</b>	Registration
<b>Description</b>	The registration component enforces the requirement that new users must provide some information about their profile (such as skills and tags needed for database filtering) before they can access the application functionalities.
<b>Objects Used</b>	User
<b>Use Cases Supported</b>	Register/Modify Profile
<b>Mobile Implementation</b>	Component is only available on Web Application

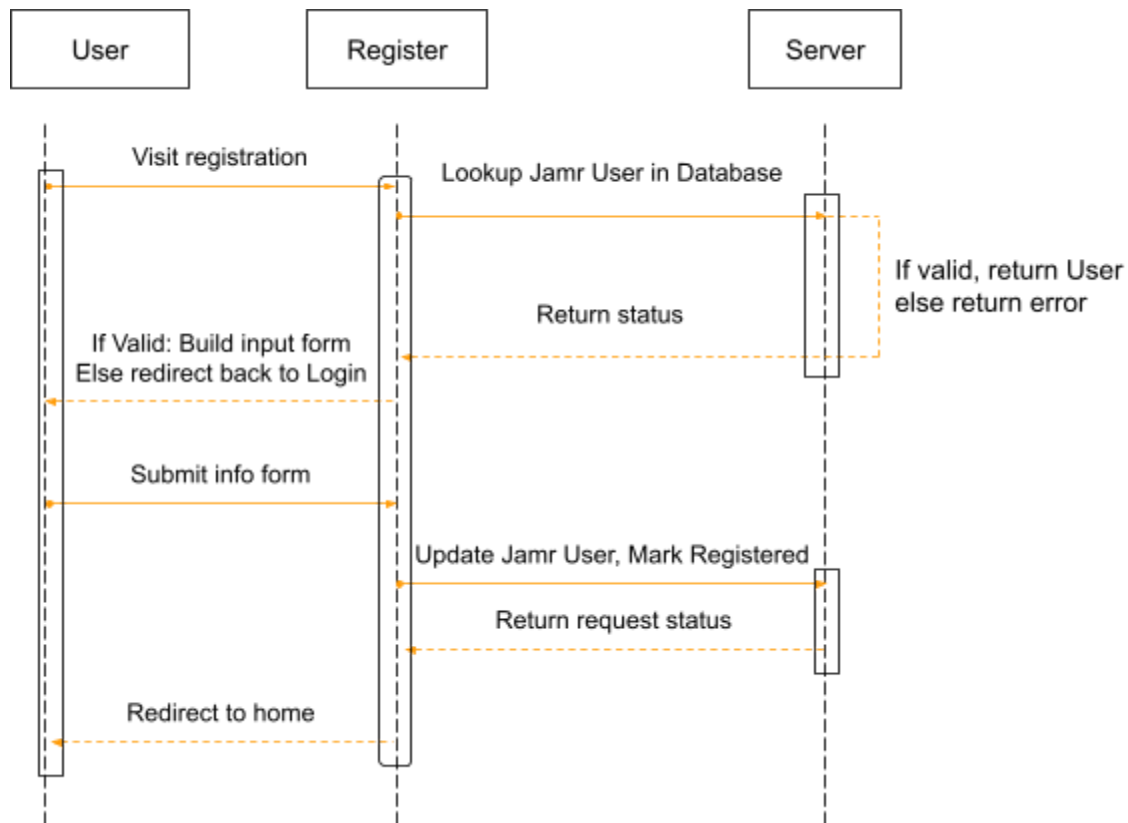


Figure 6.8 - User Registration Sequence Diagram

#### 6.4.2.4 Home Page

<b>Component</b>	Home
<b>Description</b>	The home page controls the pages that users are allowed to visit depending on their project status and allows them to view their project invites. The invite manager component is detailed in section 6.5.2.7.
<b>Objects Used</b>	User, Project
<b>Use Cases Supported</b>	Find a Project, Find Members
<b>Mobile Implementation</b>	“Create Project” component not available in native application

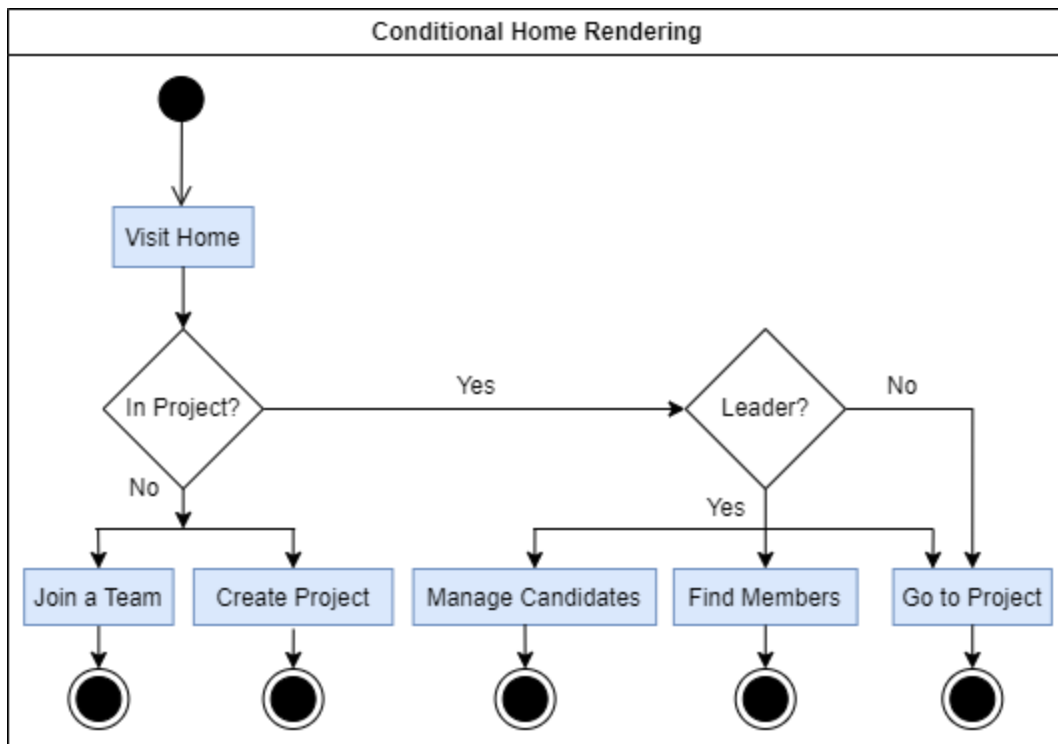


Figure 6.9 - User Registration Sequence Diagram



#### 6.4.2.5 Create Project

<b>Component</b>	CreateProject
<b>Description</b>	The CreateProject component gives users the ability to create a project and provide details such as a title, description and the deadline of the project.
<b>Objects Used</b>	User/Project
<b>Use Cases Supported</b>	Create new Project
<b>Mobile Notes</b>	Component only available on Web Application

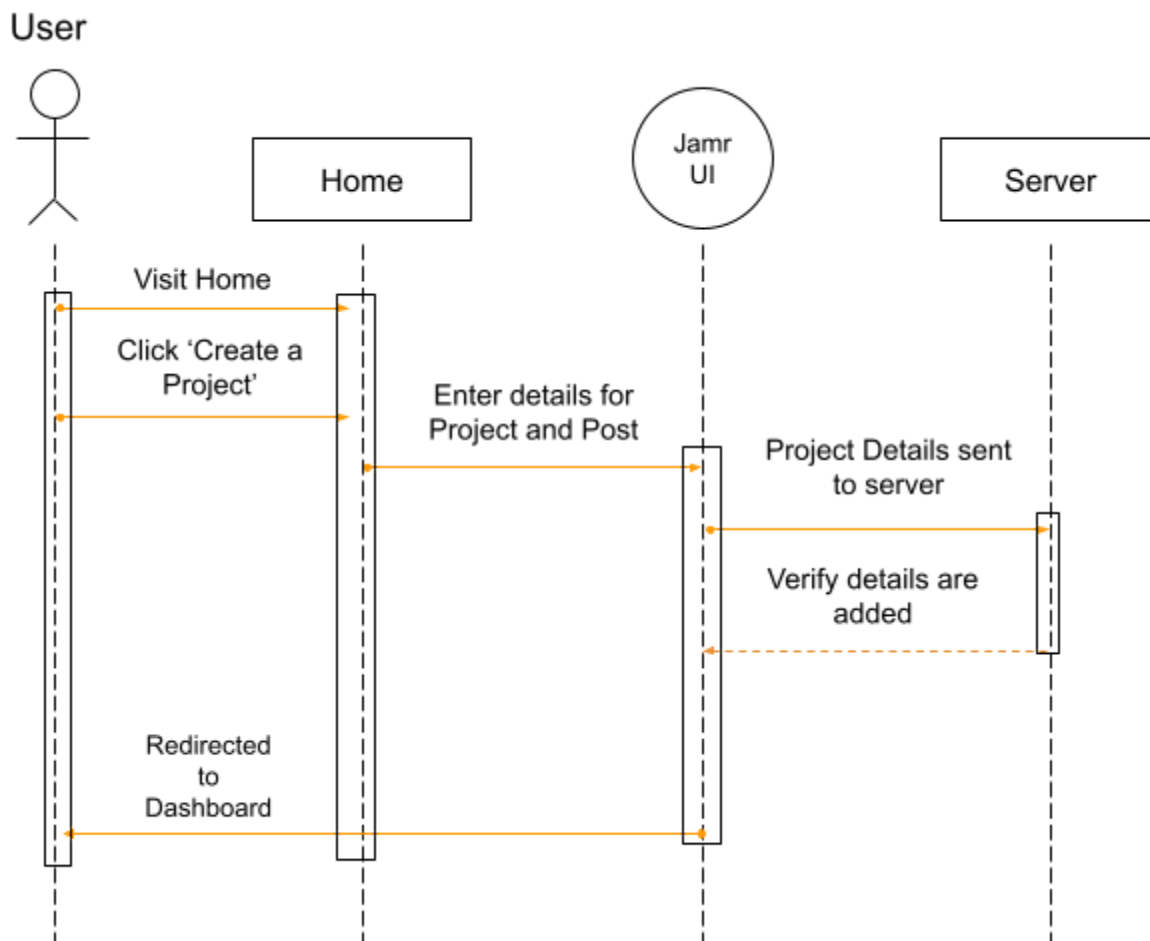


Figure 6.10 - Initial Create Project Sequence Diagram

#### 6.4.2.6 Matching

<b>Component</b>	Matching
<b>Description</b>	The Matching component displays the posted projects along with the details, giving the user to decide whether they are interested or not.
<b>Objects Used</b>	User/Project
<b>Use Cases Supported</b>	Find a Project, Find Members
<b>Mobile Notes</b>	All functionality available

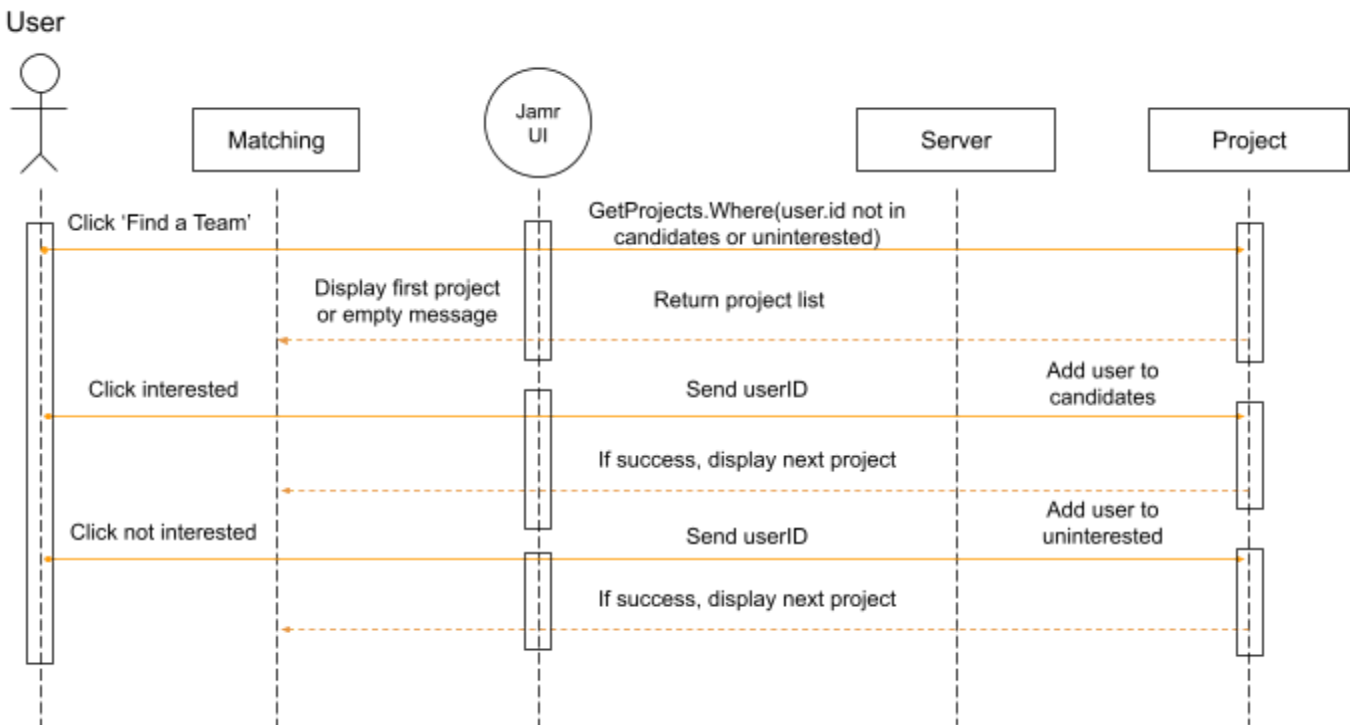


Figure 6.11 - User Find Team Sequence Diagram

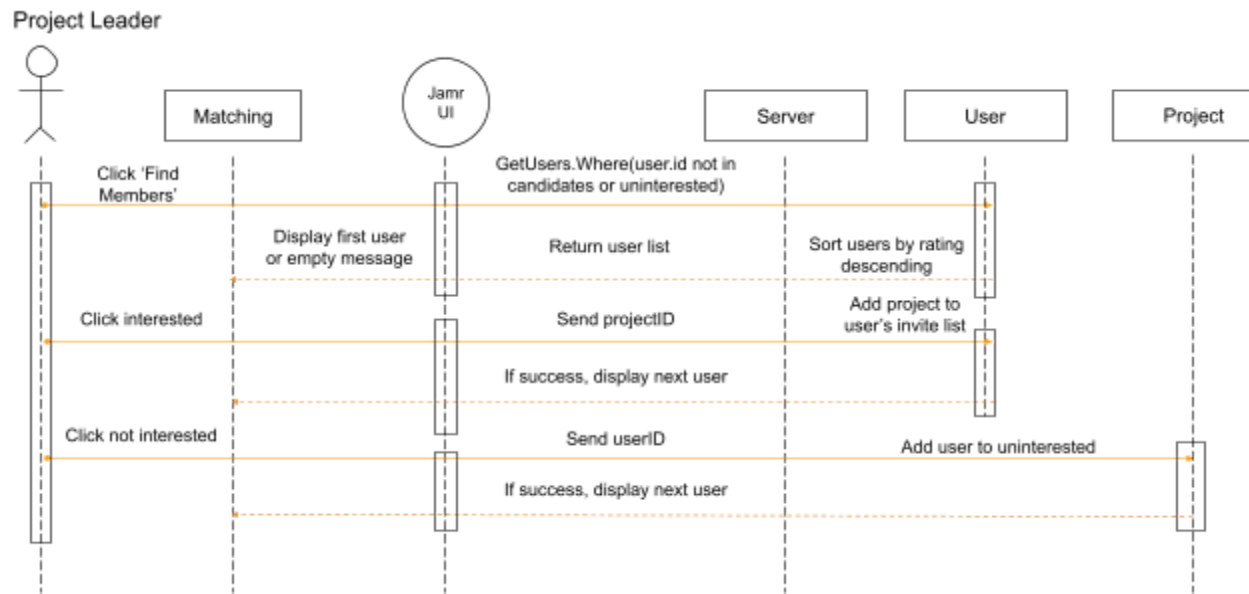


Figure 6.12 - Leader Find Member Sequence Diagram

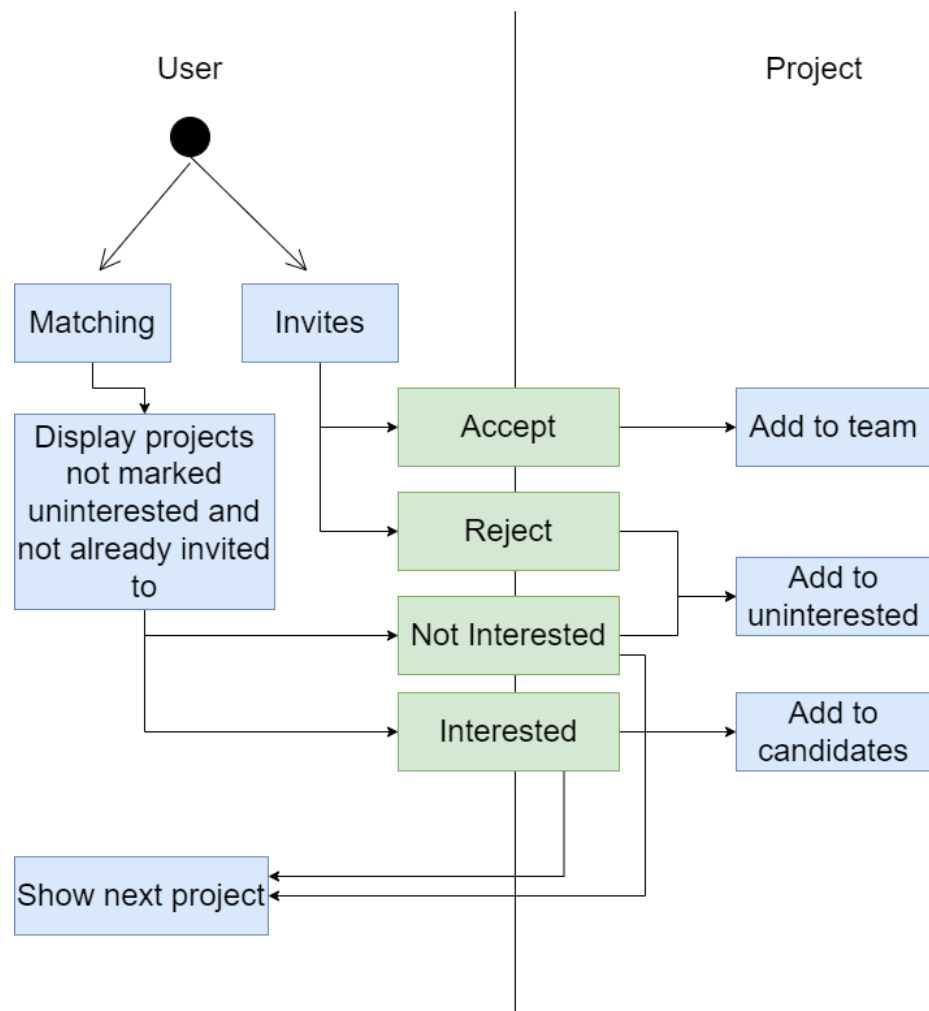


Figure 6.13 - User Matching Activity Diagram

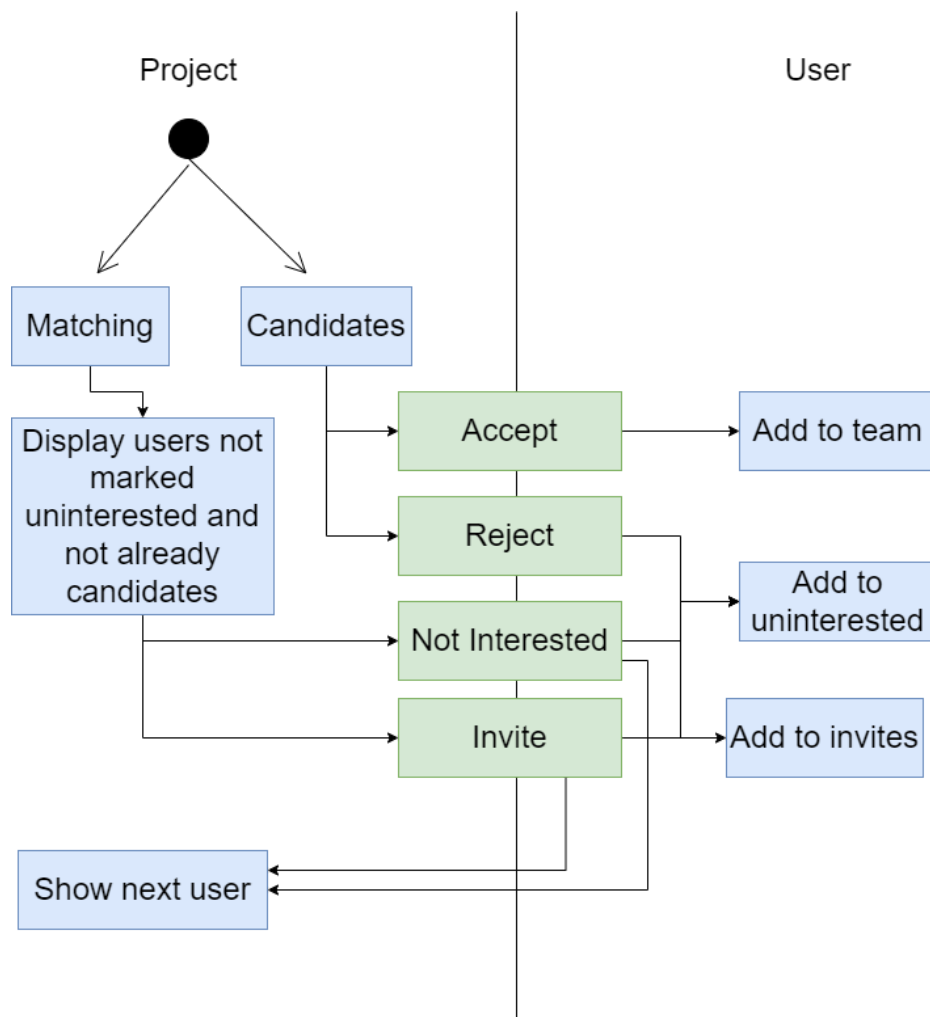


Figure 6.14 - Project Leader Matching Activity Diagram

### 6.4.2.7 Manage Invites

<b>Component</b>	Manage Invites
<b>Description</b>	This component displays all project invites the user has and allows them to view individual invitations to read more about the project. The user is also given the option to accept or decline the invitation in order to join the team they prefer.
<b>Objects Used</b>	User, Project
<b>Use Cases Supported</b>	Manage invites
<b>Mobile Notes</b>	All functionality available

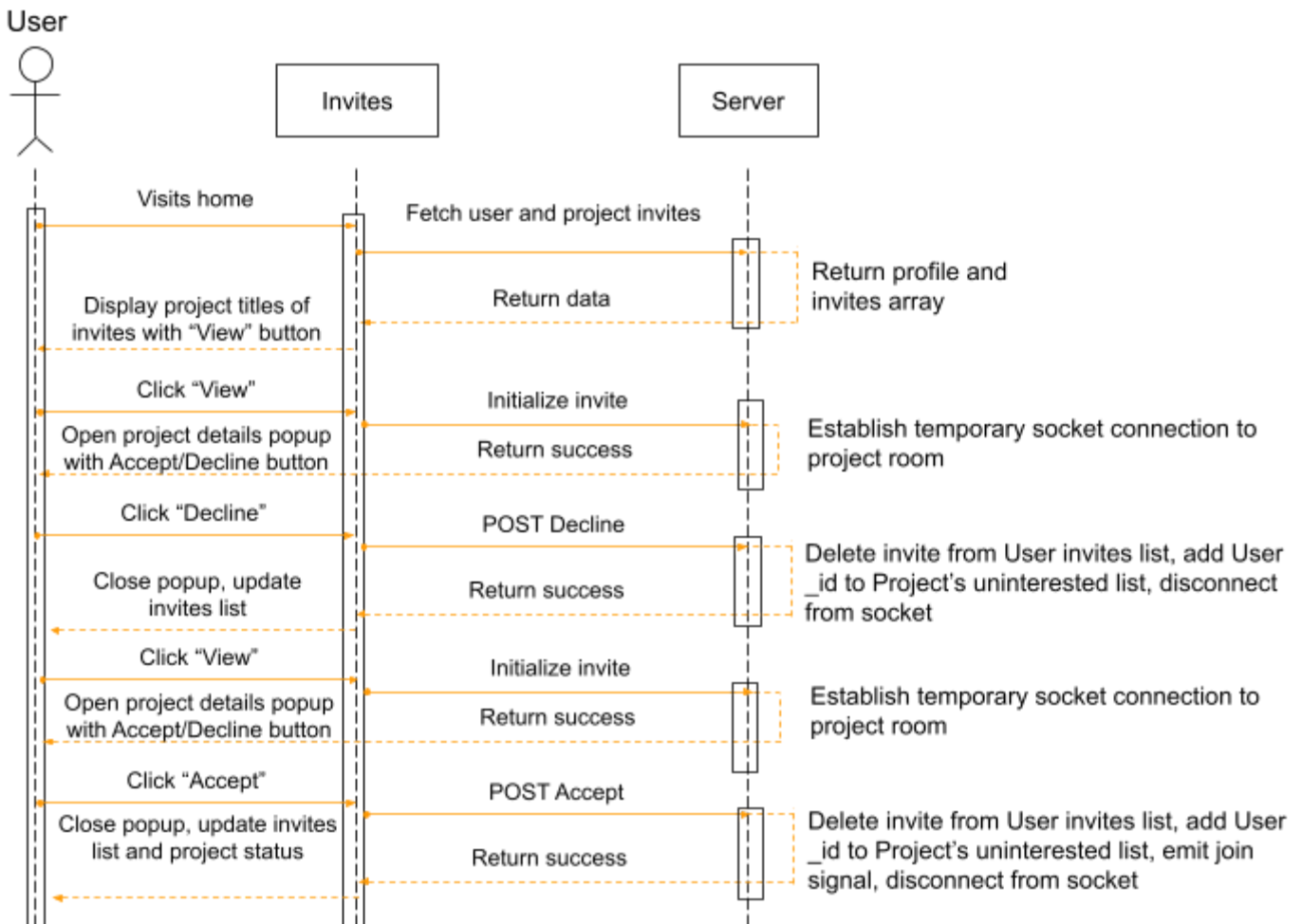


Figure 6.15 - Manage Invites Sequence Diagram

#### 6.4.2.8 Manage Candidates

<b>Component</b>	Manage Candidates
<b>Description</b>	This component is accessible only to project leaders and allows them to invite or decline potential candidates from joining the project.
<b>Objects Used</b>	User, Project
<b>Use Cases Supported</b>	Manage team members
<b>Mobile Implementation</b>	All functionality available

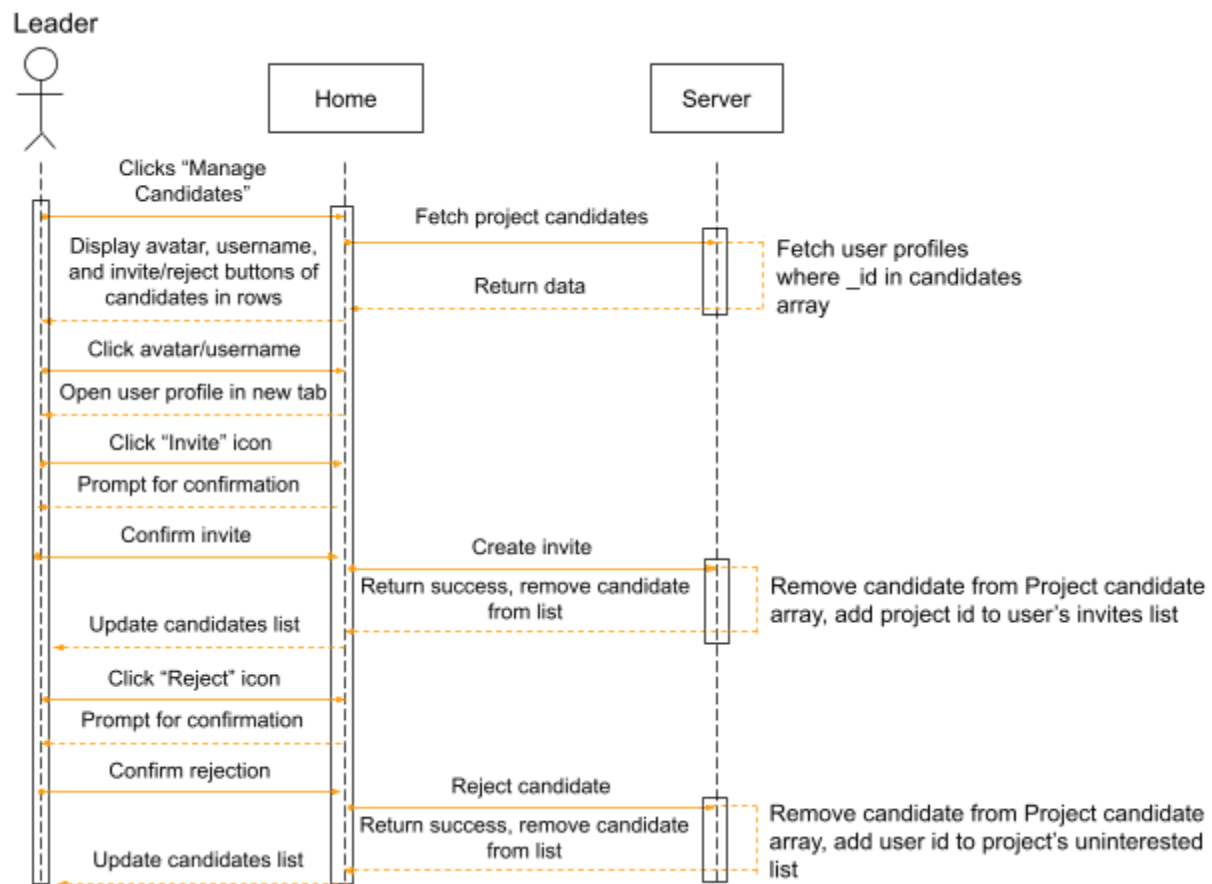


Figure 6.16 - Manage Candidates Sequence Diagram

### 6.4.2.9 Project Dashboard

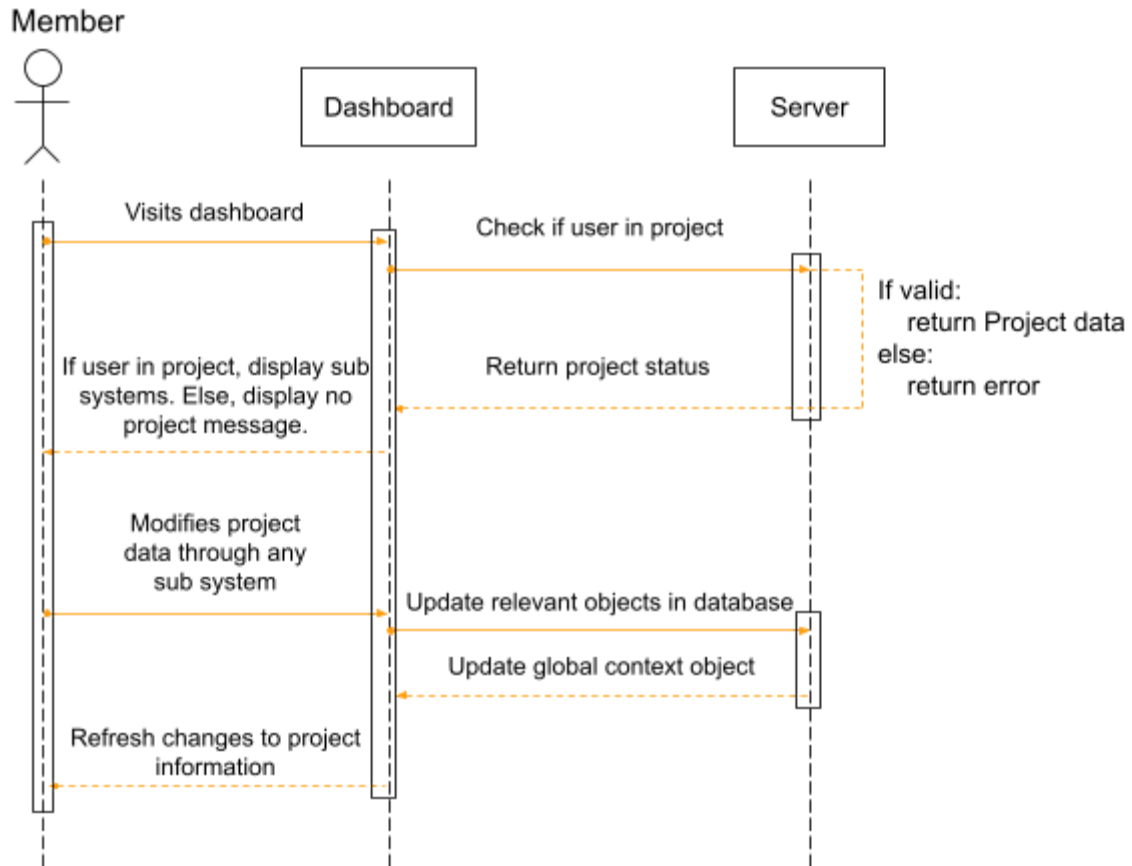


Figure 6.17 - Initial Project Dashboard Sequence Diagram

The dashboard component is a parent system containing a group of subsystems: Calendar, Todo List, Files, Development Logs, Whiteboard, and a Group Chat. The overall system provides all team member and leader use cases.



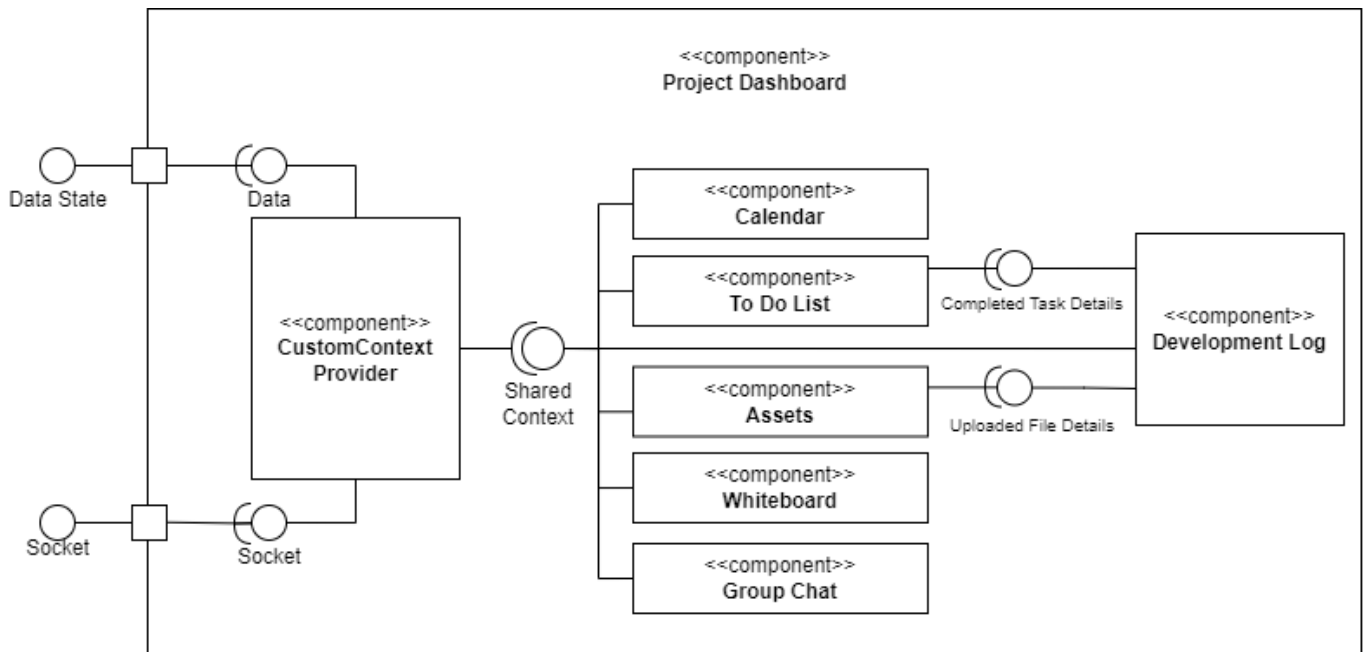


Figure 6.18 Dashboard Component Diagram

#### 6.4.2.10 Project Dashboard - Calendar

<b>Component</b>	Dashboard - Calendar
<b>Description</b>	This component is a planning tool for members to set calendar events.
<b>Objects Used</b>	User, Project, Event
<b>Use Cases Supported</b>	Calendar
<b>Mobile Implementation</b>	Component is read-only

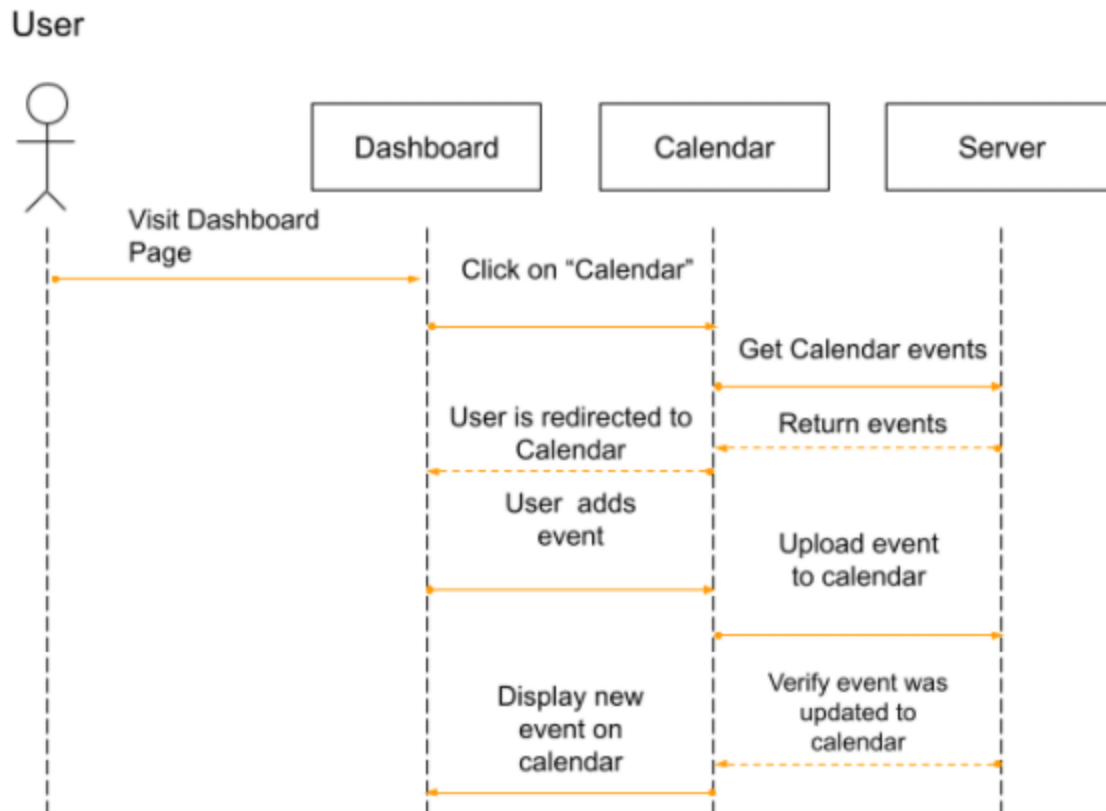


Figure 6.19 - Initial Calendar Sequence Diagram

#### 6.4.2.11 Project Dashboard - To Do List

<b>Component</b>	Dashboard - Task/To-do List
<b>Description</b>	The task list is a planning component used to track project activities. Every member can add or remove list items and mark them as completed.
<b>Objects Used</b>	User, Project, Task
<b>Use Cases Supported</b>	To do list
<b>Mobile Implementation</b>	Component is read-only

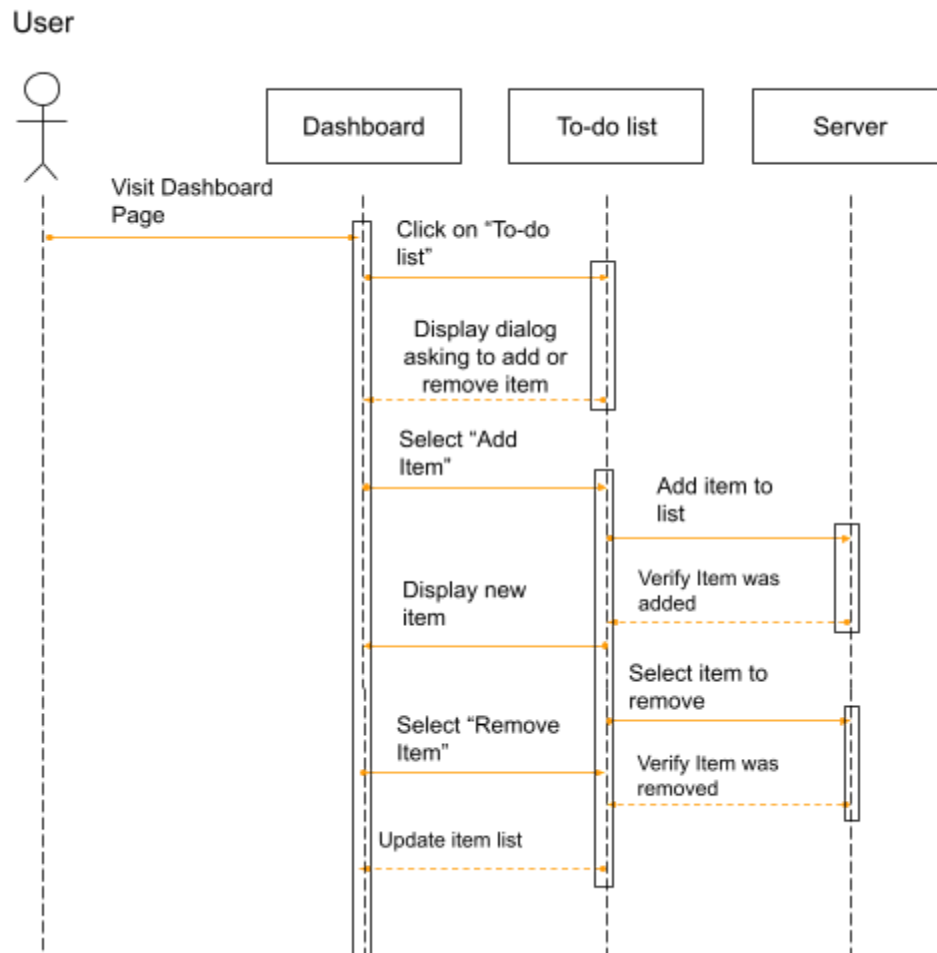


Figure 6.20 - Initial To-Do List Sequence Diagram

#### 6.4.2.12 Project Dashboard - Assets/File Sharing

<b>Component</b>	Dashboard - File Sharing
<b>Description</b>	The assets module is used for members to share project resources with other members. Every member can upload files to the central folder, along with deleting their own uploaded files. Project managers may delete any files uploaded by a member.
<b>Objects Used</b>	User, Project, File
<b>Use Cases Supported</b>	Share Files
<b>Mobile Implementation</b>	Component is read-only, files cannot be downloaded - only details can be seen

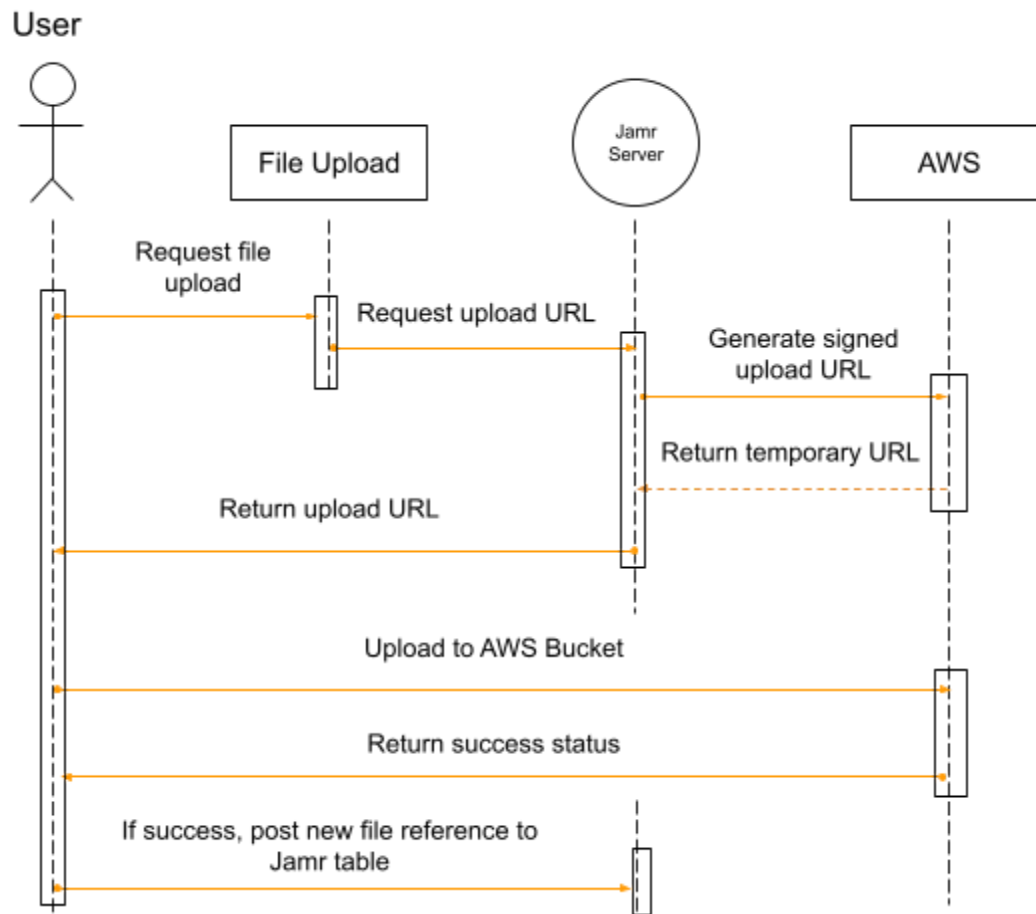


Figure 6.21 - Initial Asset Sequence Diagram

#### 6.4.2.13 Project Dashboard - Development Log

<b>Component</b>	Dashboard - Dev Log
<b>Description</b>	This component is a module to document development efforts and upload assets to the project accessible to every member.
<b>Objects Used</b>	User, Project, DevelopmentLog, File
<b>Use Cases Supported</b>	Share files, Development Log
<b>Mobile Implementation</b>	Component is read-only

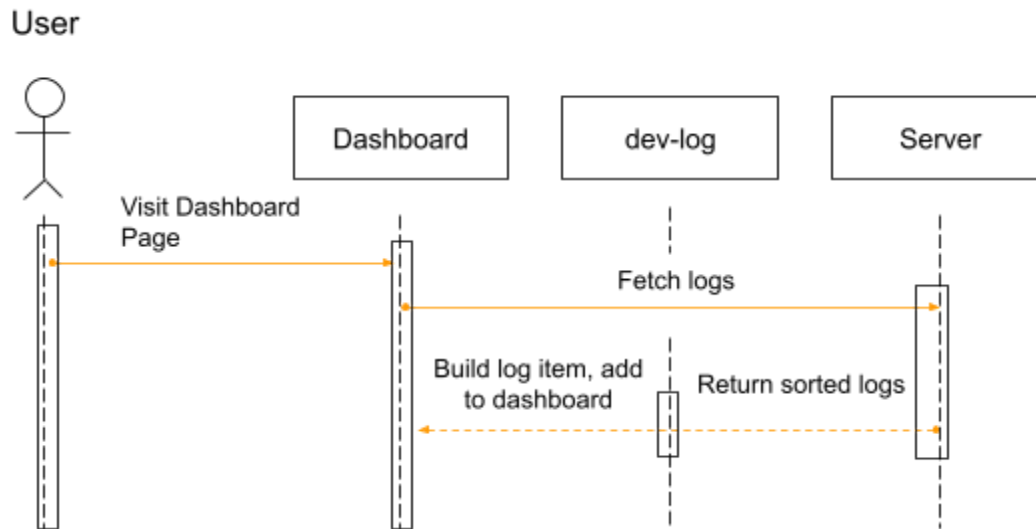


Figure 6.22 - Development Log Sequence Diagram

#### 6.4.2.14 Project Dashboard - Project Management

<b>Component</b>	Dashboard - Project Management
<b>Description</b>	This component is accessible only by team leaders and allows them to modify project class details.
<b>Objects Used</b>	Project, User
<b>Use Cases Supported</b>	Manage project, manage team members
<b>Mobile Implementation</b>	Component is only available on the Web Application

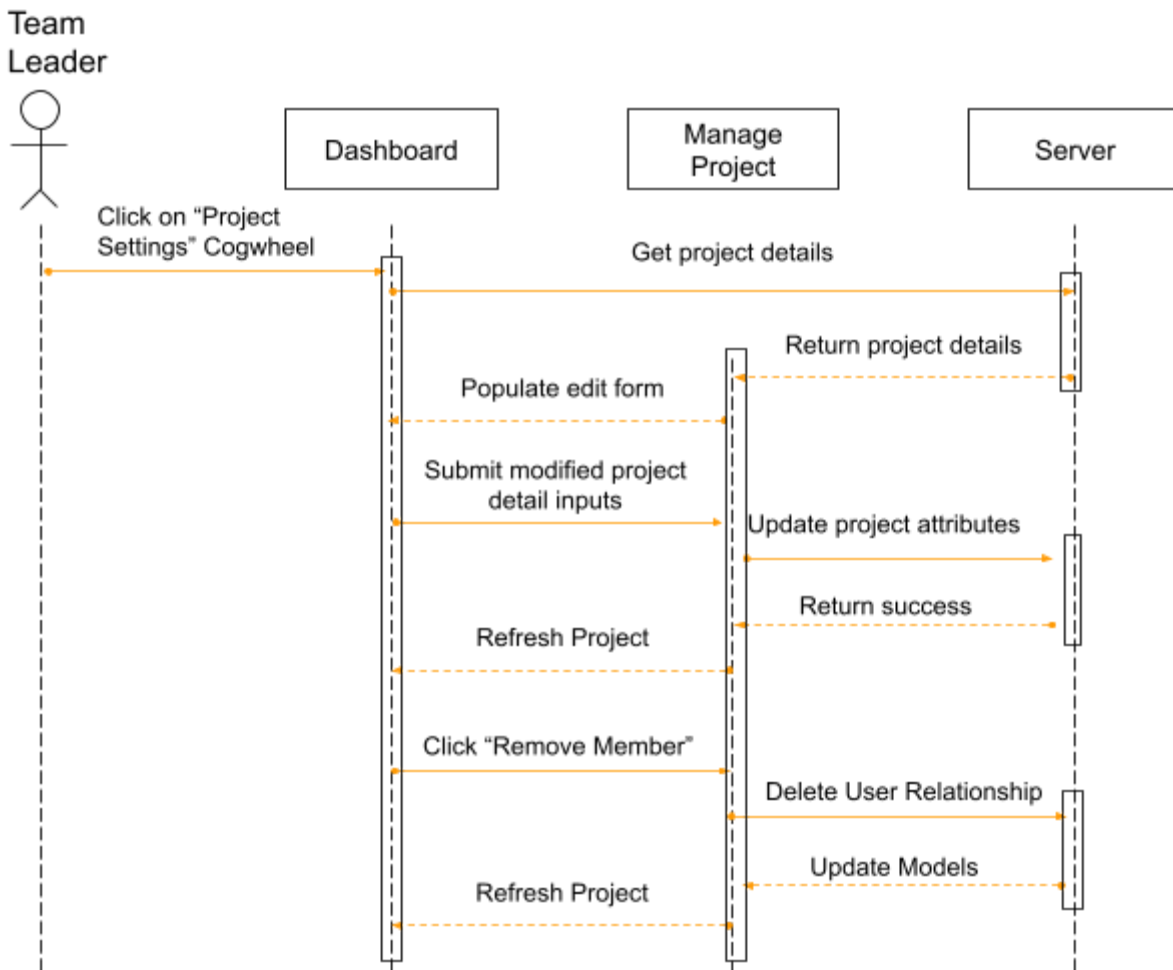


Figure 6.23 - Initial Project Management Sequence Diagram

#### 6.4.2.15 Project Dashboard - Whiteboard

<b>Component</b>	Dashboard - Whiteboard
<b>Description</b>	The whiteboard component is usable by all project members. The component displays a thumbnail of the most recent drawing and clicking on the thumbnail opens a live drawing session where members may draw simultaneously.
<b>Objects Used</b>	User, Whiteboard
<b>Use Cases Supported</b>	Whiteboard

<b>Mobile Implementation</b>	Component is read-only and only updates when a thumbnail update occurs
------------------------------	--

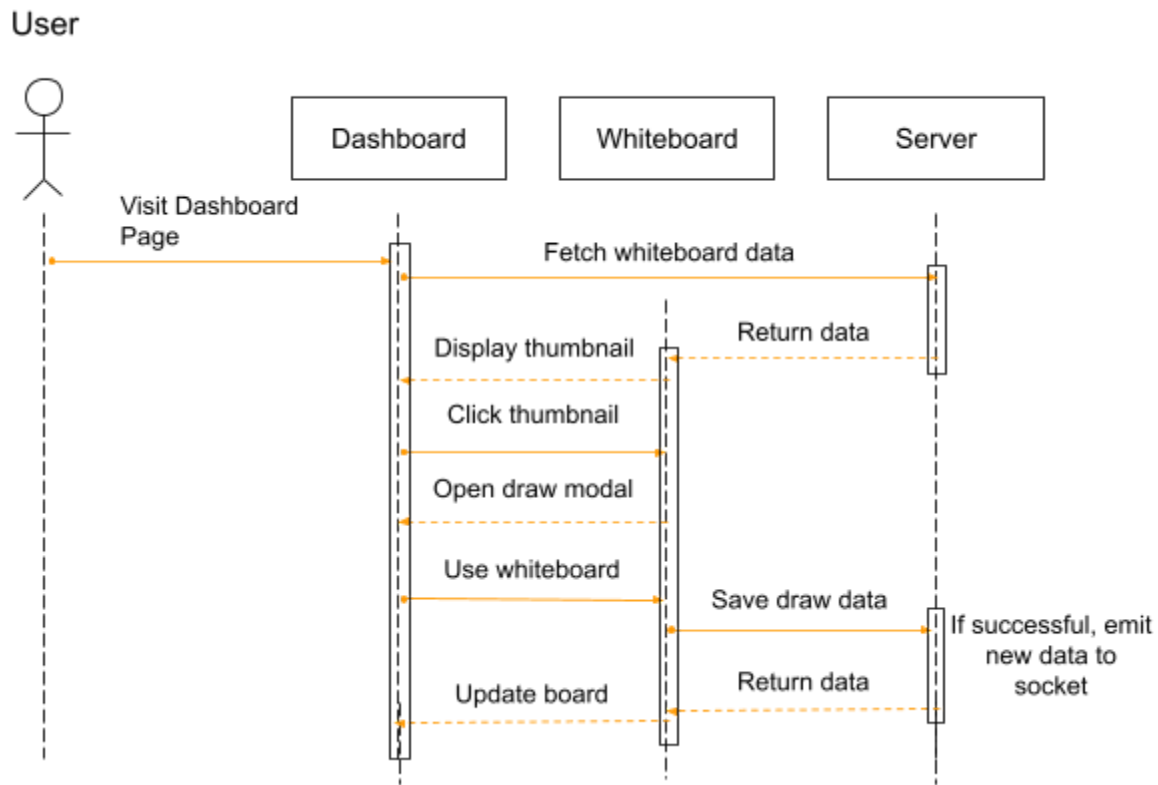


Figure 6.24 - Initial Whiteboard Sequence Diagram

#### 6.4.2.16 Project Dashboard - Group Chat

<b>Component</b>	Dashboard - Group Chat
<b>Description</b>	This component allows group members to communicate in real time via text chat. Messages are sent once they are successfully saved and the component is automatically refreshed upon receiving new ones.
<b>Objects Used</b>	User, Message
<b>Use Cases Supported</b>	Team Chat
<b>Mobile Notes</b>	All functionality available

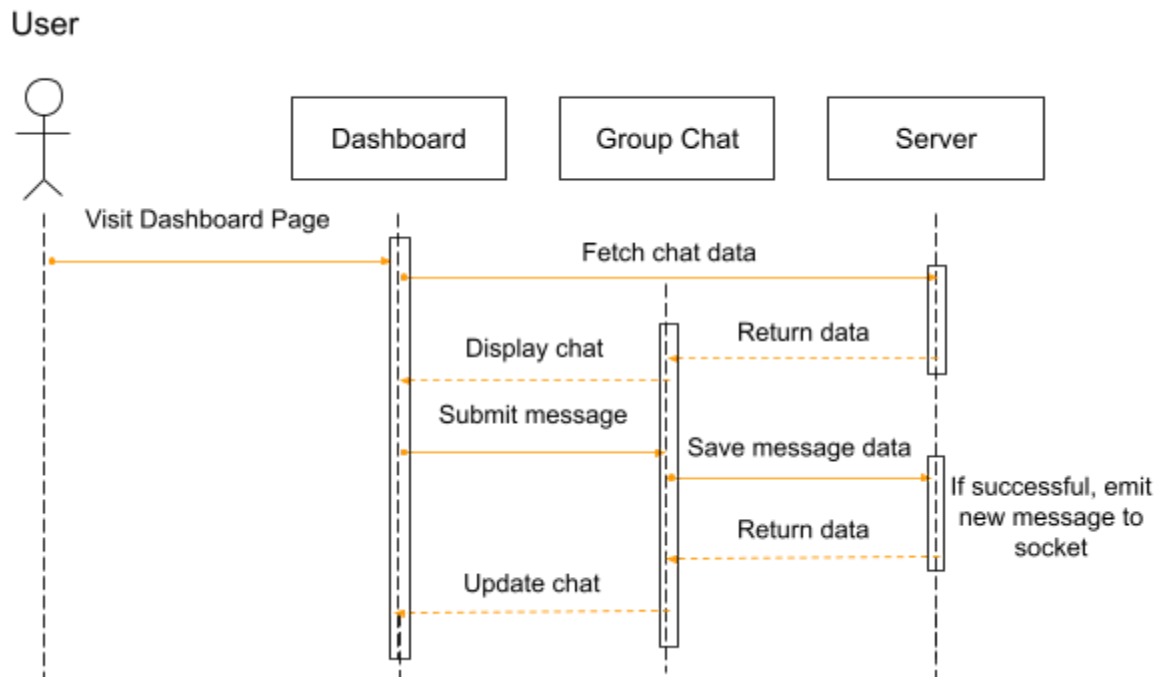


Figure 6.25 - Initial Group Chat Sequence Diagram

#### 6.4.2.17 Project Dashboard - Member Settings

<b>Component</b>	Dashboard - Member Settings
<b>Description</b>	This component is shown to all team members and is used to give ratings to current teammates or to let a member leave the project.
<b>Objects Used</b>	User
<b>Use Cases Supported</b>	Rate Team Members
<b>Mobile Notes</b>	Component not available on mobile



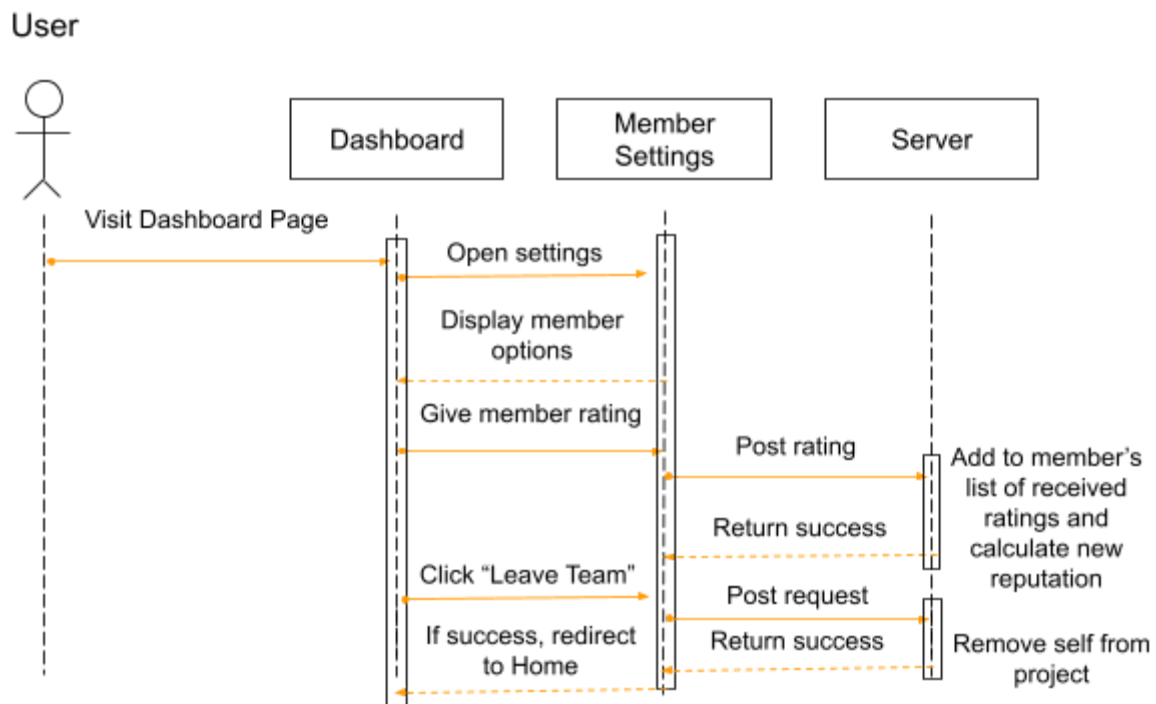


Figure 6.26 - Initial Member Settings Sequence Diagram

#### 6.4.2.18 User Settings

<b>Component</b>	User Settings
<b>Description</b>	This component allows users to modify site personalization features such as themes and gives them the ability to delete their account.
<b>Objects Used</b>	User
<b>Use Cases Supported</b>	Modify profile
<b>Mobile Notes</b>	Component not available on mobile

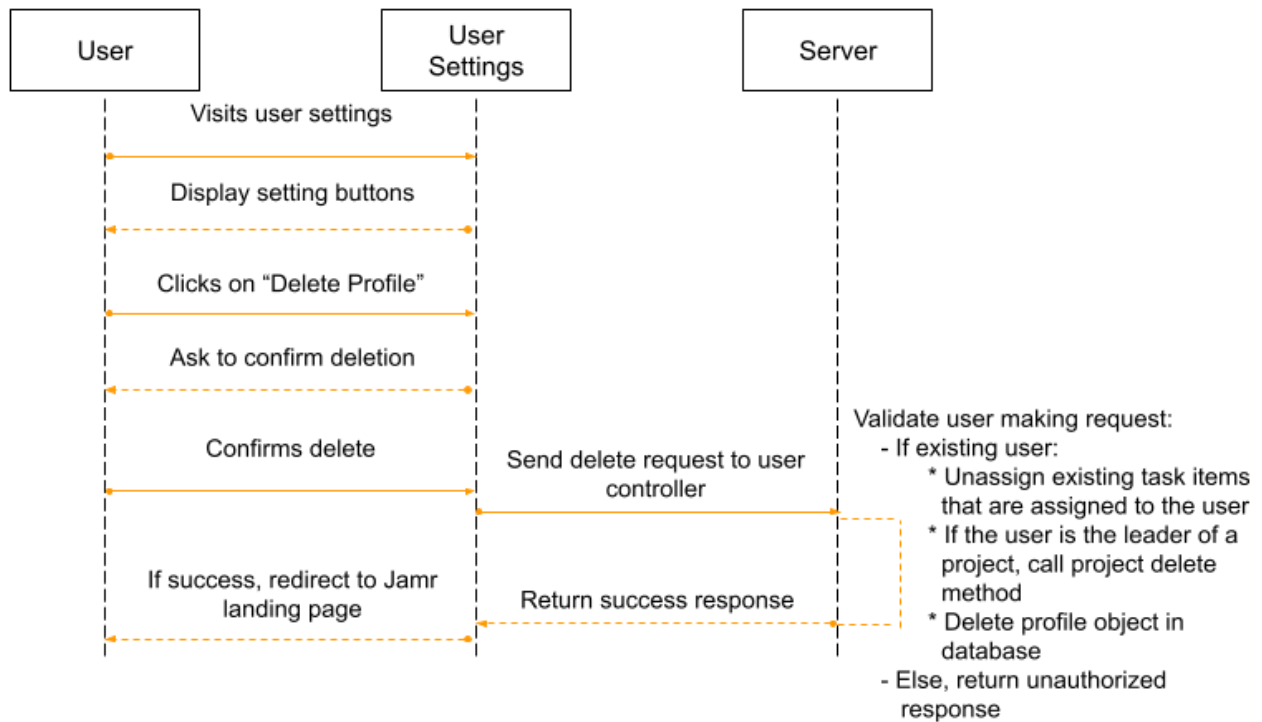


Figure 6.27 - Initial User Settings Sequence Diagram

#### 6.4.2.19 Profile

Component	Profile
<b>Description</b>	The Profile component allows users to edit their profile by changing their bio, profile picture, tags, and showcased games. Users may also view the profile of other users. If viewing the profile of another user, they may send a friend request or remove the user if they are currently connected.
<b>Objects Used</b>	User
<b>Use Cases Supported</b>	Modify profile, View Profile, Manage Friends
<b>Mobile Notes</b>	Component not available on mobile

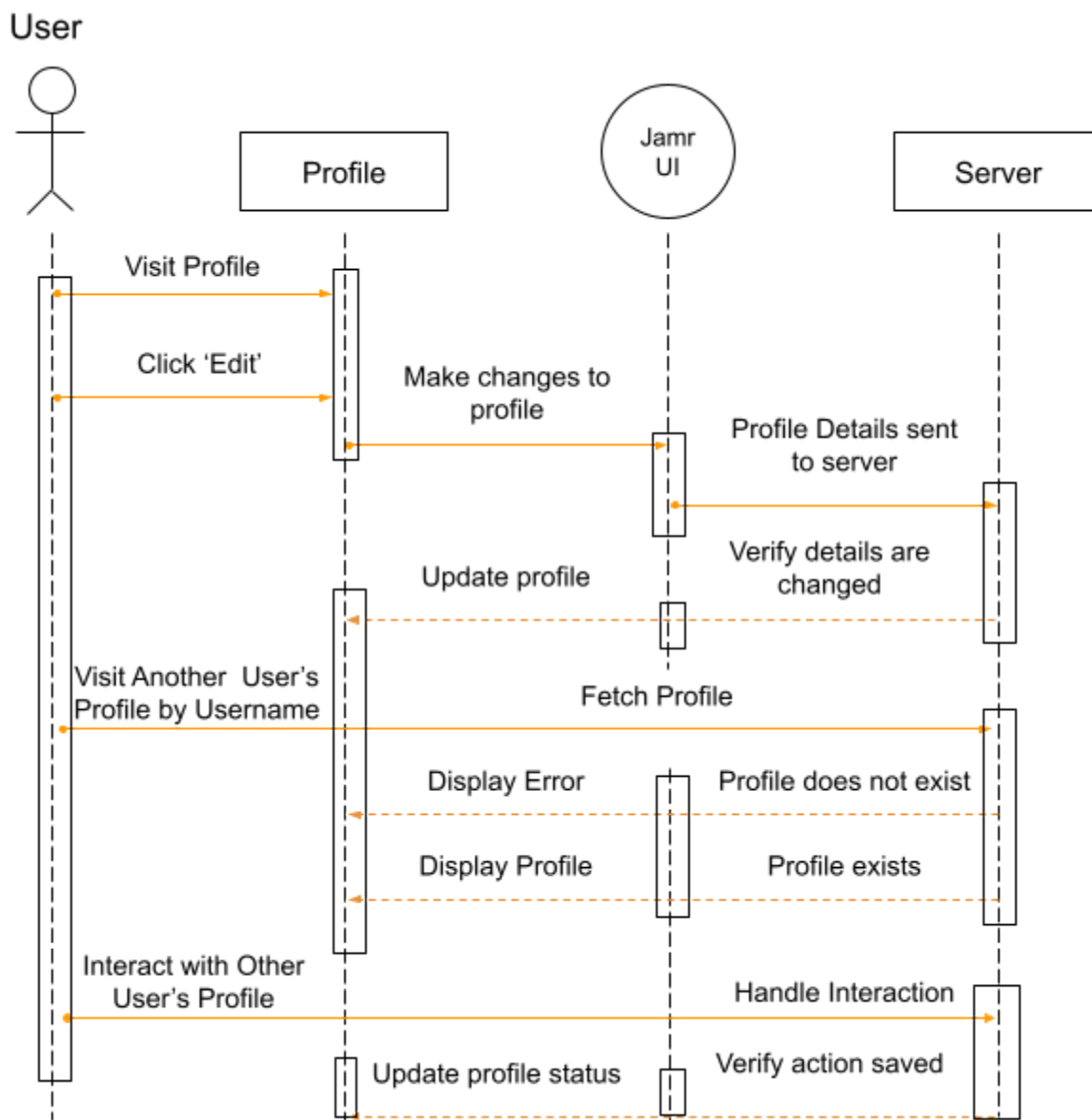


Figure 6.28 - Initial User Profile Sequence Diagram

#### 6.4.2.20 Friends

<b>Component</b>	Friends
<b>Description</b>	The Friends component allows users to add or remove users from the 'friends list'. Users can also view current users that are listed as 'friends'
<b>Objects Used</b>	User
<b>Use Cases Supported</b>	Friends
<b>Mobile Notes</b>	Component not available on mobile

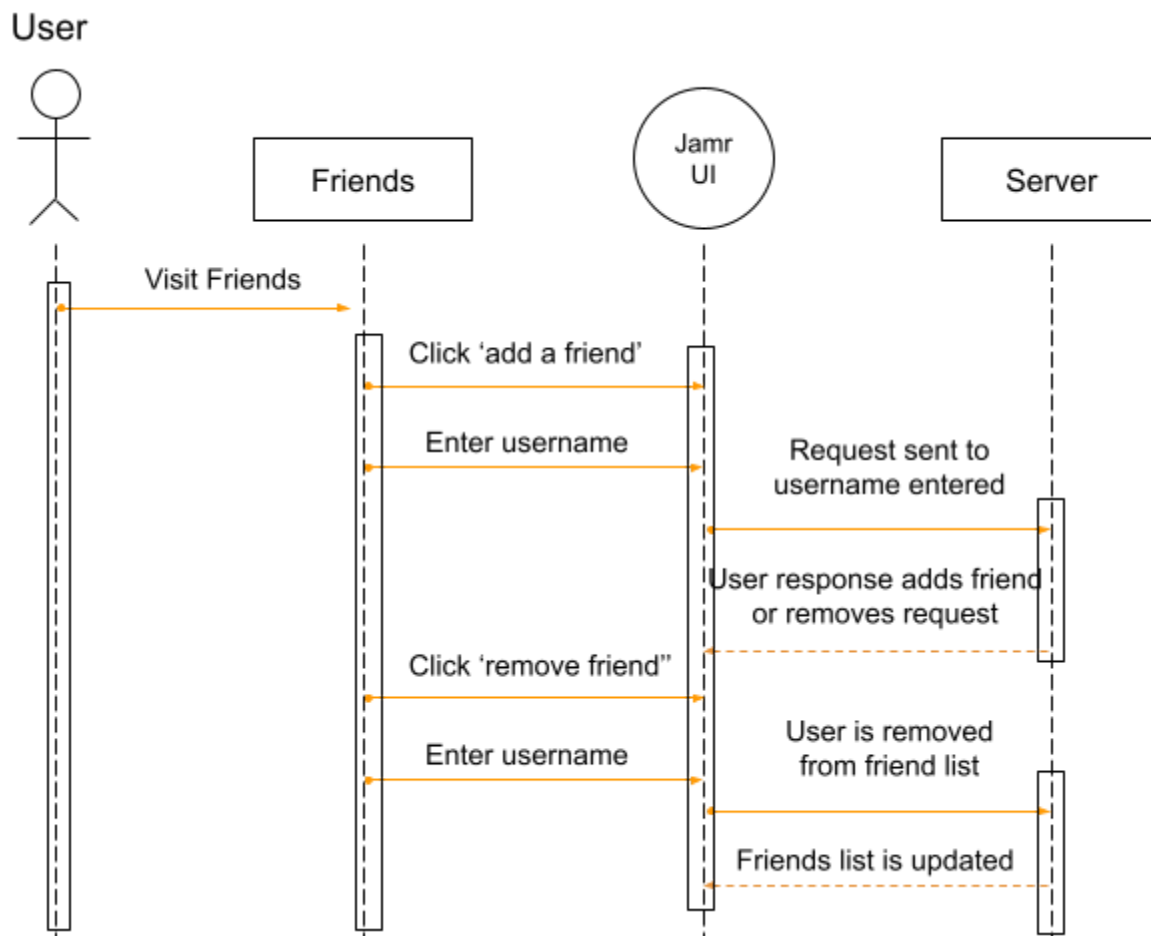


Figure 6.29 - Initial Friends Sequence Diagram

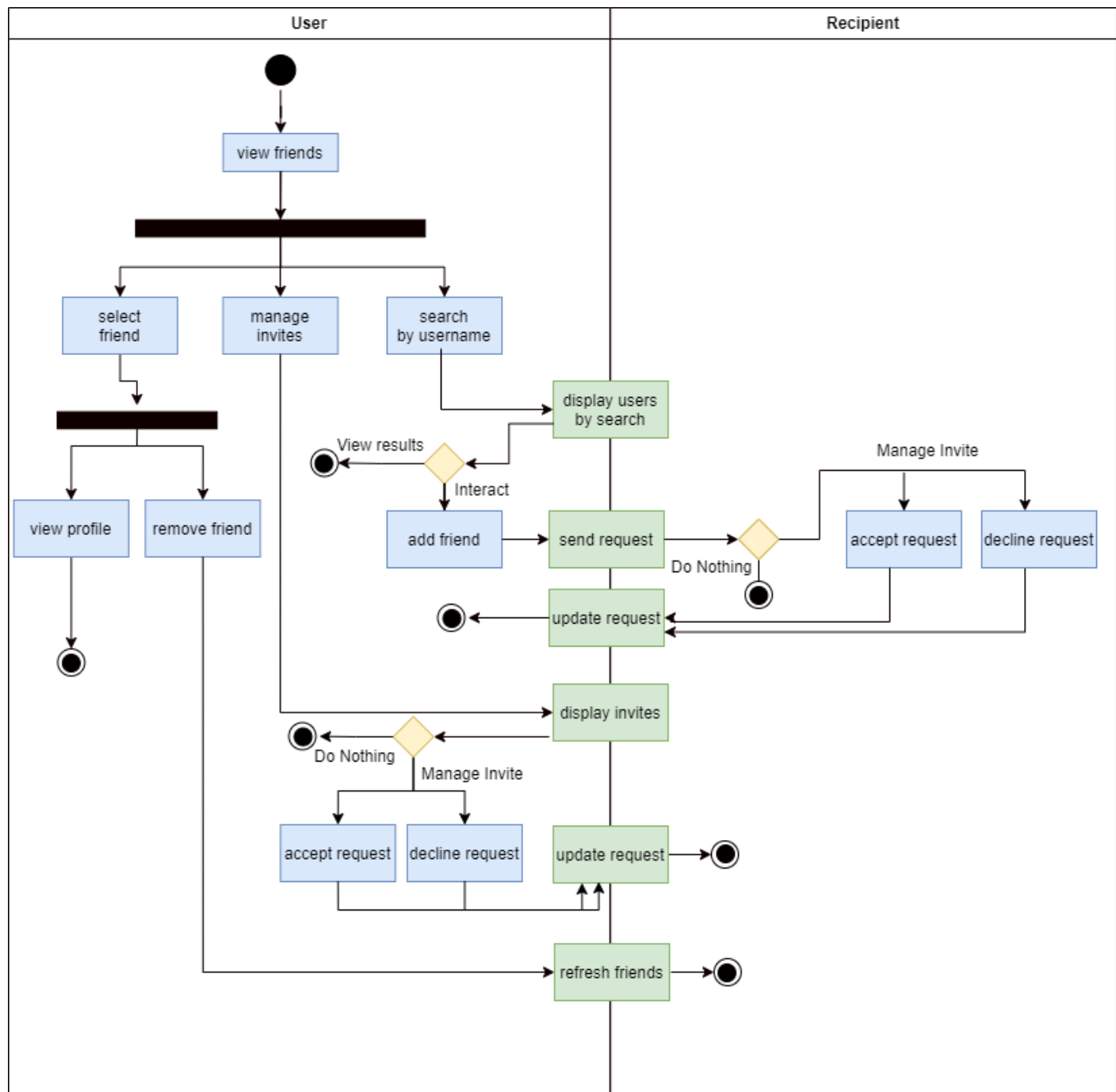


Figure 6.30 - Friends Activity Diagram

### 6.4.2.21 Messaging

<b>Component</b>	Direct Messaging
<b>Description</b>	This component is a conversation tool for users to communicate to
<b>Objects Used</b>	User, Message
<b>Use Cases Supported</b>	Direct Messaging
<b>Mobile Notes</b>	All functionality available

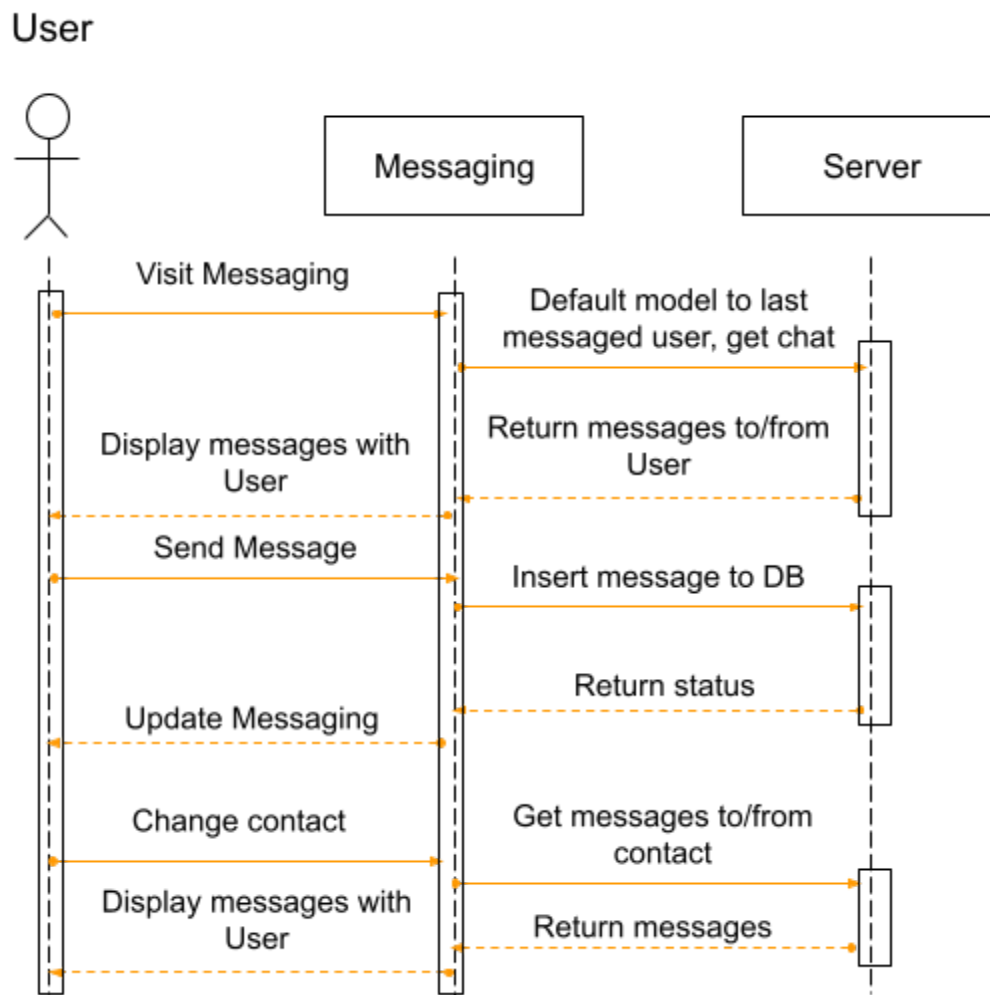


Figure 6.31 - Initial Messaging Sequence Diagram

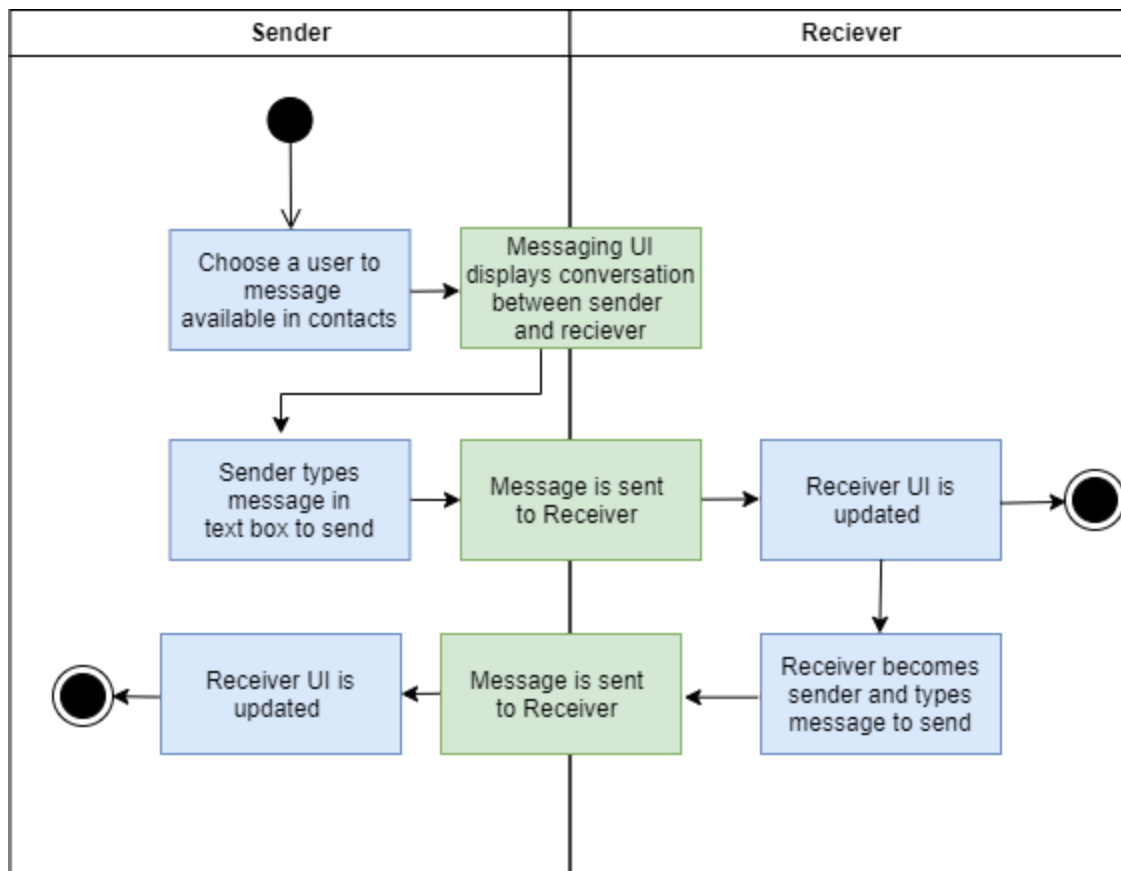


Figure 6.32 - Messaging Activity Diagram

#### 6.4.2.22 Web Navigation Bar

<b>Component</b>	Navbar
<b>Description</b>	This component provides Navigation throughout the application
<b>Objects Used</b>	User
<b>Use Cases Supported</b>	Navigation

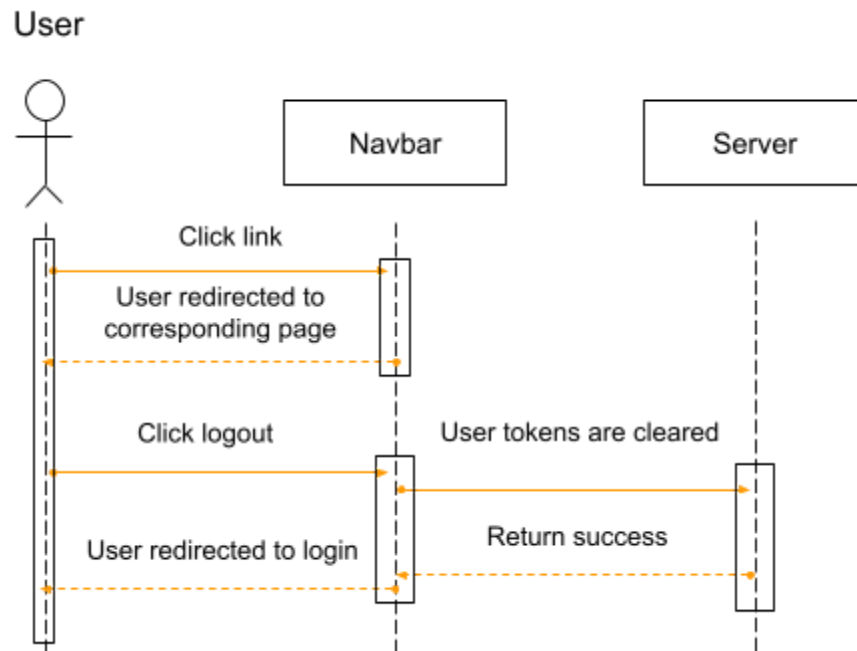


Figure 6.33 - Initial Web Navigation Bar Sequence Diagram



### 6.4.2.23 Mobile Navigation Bar

<b>Component</b>	Mobile Navbar
<b>Description</b>	This component provides Navigation throughout the native application
<b>Objects Used</b>	User
<b>Use Cases Supported</b>	Navigation

Mobile User

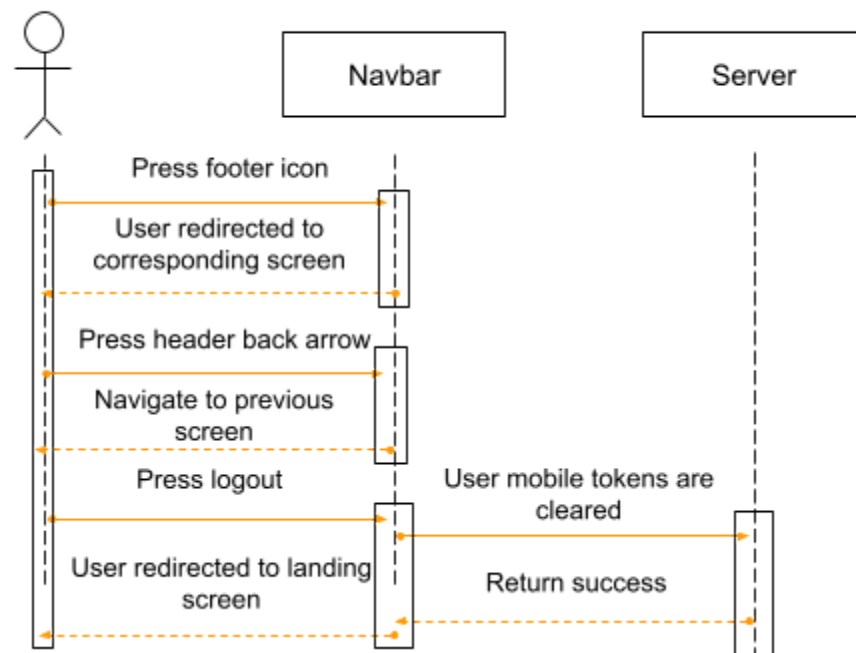


Figure 6.34 - Initial Mobile Navigation Bar Sequence Diagram

## 6.5 Overall Application Operation

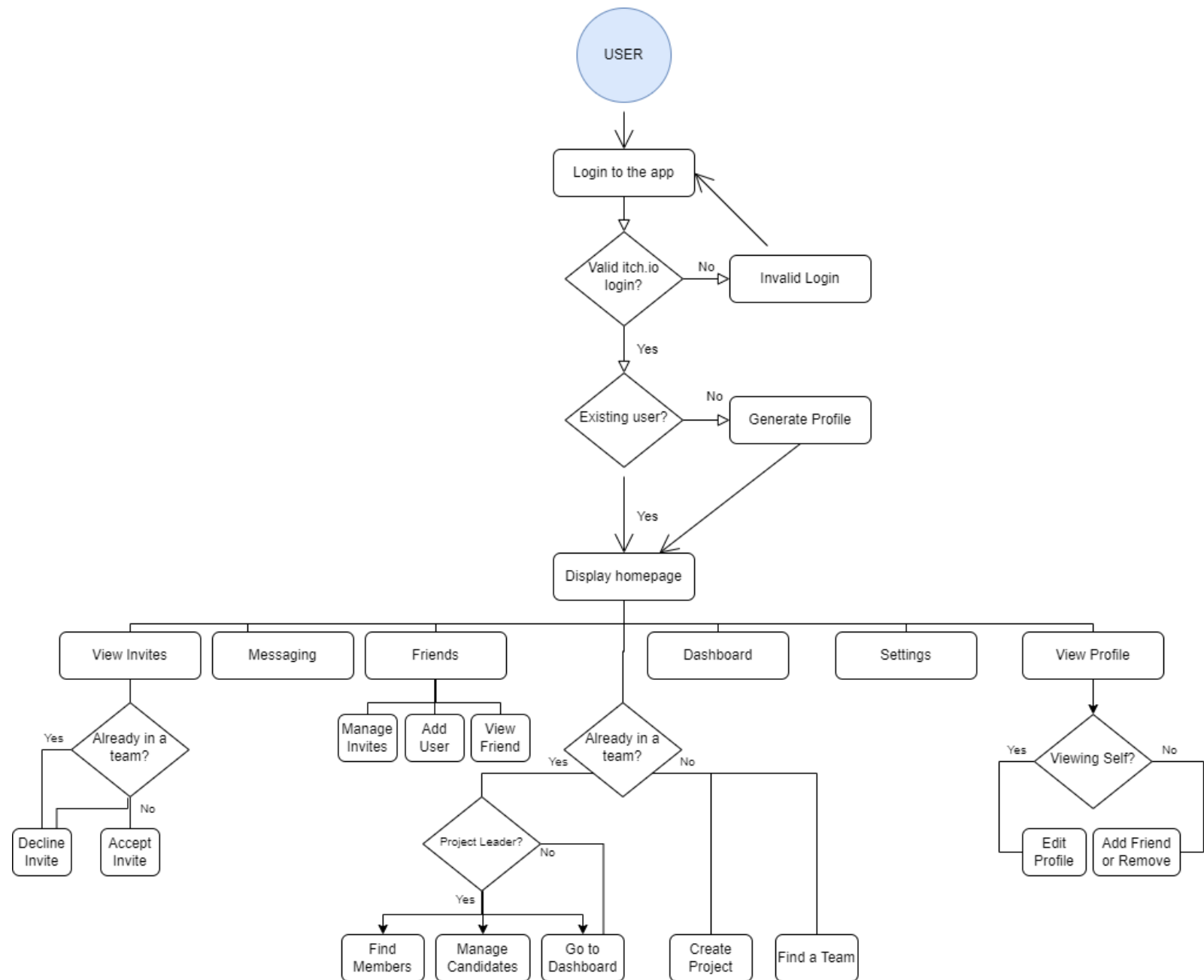


Figure 6.35 - Overall Web Application Diagram

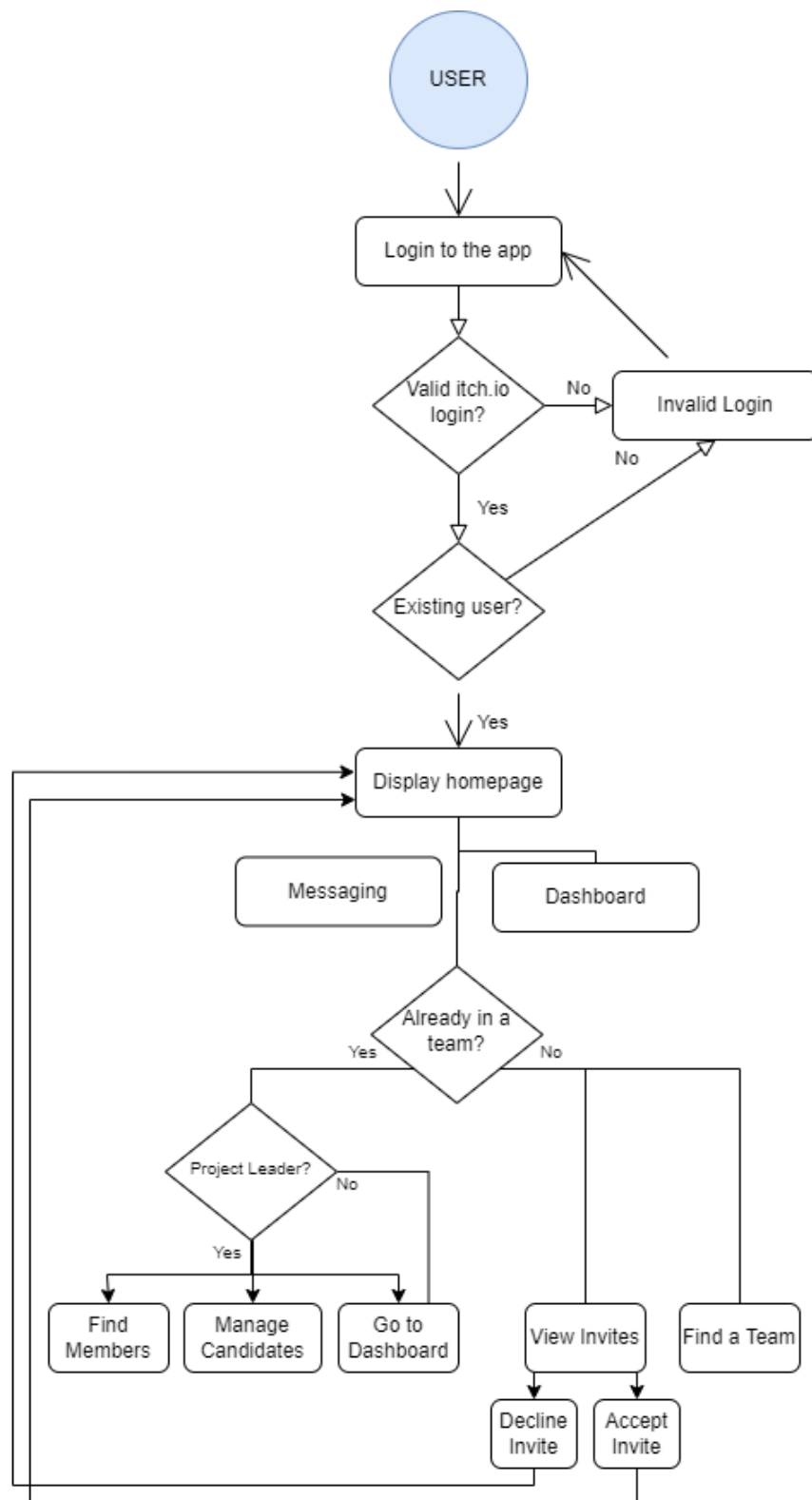


Figure 6.36 - Overall Mobile Application Diagram

## 6.6 Entity Relationship Data Model

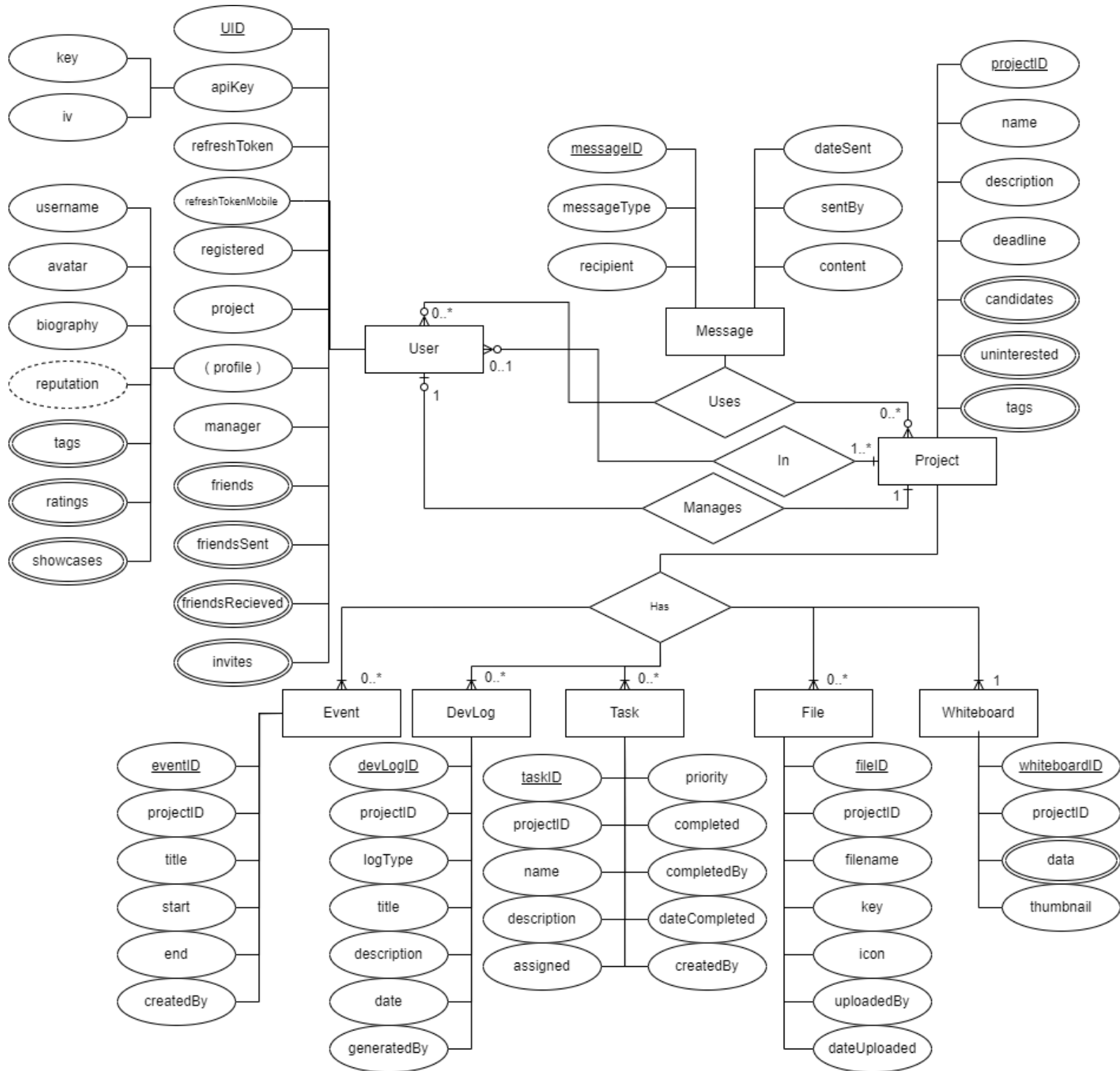


Figure 6.37 - Entity Relationship Data Model

## 7. Implementation

### 7.1 Subsystems

Functionality is achieved by connecting frontend action elements to backend method controllers through a REST API. The following sections describe the team's approach, note any revisions to the design of the components listed in the previous chapter, and a breakdown of the tasks required to implement the feature.

The mobile companion applications provide a limited subset of the functionalities available on the web application. By deploying the web application online, the mobile applications utilize the existing backend API methods defined for the web to handle authentication, logins, and data requests. Limiting functionality for mobile is achieved by removing interactive elements such as edit buttons and by defining new server methods to handle specific mobile requests. The changes in implementation are noted in the descriptions of the following subsystems.

#### 7.1.1 User Login - Web Application

##### Approach

Functional approach using keys taken from URL parameters on redirection to/from itch.io and Jamr, and passed to a UserController on the backend. The user object on Jamr is initialized with information from their public itch.io account on first time login and assigned unique data within Jamr. Their API key is finally hashed for security and stored within the database for later use. The login module is responsible for redirecting the user to the registration page if they have not previously registered their profile or to the homepage if they have an existing account.

##### Scrum Task Breakdown

User Story ID	Tasks	Owner	Estimate (Hr)	Status
1	Create itch.io OAuth application and tie to project, redirecting users on "Get Started" button click	Logan Fite	1	<b>Complete</b>
1	Fetch itch.io authentication response on redirect back to Jamr, confirm validity	Logan Fite	2	<b>Complete</b>
2	Generate JWT Access & Refresh Tokens if valid login, storing in DB and provide to user on redirect	Logan Fite	8	<b>Complete</b>
1	Redirect user to appropriate page (Home/Register) depending on user status	Logan Fite	.25	<b>Complete</b>

**User Training:**

Section 1 of User Manual

**7.1.2 User Login - Mobile Application****Approach**

The mobile application requires that the user has previously registered a Jamr account from a web browser. The same approach of using keys taken from URL parameters on redirection from the itch.io OAuth is used, but the key is sent to a different, mobile login function. This function will redirect to the homepage if the login is valid and an account exists for the user, otherwise it displays an account required page to the user with instructions to register on the web.

**Scrum Task Breakdown**

User Story ID	Tasks	Owner	Estimate (Hr)	Status
1	Create itch.io OAuth application and tie to mobile project, redirecting users on “Get Started” button click	Logan Fite	1	<b>Complete</b>
1	Fetch itch.io authentication response on redirect back to Jamr, confirm validity	Logan Fite	1	<b>Complete</b>
2	Generate Mobile JWT Access & Refresh Tokens if valid login, storing in DB and provide to user on redirect	Logan Fite	8	<b>Complete</b>
1	Redirect user to appropriate page (Home/Invalid) depending on user status	Logan Fite	.25	<b>Complete</b>

**User Training:**

Section 5.1 of User Manual

**7.1.3 User Registration****Approach**

The user registration module combines the basic profile information retrieved from the login process via itch.io OAuth with required form data asking the user to write a short biography and assign game development tags to themselves to build a complete user profile. Once the user submits this information, the server updates the user object and marks their profile as registered.

### Scrum Task Breakdown

User Story ID	Tasks	Owner	Estimate (Hr)	Status
1	Create registration page with form	Logan Fite	3	<b>Complete</b>
1	Populate user data from user to be registered, hook form entry data to object	Logan Fite	3	<b>Complete</b>
1	Add frontend validation method to ensure inputs are valid (Biography >= 50 characters)	Logan Fite	.25	<b>Complete</b>
1	On submit, update User attributes, mark as registered, redirect to home page	Logan Fite	2	<b>Complete</b>

### User Training

Section 1 of User Manual

### 7.1.4 Middleware Token Authentication

#### Revised Design

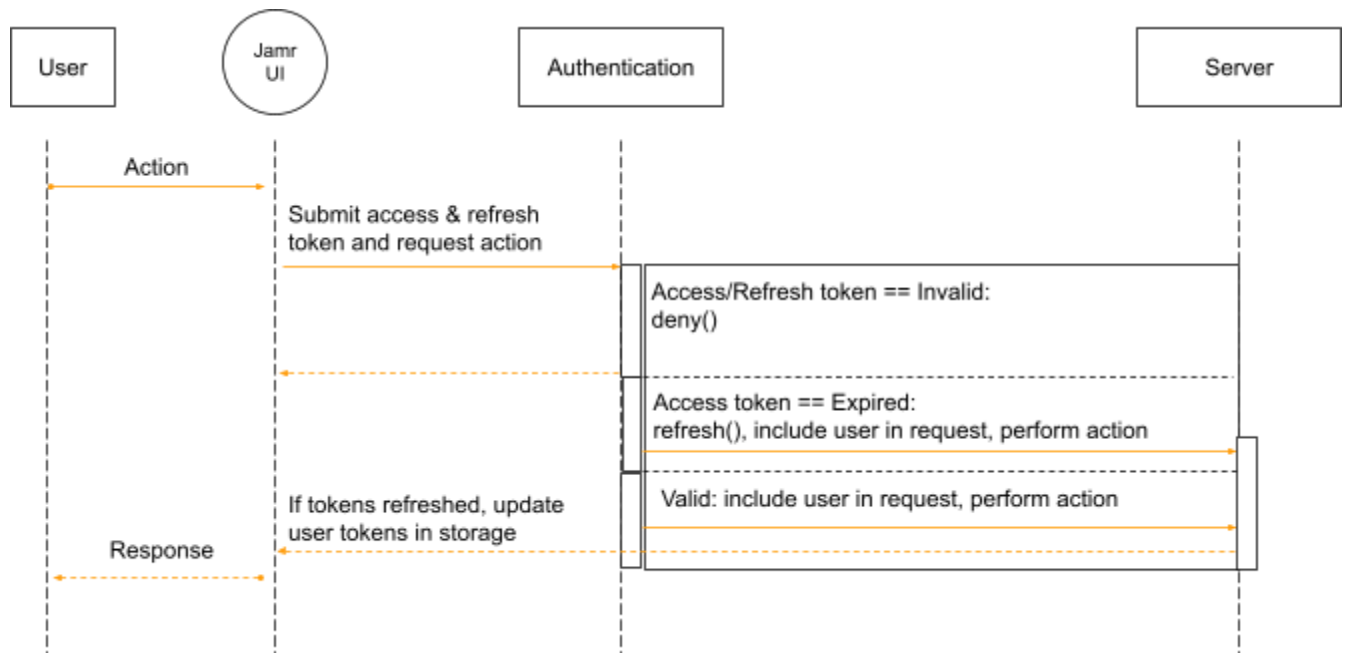


Figure 7.1 - Revised Middleware Token Authentication Sequence Diagram

### Approach

Upon registration and subsequent logins, a user is generated a secure access and refresh JSON Web Token to be stored within their browser. When the users make requests to visit application pages or use functional modules, the server first verifies the validity of these tokens by decrypting the tokens, verifying their signatures, and comparing them against the Jamr database to ensure the user has been given a matching token. As a revision, if the token is valid, the user object is also appended to the original request so that object controllers have access to information about the user making the request. To reflect modern security standards, the access tokens are short lived (currently 10 minutes) and need to be regenerated using a valid refresh token once expired. When a user logs out, their refresh token is revoked by removing it from the database for additional security.

### Mobile Implementation Notes

The mobile application uses the same authentication system, but stores a mobile specific refresh token in order to keep login sessions active on the web when logging out of the mobile application.

### Scrum Task Breakdown

User Story ID	Tasks	Owner	Estimate (Hr)	Status
2	Generate verifiable access tokens	Logan Fite	5	<b>Complete</b>
2	Set access tokens to expire, generate long-lasting refresh tokens that can be used to renew expired tokens	Logan Fite	8	<b>Complete</b>
2	Write modular function to be included across all requests, returning appropriate status code or passing request to endpoint along with user information	Logan Fite	10	<b>Complete</b>
2	Create modular token utility functions imported on each page to verify, generate, and delete tokens from user's browser	Logan Fite	3	<b>Complete</b>



**Backend/API Testing & Results**

Actions	Expected Behavior	Result	Notes
Request made to protected endpoint without access or refresh token	401 Status	<b>Success</b>	
Request made to protected endpoint with invalid access token	403 Status - Include permission error in response	<b>Success</b>	
Request made to protected endpoint with expired access token	403 Status - Include expiration notice in response	<b>Success</b>	
Request made with valid token	API Request forwarded to target endpoint	<b>Success</b>	

**User Training**

Not applicable

**7.1.5 Home Page****Approach**

The home page consists of a dynamically rendered action button group depending on the user's project status and a list of project invites. Modifying which buttons are available to the user controls the actions they are allowed to make to prevent unwanted scenarios (such as visiting the "Find a Project" page as a project leader).

**Mobile Implementation Notes**

The mobile application uses the same authentication system, but stores a mobile specific refresh token in order to keep login sessions active on the web when logging out of the mobile application.

**Scrum Backlog**

User Story ID	Tasks	Owner	Estimate (Hr)	Status
12	Create blank home page container	Logan Fite	.5	<b>Complete</b>
12	Create project status component that dynamically returns the following button groups depending on user's project status: <ul style="list-style-type: none"> <li>- If user not in project: Show "Find a Project" &amp; "Create a Project" buttons</li> <li>- If user is a member of a project: Show "Go to Project" button</li> <li>- If user is a project leader: Show "Find Members", "Manage Candidates", and "Go to Project" button</li> </ul>	Logan Fite	3	<b>Complete</b>
12	Add invites list component to homepage card	Logan Fite	3	<b>Complete</b>
20	Create home page screen for mobile application mirroring the web design	Logan Fite	7	<b>Complete</b>
16	Make web page responsive for varying page sizes	Jonathan Myers	1	<b>Complete</b>

**User Training**

Section 2 of User Manual for Web  
Sections 5.2-5.4 for Mobile

**7.1.6 Create Project****Approach**

Upon creation of the project the details that are entered are handled with the ProjectController which posts the details to the database, storing the project's details and linking them to the project creator.

**Mobile Implementation Notes**

N/A - Component not available on mobile

**Scrum Backlog**

User Story ID	Tasks	Owner	Estimate (Hr)	Status
3	Create page with option to create a project along with UI for creating the project	Jonathan Myers	8	<b>Complete</b>
3	Create ProjectController for handling project Details and sending them to the database.	Jonathan Myers	1.5	<b>Complete</b>
3	Add frontend validation method to ensure inputs are valid (Project name not null, description > 50 char, deadline greater than present time)	Logan Fite	.5	<b>Complete</b>
3	Link Project Creator to the Project being created	Jonathan Myers	1.5	<b>Complete</b>
16	Make web page responsive for varying page sizes	Jonathan Myers	1	<b>Complete</b>

**User Training**

Section 3.1 of User Manual

**7.1.7 Find a Team****Approach**

Creating a ProjectController to render the individual projects that have been created for users to decide if they are interested or not, when a user chooses interested their profile is sent to the project creator for review. Either button renders a new project for the user to see.

**Mobile Implementation Notes**

No major user interface or functional differences between the web and mobile implementation.

**Scrum Backlog**

User Story ID	Tasks	Owner	Estimate (Hr)	Status
4	Create page with option to find a team along with UI for generating the posted projects and its details	Jonathan Myers	6	<b>Complete</b>
4	Create ProjectController method to fetch the posted projects	Jonathan Myers	2	<b>Complete</b>
4	Add buttons for sorting through posted projects	Jonathan Myers	2	<b>Complete</b>
5	Send project creator users that are interested in the project	Jonathan Myers	4	<b>Complete</b>
16	Make web page responsive for varying page sizes	Jonathan Myers	1	<b>Complete</b>
21	Create Find a Team page screen for mobile application mirroring the web design	Logan Fite	5	<b>Complete</b>

**User Training**

Section 2.1 of User Manual for Web

Sections 5.2.1 of User Manual for Mobile

**7.1.8 Find Members****Approach**

Component to display user objects in an organized fashion for project leaders to decide if they are interested or not in the user. When a leader chooses interested, a project invite is sent to the currently displayed user. Either button renders a new user for the leader to see until no more are available and an empty list message is displayed.

**Mobile Implementation Notes**

No major user interface or functional differences between the web and mobile implementation.

### Scrum Backlog

User Story ID	Tasks	Owner	Estimate (Hr)	Status
5	Create user display template	Logan Fite	3	<b>Complete</b>
5	Create matching controller method to find all users matching the criteria: <ul style="list-style-type: none"> <li>- Leader has not previously marked interested/uninterested on the user</li> <li>- User leader has not marked interested/uninterested on the project</li> <li>- User has not previously been removed from the project</li> <li>- There is at least one common tag between the user's skills and the project's "Looking for"</li> </ul>	Logan Fite	2	<b>Complete</b>
5	Create leader interested & uninterested matching controller methods to send invite or add to uninterested list	Logan Fite	2	<b>Complete</b>
16	Make web page responsive for varying page sizes	Logan Fite	1	<b>Complete</b>
22	Create page screen for mobile application mirroring the web design	Logan Fite	5	<b>Complete</b>

### User Training

Section 3.2 of User Manual for Web

Sections 5.4 of User Manual for Mobile

### 7.1.9 View Project Invites

#### Approach

Displays invite project titles in rows with a View button that populates a popup with project information and an Accept & Decline button. Additional logic is added to ensure that users who are a member or a leader of a project cannot accept an invite until they leave their current one. Accepting the invite will add the user to the project and declining the invite will delete the invitation along with adding the user to the project's uninterested list so that neither party will see each other in the matching component.

### Mobile Implementation Notes

Instead of displaying a list of invites on the homepage with a popup modal to view specific information about the invite, the mobile application separates these views into unique screens. The project invites screen is nested within the home page and accessible under the “View Invites” button. When the “View” button is pressed, the user is navigated to a new screen that is based on the project matching screen view that displays the project information and replaces the “Interested” and “Uninterested” buttons with “Accept” and “Decline”.

### Scrum Backlog

User Story ID	Tasks	Owner	Estimate (Hr)	Status
4	Create fetch invites controller method	Logan Fite	.25	<b>Complete</b>
4	Create displayable row list of invites	Logan Fite	.5	<b>Complete</b>
4	Create popup modal template with invite information and accept/decline action buttons and disable join button if user in project	Logan Fite	2.5	<b>Complete</b>
4	Connect open method of popup to socket server and add method to emit join signal if accepted	Logan Fite	2	<b>Complete</b>
4	Create and link action buttons to controller methods to join project or decline and add to uninterested list	Logan Fite	1	<b>Complete</b>
4	Create page screen for mobile application mirroring the web design	Logan Fite	5	<b>Complete</b>

### User Training

Section 2.2 of User Manual for Web

Sections 5.2.1 of User Manual for Mobile

### 7.1.10 Manage Candidates

#### Approach

Displays users who have marked an interest in a leader’s project in rows inside of a popup modal with a hyperlink to the candidate’s profile and an Accept & Decline action button. Accepting the candidate will invite the user to the project, and rejecting the candidate will remove the candidate along with adding the user to the project’s uninterested list so that neither party will see each other in the matching component.

### Mobile Implementation Notes

Instead of displaying a list of candidates inside of a popup modal to view specific information about the users, the mobile application separates these views into unique screens. The candidates screen is nested within the homepage and accessible under the “Manage Candidates” button. When the “View” button is pressed, the user is navigated to a new screen that is based on the project matching screen view that displays the project information and replaces the “Interested” and “Uninterested” buttons with “Invite” and “Reject”.

### Scrum Backlog

User Story ID	Tasks	Owner	Estimate (Hr)	Status
5	Create controller method to fetch project candidate profiles	Logan Fite	.5	<b>Complete</b>
5	Create display component to list all profiles in rows with accept/reject action buttons	Logan Fite	1	<b>Complete</b>
5	Create controller method to send invite if accepted or to add user to project’s uninterested list if rejected	Logan Fite	.5	<b>Complete</b>
5	Create page screen for mobile application mirroring the web design	Logan Fite	5	<b>Complete</b>

### User Training

Sections 3.2, 4.7.1.1 of User Manual for Web  
Section 5.4 of User Manual for Mobile

### 7.1.11 Project Dashboard

#### Revised Design

<b>Component</b>	Project Dashboard
<b>Description</b>	The dashboard component is the parent component containing all subsystems needed for project management.
<b>Objects Used</b>	User, Project, Event, File, Task, Message, Development Log
<b>Use Cases Supported</b>	To Do List, Share Files, Development Log, Team chat, Calendar, Manage Project, Manage team members

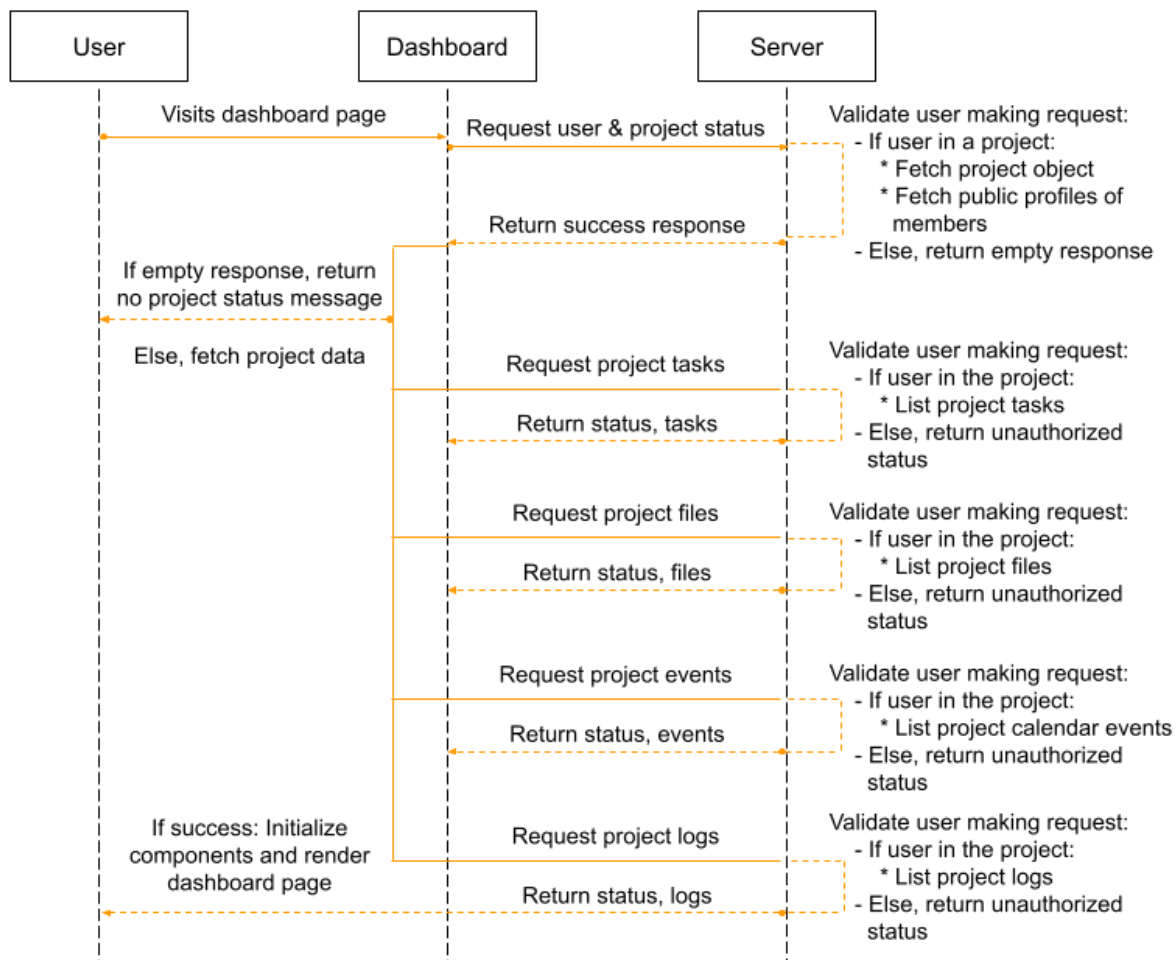


Figure 7.2 - Revised Project Dashboard Sequence Diagram

### Approach

On access to the dashboard, the system is responsible for loading the relevant project information for users and passing the project data to its subsystems. Instead of directly passing data to components as props, the dashboard initializes a global context state object usable by all sub-systems. With this approach, each system has access to and can modify the same data causing updates to other components for data consistency and simpler data management. The group chat messages are excluded from the context to support a generalized implementation structure that allows it to be reused for private messaging. The following is a system context diagram representing the connection between the dashboard subsystems:



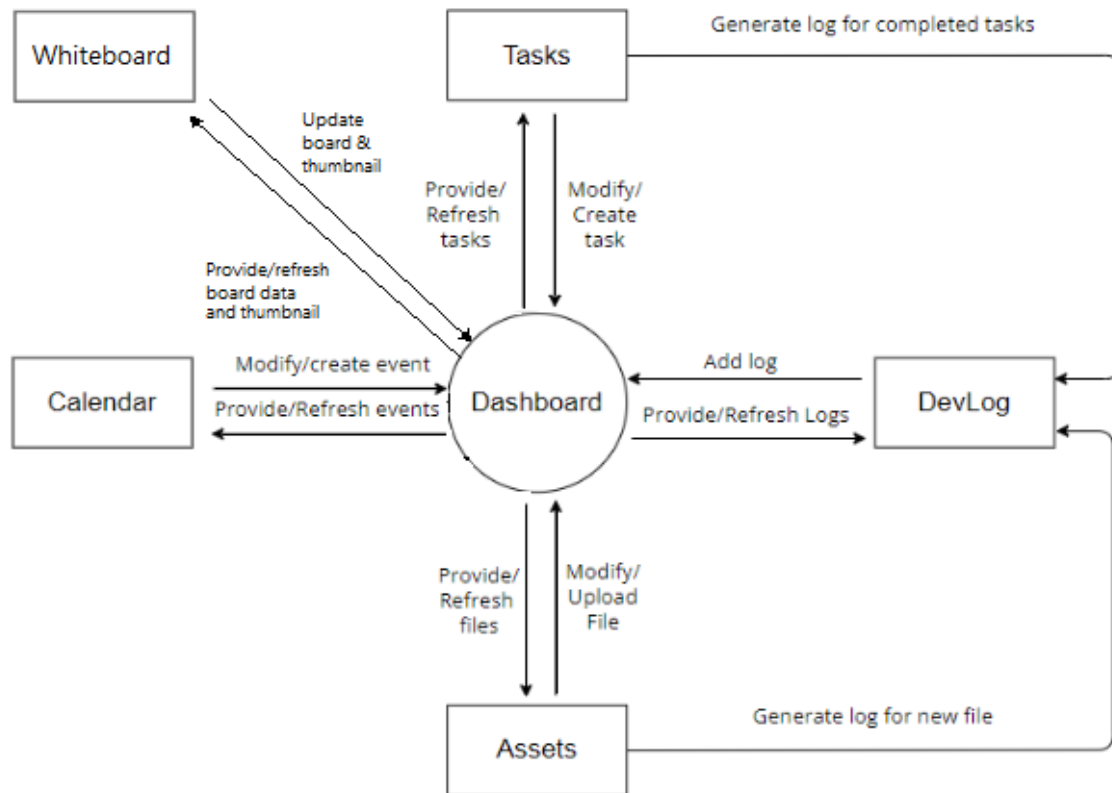


Figure 7.3 - Subsystem Communication Diagram for Project Dashboard

### Mobile Implementation Notes

The mobile version of the project dashboard has the largest visual change in comparison between any other pages between the web and mobile application. Instead of showing every component in the page at once, the native application separates each dashboard component into their own screens and uses a nested navigator to allow the member to toggle between each of the screens. The navigation header contains the project title and deadline, and the footer houses six icons representing the six components available within the dashboard that open their respective pages when pressed. The mobile application fetches and updates project component data in the same fashion as the web application.

**Scrum Task Breakdown**

User Story ID	Tasks	Owner	Estimate (Hr)	Status
6	Create parent dashboard page containing project information and group of individual dashboard components	Logan Fite	12	<b>Complete</b>
6	Create identifier method to display the current user's project if they are in one or an error message if they are not a member of a project with links to create a new project or visit the matching page	Logan Fite, Daniel Hill	3	<b>Complete</b>
6	Create project CRUD methods with protected permissions to retrieve/serve/edit the general project data and the tied objects of its subcomponents	Logan Fite	15	<b>Complete</b>
16	Make web page responsive for varying page sizes	Logan Fite	2	<b>Complete</b>

**User Training**

Section 4 of User Manual for Web

Section 5.3.1 of User Manual for Mobile

### 7.1.12 Project Dashboard - Calendar

#### Revised Design

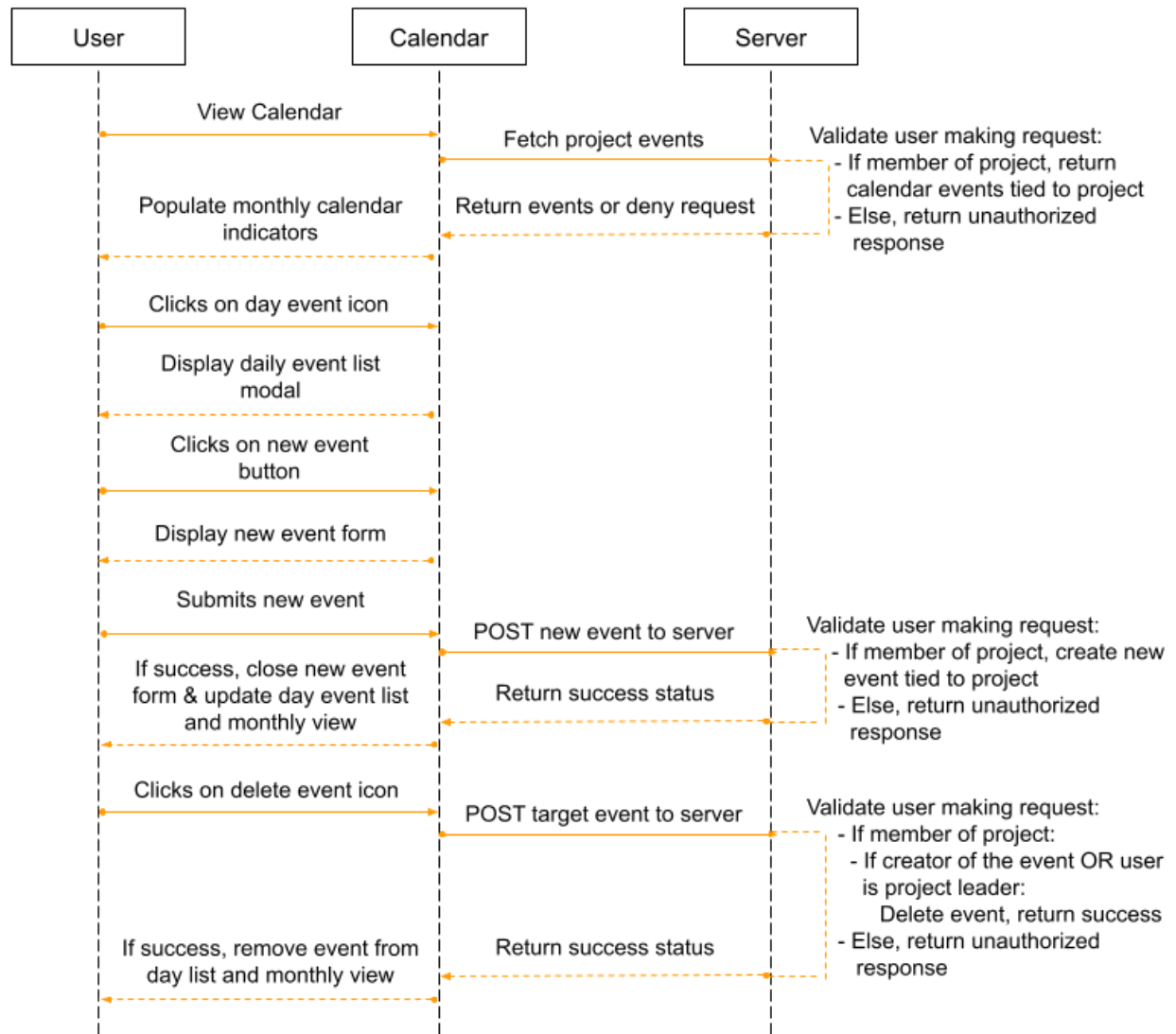


Figure 7.4 - Revised Calendar Sequence Diagram

#### Approach

The calendar component displays calendar Event objects for each project to team members. The events are displayed on a monthly overview calendar and within a popup modal list where they can be edited. To add an event, the user specifies the day of the event by clicking on the date and the name through a form input. The data is wrapped and sent to a backend API that ties the event to the project. Finally, upon all create, edit, and delete functions, an update message is sent to the socket server to update the items for other users currently viewing the dashboard.

### Mobile Implementation Notes

Because the component is read-only on mobile, users cannot select a date that does not already have a pink circle icon indicator. Pressing on date indicators opens a read-only modal of the event list for the date selected.

### Scrum Task Breakdown

User Story ID	Tasks	Owner	Estimate (Hr)	Status
8	Provide monthly calendar overview with clickable dates	Logan Fite	10	<b>Complete</b>
8	Create popup modal for chosen date, populate current events	Logan Fite	6	<b>Complete</b>
8	Provide insert form for new event, create object upon submittal & refresh	Logan Fite	5	<b>Complete</b>
8	Add inline edit and delete buttons to modify the event	Logan Fite	5	<b>Complete</b>
8	Add CRUD operation permissions only to users within the project with appropriate socket signals emitted upon completion	Logan Fite	2	<b>Complete</b>
8	Disable edit/delete buttons for users who did not create the event and are not the project leader	Daniel Hill	1	<b>Complete</b>
16	Make web page responsive for varying page sizes	Logan Fite	1	<b>Complete</b>
25	Create mobile component to import to project screen in mobile application that also connects to socket server	Logan Fite	6	<b>Complete</b>

### User Training:

Section 4.2 of User Manual for Web

Section 5.3.1 of User Manual for Mobile

### 7.1.13 Project Dashboard - To Do List

#### Revised Design

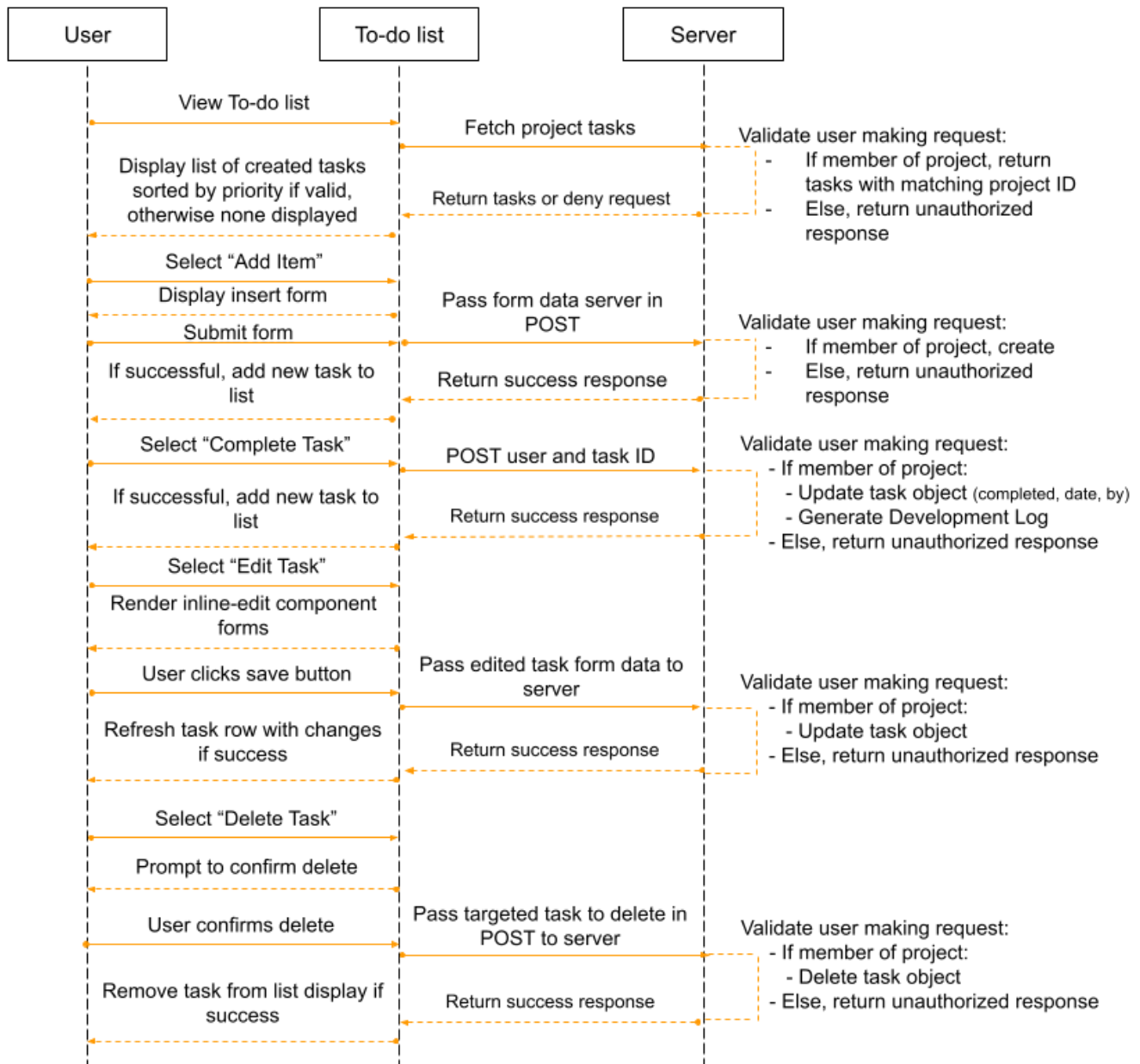


Figure 7.5 - Revised To-Do List Sequence Diagram

### Approach

The to-do list module displays task items not yet completed to users in a sorted-by-priority group of cards for an easy indicator of project status. A popup modal is available with detailed task list rows with inline buttons to modify the task object, delete a task, complete the task, and a create form to insert new task objects into the database. Upon all create, edit, complete, and delete functions, an update message is sent to the socket server to update the item for other users currently viewing the dashboard.

### Mobile Implementation Notes

Because the component is read-only on mobile, the event manager modal is not accessible on mobile. Instead, the user is provided the ability to expand the simple card items to see additional information about the tasks such as description. As a result, only pending tasks are visible on mobile.

### Scrum Task Breakdown

User Story ID	Tasks	Owner	Estimate (Hr)	Status
7	Create task cards with icon indicator and assignment information	Logan Fite	4	<b>Complete</b>
7	Create parent container which sorts and displays card on dashboard	Logan Fite	6	<b>Complete</b>
7	Provide insert form for new event, create object upon submittal & refresh	Logan Fite	5	<b>Complete</b>
7	Add front-end input validation method and messages to ensure tasks have a title	Logan Fite	1	<b>Complete</b>
7	Add inline edit and delete buttons to modify the task and CRUD operations to the task controller with appropriate socket signals emitted upon completion	Logan Fite	5	<b>Complete</b>
7	Add a separate tab for tasks marked completed	Logan Fite	2	<b>Complete</b>
7	Add Dev Log creation when task marked completed	Logan Fite	.25	<b>Complete</b>
7	Disable edit & delete buttons on tasks	Daniel Hill	1	<b>Complete</b>

	that are not assigned to user unless they are the project leader or created the task			
16	Make web page responsive for varying page sizes	Logan Fite	1	<b>Complete</b>
24	Create mobile component to import to project screen in mobile application that also connects to socket server	Logan Fite	5	<b>Complete</b>

**User Training:**

Section 4.1 of User Manual for Web

Section 5.3.1 of User Manual for Mobile

**7.1.14 Project Dashboard - Assets/File Sharing****Revised Design**

Diagram on next page

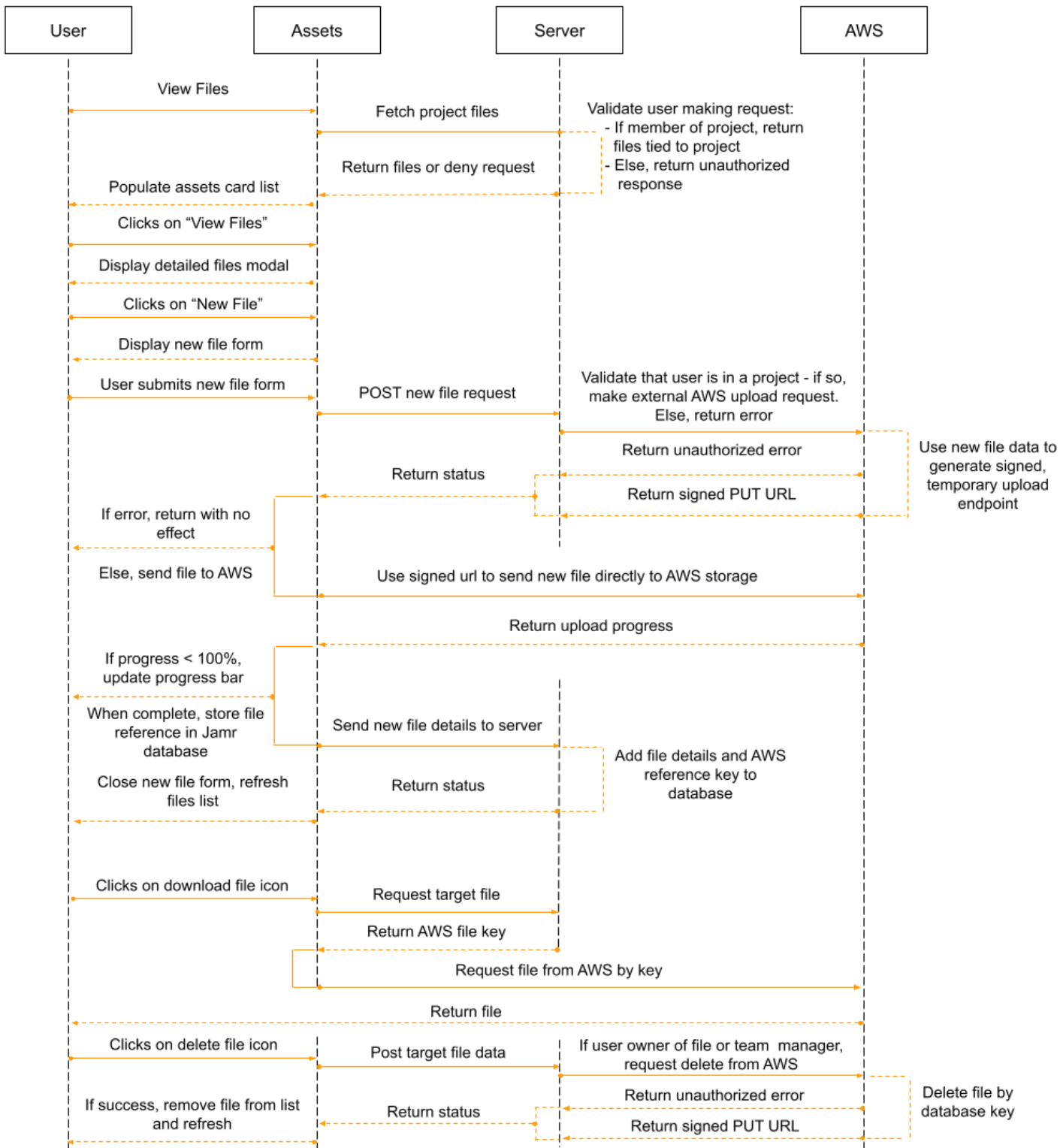


Figure 7.6 - Revised Asset Sequence Diagram



### Approach

The file sharing component allows members to upload and download files to a shared assets folder for all team members. Jamr's role is to generate and provide secure access links to the external cloud hosting for the user to handle requests from their own browser, instead of through the Jamr backend. This near-serverless approach removes any potential performance impact for both Jamr and the client. When the user indicates a file upload request to the server, a secure, signed endpoint lasting 60 seconds is generated through Amazon's AWS platform. The user sends a PUT request of the file information to the URL provided by the server upon success. Successful upload to the cloud hosting service creates an entry in the files table within Jamr to reference external files. Downloads are handled in the same manner: Users request to download a file, the server will find the appropriate file entry within Jamr's Simple Storage Service (S3) account, and return a link that the application will automatically submit a GET request to download from the user's browser. Finally, upon all create, edit, and delete functions, an update message is sent to the socket server to update the item for other users currently viewing the dashboard.

### Mobile Implementation Notes

Because the component is read-only on mobile, the asset manager modal is not accessible on mobile. Instead, the user is provided the ability to expand the simple card items to see additional information about the files such as who uploaded the file and the time uploaded.

### Scrum Task Breakdown

User Story ID	Tasks	Owner	Estimate (Hr)	Status
9	Connect application to AWS S3	Logan Fite	5	<b>Complete</b>
9	Create upload module with file input	Logan Fite	3	<b>Complete</b>
9	Add front-end input validation method to require files < 25mb, filename < 50 characters, and file not null	Logan Fite	1.5	<b>Complete</b>
9	Create backend method for generating presigned URL for client to post file to S3 bucket	Logan Fite	3	<b>Complete</b>
9	Store reference to uploaded file in Jamr database	Logan Fite	1	<b>Complete</b>
9	Create download/delete methods for files	Logan Fite	4	<b>Complete</b>

	with appropriate socket signals emitted upon completion			
9	Hide delete file button if file not uploaded by user or user not project leader	Daniel Hill	1	<b>Complete</b>
16	Make web page responsive for varying page sizes	Logan Fite	.5	<b>Complete</b>
26	Create mobile component to import to project screen in mobile application that also connects to socket server	Logan Fite	5	<b>Complete</b>

### User Training:

Section 4.5 of User Manual for Web

Section 5.3.1 of User Manual for Mobile

## 7.1.15 Project Dashboard - Development Log

### Revised Design

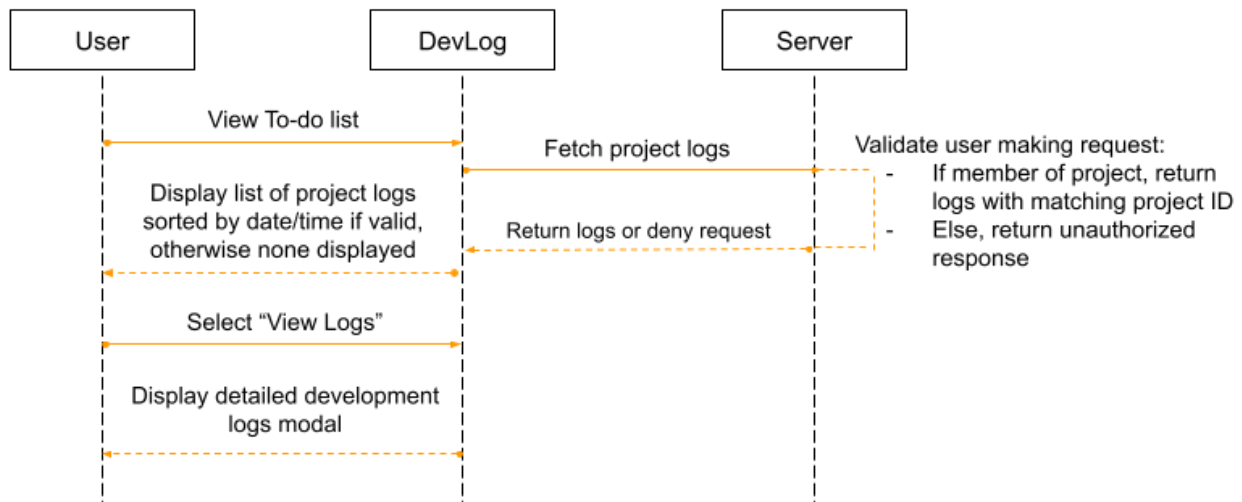


Figure 7.7 - Revised Development Log Sequence Diagram

### Revision

Upon the decision to postpone the collaborative whiteboard to the second semester, development logs became a standalone tool by separating from the file uploading functionality, which is now another standalone component named assets.

### Approach

The development log is a read-only list of project highlights, such as adding a new member, creating a new task, marking tasks completed, and uploading files. It is primarily used as a measurement tool to determine the group's overall progress and these objects are created automatically through the events mentioned above. Upon creating a log, an update message is sent to the socket server to update the item for other users currently viewing the dashboard.

### Mobile Implementation Notes

The development logs modal is not accessible on mobile. Instead, the user is provided the ability to expand the simple card items to see additional information about the files such as who it was generated by and the time generated.

### Scrum Task Breakdown

User Story ID	Tasks	Owner	Estimate (Hr)	Status
11	Create development log model	Logan Fite	.5	<b>Complete</b>
11	Create development log display cards	Logan Fite	1	<b>Complete</b>
11	Create detailed information modal for development log details	Logan Fite	2	<b>Complete</b>
11	Extend asset/task completion methods to generate development log with appropriate socket signals emitted upon completion	Logan Fite	1	<b>Complete</b>
16	Make web page responsive for varying page sizes	Logan Fite	.5	<b>Complete</b>
28	Create mobile component to import to project screen in mobile application that also connects to socket server	Logan Fite	5	<b>Complete</b>

### User Training:

Section 4.3 of User Manual for Web

Section 5.3.1 of User Manual for Mobile

### 7.1.16 Project Dashboard - Whiteboard

#### Revised Design

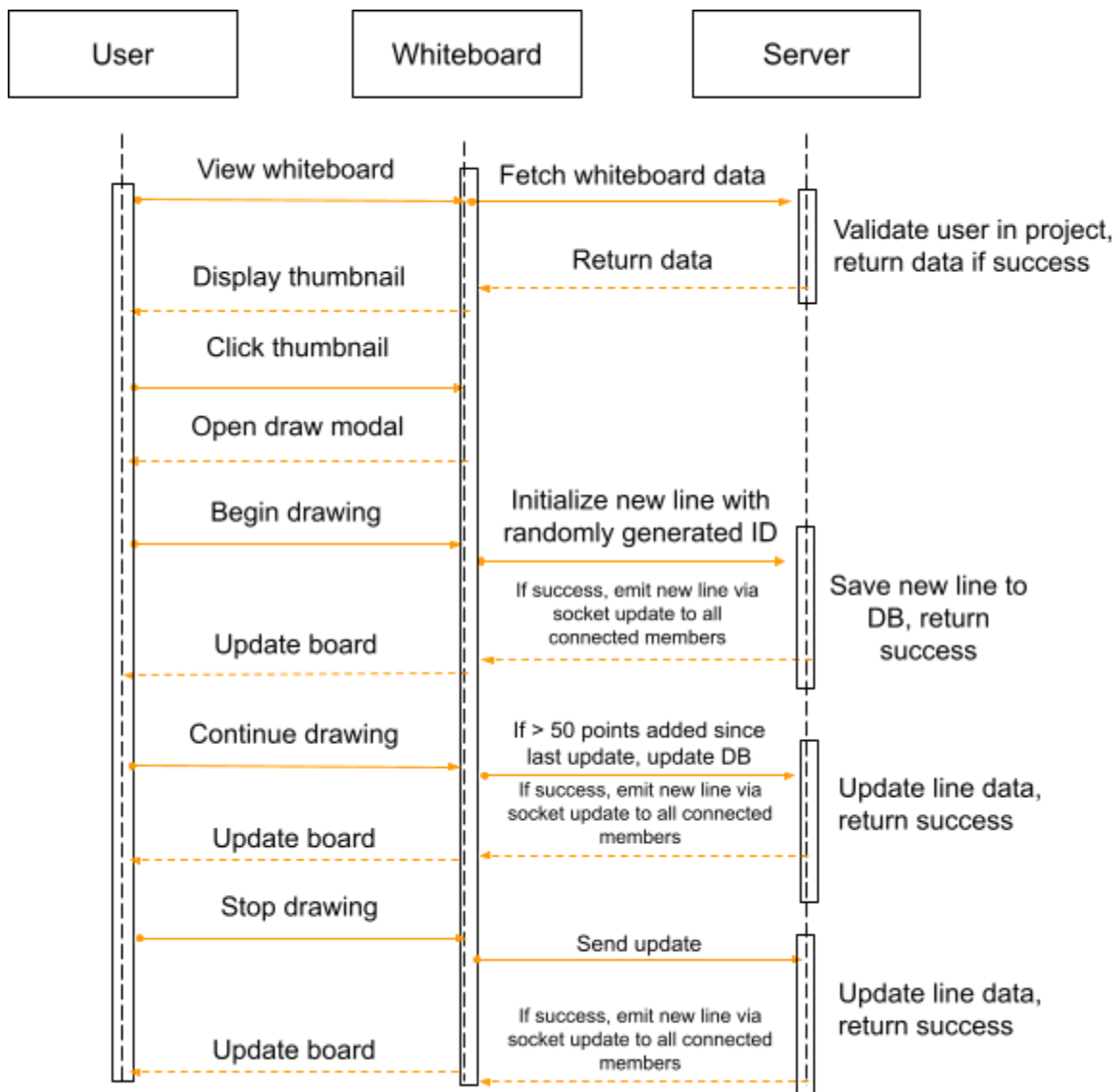


Figure 7.8 - Revised Whiteboard Sequence Diagram

#### Revision

For better optimization, the whiteboard component updates its database entry at a limited rate of every 50 coordinates added or upon completing the drawing (mouse release). An initial entry is required on the user's first mouse click to ensure that all connected users receive the initialized line and can successfully update changes to the line.

### Approach

The project whiteboard follows a tutorial from [https://konvajs.org/docs/react/Free\\_Drawing.html](https://konvajs.org/docs/react/Free_Drawing.html) to utilize drawing capabilities on a Konva stage. Additional React states controlled by a color input picker and value slider are added to control line color and width accordingly.

To support drawing from multiple users simultaneously through a socket connection, individual line data is stored in a complex JSON object as an object identified by a randomly generated key rather than a primitive array of lines in the frontend component. This structure allows the line objects to be accessed and updated directly (initialized on mouse down, point data updated during drawing) without needing to implement a search function to identify the correct element to update while another user draws or to manage an array of objects during runtime. In the backend, these line objects are stored within the database as an array of objects for a simple fetch call - it is the responsibility of the frontend to manage them efficiently.

Finally, upon mouse release, a thumbnail object is generated by converting the drawing data to a base64 image to better display the current status of the whiteboard to users who are not connected to the drawing modal or are viewing on the mobile application.

### Mobile Implementation Notes

The whiteboard drawing modal is not accessible on mobile. Instead, the thumbnail is displayed and receives updates when a thumbnail update action occurs on the web (upon mouse release of drawing a line and on board clear).

### Scrum Task Breakdown

User Story ID	Tasks	Owner	Estimate (Hr)	Status
15	Create whiteboard thumbnail card	Logan Fite	.25	<b>Complete</b>
15	Create whiteboard modal with drawing capabilities and custom mouse pointer	Logan Fite	15	<b>Complete</b>
15	Create toolbar with adjustable line color, tool, and width	Logan Fite	5	<b>Complete</b>
15	Connect draw functions to socket server to enable simultaneous drawing	Logan Fite	10	<b>Complete</b>
29	Create mobile component to import to project screen in mobile application that	Logan Fite	5	<b>Complete</b>

	also connects to socket server			
--	--------------------------------	--	--	--

**User Training:**

Section 4.4 of User Manual for Web

Section 5.3.1 of User Manual for Mobile

**7.1.17 Project Dashboard - Group Chat****Approach**

The group chat component initializes a socket connection and emits a request to join the project room in the socket server. If successful, the server responds with a list of message objects with the messageType “Group” and recipient of the Project ID. Upon receiving this data, the component sorts and formats the messages so that messages sent by the user are right-aligned with a purple background while messages from other members are left-aligned with a white background.

**Mobile Implementation Notes**

The group chat is fully functional on the mobile application and functions similar to the web application.

**Scrum Backlog**

User Story ID	Tasks	Owner	Estimate (Hr)	Status
10	Create reverse-order chat container, text input, and submit button for sending messages	Logan Fite	5	<b>Completed</b>
10	Connect component to socket and fetch all project message and populate chat container in order sent	Logan Fite	5	<b>Completed</b>
10	Validate text input on submission, save to database, and emit signal if successful for recipients to receive update	Logan Fite	1	<b>Completed</b>
16	Make web page responsive for varying page sizes	Logan Fite	.5	<b>Complete</b>
27	Create similar components in mobile	Logan Fite	5	<b>Completed</b>

	application with same functionalities inside of the Messaging screen			
--	--	--	--	--

**User Training:**

Section 4.6 of User Manual for Web

Section 5.3.1 of User Manual for Mobile

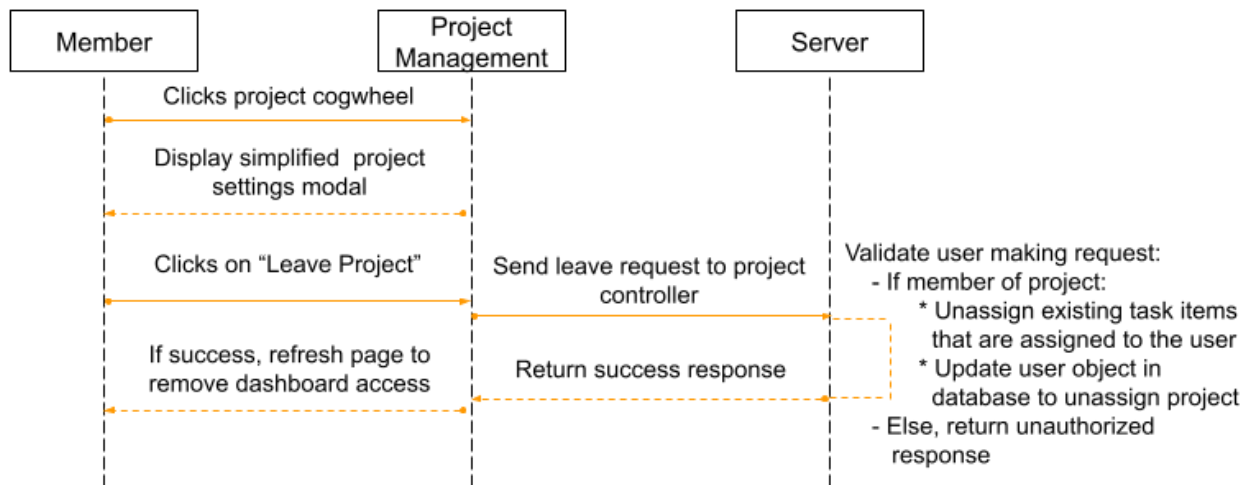
**7.1.18 Project Dashboard - Project Management****Revised Design**

Figure 7.9 - Revised Project Management Sequence Diagram (Member)

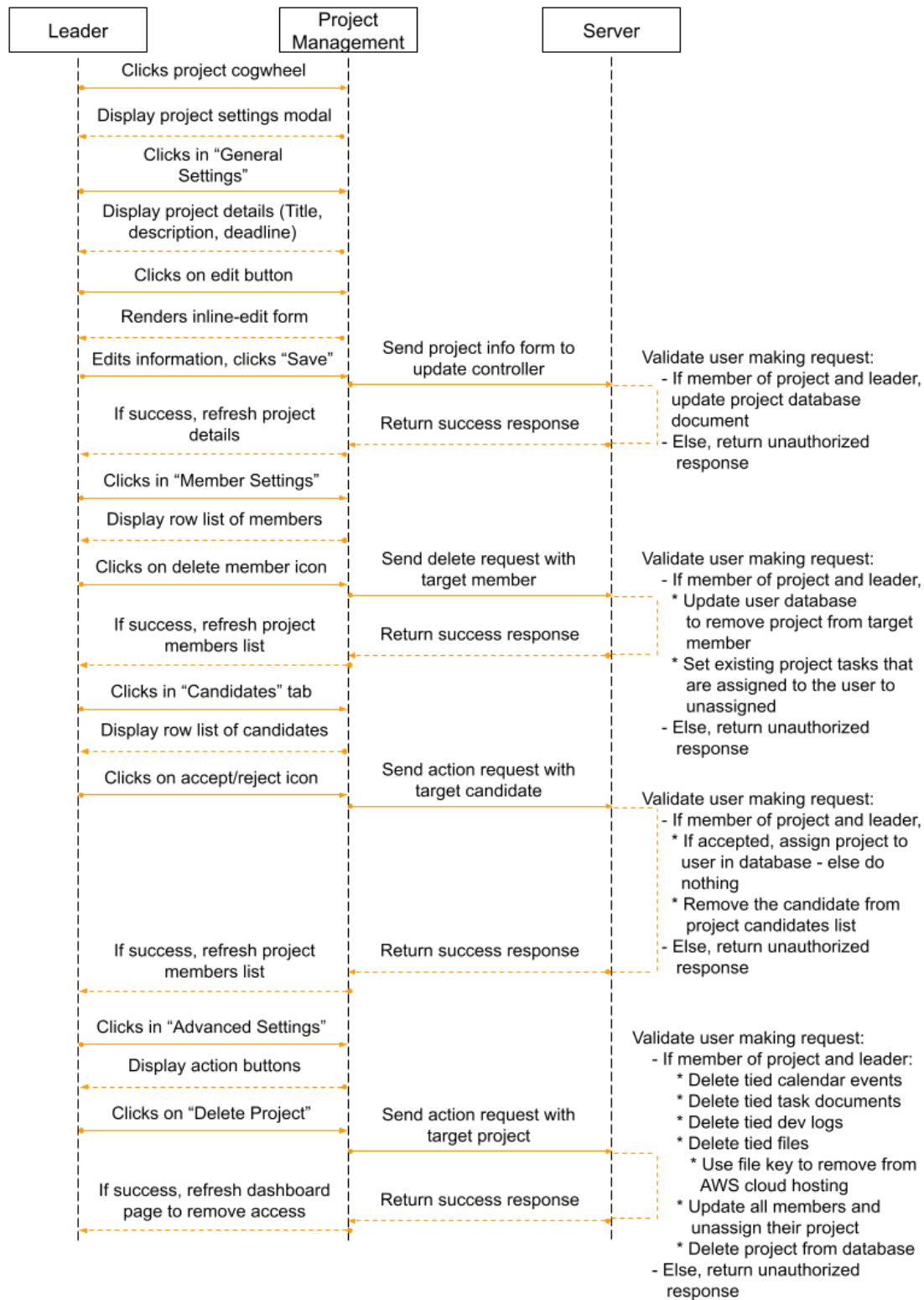


Figure 7.10 - Revised Project Management Sequence Diagram (Leader)



### Revision

Users have been given a simplified version of the project management modal so that they are able to leave their current project.

### Approach

The project management module is a permissions-based popup modal accessible by both the project leader and members. Members have access to a simplified modal that allows them to leave the project, while leaders can access a detailed settings menu with 4 tabs dedicated to modifying different aspects of the project. Leaders may update the project database entry by editing information forms such as project Title, Description, and Deadline. Leaders also have control over member management by removing existing users or inviting candidates who appear from the matching functionality. The leader may also delete the project, which removes all project information from Jamr & its associated files on AWS cloud storage and unassigned all members from the project.

### Mobile Implementation Notes

Project settings are not available on the mobile application.

### Scrum Task Breakdown

User Story ID	Tasks	Owner	Estimate (Hr)	Status
5	Create manager actions modal for editing project details	Logan Fite, Jonathan Myers	6	<b>Complete</b>
5	Create project details tab with inline editing module and update controller method and connect to socket server to emit any successful changes	Logan Fite	4	<b>Complete</b>
5	Create members page with populated rows and removal button and corresponding controller method and connect to socket server to emit any successful changes	Logan Fite	3	<b>Complete</b>
5	Create candidates tab with populated rows and accept/reject action buttons and corresponding controller methods	Jonathan Myers	4	<b>Complete</b>

5	Create advanced settings tab with delete project button and corresponding controller method and connect to socket server to emit any successful changes	Logan Fite	3	<b>Complete</b>
5	Create simplified members action modal with leave project button and corresponding controller method and connect to socket server to emit any successful changes	Logan Fite	2	<b>Complete</b>

### User Training:

Section 4.7 of User Manual

### 7.1.19 User Settings

#### Approach

The user settings page allows the user to modify their light mode/dark mode accessibility setting and delete their Jamr account. Theme settings are stored in the user's browser and accessed by a context function that checks the value of the setting. Themes are supported in every component by conditionally applying light or dark styling classes to components. Deleting an account deletes the user object and any references to the user within a project.

### Mobile Implementation Notes

User settings are not available on the mobile application.

### Scrum Task Breakdown

User Story ID	Tasks	Owner	Estimate (Hr)	Status
1	Create "Delete Profile" button, warning message and confirmation modal	Logan Fite	.25	<b>Complete</b>
1	Create profile deletion method to remove User object from Jamr, update project dashboard elements, and deleting project and project items cascade if user owns a project	Logan Fite	3	<b>Complete</b>
17	Create light mode/dark mode switch that stores the preference in user browser	Logan Fite	3	<b>Complete</b>

17	Add dynamic class name styles and update all components to check and assign the appropriate class name to page elements	Logan Fite, Jonathan Myers	15	<b>Complete</b>
----	---	----------------------------	----	-----------------

### User Training:

Section 7 of User Manual

### 7.1.20 User Profile

#### Approach

The user that is displayed on the profile page is dependent upon the username entered in the page URL. The application pulls the username following the last slash in the pattern jamrdemo.herokuapp.com/profile/**username** and fetches the profile data corresponding to the username. If a profile page is accessed that does not have a user, a message will be displayed. Upon visiting from the navigation bar, the default information displayed will be based on what the user enters during registration. If the user visits the profile of another user, they may read their profile information and click any of the following action buttons: Add Friend, Remove Friend. If the user is visiting their own profile, the user can decide to edit the information provided or add additional elements such as itch.io item showcases.

### Mobile Implementation Notes

Viewing profiles is not supported on the mobile application.

### Scrum Backlog

User Story ID	Tasks	Owner	Estimate (Hr)	Status
13	Create dynamically accessed profile page with URL argument	Logan Fite, Jonathan Myers	6	<b>Complete</b>
13	Create User Controller for displaying user information and formatting display of profile information with dynamically displayed action buttons	Logan Fite	15	<b>Complete</b>
13	Create edit function for changing the profile details with input validation	Logan Fite	5	<b>Complete</b>

13	Create itch.io showcase modal and add forms with validated inputs	Logan Fite	8	<b>Complete</b>
16	Make web page responsive for varying page sizes	Logan Fite	1	<b>Complete</b>

**User Training:**

Section 2.4 of User Manual

**7.1.21 Friends****Approach**

The Friends page displays the current users that are tagged as ‘friends’. The option to remove or add friends will be provided using a UserController to search users as well as display them.

**Mobile Implementation Notes**

Adding or managing friends is not supported on the mobile application.

**Scrum Backlog**

User Story ID	Tasks	Owner	Estimate (Hr)	Status
14	Create Friends page with option to add or remove ‘friends’ with UI for displaying ‘friends’	Logan Fite	8	<b>Complete</b>
14	Create FriendController method for fetching users ‘friends list’ and a component function for displaying them	Logan Fite	3	<b>Complete</b>
14	Create FriendController method for searching users by username and displaying results in a table with conditionally rendered action buttons to send or accept/deny requests	Logan Fite	5	<b>Complete</b>
14	Create FriendController for adding and removing ‘friends’	Logan Fite	3	<b>Complete</b>
14	Create action modal to view friend profile or remove friend	Logan Fite	1	<b>Complete</b>

**User Training:**

Section 2.3 of User Manual

**7.1.22 Messaging****Approach**

The messaging page requires that the user has at least one friend added. If that condition is met, it initializes a socket connection with a default contact (most recently messaged or added) and requests existing messages. If successful, the server responds with a list of message objects where the messageType is “Private”, the recipient is either the user or contact, and the sentBy \_id is either the user or recipient . Upon receiving this data, the component sorts and formats the messages so that messages sent by the user are right-aligned with a purple background while messages from other members are left-aligned with a white background.

**Mobile Implementation Notes**

The private messaging screen is fully functional on the mobile application. Instead of displaying the chat box and a contacts list side by side, the native application houses the contacts list and switcher inside of a fullscreen component that is accessible by pressing on a hamburger menu icon in the top right of the messaging screen. Users may change contacts to manage multiple messaging sessions.

**Scrum Backlog**

User Story ID	Tasks	Owner	Estimate (Hr)	Status
14	Create a reverse-order chat container, text input, submit button, and contacts list for messaging template. Default selected contact to most recently messaged or display no contacts info if none added.	Logan Fite	10	<b>Completed</b>
14	Connect messaging page to socket and fetch all messages between user and recipient and populate chat container in order sent	Logan Fite	5	<b>Completed</b>
14	Validate text input on submission (not null), save to database, and emit signal if successful for recipient to receive update	Logan Fite	1	<b>Completed</b>
14	Add contact switching by clicking on an	Logan Fite	2	<b>Completed</b>

	existing contact, reload messages			
19	Create similar components in mobile application with same functionalities inside of the Messaging screen	Logan Fite	7	<b>Completed</b>
16	Make web page responsive for varying page sizes	Jonathan Myers	5	<b>Complete</b>

**User Training:**

Section 2.3 of User Manual

**7.1.23 Web Navigation Bar****Approach**

The navigation bar is a list of icons and titles that are rendered on every page of the application for ease of navigation throughout the web application. Clicking on any of the links will redirect the user to the target page. Additional functionality is included in the Logout button which revokes web access and refresh tokens from the user's storage and removes their entries in the database User object for additional security.

**Scrum Backlog**

User Story ID	Tasks	Owner	Estimate (Hr)	Status
N/A	Create navigation bar with corresponding links	Logan Fite, Jonathan Myers	3	<b>Complete</b>
N/A	Render the navigation bar on every page	Daniel Hill	.5	<b>Complete</b>
1	Add functionality to logout button	Jonathan Myers	.5	<b>Complete</b>
16	Create condensed navigation bar for smaller resolution screens, completely hiding it on mobile resolutions with a toggle button	Logan Fite	2.5	<b>Complete</b>

### 7.1.24 Mobile Navigation Bar

#### Approach

Unlike the web application, the mobile application uses navigation wrappers defining a list of reachable screens rather than pages. By nesting and conditionally rendering navigation wrappers, the mobile application controls which screens are accessible to the user. Additional functionality is included in the Logout button which revokes mobile access and refresh tokens from the user's storage and removes their entries in the database User object for additional security. Below is the nesting structure of navigation screens:

<pre> &lt;App&gt;   Logged In?     &lt;Tab Navigator&gt;       &lt;Home Screen&gt;         &lt;Home Navigator&gt;           Home/Matching/Invites List/ Invite/Candidate         &lt;/Home Navigator&gt;       &lt;Home Screen&gt;       &lt;Messaging Screen /&gt;       &lt;Dashboard Screen&gt;         &lt;Dashboard Navigator&gt;           To Do/Calendar/Logs/Assets/Whiteboard/Chat         &lt;Dashboard Navigator&gt;       &lt;Dashboard Screen&gt;     &lt;Tab Navigator&gt;   Not Logged In?     &lt;Stack Navigator&gt;       Landing/Login/Error     &lt;Stack Navigator&gt; &lt;/App&gt; </pre>	<p><b>Tab Navigator:</b> Provides the Jamr logo header with logout capability and bottom tabs to navigate to Messaging, Home, and Dashboard.</p> <p><b>Home Navigator:</b> Used to navigate into home subpages such as matching and viewing invites or candidates. No footer, but a header with a back button to reach the root Home page</p> <p><b>Dashboard Navigator:</b> Used to navigate between dashboard components of a project. Header is the project title and time left, footer is a list of component icons to switch between all 6 components.</p>
---	---

#### Scrum Backlog

User Story ID	Tasks	Owner	Estimate (Hr)	Status
18	Create navigator group for user not logged in (Landing, Login, Details, Invalid)	Logan Fite	5	<b>Complete</b>
19, 20, 23	Create main application navigator group	Logan Fite	10	<b>Complete</b>
23	Create dashboard navigator group	Logan Fite	5	<b>Complete</b>

2	Add methods to fetch and validate User tokens, conditionally rendering the appropriate navigator group depending upon status of the token	Logan Fite	8	<b>Complete</b>
---	---	------------	---	-----------------

## 7.2 Technologies

Below are short descriptions and code samples illustrating how the specified required technologies were used for implementation of the project. As a result of using the MERN stack, the primary technologies used by this project are JavaScript based.

### 7.2.1 MongoDB

Application objects are defined by schema models within the server application, which are used during data creation/retrieval to specify the storage format in database collections.

```
const projectSchema = new Schema({
  projectID: {
    type: Number
  },
  name: {
    type: String
  },
  description: {
    type: String
  },
  deadline: {
    type: Date
  },
  candidates: {
    type: Array,
    default: []
  },
  uninterested: {
    type: Array,
    default: []
  },
  tags: {
    type: Array,
    default: []
  }
});
projectSchema.plugin(AutoIncrement,
  {inc_field: 'projectID'});
```

```
{
  _id: ObjectId("6255cb3b5d9533f8353549c5"),
  name: "Example Project",
  description: "This is a project for demonstration purposes to test the features avai...",
  deadline: 2022-05-15T17:00:14.000+00:00,
  > candidates: Array
  > uninterested: Array
  > tags: Array
  projectID: 13
  __v: 0
}
```

Figure 7.11 - Mongo Schema & Collection Object

### 7.2.2 Express JS

Express provides support for passing data from the frontend application to the backend server and vice-versa, along with user routing through endpoint controllers. These routes specify backend methods to perform upon receiving a request.



```
const router = express.Router();
// User API Routes
router.post('/user/login', UserController.login);
router.post('/user/setup', UserController.getRegistrationUser);
router.post('/user/register', UserController.registerUser);
router.post('/user/logout', middlewareTokenAuth, UserController.logout);
router.post('/user/deleteProfile', middlewareTokenAuth, UserController.deleteProfile);
router.post('/user/leaveProject', middlewareTokenAuth, UserController.leaveProject);
```

Figure 7.12 - Express Routes

### 7.2.3 React JS

React allows frontend user interfaces with strong support for dynamic interactions to be built. Some features of react essential to the frontend user pages are conditional rendering and hooks, which allow for efficient and complex data handling.

```
const [user, setUser] = useState({});
useEffect(() => {
  (async () => {
    try {
      const allowed = await ValidToken(GetAccessToken(), GetRefreshToken());
      if (allowed == false) history.push('/');
      const response = await axios.get('/user/getUserProfile', {headers: {'authorization': 'Bearer ' + GetAccessToken().toString()}});
      console.log(response);
      if (response.data) {
        setUser(response.data.profile.profile);
      }
    } catch (error) {
      console.log(error);
    }
  })();
}, []);
return (
  <div className='find-Member'>
    <Container>
      <Col className='prof-avatar' xs={6}><img id='prof-Avatar' src={user.avatar} height= '125rem' width= '125rem'></img></Col>
      <Col className='prof-username' xs={6}><h1>{user.username}</h1><h2>{user.biography}</h2></Col>
      <Col className='prof-tags' xs={6}></Col>
    </Container>
  </div>
);
```

Figure 7.13 - React Page for Displaying User Profile

### 7.2.4 React Native

React Native components allow the same flexibility as React components to be used inside of mobile applications. React Native further improves development efforts by supporting a “Write once, run anywhere” build structure that allows the application to build for both Android & iOS from the same source code. As a result, most of the implementation of the native application was spent translating the existing web code rather than completely rewriting it, apart from the implementation decisions mentioned in the previous sections.

```
return (
  <View style={HomeStyle.container}>
    <Text style={HomeStyle.title}>Welcome to Jamr</Text>
    <Text style={HomeStyle.description}>This companion app requires an existing Jamr account.</Text>
    <Text style={HomeStyle.description}>Register an account online at jamrdemo.herokuapp.com</Text>
    <TouchableOpacity
      style={HomeStyle.button}
      onPress={() => {navigation.navigate("Login");}}
    >
      <Text style={{fontSize:20,fontWeight:'bold',color:'white'}}>LOG IN</Text>
    </TouchableOpacity>
  </View>
);
```

Figure 7.14 - Example of a React Native Render Method

### 7.2.5 Node JS

Node JS is the javascript environment that performs the server-side code. In the context of Jamr, Node is used to initialize the socket server for real time collaboration and communication, along with providing a REST API for responding to requests. The backend methods are essential to executing all requests that the user makes through the frontend via controllers.

```
async getEvents(req, res) {
  try {
    // Get project and user making request
    const { pid } = req.params;
    const user = req.user;
    // Before returning events, check that user is allowed to list them
    const projectCheck = await UtilityController.userInProject(pid,user);
    if (!projectCheck.inProject) return res.sendStatus(403);
    // If user is allowed, return list of events for the project
    const events = await Event.find({projectID: pid});
    return res.status(200).send(events);
  }
  catch (error) {
    res.status(500).json({ error: error.toString() }).send();
  }
},
```

Figure 7.15 - Server Code to Get Project Calendar Events

### 7.2.6 Axios

The web and mobile applications rely upon Axios to send server requests from the front-end interface. GET and POST requests are defined in the action methods tied to buttons and their responses are used to update page information.

```
const pid = props.project.projectID;
const getCandidates = await axios.get(`/project/candidateProfile/${pid}`,
  {headers: {'authorization': 'Bearer ' + GetAccessToken().toString()}});
if (getCandidates.status === 200) {
  setCandidates(getCandidates.data.candidateProfiles);
}
```

Figure 7.16 - Axios GET request to fetch candidate profiles for a project

```
const response = await axios.post(`/project/addCandidate/${pid}`,
  {data: candidate},
  {headers: {'authorization': 'Bearer ' + GetAccessToken().toString()}});
if (response.status === 200) {
  await contextObj.refreshProject.refreshProject();
  await props.refreshCandidates();
}
```

Figure 7.17 - Axios POST request to add a new candidate to a project

## **8. Testing**

### **8.1 Testing Plan**

#### **8.1.1 Accessibility Testing**

Accessibility is provided by ensuring that the application is both responsive and usable across multiple web browsers and screen dimensions. These requirements will be tested manually by accessing the application on Microsoft Edge, Google Chrome, Firefox, and Safari. The built-in device toolbar included in Firefox and Google Chrome allows the application to be viewed in desktop, laptop, and mobile preset dimensions along with an adjustable resizer to confirm that the app consistently scales.

Further accessibility features are added to the web application by a light mode/dark mode theme switch. Manual testing will be done to ensure that all application pages and components are responsive to the theme chosen by the user for better visibility.

#### **8.1.2 User Acceptance Testing**

The primary goal of the application to achieve efficient collaboration between users is best tested by an internal group of end users. External feedback covering the application from registration to project creation including any bugs encountered during the process will help spot errors not previously accounted for, and an increased user base will help determine the efficiency and correctness of the real time components hosted within the application. Testers will also be given access to the native app version of Jamr to ensure the companion app functions as expected. Invites to the application will be controlled to local colleagues and bug reports will be collected through a Google form.

#### **8.1.3 Unit Testing**

Unit testing is the most important tool for ensuring the application components are working correctly. Each component that is responsible for achieving a use case requirement will be tested with valid and invalid actions wherever possible to determine robustness. Following this page are the test cases for the components responsible for the use case requirements specified earlier.

## 8.2 Test Case Results

System Test Case:

USER STORY: Itch.io OAuth Login/Registration

<b>Purpose:</b> Ensure that login behavior works as intended for valid and invalid logins
<b>Prerequisites for this test:</b> User has created an itch.io account
<b>Software Versions:</b> Browser: Edge, Safari, Chrome, Firefox Operating Systems: iOS, Android Database: MongoDB Atlas Cloud Database
<b>Required Configuration:</b> No special setup needed
<b>NOTES and RESULTS:</b> Mobile test starts from landing page, all tests passed

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/FAIL
<b>Attempt to login to deployed website with an existing Jamr Account - Web &amp; iOS/Android</b>					
1	Visit jamrdemo.herokuapp.com	App landing page returned	Landing page displayed		PASS
2	Click “Get Started” button	Prompted for site password	Site password popup		PASS
3	Enter site password	Redirected to itch.io OAuth	Redirected		PASS
4	Click on Authorize button	Redirected back into Jamr application	Redirected back to Jamr homepage	5.4.2.1.1 Login, 5.4.2.4.1 Mobile Login	PASS
<b>Alternative Flow 1: User has an itch.io account, but no Jamr account - Web Only</b>					
6	Visit jamrdemo.herokuapp.com	App landing page returned	Landing page displayed		PASS
7	Click “Get Started” button	Prompted for site password	Site password popup		PASS
	Enter site password	Redirected to itch.io OAuth	Redirected		PASS
8	Click on Authorize button	Redirected back into Jamr application registration	Redirected back to Jamr homepage		PASS
9	View registration form	Registration form displayed	Register page displayed		PASS
10	Submit empty/invalid form	Error message displayed, no redirect	Biography required message displayed		PASS
11	Submit valid form	Inputs saved, account marked registered, user redirected to home	Redirected to home page	5.4.2.1.1 Register	PASS
<b>Alternative Flow 2: User provides invalid itch.io OAuth key upon login - Web &amp; iOS/Android</b>					
13	Visit login page with modified api key	Error logged, user redirected to landing page	500 Error logged, redirected to landing page		PASS

System Test Case:

USER STORY: Find a Project

<b>Purpose:</b> Confirm that registered users are able to match with a project
<b>Prerequisites for this test:</b> User has an itch.io account, has registered for Jamr, and is not currently in a project.
<b>Software Versions:</b> Browser: Edge, Safari, Chrome, Firefox Operating Systems: iOS, Android Database: MongoDB Atlas Cloud Database
<b>Required Configuration:</b> No special setup needed

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/ FAIL
<b>Find a Project Matching Functionality - Web &amp; iOS/Android</b>					
1	User visits home page	Home page displayed	Home displayed		PASS
2	User clicks “Join a Team” button	Redirected to matching page	Redirected to project matching page	5.4.2.1.4 Find a Project, 5.4.2.4.3 Find a Project	PASS
<b>Alternative Flow 1: No projects are available (no applicable projects matching user’s tags, no projects have been created, or all have been voted on)</b>					
3	View display message, no actions possible	Display no projects screen	Empty page with “No available projects” message		PASS
<b>Alternative Flow 2: Projects are available</b>					
4	View first project in the matching list	Project(s) displayed match the following criteria: <ul style="list-style-type: none"> <li>- User has not previously marked interested/uninterested on the project</li> <li>- Project leader has not marked interested/uninterested on the user</li> <li>- User has not previously been removed from the project</li> <li>- There is at least one common tag between the user’s skills and the project’s “Looking for”</li> </ul>	Projects filtered correctly	5.4.2.1.4 Find a Project, 5.4.2.4.3 Find a Project	PASS
5	User selects “Interested”	User added to project’s candidate list, next project or empty projects message displayed	Added to candidates list, next project loaded	5.4.2.1.4 Find a Project, 5.4.2.4.3 Find a Project	PASS

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/ FAIL
6	User selects “Uninterested”	User added to project’s uninterested list, next project or empty projects message displayed	Added to uninterested list, next project loaded	5.4.2.1.4 Find a Project, 5.4.2.4.3 Find a Project	PASS

System Test Case:

USER STORY: Home Page

**Purpose:** Dynamically display the actions available to users depending on their project status**Prerequisites for this test:**

Flow 1: User has an itch.io account, has registered for Jamr, and has received two project invites

Flow 2: Project has at least two candidates

**Software Versions:**

Browser: Edge, Safari, Chrome, Firefox

Operating Systems: iOS, Android

Database: MongoDB Atlas Cloud Database

**TEST SCRIPT STEPS/RESULTS**

STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/FAIL
<b>Home Functionality – Web &amp; iOS/Android</b>					
<b>Flow 1: User is not in a project</b>					
1	Visits homepage	Buttons “Find a Project”, “Create a Project” and Invites table displayed	Buttons “Find a Project”, “Create a Project” and Invites table displayed		PASS
2	Click “Join a Team”	Redirected to Find a Project matching page	Redirected to Find a Project matching page		PASS
3	Go back, click “Create a Project”	Redirected to Create a Project page, new project form displayed	Redirected to Create a Project page, new project form displayed		PASS
4	Go back, click “View” on a project invite	Project summary displayed with Accept/Decline action buttons	Project summary displayed with Accept/Decline action buttons	5.4.2.1.5 Manage Invites, 5.4.2.4.5 Mobile Manage Invites	PASS
5	Click “Decline” action button	Project removed from User’s invites list and User added to project’s uninterested list	Project removed from User’s invites list and User added to project’s uninterested list	5.4.2.1.5 Manage Invites, 5.4.2.4.5 Mobile Manage Invites	PASS
6	Open other invite, click “Accept” action button	User added to team, homepage refreshed and status updated	User added to team, homepage refreshed and status updated	5.4.2.1.5 Manage Invites, 5.4.2.4.5 Mobile Manage Invites	PASS
<b>Alternative Flow 2: User is the leader of a project</b>					
7	Visits homepage	Buttons “Find Members”, “Manage Candidates”, “Go to Project” and Invites table displayed	Buttons “Find Members”, “Manage Candidates”, “Go to Project” and Invites table displayed		PASS
8	Click “Find Members”	Redirected to Find Members matching page	Redirected to Find Members matching page	5.4.2.3.4 Find Members, 5.4.2.5.4 Mobile Find Members	PASS
9	Go back, click “Manage Candidates”	Candidates modal opened with list of current candidates and action buttons to invite/reject them	Candidates modal opened with list of current candidates and action buttons to invite/reject them	5.4.2.3.2 Manage team members, 5.4.2.5.3 Manage Members	PASS



TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/FAIL
		or “No current” candidates message displayed	or “No current” candidates message displayed		
10	Click “Green Plus” action button on Web or “Invite” on mobile	Project invite sent to candidate	Approve user Modal opens and asks the user to confirm or cancel the invite on the web. Upon confirming the invite, the candidate receives an invite and is removed from the candidate list.	5.4.2.3.2 Manage team members, 5.4.2.5.3 Manage Members	PASS
11	Click “Red X” action button or “Reject” on mobile	Candidate removed from list	Decline User Modal opens and asks the user to confirm or cancel the decline. Upon confirming the decline, the candidate is removed from the candidates list.	5.4.2.3.2 Manage team members, 5.4.2.5.3 Manage Members	PASS
12	Close modal, click View on invite	On web: Project summary displayed, Accept button disabled  Mobile: Invites not accessible while in a project	Project summary is displayed and the user can not click accept while in another project. The user may decline the invite or close the invite modal.	5.4.2.1.5 Manage Invites	PASS
13	Close modal, click ”Go to Project”	Redirected to project dashboard page	Redirected to project dashboard page	5.4.2.2 Dashboard, 5.4.2.5.1 Team Dashboard	PASS
Alternative Flow 3: User is a member of a project					
14	Visits homepage	Buttons “Go to Project” and Invites table displayed	Buttons “Go to Project” and Invites table displayed		PASS
15	Click ”Go to Project”	Redirected to project dashboard page	Redirected to project dashboard page	5.4.2.2 Dashboard, 5.4.2.5.1 Team Dashboard	PASS
16	Go back, click “View” on an invite	On web: Project summary displayed, Accept button disabled  Mobile: Invites not accessible while in a project	Project summary is displayed and the user can not click accept while in another project. The user may decline the invite or close the invite modal.	5.4.2.1.5 Manage Invites	PASS

System Test Case:

USER STORY: Create a Project

<b>Purpose:</b> Confirm that registered users are able to create a project
<b>Prerequisites for this test:</b> User has an itch.io account, has registered for Jamr, and is not currently in a project.
<b>Software Versions:</b> Browser: Edge, Safari, Chrome, Firefox Database: MongoDB Atlas Cloud Database
<b>Required Configuration:</b> No special setup needed
<b>NOTES and RESULTS:</b> Functionality only available on web, all tests passed

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/ FAIL
<b>Create a Project Functionality - Web Only</b>					
1	User visits home page	Home page displayed	Home page displayed		PASS
2	User clicks "Create a Project" button	Redirected to create project form	Redirected to create project form		PASS
3	User submits empty form	Prompted to fill out the form properly	Project name required, description required, valid deadline required messaged appear		PASS
4	User submits filled out form	Project is created with proper information and the user is taken to the project dashboard where they have owner privileges over the project	Project created, user redirected to dashboard	<b>5.4.2.1.4 Create a Project</b>	PASS

System Test Case:

USER STORY: View and edit personal profile

<b>Purpose:</b> Confirm that registered users are able to create a project
<b>Prerequisites for this test:</b> User has an itch.io account and has registered for Jamr. User has content posted on their itch.io profile.
<b>Software Versions:</b> Browser: Edge, Safari, Chrome, Firefox Database: MongoDB Atlas Cloud Database
<b>Required Configuration:</b> No special setup needed
<b>NOTES and RESULTS:</b> Functionality only available on web, all tests passed

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/ FAIL
<b>View profile functionality - Web Only</b>					
1	User clicks profile section on navbar	Users personal profile is displayed with the information that they filled out when registering.	User's profile is displayed	5.4.2.1.2 View own profile	PASS
<b>Edit profile functionality - Web Only</b>					
2	Click the "Edit Profile" button on the personal profile page	Edit modal is shown.	Edit modal is shown.		PASS
3	Click replace avatar button	Local file system is brought up	Local file system popup opened		PASS
4	Cancel upload	File explorer closes and nothing happens.	File explorer closes and nothing happens.		PASS
5	Click replace avatar and upload image	File is uploaded and becomes users new avatar	File is uploaded and becomes users new avatar		PASS
6	Change checkbox tags	Checkboxes update	Checkboxes update		PASS
7	Clear description and click save	Prompted to add 50+ character description	Short biography (50+) characters required message appears		PASS
8	Add description and save	Profile is updated with new tags and description	Profile is updated with new tags and description	5.4.2.1.3 Edit profile	PASS
<b>Add showcase - Web Only</b>					
9	Click add showcase button on personal profile	Modal pops up displaying content from the users itch.io profile	Modal pops up displaying content from the users itch.io profile		PASS
10	Click on a piece of content	Modal shows text box to fill out a description	Modal shows text box to fill out a description		PASS
11	Click "Back to Select"	Goes back to content from the users itch.io profile	Goes back to content from the users itch.io profile		PASS
12	Select content and submit with no description	Prompted to fill out the showcase properly	Short description required error message displayed		PASS
13	Add valid description and click "Add Showcase"	Modal closed, showcase added to profile	Modal closed, showcase added to profile	5.4.2.1.3 Edit profile	PASS

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/ FAIL
14	Click on showcase content thumbnail image	User is taken to the itch.io page of that content	New tab opened of itch.io page		PASS
15	Click add showcase button again	Content in showcase should display grayed out	Icon gray and unclickable		PASS
<b>Edit and delete showcase - Web Only</b>					
16	Click edit cogwheel icon on showcase	Modal pops up displaying the description in a text box	Modal pops up displaying the description in a text box		PASS
17	Change description and click save changes	Changed description now appears on users profile	Changed description now appears on users profile	5.4.2.1.3 Edit profile	PASS
18	Click delete button on showcase	Modal pops up asking to confirm deletion	Modal pops up asking to confirm deletion		PASS
19	Click confirm	Showcase is removed from profile	Showcase is removed from profile	5.4.2.1.3 Edit profile	PASS

System Test Case:

USER STORY: Manage friends and view user profiles

<b>Purpose:</b> Allow users to manage friends and view other users profiles
<b>Prerequisites for this test:</b> User has an itch.io account and has registered for Jamr. Several friend invites. In a project. Assumes that user “loganfite” is registered within the application and is not currently friends with the user.
<b>Software Versions:</b> Browser: Edge, Safari, Chrome, Firefox Database: MongoDB Atlas Cloud Database
<b>NOTES and RESULTS:</b> Functionality only available on web, all tests passed

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/ FAIL
<b>Manage friends and view other user profiles - Web Only</b>					
<b>Flow 1: Manage friends</b>					
1	Navigate to the friends page	Displays friends page	Displays friends page		PASS
2	Click add friend button	Displays modal with search box	Displays modal with search box		PASS
3	Begin typing a username (loganfite)	Displays loganfite profile with add friend button	Displays loganfite profile with add friend button		PASS
4	Click “Add Friend” button	Friend request sent, button turns into “Request Sent” text	Friend request sent, button turns into “Request Sent”	5.4.2.1.8 Manage Friends	PASS
5	Close add friend modal, click “Invites” button	Displays modal with users that sent requests with an accept/reject button	Displays modal with users that sent requests with an accept/reject button		PASS
6	Click “Decline” on a request row	User will not be added to friends, invite is removed from the list	User will not be added to friends, invite is removed from the list	5.4.2.1.8 Manage Friends	PASS
7	Click “Accept” on another invite	User will be added to friends, invite is removed from the modal list	User will be added to friends, invite is removed from the modal list	5.4.2.1.8 Manage Friends	PASS
8	Navigate to the messages page	Messages page should display added friends in contacts list	Messages page should display added friends in contacts list		PASS
9	Navigate to the friends page	Friends list should display all added friends	Friends list should display all added friends		PASS
10	Click on a friend	Modal is shown with options to view profile and remove friend	Modal is shown with options to view profile and remove friend		PASS
11	Click remove friend	Friend is removed from friends list	Friend is removed from friends list	5.4.2.1.8 Manage Friends	PASS
<b>Alternative Flow 2: Manage friends through user profiles</b>					
12	Navigate to a user's profile through the project page.	Users profile is displayed with their avatar, name, description, tags, and showcases	Users profile is displayed with their avatar, name, description, tags, and showcases	5.4.2.1.2 View another user's profile	PASS

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/ FAIL
13	Click add friend button	Other user will be sent an invite, button turns into “Request Sent”	Other user will be sent an invite, button turns into “Request Sent”	<b>5.4.2.1.8 Manage Friends</b>	<b>PASS</b>
14	Navigate to a user that invited you through a direct link	Users profile is displayed with their avatar, name, description, tags, and showcases	Users profile is displayed with their avatar, name, description, tags, and showcases	<b>5.4.2.1.2 View another user’s profile</b>	<b>PASS</b>
15	Click accept to their friend request	User will be added to friends list, action buttons updated to single “Remove Friend” button	User added to friends list, action buttons updated to single “Remove Friend” button	<b>5.4.2.1.8 Manage Friends</b>	<b>PASS</b>

System Test Case:

USER STORY: Message friends

<b>Purpose:</b> Allow users to message friends
<b>Prerequisites for this test:</b> User has an itch.io account and has registered for Jamr. Friend added.
<b>Software Versions:</b> Browser: Edge, Safari, Chrome, Firefox Database: MongoDB Atlas Cloud Database
<b>NOTES and RESULTS:</b> On mobile, contacts list is available by selecting the hamburger menu from the top right corner. Pressing on the “chatting with” box on mobile will not redirect to user profile.

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/ FAIL
<b>Message friends- Web &amp; iOS/Android</b>					
<b>Flow 1: Message Friend</b>					
1	Navigate to the messaging page	Messaging page is displayed with added friends shown in contacts	Messaging page is displayed with added friends shown in contacts		PASS
2	Click on contact	Message page shows text box and other users name	Message page shows text box and other users name		PASS
3	Type in textbox and submit	Message is sent to the other user and displayed in the page	Message is sent to the other user and displayed in the page	5.4.2.1.7 Direct Messaging, 5.4.2.4.5 Mobile Direct Messaging	PASS
4	Receive message from other user	Message is displayed, timestamp is correct, and the messages are ordered by timestamps.	Message is displayed, timestamp is correct, and the messages are ordered by timestamps.	5.4.2.1.7 Direct Messaging, 5.4.2.4.5 Mobile Direct Messaging	PASS
5	On web: Navigate to the friends profile by clicking “Chatting with” box	Friends profile is displayed with proper name, tags, description and showcases	Friends profile is displayed with proper name, tags, description and showcases	5.4.2.1.2 View Profile	PASS
6	Click remove friend	Friend is removed from friends list	Friend is removed from friends list	5.4.2.1.8 Manage Friends	PASS
7	Navigate back to messaging page	Message with removed friend is now gone	Message with removed friend is now gone		PASS

System Test Case:  
USER STORY: Settings

<b>Purpose:</b> Allow users to change theme and delete profile
<b>Prerequisites for this test:</b> User has an itch.io account and has registered for Jamr. Friend added with messages sent. Project leader.
<b>Software Versions:</b> Browser: Edge, Safari, Chrome, Firefox Database: MongoDB Atlas Cloud Database
<b>NOTES and RESULTS:</b> The first round of testing identified a case where the Find a Team project description text was white and could not be read on light mode.

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/FAIL
<b>Settings functionality- Web Only</b>					
<b>Flow 1: Theme</b>					
1	Navigate to the settings page	Settings page is displayed with a switch to change theme and button to delete profile	Settings page is displayed with a switch to change theme and button to delete profile		PASS
2	Click the “Application Theme” switch	Page should change from dark theme to light theme or from light theme to dark theme	Page changes from dark theme to light theme and from light theme to dark theme	5.4.2.1.9 Settings	PASS
3	Navigate through all pages and profiles	Colors should match selected theme	Colors match selected theme		PASS
<b>Flow 1: Delete profile</b>					
4	Click delete profile button	Profile is deleted, user redirected to landing page	Profile is deleted, user redirected to landing page	5.4.2.1.9 Settings	PASS
5	Navigate to deleted users profile via URL from another user account	User not found message should be displayed	User not found message displayed		PASS
6	Navigate to project dashboard as a member of deleted users project	Not in a project message should be displayed	Not in a project page displayed		PASS
7	Navigate to friends list of a friend of the deleted user	Should no longer appear on friends list	Friend no longer appears		PASS
8	Navigate to messages of a friend with deleted user	Should no longer appear in contacts and messages are deleted	Contact no longer appears, messages deleted		PASS



System Test Case:

USER STORY: Find Members

<b>Purpose:</b> Confirm that project leaders are able to match with registered users
<b>Prerequisites for this test:</b> User has an itch.io account, has registered for Jamr, and is the leader of a project
<b>Software Versions:</b> Browser: Edge, Safari, Chrome, Firefox Operating Systems: iOS, Android Database: MongoDB Atlas Cloud Database
<b>Required Configuration:</b> No special setup needed

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/FAIL
<b>Find Members Matching Functionality - Web &amp; iOS/Android</b>					
1	User visits home page	Home page displayed	Home page displayed		PASS
2	User clicks Find Members button	Redirected to matching page	Redirected to matching page		PASS
<b>Alternative Flow 1: No members are available (no applicable members matching project's tags or all members have been previously voted on)</b>					
3	View display message, no actions possible	Display no members screen	No members screen displayed		PASS
<b>Alternative Flow 2: Members are available</b>					
4	View first project in the matching list	Members(s) displayed match the following criteria ordered by descending Jamr reputation: <ul style="list-style-type: none"> <li>- Leader has not previously marked interested/uninterested on the user</li> <li>- User leader has not marked interested/uninterested on the project</li> <li>- User has not previously been removed from the project</li> <li>- There is at least one common tag between the user's skills and the project's "Looking for"</li> </ul>	Appropriate members displayed	5.4.2.3.4 Find Members, 5.4.2.5.4 Mobile Find Members	PASS
5	Leader selects "Interested"	User is sent a project invite, next project or empty users message displayed	User is sent a project invite, next project or empty users message displayed	5.4.2.3.4 Find Members, 5.4.2.5.4 Mobile Find Members	PASS
6	Leader selects "Uninterested"	User added to project's uninterested list, next project or empty users message displayed	User added to project's uninterested list, next project or empty users message displayed	5.4.2.3.4 Find Members, 5.4.2.5.4 Mobile Find Members	PASS

System Test Case:

USER STORY: Use Project Calendar

<b>Purpose:</b> Confirm that team leader and members are able to use the project calendar component
<b>Prerequisites for this test:</b> User is the leader or a member of a project Another user account needed who is a member of the project
<b>Software Versions:</b> Browser: Edge, Safari, Chrome, Firefox Operating Systems: iOS, Android Database: MongoDB Atlas Cloud Database
<b>Required Configuration:</b> No special setup needed
<b>NOTES and RESULTS:</b> Mobile calendar can only select days with an indicator on them, marking at least one event present. This functionality is intentional since the component is read only on mobile. Initial testing identified an error where users could not see events from other users in a different timezone. All tests now pass.

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/FAIL
<b>Dashboard View Calendar Functionality - Web &amp; iOS/Android – Reqmts Validated: 5.4.2.1.X</b>					
1	User visits dashboard from homepage or navigation bar	Project dashboard displayed	Project dashboard displayed		PASS
2	User clicks on a day on the calendar	Mobile: - If the day has at least one event, opens a modal with a list of events for the day  Web: - Opens event manager list with any existing dates in a row with edit and delete buttons and an add new event button	Mobile: - If the day has at least one event, opens a modal with a list of events for the day  Web: - Opens event manager list with any existing dates in a row with edit and delete buttons and an add new event button	<b>5.4.2.2.5 Calendar, 5.4.2.5.1 Mobile Team Dashboard</b>	PASS
<b>Create calendar event functionality - Web Only</b>					
3	Leader clicks “New Event” button	New event form opened asking for event title	New event form opened asking for event title		PASS
4	Click on Cancel button	Event form closed	New Event form closed		PASS
5	Leader clicks “New Event” and “Save” with empty title	Title required prompt displayed	Event title required error message displayed		PASS
6	Leader enters a title, clicks “Save”	Event saved, icon added to calendar day, new event form closed, event list updated, new event communicated to any other members currently viewing the dashboard	Event saved, icon added to calendar day, new event form closed, event list updated, new event communicated to any	<b>5.4.2.2.5 Calendar</b>	PASS

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/FAIL
			other members currently viewing the dashboard		
<b>Edit/Delete calendar event functionality - Web Only</b>					
7	User clicks on a calendar day to view event list	Event manager modal opens, displays existing events as rows with a title and action buttons: <ul style="list-style-type: none"> <li>- If the event was created by the user or the user is the project leader, both Edit and Delete buttons are active</li> <li>- If the event was not created by the user and the user is not a leader, both buttons are disabled</li> </ul>	Buttons disabled for member, buttons active for leader	5.4.2.2.5 Calendar	PASS
8	Leader clicks edit button	Project title becomes editable, action buttons become Save and Cancel	Project title becomes editable, action buttons become Save and Cancel		PASS
9	Leader clicks cancel	Event unedited, action buttons return to Edit & Delete	Event unedited, action buttons return to Edit & Delete		PASS
10	Leader clicks edit button and submits empty title	Event title required message displayed	Event title required error message displayed		PASS
11	Leader enters a new title and clicks save	Event title updated in event list, action buttons returned to Edit & Delete, change communicated and event updated for any other members currently viewing the event list	Event title updated in event list, action buttons returned to Edit & Delete, change communicated and event updated for any other members currently viewing the event list	5.4.2.2.5 Calendar	PASS
12	Leader clicks delete button	Confirmation prompt displayed with Confirm & Cancel buttons	Confirmation prompt displayed with Confirm & Cancel buttons		PASS
13	Leader clicks cancel	Prompt closed, no action	Prompt closed, no action		PASS
14	Leader clicks delete and confirms the action	Event removed from event list along with dot indicator on card view if it is the only event for that day, change communicated and event updated for any other members currently viewing the event list or on dashboard	Event removed from event list along with dot indicator on card view if it is the only event for that day, change communicated and event updated for any other members currently viewing the event list or on dashboard	5.4.2.2.5 Calendar	PASS

System Test Case:

USER STORY: Use Project To Do List

<b>Purpose:</b> Confirm that team leader and members are able to use the project to do component
<b>Prerequisites for this test:</b> User is the leader of a project Another user account needed who is a member of the project
<b>Software Versions:</b> Browser: Edge, Safari, Chrome, Firefox Operating Systems: iOS, Android Database: MongoDB Atlas Cloud Database
<b>Required Configuration:</b> No special setup needed
<b>NOTES and RESULTS:</b> Mobile users may only read task information and cannot open the task manager unless on web. All tests passed.

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/FAIL
<b>View To Do Functionality – Web &amp; iOS/Android</b>					
1	Visit dashboard from homepage or navigation bar	Project dashboard displayed, To Do cards list populated with tasks not marked completed	Project dashboard displayed, To Do cards list populated with tasks not marked completed	5.4.2.2.1 To Do List, 5.4.2.5.1 Mobile Team Dashboard	PASS
<b>Add Task - Web Only</b>					
2	User clicks on View Tasks button	Task Manager modal displayed	Task Manager modal displayed	5.4.2.2.1 To Do List	PASS
3	Click on New Task button	New task form displayed	New task form displayed		PASS
4	Submit empty/invalid form	Task Title required prompt displayed	Task title required error message displayed		PASS
5	Enter a title, click “Save”	Task saved, card added to calendar day, new event form closed, event list updated, new event communicated to any other members currently viewing the dashboard	Task saved, card added to calendar day, new event form closed, event list updated, new event communicated to any other members currently viewing the dashboard	5.4.2.2.1 To Do List	PASS
<b>Edit/Delete Task as Project Leader or Task Creator - Web Only</b>					
6	Click on expand task icon	Task expanded, revealing assignment and priority information, along with Edit and Delete action buttons	Task expanded, revealing assignment and priority information, along with Edit and Delete action buttons		PASS
7	Click edit button	Task title and description become text inputs, assign to and priority text become dropdown selectors, action buttons become save and delete	Task title and description become text inputs, assign to and priority text become dropdown selectors, action buttons become save and delete		PASS
8	Click cancel	Task unedited, card reverted to previous view	Task unedited, card reverted to previous view		PASS

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/FAIL
9	Click edit button and submit empty title	Task title required message displayed	Task title required message displayed		PASS
10	Save valid changes	Task title, description, assignment, and priority updated in task list, action buttons returned to Edit & Delete, change communicated and event updated for any other members currently viewing the event list	Task title, description, assignment, and priority updated in task list, action buttons returned to Edit & Delete, change communicated and event updated for any other members currently viewing the event list	5.4.2.2.1 To Do List	PASS
11	Click delete button	Confirmation prompt displayed with Confirm & Cancel buttons	Confirmation prompt displayed with Confirm & Cancel buttons		PASS
12	Click cancel	Prompt closed, no action	Prompt closed, no action		
13	Click delete and confirm the action	Task removed from list, change communicated and event updated for any other members currently viewing the task manager or visiting dashboard	Task removed from list, change communicated and event updated for any other members currently viewing the task manager or visiting dashboard	5.4.2.2.1 To Do List	PASS
<b>Edit unassigned task or task assigned to member - Web Only</b>					
14	Expand task assigned to user or a task marked unassigned that they did not create	Additional task information shown, edit button active, delete button disabled	Additional task information shown, edit button active, delete button disabled		PASS
15	Click edit button	Task title and description become text inputs, assign to and priority text become dropdown selectors, action buttons become save and delete	Task title and description become text inputs, assign to and priority text become dropdown selectors, action buttons become save and delete		PASS
16	Click cancel	Task unedited, card reverted to previous view	Task unedited, card reverted to previous view		PASS
17	Click edit button and submit empty title	Task title required message displayed	Task title required message displayed		PASS
18	Save valid changes	Task title, description, assignment, and priority updated in task list, action buttons returned to Edit & Delete, change communicated and event updated for any other members currently viewing the event list	Task title, description, assignment, and priority updated in task list, action buttons returned to Edit & Delete, change communicated and event updated for any other members currently viewing the event list	5.4.2.2.1 To Do List	PASS
<b>Complete Task - Web Only</b>					

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/FAIL
19	Click Complete Task button	Confirmation prompt displayed with Confirm and Cancel action buttons	Confirmation prompt displayed with Confirm and Cancel action buttons		PASS
20	Click cancel	Prompt closed, no other actions	Prompt closed, no other actions		PASS
21	Click Complete and Confirm buttons	Task removed from To Do tab and moved into Completed tab, development log created, change communicated and event updated for any other members currently viewing the task manager or visiting dashboard	Task removed from To Do tab and moved into Completed tab, development log created, change communicated and event updated for any other members currently viewing the task manager or visiting dashboard	5.4.2.2.1 To Do List	PASS

System Test Case:

USER STORY: Use Project Assets

<b>Purpose:</b> Confirm that team leader and members are able to upload, download, and delete project files
<b>Prerequisites for this test:</b> User is the leader or a member of a project Another user account needed who is a member of the project
<b>Software Versions:</b> Browser: Edge, Safari, Chrome, Firefox Operating Systems: iOS, Android Database: MongoDB Atlas Cloud Database
<b>Required Configuration:</b> No special setup needed
<b>NOTES and RESULTS:</b> Testing identified an error where an input validation message does not display.

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/FAIL
<b>View Assets Functionality - Web &amp; iOS/Android – Reqmts Validated: 5.4.2.1.X</b>					
1	User visits dashboard from homepage or navigation bar	Assets card list populated with uploaded filenames	Assets card list populated with uploaded filenames	5.4.2.2.2 Share Files, 5.4.2.5.1 Mobile Team Dashboard	PASS
<b>Upload file functionality - Web Only</b>					
2	Click View Files	File manager popup opened, existing files displayed by row with file information and action buttons  If user is project leader or uploaded the file, trash icon visible next to files	File manager popup opened, existing files displayed by row with file information and action buttons  If user is project leader or uploaded the file, trash icon visible next to files	5.4.2.2.2 Share Files	PASS
3	Click New File	Upload file form displayed	Upload file form displayed		PASS
4	Click Cancel	Form closed	Form closed		PASS
5	Click New File & Upload	File required error message displayed	File required error message displayed		PASS
6	Click Choose File, select a file with filename > 50 characters and click Upload	Error message displayed requiring shorter filename	Filename too long error message displayed		PASS
7	Click Choose File, select a file > 25mb and click Upload	File size error displayed	File must be less than 25mb error message displayed		PASS
8	Select valid file and click Upload	Progress bar begins updating, form closes upon completion, file added to row list, development log generated and file updated for any other members currently viewing	Progress bar begins updating, form closes upon completion, file added to row list, development log generated and file	5.4.2.2.2 Share Files	PASS

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/ FAIL
		the task manager or visiting dashboard	updated for any other members currently viewing the task manager or visiting dashboard		
<b>Download File - Web Only</b>					
9	Click View Files	File manager popup opened, existing files displayed by row	File manager popup opened, existing files displayed by row	5.4.2.2.2 Share Files	PASS
10	Click blue download icon	File begins downloading	File begins downloading	5.4.2.2.2 Share Files	PASS
<b>Delete File as Leader or File Creator - Web Only</b>					
11	Click View Files	File manager popup opened, existing files displayed by row	File manager popup opened, existing files displayed by row	5.4.2.2.2 Share Files	PASS
12	Click red trash icon	Prompt for confirmation	Prompted for confirmation		PASS
13	Click cancel	Prompt closed, no action	Prompt closed, no action		PASS
14	Click red trash icon, confirm delete	File removed from File Manager and Assets list, change communicated and updated for any other members currently visiting dashboard	File removed from File Manager and Assets list, change communicated and updated for any other members currently visiting dashboard	5.4.2.2.2 Share Files	PASS



System Test Case:

USER STORY: Use Project Whiteboard

<b>Purpose:</b> Confirm that team leader and members are able to view and use project whiteboard
<b>Prerequisites for this test:</b> User is the leader or a member of a project Another user account needed who is a member of the project
<b>Software Versions:</b> Browser: Edge, Safari, Chrome, Firefox Operating Systems: iOS, Android Database: MongoDB Atlas Cloud Database
<b>Required Configuration:</b> No special setup needed
<b>NOTES and RESULTS:</b> The mobile application whiteboard does not update in real time - it only updates when the whiteboard is cleared or once another user releases their mouse after drawing a line.

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/FAIL
<b>View Whiteboard Thumbnail Functionality - Web &amp; iOS/Android</b>					
1	User visits dashboard from homepage or navigation bar	Whiteboard thumbnail displayed	Whiteboard thumbnail displayed	5.4.2.2.6 Whiteboard, 5.4.2.5.1 Mobile Team Dashboard	PASS
<b>Edit Whiteboard - Web Only</b>					
2	Click whiteboard thumbnail	Interactive whiteboard popup opened	Interactive whiteboard popup opened	5.4.2.2.6 Whiteboard	PASS
3	Draw by clicking and holding and dragging cursor	Lines added to popup whiteboard, real-time updates sent to any other users viewing the module	Lines added to popup whiteboard, real-time updates sent to any other users viewing the module	5.4.2.2.6 Whiteboard	PASS
4	Stop drawing	Save thumbnail data and send image update to users viewing dashboard	Thumbnail image update to users viewing dashboard	5.4.2.2.6 Whiteboard	PASS
5	Adjust pen color and brush size, draw new lines	Toolbar and cursor updated, real-time updates sent to any other users viewing the module during drawing and thumbnail data saved and updated to users viewing dashboard	Toolbar and cursor updated, real-time updates sent to any other users viewing the module during drawing and thumbnail data saved and updated to users viewing dashboard		PASS
6	Select eraser tool, erase line(s)	Cursor updated to pink circle, lines removed, real-time updates sent to any other users viewing the module	Cursor updated to pink circle, lines removed, real-time updates sent to any other users viewing the module	5.4.2.2.6 Whiteboard	PASS
7	Stop erasing	Save thumbnail data and send image update to users viewing dashboard	Image updated to users viewing dashboard		PASS
8	Click Clear button	Whiteboard data cleared in real-time popup and thumbnail image emptied on dashboard	Whiteboard data cleared in real-time popup and thumbnail image emptied on dashboard	5.4.2.2.6 Whiteboard	PASS

<b>TEST SCRIPT STEPS/RESULTS</b>					
<b>STEP</b>	<b>TEST STEP/INPUT</b>	<b>EXPECTED RESULTS</b>	<b>ACTUAL RESULTS</b>	<b>Requirements Validated</b>	<b>PASS/FAIL</b>
9	Another member draws on the whiteboard	Drawing updated in real-time with thumbnail image updating when the other user stops drawing	Drawing updated in real-time with thumbnail image updating when the other user stops drawing	<b>5.4.2.2.6 Whiteboard</b>	<b>PASS</b>
10	Click Close	Whiteboard thumbnail updated for user, popup closed	Whiteboard thumbnail updated for user, popup closed	<b>5.4.2.2.6 Whiteboard</b>	<b>PASS</b>

System Test Case:

USER STORY: Use Project Group Chat

<b>Purpose:</b> Confirm that team leader and members are able to communicate through the project group chat
<b>Prerequisites for this test:</b> At least two member accounts in the same project needed to perform test
<b>Software Versions:</b> Browser: Edge, Safari, Chrome, Firefox Operating Systems: iOS, Android Database: MongoDB Atlas Cloud Database
<b>Required Configuration:</b> No special setup needed

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/ FAIL
<b>Group Chat Functionality - Web &amp; iOS/Android</b>					
1	User visits dashboard from homepage or navigation bar	Group chat card populated with existing chat messages	Group chat card populated with existing chat messages	5.4.2.2.4 Team chat, 5.4.2.5.2 Mobile Team chat	PASS
2	Type in textbox and submit	Message saved in database and displayed to other members viewing chat - Message sent by self right aligned and contained in purple box	Message saved in database and displayed to other members viewing chat - Message sent by self right aligned and contained in purple box	5.4.2.2.4 Team chat, 5.4.2.5.2 Mobile Team chat	PASS
3	Receive message from another team member	Message displayed in correct order, messages from teammates left aligned and contained in white box	Message displayed in correct order, messages from teammates left aligned and contained in white box	5.4.2.2.4 Team chat, 5.4.2.5.2 Mobile Team chat	PASS

System Test Case:

USER STORY: View Project Development Logs

<b>Purpose:</b> Confirm that team leader and members are able to gauge project progress by viewing development logs
<b>Prerequisites for this test:</b> User is the leader or a member of a project and at least one log has been automatically created from prior task completion or file uploaded
<b>Software Versions:</b> Browser: Edge, Safari, Chrome, Firefox Operating Systems: iOS, Android Database: MongoDB Atlas Cloud Database
<b>Required Configuration:</b> No special setup needed

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/FAIL
<b>Group Chat Functionality - Web &amp; iOS/Android</b>					
1	User visits dashboard from homepage or navigation bar	Development Logs card populated with existing project logs ordered by date	Development Logs card populated with existing project logs ordered by date	5.4.2.2.3 Development log, 5.4.2.5.1 Mobile Team Dashboard	PASS
2	Expand log	View additional information such as type of log, file/task name associated, the user who generated it, and datetime generated	View additional information such as type of log, file/task name associated, the user who generated it, and datetime generated	5.4.2.2.3 Development log, 5.4.2.5.1 Mobile Team Dashboard	PASS

System Test Case:

USER STORY: Manage Project Settings

<b>Purpose:</b> Confirm that team leader and members are able to modify appropriate project settings
<b>Prerequisites for this test:</b> Requires two accounts, one leader and one member to test their respective settings. Assumes user is already on the project dashboard page. Project requires multiple candidates.
<b>Software Versions:</b> Browser: Edge, Safari, Chrome, Firefox Database: MongoDB Atlas Cloud Database
<b>Required Configuration:</b> No special setup needed
<b>NOTES and RESULTS:</b> Project settings not available on iOS & Android application. Testing identified an error where user profiles open in the current tab instead of in a new tab.

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/FAIL
<b>Leader Project Settings - General Tab - Web Only</b>					
1	Leader clicks on cogwheel icon in the top right	Project settings modal opened defaulting to general settings tab	Project settings modal opened defaulting to general settings tab		PASS
2	Click edit button	Project name, description, and deadline become editable inputs	Project name, description, and deadline become editable inputs		PASS
3	Click cancel button	Display returned to text	Display returned to text		PASS
4	Click save	New inputs validated under the criteria: <ul style="list-style-type: none"> <li>- Project name not empty</li> <li>- Description &gt;= 50 Chars</li> <li>- Deadline greater than current time</li> </ul> Error message displayed if any inputs invalid, unable to save until valid. Updates reflect for users currently viewing the dashboard.	Valid inputs saved, appropriate error messages shown, successful changes broadcast to connected devices viewing the dashboard	<b>5.4.2.3.1 Manage Project</b>	PASS
<b>Leader Project Settings - Members Tab - Web Only</b>					
5	Leader clicks on cogwheel icon in the top right	Project settings modal opened defaulting to general settings tab	Project settings modal opened defaulting to general settings tab		PASS
6	Click Members tab	Current members displayed in rows with rating and removal action button	Current members displayed in rows with rating and removal action button		PASS
7	Click member profile name/avatar	User's profile opened in new tab	User's profile opened in new tab	<b>5.4.2.1.2 View Profile</b>	PASS
8	Close tab, click "Give Rating"	5 gray stars displayed and fill gold on hover from 1 to 5	5 gray stars displayed and fill gold on hover from 1 to 5		PASS

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/FAIL
9	Click one of the stars	View updated to reflect rating given to user, user's overall reputation updated	View updated to reflect rating given to user, user's overall reputation updated	5.4.2.3.3 Rate Team Members	PASS
10	Click another rating for the same member	Rating adjusted	Rating adjusted	5.4.2.3.3 Rate Team Members	PASS
11	Click red trash icon next to member	Confirmation message displayed prompting deletion	Confirmation message displayed prompting deletion		PASS
12	Click cancel	Prompt removed, no further action	Prompt removed, no further action		PASS
13	Click trash icon and confirm	User removed from project, unassigned from any existing tasks	User removed from project, unassigned from any existing tasks	5.4.2.3.2 Manage team members	PASS
Leader Project Settings - Candidates Tab - Web Only					
14	Leader clicks on cogwheel icon in the top right	Project settings modal opened defaulting to general settings tab	Project settings modal opened defaulting to general settings tab		PASS
15	Click Candidates tab	Candidate profiles (users who have marked interested in the project via matching) displayed in rows with add and remove action buttons	Candidate profiles (users who have marked interested in the project via matching) displayed in rows with add and remove action buttons		PASS
16	Click candidate name/avatar	Candidate's profile opened in new tab	Candidate's profile opened in new tab	5.4.2.1.2 View Profile	PASS
17	Close tab, click green add icon	Display confirmation popup to send project invite to candidate	Confirmation popup to send project invite to candidate		PASS
18	Click cancel	Prompt closed, no action	Prompt closed, no action		PASS
19	Click green add icon, confirm invite	Candidate removed from row list, project invite sent to candidate	Candidate removed from row list, project invite sent to candidate	5.4.2.3.2 Manage team members	PASS
20	Click red trash icon on next candidate	Display confirmation message to remove candidate	Confirmation message to remove candidate displayed		PASS
21	Confirm removal	Candidate removed from list, added to project's uninterested users list	Candidate removed from list, added to project's uninterested users list	5.4.2.3.2 Manage team members	PASS
Leader Project Settings - Advanced Tab - Web Only					
22	Leader clicks on cogwheel icon in the top right	Project settings modal opened defaulting to general settings tab	Project settings modal opened defaulting to general settings tab		PASS
23	Click Advanced tab	Project "looking for" tags displayed and set according to leader's preference	Project "looking for" tags displayed and set according to leader's preference		PASS
24	Click on tags	Update project preferences and check/uncheck tag on screen	Project preferences updated and check/uncheck tag on screen	5.4.2.3.1 Manage Project	PASS
25	Click delete project	Prompt for confirmation	Deletion confirmation popup opened		PASS
26	Click cancel	Prompt closed, no action	Prompt closed, no action		PASS

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/ FAIL
27	Click delete and confirm	Project deleted, all events, whiteboard, tasks, asset/, devlogs, message data deleted, all members removed from project, project page refreshed	Project deleted, all events, whiteboard, tasks, asset/, devlogs, message data deleted, all members removed from project, project page refreshed	5.4.2.3.1 Manage Project	PASS
Member Project Settings - Web Only					
28	Member clicks on cogwheel icon in the top right	Member settings modal opened displaying team members, rating button, and leave project button	Member Settings modal opens displaying current members currently working on the project. The user can rate each member individually, view each member's profile page or click the "Leave Project" button.		PASS
29	Click on member profile avatar/username	Member's profile opened in new tab	Member's profile opened in new tab	5.4.2.1.2 View Profile	PASS
30	Close tab, click give rating button	5 gray stars displayed and fill gold on hover from 1 to 5	After clicking the "give rating" button five gray stars display with color changing to gold on hover.		PASS
31	Click one of the stars	View updated to reflect rating given to user, user's overall reputation updated	After a user is given a rating, the stars remain filled with gold color up to the highest star chosen. User who is rated overall rating changes on profile.	5.4.2.2.7 Member Settings	PASS
32	Click Leave Project	Confirmation popup displayed	Confirm Leave Project modal opens up and prompts the user to confirm or cancel this action.		PASS
33	Click cancel	Popup closed, no action	Confirm Leave Project modal is closed and user remains on Member Settings.		PASS
34	Click Leave Project and Confirm	User redirected to dashboard page where informed they are no longer in a project, given links to match or create a project	User Dashboard page is refreshed and no project information or details are no longer available. The user is informed they are no longer part of a project and are given the option to start matching or create a new project.	5.4.2.2.7 Member Settings	PASS

System Test Case:  
USER STORY: Logout

<b>Purpose:</b> Allow users to logout of the website
<b>Prerequisites for this test:</b> User has an itch.io account and has registered for Jamr. Logged in.
<b>Software Versions:</b> Browser: Edge, Safari, Chrome, Firefox, Operating System: iOS, Android Database: MongoDB Atlas Cloud Database
<b>NOTES and RESULTS:</b> Web and mobile have separate tokens that need to be checked upon testing. All tests passed.

TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/ FAIL
<b>Logout functionality - Web &amp; iOS/Android</b>					
<b>Flow 1: Logout</b>					
1	Click the logout button on the navbar	Users are taken back to the landing page and their token is revoked.	User is redirected to the landing page and their token is revoked.	<b>5.4.2.1.10 Logout, 5.4.2.4.2 Mobile Logout</b>	<b>PASS</b>
2	Click get started button	User is prompted to log back into Jamr	User is redirected to Itch.io authentication where they are able to login to Jamr application again.		<b>PASS</b>



## 9. Results

Below are screenshots and descriptions of some of the primary functionalities of the applications. For a complete list of screenshots and detailed functionality descriptions, refer to the attached User Manual.

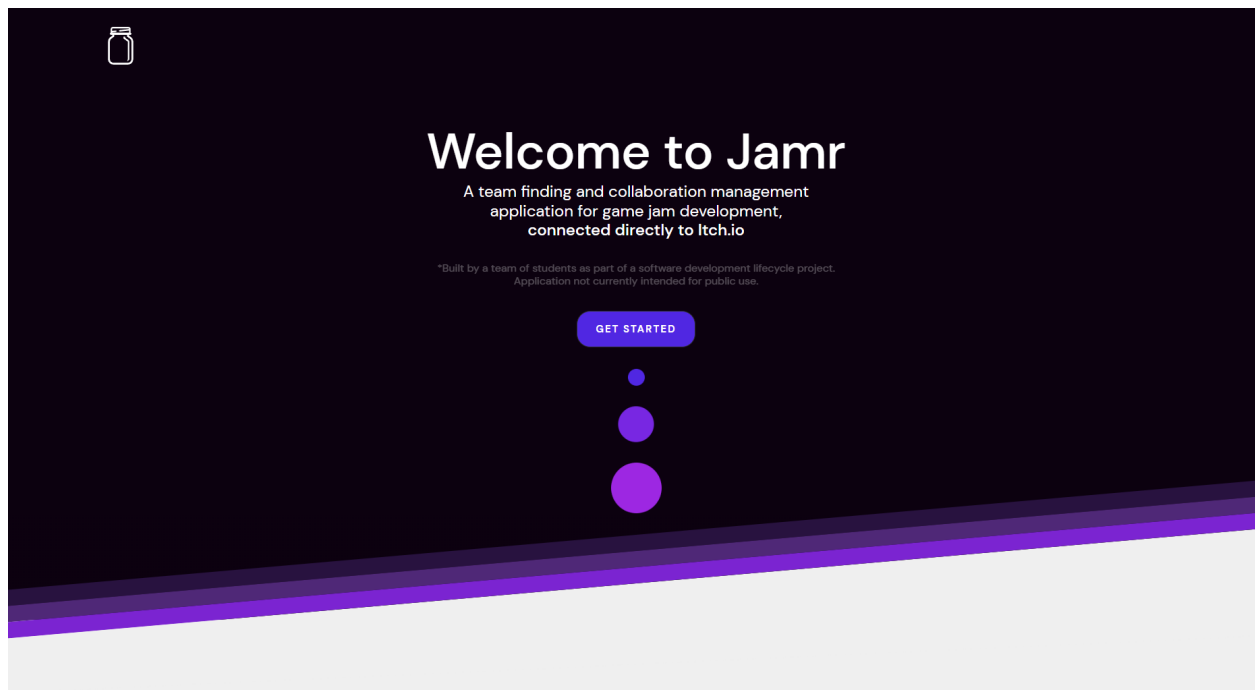
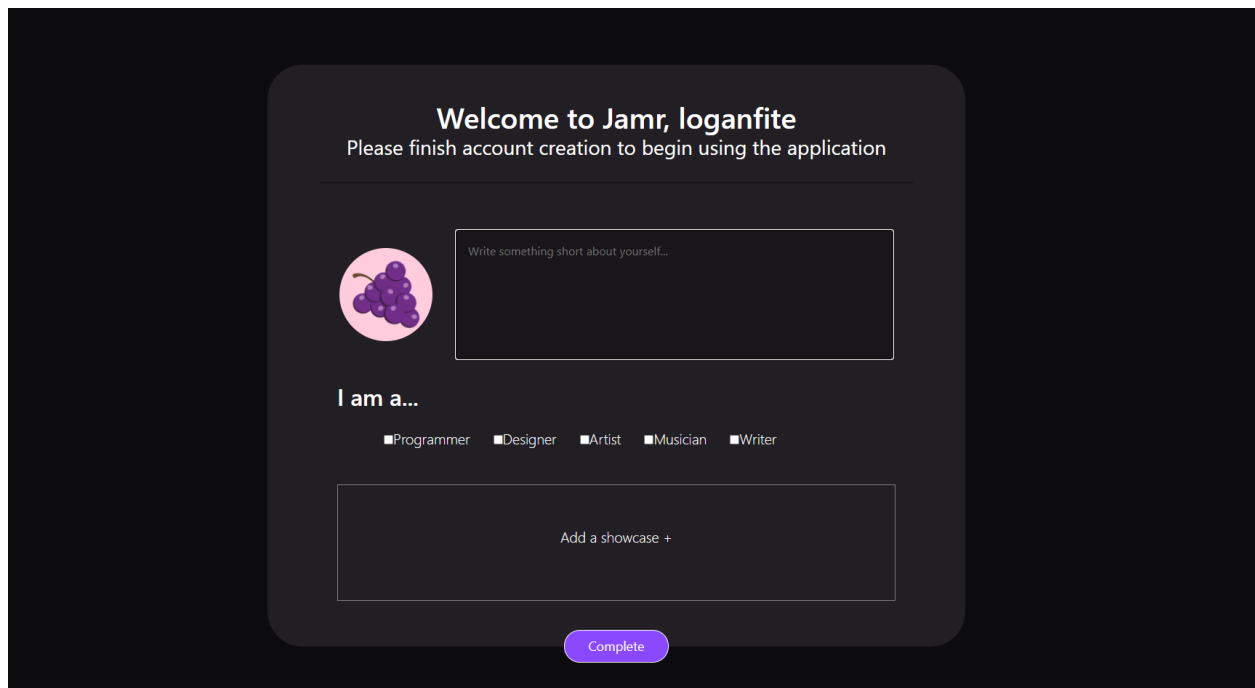


Figure 9.1 - Landing Page


The landing page is the first page users will see when they visit the webpage. This has to make an impact on the user and make them want to experience the application, and guide them through it with ease. By clicking the get started button, the user will be prompted to login to the application through itch.io's OAuth system. The landing page also contains other information about the applications uses and goals. At the bottom of the landing page, users can submit bug reports using a Google Form.



The image shows a user registration page for 'Jamr, loganfite'. The page has a dark background with a central light gray rounded rectangle containing the registration form. At the top, it says 'Welcome to Jamr, loganfite' and 'Please finish account creation to begin using the application'. Below this is a profile picture placeholder (a bunch of purple grapes) and a text input field for a short bio. Underneath is a section 'I am a...' with five radio button options: Programmer, Designer, Artist, Musician, and Writer. Below these is a showcase section with a placeholder and a '+'. At the bottom is a purple 'Complete' button.

Welcome to Jamr, loganfite

Please finish account creation to begin using the application



Write something short about yourself...

I am a...

☐ Programmer ☐ Designer ☐ Artist ☐ Musician ☐ Writer

Add a showcase +

Complete

Figure 9.2 - User Registration Page

The user registration page successfully allows users to create a Jamr account through the itch.io OAuth login system. This page successfully receives additional user information that will become useful for future matching functionality through self-assigned tags and a short user biography. The application requires that all users complete this stage before being allowed to use the full features of the application, and does not appear again after subsequent logins once registration is complete.

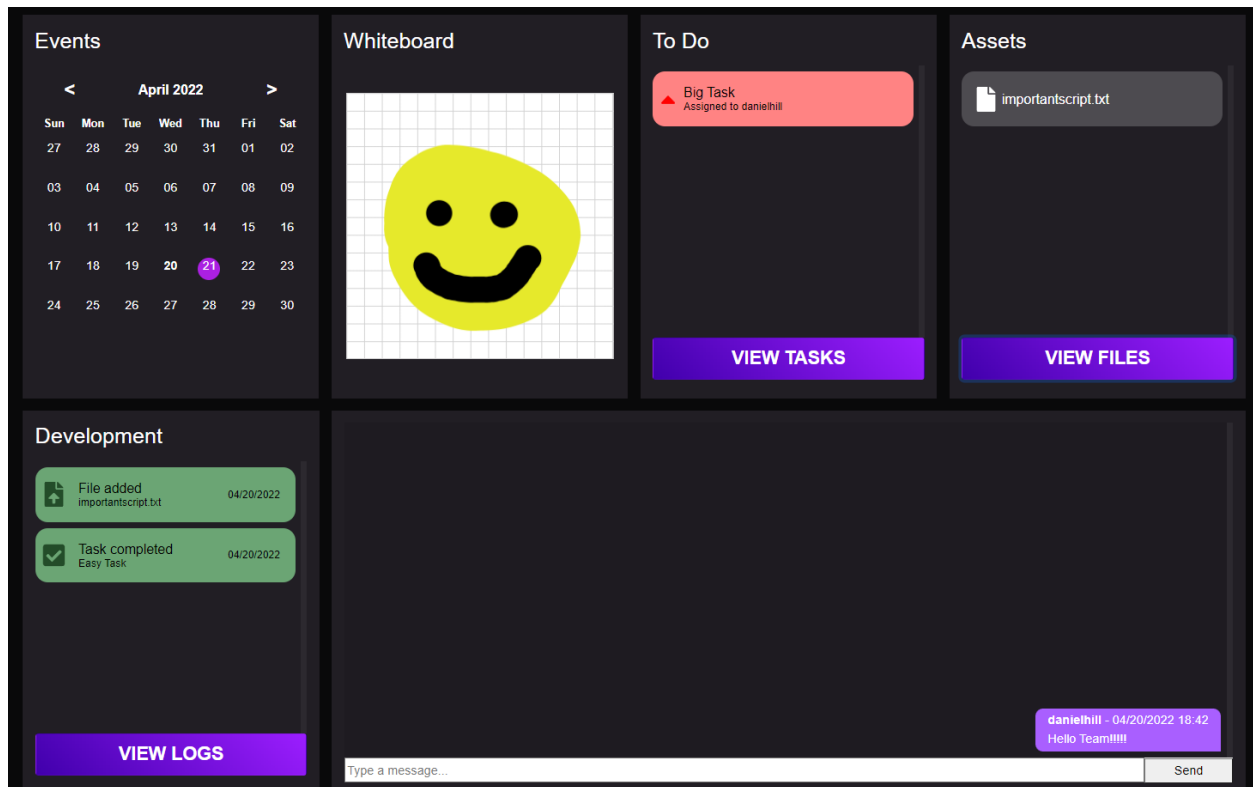


Figure 9.3 - Project Dashboard - All Components (Web)

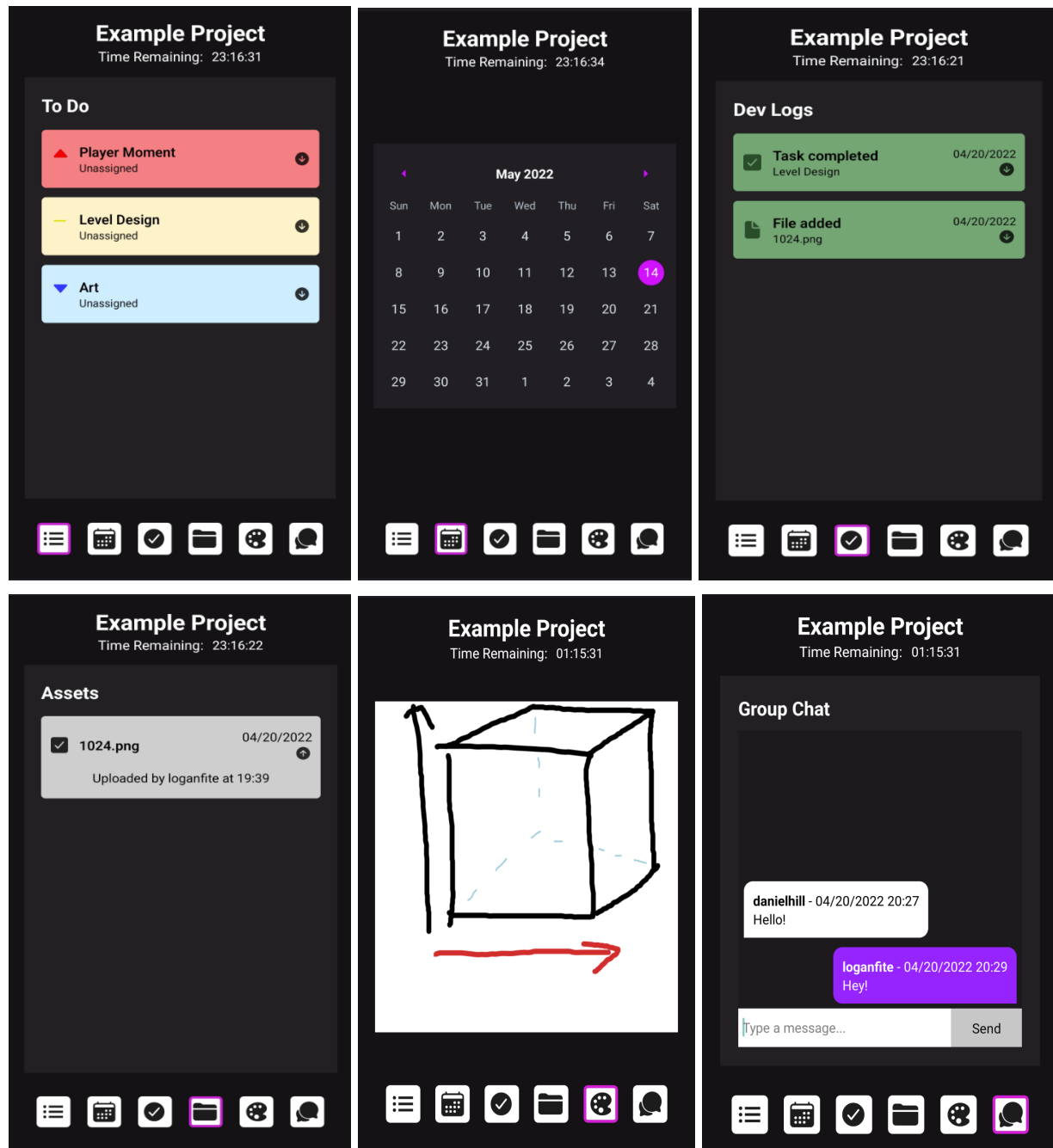


Figure 9.4 - Project Dashboard - All Components (Mobile)

The project dashboard is where the main user functionalities take place. The user is able to interact with various modules that their team members can see as well. These interactions allow for the overall collaboration functionality of the project dashboard.

On the web application, the calendar module inside the project dashboard allows the user to add events on certain dates that will add a vibrant marker for other team members to see, so that they will be informed about the event. This will help keep users on the same page about upcoming and previous events and deadlines. Update and delete functionalities for the calendar events are also completed and updated in real time to keep members aware of any changes.

The To Do List module allows users to create tasks of varying priorities. They can assign the tasks to another team member, or they can take a task for themselves if no one else is working on it. The module keeps everyone up to date on what is being worked on by each team member and the level of importance of each item. Tasks may be edited, reassigned, deleted, or marked completed on the web application. All actions update in real time for other users viewing the dashboard.

The development log module keeps a log of all completed tasks and changes to the project. This makes sure everyone in the team knows when changes are made. There could be dozens of tasks being worked on at any given point, and it is crucial that every member knows when any of these changes are made, as it can affect their workflow and workload. Every time a task is completed or a file is uploaded, a development log is automatically generated and displayed to project members. All actions update in real time for other users viewing the dashboard. All actions update in real time for other users viewing the dashboard.

The file sharing module allows team members to upload assets to a server for the other members to download. This allows artists or musicians to upload models or audio files for the developers to download and implement into code. The user who uploaded the file or the project leader may delete the file from the assets folder. All actions update in real time for other users viewing the dashboard. For native users, the assets list is read-only.

The team chat module keeps all members in contact with each other in real time. It allows every user to send messages to and read messages from every other user in the project. This allows for a constant stream of communication so that no members are ever lost.

The whiteboard module allows users to collaborate together on a real time virtual whiteboard. The canvas is shared between users in a project and lets each of them draw on it at the same time. This allows for collaborating on prototyping art, designing components, and brainstorming. The native view is read only and updates only upon thumbnail updates.

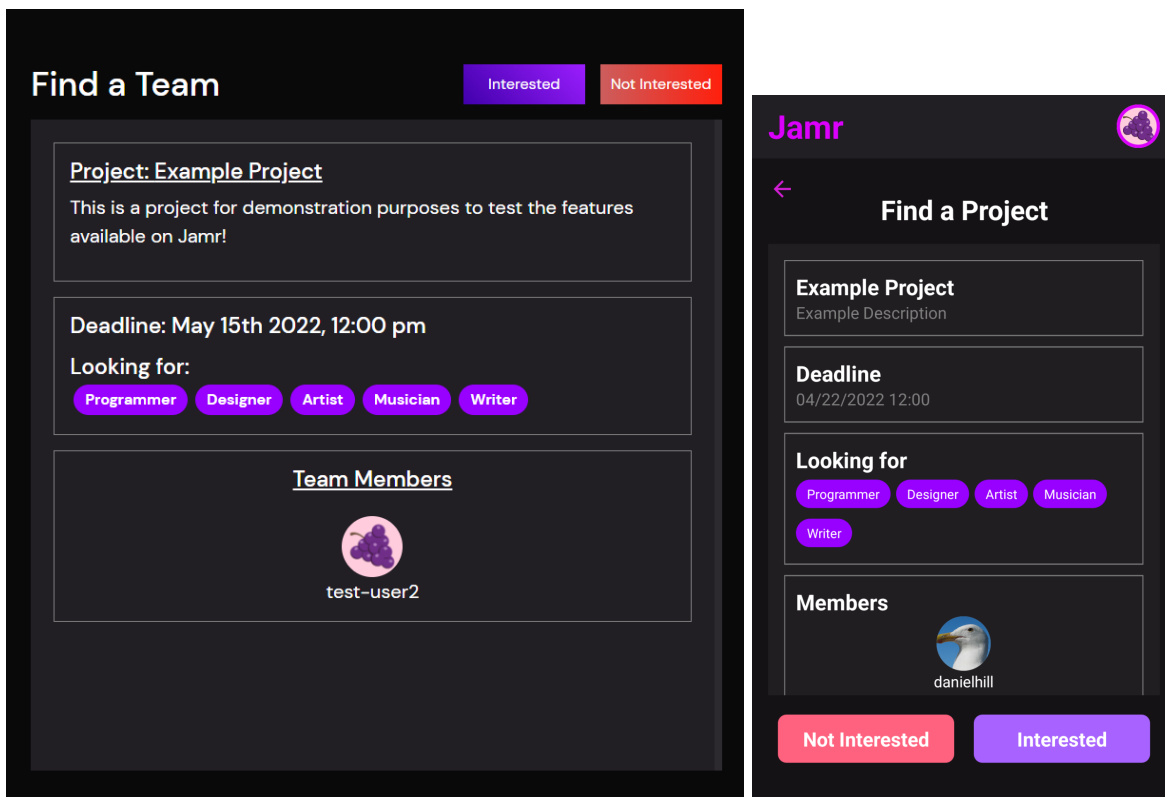
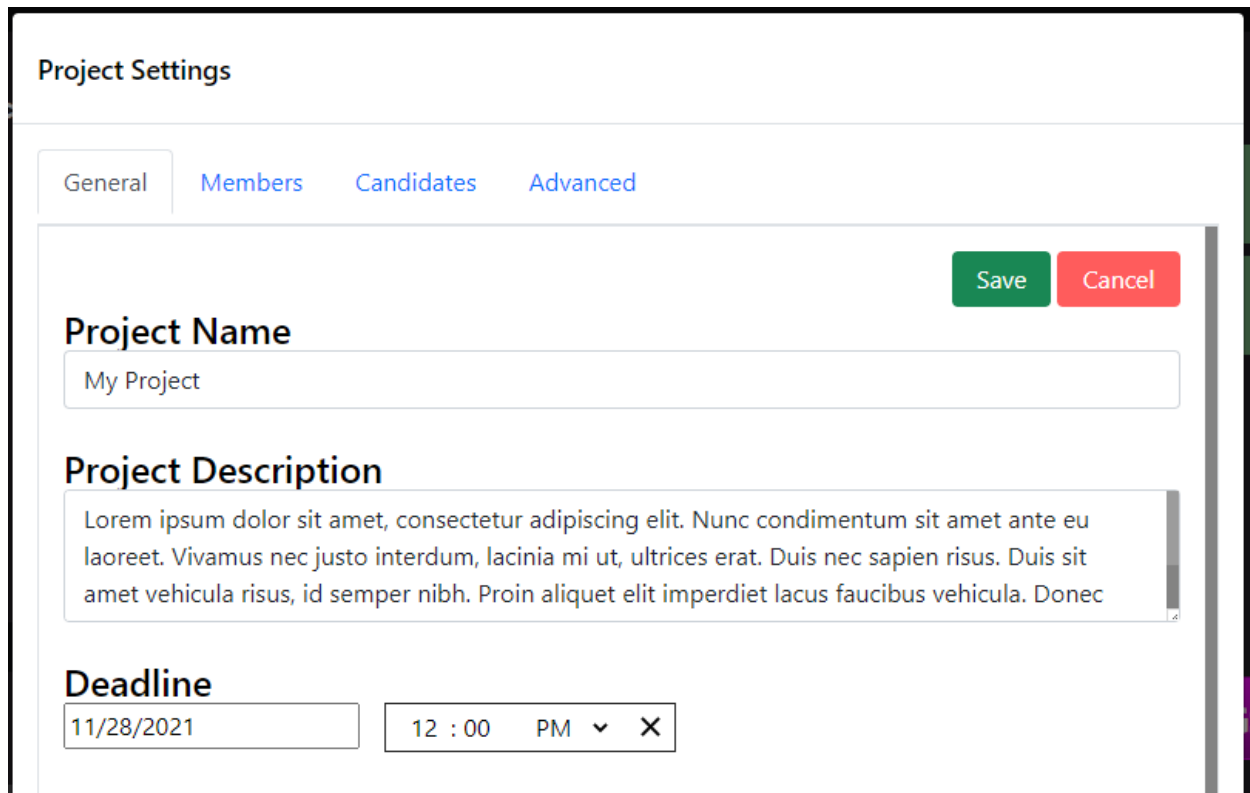


Figure 9.5 - Matching Page - Web & Mobile Views

When a user is searching for a project, the page above will be shown. The user can either select “interested” or “not interested”, if the user clicks on “not interested” the project will be removed from the list of projects to be rendered and a new project will be shown if available. If the user clicks on “interested” That user is added to a list of candidates that is accessed by the project creator, where they can decide whether to send the candidate an invite to the project or reject the request.



**Project Settings**

General Members Candidates Advanced

Save Cancel

**Project Name**

My Project

**Project Description**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc condimentum sit amet ante eu laoreet. Vivamus nec justo interdum, lacinia mi ut, ultrices erat. Duis nec sapien risus. Duis sit amet vehicula risus, id semper nibh. Proin aliquet elit imperdiet lacus faucibus vehicula. Donec

**Deadline**

11/28/2021 12 : 00 PM X

Figure 9.6 - Project Management

The project creator has access to the project settings. These settings allow the leader to change the basic information of the project, such as the name, description, and deadline.

In the candidates tab, all users that have selected “interested” on the project will be rendered. The project creator can then click the add member button (green) to give the candidate access to the project and make them a member of the team, while also removing them from the candidate list. Or, the project creator can click the reject candidate button (red) removing the candidate from the candidates list shown in the project settings.

After this, the leader can manage and remove accepted members through the members tab.

The advanced tab holds the option to delete the project permanently, removing all users from the project in doing so.

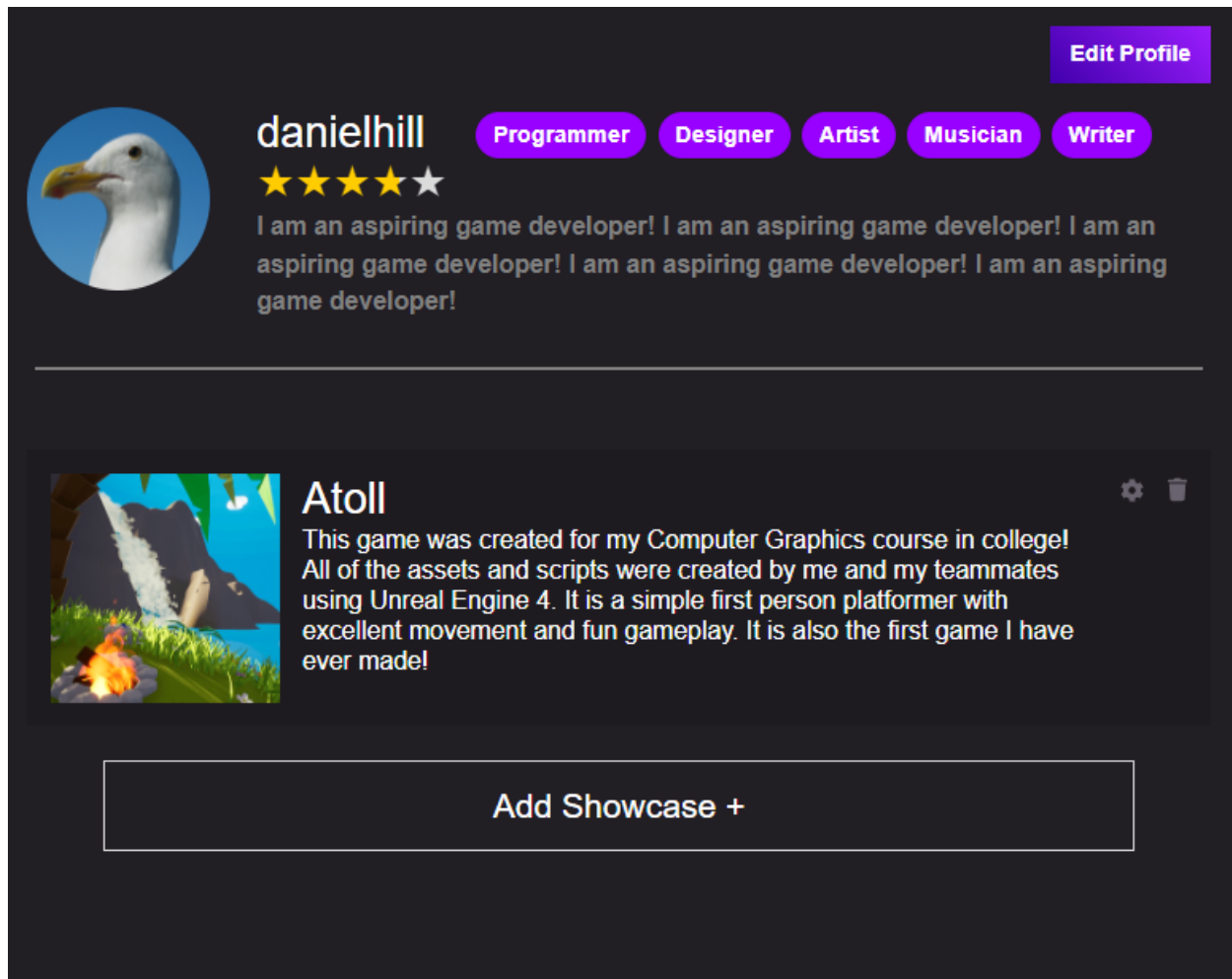


Figure 9.7 - Profile Page

The profile page allows users to customize and showcase information from their past projects. All profiles have permanent links that can be accessed by all users. It displays the information that the user had input at registration, such as their itch.io name, their description, and their tags. The users are able to add showcases of their itch.io content to their profile and write a description about what the project was. Other users are able to click on this and view the actual project on itch.io's website. There is also a star based rating system shown on user's profiles. If a user is in a project with another user, they are able to rate each other, and this information is used by the matching functionality to recommend better prospects first.



## 10. Summary and Future Work

The team succeeded in completing and delivering functional prototypes for the web and mobile platforms. All planned functionalities were completed and tested within the given timeframe. Future work includes handling bugs from user submitted reports to maintain the product and to further test the application for greater scalability and security before releasing the application to be publicly tested by the game jam community.

The web application is currently hosted on <https://jamrdemo.herokuapp.com/> and may be tested internally with the following site password:

**J@mr-WTcs43912+LJD\_2022sp!**

The mobile applications may only be downloaded through direct invitation to internal Android and iOS testing groups managed by Logan Fite. Requests for invitation may be made through email.

The following repositories were used to complete the applications for this project:

**Web Application:** <https://github.com/loganfite/Jamr>

**Native Application:** <https://github.com/loganfite/Jamr-Native>

Due to the project configurations containing sensitive connection strings and signing secrets, both repositories are set to private for safety. Requests for invitation to the repositories may also be made by email to Logan Fite.

## 11. References

itch.io. (n.d.). *Itch.io API*. itch.io. Retrieved 2021, from <https://itch.io/docs/api/overview>.

Facebook Inc. (n.d.). *Create a new react app*. React. Retrieved 2021, from <https://reactjs.org/docs/create-a-new-react-app.html>.

Facebook Inc. (n.d.). *Introducing hooks*. React. Retrieved 2021, from <https://reactjs.org/docs/hooks-intro.html>.

The Axios Project. (n.d.). *Getting started*. Getting Started | Axios Docs. Retrieved 2021, from <https://axios-http.com/docs/intro>.

Mongoose. (n.d.). *Api docs*. Mongoose v6.0.13: API docs. Retrieved 2021, from <https://mongoosejs.com/docs/api.html>.

Amazon Web Services, Inc. (n.d.). *Getting Started with the AWS SDK for JavaScript*. Retrieved 2021, from <https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/getting-started.html>.

Amazon Web Services, Inc. (n.d.). *Creating and Using Amazon S3 Buckets*. Creating and Using Amazon S3 Buckets - AWS SDK for JavaScript. Retrieved 2021, from <https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/s3-example-creating-buckets.html#s3-example-creating-buckets-upload-file>.

React Hook Form. (n.d.). *API documentation*. Documentation | React Hook Form - Simple React forms validation. Retrieved 2021, from <https://react-hook-form.com/api>.

React Bootstrap. (n.d.). *Introduction*. React-Bootstrap Documentation. Retrieved 2021, from <https://react-bootstrap.github.io/getting-started/introduction>.

Socket.IO. (2022, April 1). *Introduction*. SocketIO RSS. Retrieved 2022, from <https://socket.io/docs/v4/>

Alfred Genkin on Jul 5, & Genkin, A. (2021, December 1). *Five methods for five-star ratings: CSS-tricks*. CSS. Retrieved 2022, from <https://css-tricks.com/five-methods-for-five-star-ratings/>

Create React App. (2022, January 12). *Deployment*. Create React App. Retrieved 2022, from <https://create-react-app.dev/docs/deployment/>

Lavrenov, A. (2020, November 23). *React-KONVA - declarative canvas components for react*. Konva.js - JavaScript 2d canvas library. Retrieved 2022, from <https://konvajs.org/docs/react/Intro.html>

Meta Platforms, Inc. (2022, April 11). *Setting up the development environment · REACT NATIVE*. React Native RSS. Retrieved 2022, from <https://reactnative.dev/docs/environment-setup>

Meta Platforms, Inc. (2022, March 30). *Introduction · REACT NATIVE*. React Native RSS. Retrieved April 20, 2022, from <https://reactnative.dev/docs/getting-started>

Meta Platforms, Inc. (2022, March 30). *Publishing to google play store · REACT native*. React Native RSS. Retrieved 2022, from <https://reactnative.dev/docs/signed-apk-android>

Raj, M. (2021, January 16). *Dark mode for react application using context API and Hooks*. Section. Retrieved 2022, from <https://www.section.io/engineering-education/dark-mode-for-react-app-using-context-api-and-hooks/>

React Navigation. (n.d.). *Getting started*. React navigation RSS. Retrieved 2022, from <https://reactnavigation.org/docs/getting-started>

Shahid. (2019). *Encrypt and decrypt data in Node.js*. CodeForGeek. Retrieved 2022, from <https://codeforgeek.com/encrypt-and-decrypt-data-in-node-js/>

## 12. Scrum Report

### 12.1 User Stories

Story ID	User Story	Estimation (Hr)	Priority	Status
1	Allow users to login with their itch.io account and generate a profile for them with their existing information, as well as logout from and delete their account.	23.75	1	Complete
2	Securely handle and authenticate all user actions through the use of JSON Web Tokens	34	1	Complete
3	Allow users to create a project entry with user generated content for describing and tagging the entry	11.5	2	Complete
4	Allow users to view projects with matching tags and manage invitations to join a project	19.25	2	Complete
5	Allow project creators to view users interested in their project or users with matching skills and decide whether or not to add them to the project as well as edit the existing project information	45	2	Complete
6	Give all users in a project access to a dashboard that contains collaboration and management modules	30	2	Complete
7	Create to-do list module for dashboard	24.25	2	Complete
8	Create calendar module for dashboard	29	2	Complete
9	Create file sharing module for dashboard	18.5	2	Complete
10	Create group messaging module for dashboard	11	2	Complete
11	Create dev log module for dashboard	4.5	2	Complete
12	Create a homepage with conditionally displayed action buttons to find a project, create project, find members, and manage invitations depending on project status	6.5	2	Complete

13	Allow users to customize and edit their profile and view other user's profiles	34	3	Complete
14	Add messaging and friends feature to allow communication between users	38	3	Complete
15	Create whiteboard module for dashboard	30.25	3	Complete
16	Make UI compliant with a variety of resolutions, browsers, and devices	14	3	Complete
17	Add accessibility settings such as light mode	18	3	Complete
18	Create native application login	15.25	3	Complete
19	Create messaging page for native application	10	3	Complete
20	Create home page with matching and invite/candidate management for native application	12.5	3	Complete
21	Create Find a Project matching page for native application	5	3	Complete
22	Create Find Members matching page for native application	5	3	Complete
23	Create dashboard page for native application	8	3	Complete
24	Create read-only to-do list component for dashboard screen on mobile	5	3	Complete
25	Create read-only calendar component for dashboard screen on mobile	6	3	Complete
26	Create read-only file sharing component for dashboard screen on mobile	5	3	Complete
27	Create group messaging component for dashboard screen on mobile	5	3	Complete
28	Create read-only development log component for dashboard screen on mobile	5	3	Complete
29	Create read-only whiteboard thumbnail module	5	3	Complete

	for dashboard screen on mobile			
--	--------------------------------	--	--	--