# *Software Engineering Software Requirements Specification (SRS) Document*

**Hooplaza**

**09/21/2023**

**Version 1.0**

**By: Logan Roberts, Ethan Mongelli, Kol Herget**

**[Honor Code]**

# Table of Contents

# 1. Introduction

## 1.1. Purpose

HooPlaza is a neighborhood social networking service. It is designed as a community hub for events, community projects, and clubs. The main objective of this app is to connect community members and ease car-dependency by allowing users to join in activities that are close-by, instead of far away from their neighborhood. HooPlaza is a Web Application that can be reached via the internet, on a web browser. It is designed to allow users to connect to an admin to create a new community and, as a moderator, appoint other users as moderators and keep the community organized. It allows users to join clubs and RSVP to events they may not have otherwise known about.

## 1.2. Document Conventions

The purpose of this Software Requirements Document (SRD) is to describe the requirements of Hooplaza from both a user experience and developer oriented points of view. User-oriented sections will detail the system from the user's perspective. These requirements include descriptions of the different types of users and the actions able to be performed by each. Developer requirements describe the system from a software developers perspective. These requirements include detailed descriptions for the architecture, function, performance, and other necessary requirements.

## 1.3. Definitions, Acronyms, and Abbreviations

| | |
|---|---|
| Java | A programming language originally developed by James Gosling at Sun Microsystems. This is the language that will be used to develop Hooplaza. |
| MySQL | Open-source relational database management system. |
| HTML | Hypertext Markup Language. This is the code that will be used to structure and design the web application and its content. |
| SpringBoot | An open-source Java-based framework used to create a micro Service. This will be used to create and run our application. |
| MVC | Model-View-Controller. This is the architectural pattern that will be used to implement our system. |
| Spring Web | Will be used to build our web application by using Spring MVC. This is one of the dependencies of our system. |
| Thymeleaf | A modern server-side Java template engine for our web environment. This is one of the dependencies of our system. |
| Intellij | An integrated development environment (IDE) for Java. This is where our system will be created. |
| API | Application Programming Interface. This will be used to implement a function within the software where the current date and time is displayed on the homepage. |
| Communities | The organizational units within the application. Users will be able to interact with each other within them. Communities should correspond to physical locations such as neighborhoods. |

## 1.4. Intended Audience

Describe which part of the SRS document is intended for which reader. Include a list of all stakeholders ofthe project, developers, project managers, and users for better clarity.]

Developers: Developers will benefit from sections 2.x, 3.x, 4.x, 5.x, and 6.x of the SRD. Every one of these sections contains data that will be beneficial to the development of Hooplaza.

Project Managers: Project managers will benefit from sections 2.x, 3.x, 5.[5-9], 6.x, as well as 7.x. These sections contain information that are beneficial for understanding the project without unnecessary detail related to the implementation.

Users: Users will benefit from sections, 2.x, 5.7, 5.8, and 7.x as these sections contain data relevant to the use of Hooplaza. They include information that might help a user better understand how to make full use of the platform.

## 1.5. Project Scope

The goal of the software is to provide a centralized platform for users to become connected to their local communities and become engaged in activities. User adoption will lead to a platform for expanded services that will benefit the business as well as the users.

Possible monetizable platform features include:

- Integrated booking of services such as food trucks, or local entertainers.
- Event planning help lines.
- Adding the ability to tip active event organizers, or other active community members.

## 1.6. Technology Challenges

Some of the technology challenges include having to learn how to use and manage databases for the persistent storage of complex, linked data. Combined with having to learn javascript in order to use the Google Maps API which none of us have used before is incredibly challenging.

## 1.7. References

[Mention books, articles, web sites, worksheets, people who are sources of information about the application domain, etc. Use proper and complete reference notation. Give links to documents as appropriate. You should use the APA Documentation model (Alred, 2003, p. 144).]

Google. (n.d.). Google Maps Platform Documentation. Google Maps Platform. https://developers.google.com/maps/documentation

# 2. General Description

## 2.1. Product Perspective

HooPlaza was conceptualized by a preemptive step away from car dependency and to bring local communities together. The idea is driven by the condition to improve everyday local socialization and interactions.

## 2.2. Product Features

HooPlaza will generate location-based communities at a users request. It will allow moderators to add members to their community, appoint others to moderators. If a user tries to create a community within an already established community location, the app will notify moderators of the established community that someone would like to be a part of it. Members will be able to create posts to be seen by everyone in their community. Events with locations can be put on the map for users to view.

## 2.3. User Class and Characteristics

Admin: Runs operations such as creating communities within a location. Directs users to their community as needed. Appoints/removes moderators.
Moderator: Organizes their respective community. Adds/kicks users from the community
Basic user: Creates posts. RSVPs to events. Requests create community

## 2.4. Operating Environment

This application is designed to work on the web across PC and Laptop devices.

## 2.5. Constraints

N/A

## 2.6. Assumptions and Dependencies

Assumptions: the API will allow for a visual map of a community area.

# 3. Functional Requirements

## 3.1. Primary

- FR0: Users should be able to view and join communities. The user should only be able to interact in communities they have joined.
- FR1: Users should be able to make, and read posts in communities they have joined.
- FR2: Users should be able to request for a new community to be made if one for the location they wish is not available.
- FR3: Each community should have moderators which have the power to remove users from a group, as well as add other moderators.
- FR4: There  should be site admins that have the ability to approve community creation requests, and ban users from the site.

## 3.2. Secondary

- Duplicate communities should not exist

- Authorization scheme such that no users will be able to perform actions they are not supposed to. Authentication should be done through passwords.

# 4. Technical Requirements

## 4.1 Operating System and Compatibility

The system should be compatible with any system that is able to interact with traditional web pages.

## 4.2 Interface Requirements

### 4.2.1 User Interfaces

There will be a button to make posts. There will be three feed sections: Posts, Events, and Announcements, and Announcements will be viewable from any screen of the app where the user is in a community. There will be a hamburger button that holds all communities the user is a part of, and the user can switch between these communities.

### 4.2.2 Hardware Interfaces

The application will run on any device able to access the internet and interact with and display webpages. Examples of such devices include smart phones, tablets, laptops, desktop computers and more.

### 4.2.3 Communications Interfaces

It must be able to connect to the internet as well as some databases for persistent information storage. HTTP will be used for any internet based communication, and the Google maps API should be accessible for location lookup and embedding.

### 4.2.4 Software Interfaces

The frontend will utilize JavaScript and Springboot Thymeleaf. The JPA (Java Persistence API) will be used for the backend database functionality. Springboot with java will be used to connect the frontend and the backend.

# 5. Non-Functional Requirements

## 5.1. Performance Requirements

- NFR0: The local version of the database should use less than 100MB of memory
- NFR1: The system should use less than 200MB of memory including the database.
- NFR2: Posts should appear to other users within one minute of a given user posting it.

## 5.2. Safety Requirements

- NFR3: All user input should be sanitized to prevent users from performing SQL injection attacks, such that malicious actors do not have access to the locations and times of events within communities.

## 5.3. Security Requirements

- NFR4: Only users within a comunity should be able to access or make posts withing a community.
- NFR5: Users should be required to authenticate before they can access the system.

## 5.4. Software Quality Attributes

### 5.4.1. Availability

HooPlaza must be usable with multiple communities filled with multiple users & moderators

### 5.4.2. Correctness

N/A

### 5.4.3. Maintainability

Will be maintained by developers and admins

### 5.4.4. Reusability

N/A

### 5.4.5. Portability

As a web application, HooPlaza will be usable in any internet browser

## 5.5. Process Requirements

### 5.5.1. Development Process Used

Scrum

### 5.5.2. Time Constraints

Must be finished by end of November. All developers have different availabilities, so planning meetings may be difficult

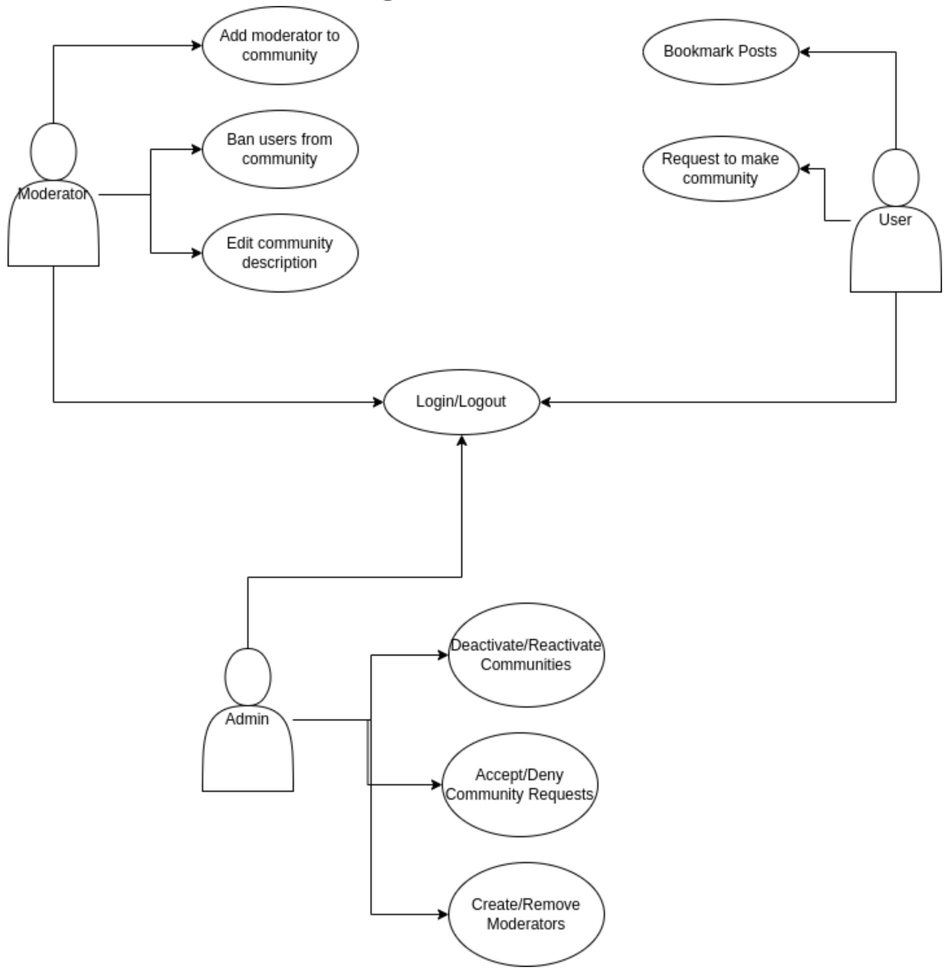
### 5.5.3. Cost and Delivery Date

$0. Estimated delivery date: December 5

## 5.6. Other Requirements

TBD

## 5.7. Use-Case Model Diagram


Add moderator to community
Ban users from community
Edit community description
Moderator
Bookmark Posts
Request to make community
User
Login/Logout
Admin
Deactivate/Reactivate Communities
Accept/Deny Community Requests
Create/Remove Moderators

## 5.8. Use-Case Model Descriptions

### 5.8.1. Actor: Admin (Logan Roberts)

– **Use-Case Name**: Create Community
– **Use-Case Name**: Delete Community

### 5.8.2. Actor: Moderator (Ethan Mongelli)

- **Use-Case Name**: Make Announcement
- **Use-Case Name**: Ban User

### 5.8.3. Actor: Basic User (Kol Herget)

- **Use-Case Name**: RSVP Event
- **Use-Case Name**: Request to Create Community

## 5.9. Use-Case Model Scenarios

### 5.9.1. Actor: Admin (Logan)

- **Use-Case Name**: Create Community
    - **Initial Assumption**: There may or may not be an existing community at the requested location.
    - **Normal**: Location information is stored in a community on a MySQL server. Community is made and requester is given Moderator role in that community

- **What Can Go Wrong:** Community is already created in the requested location
    - **Other Activities**: N/A
    - **System State on Completion**: New community is backed up, as well as requester's moderator status.
- **Use-Case Name**: Delete Community
    - **Initial Assumption**: The community exists.
    - **Normal**: Community information is deleted, and all community members are removed from it. The community is deleted.
    - **What Can Go Wrong**: The community does not exist.
    - **Other Activities**: N/A
    - **System State on Completion**: Updates are backed up

**5.9.2. Actor: Moderator (Ethan)**

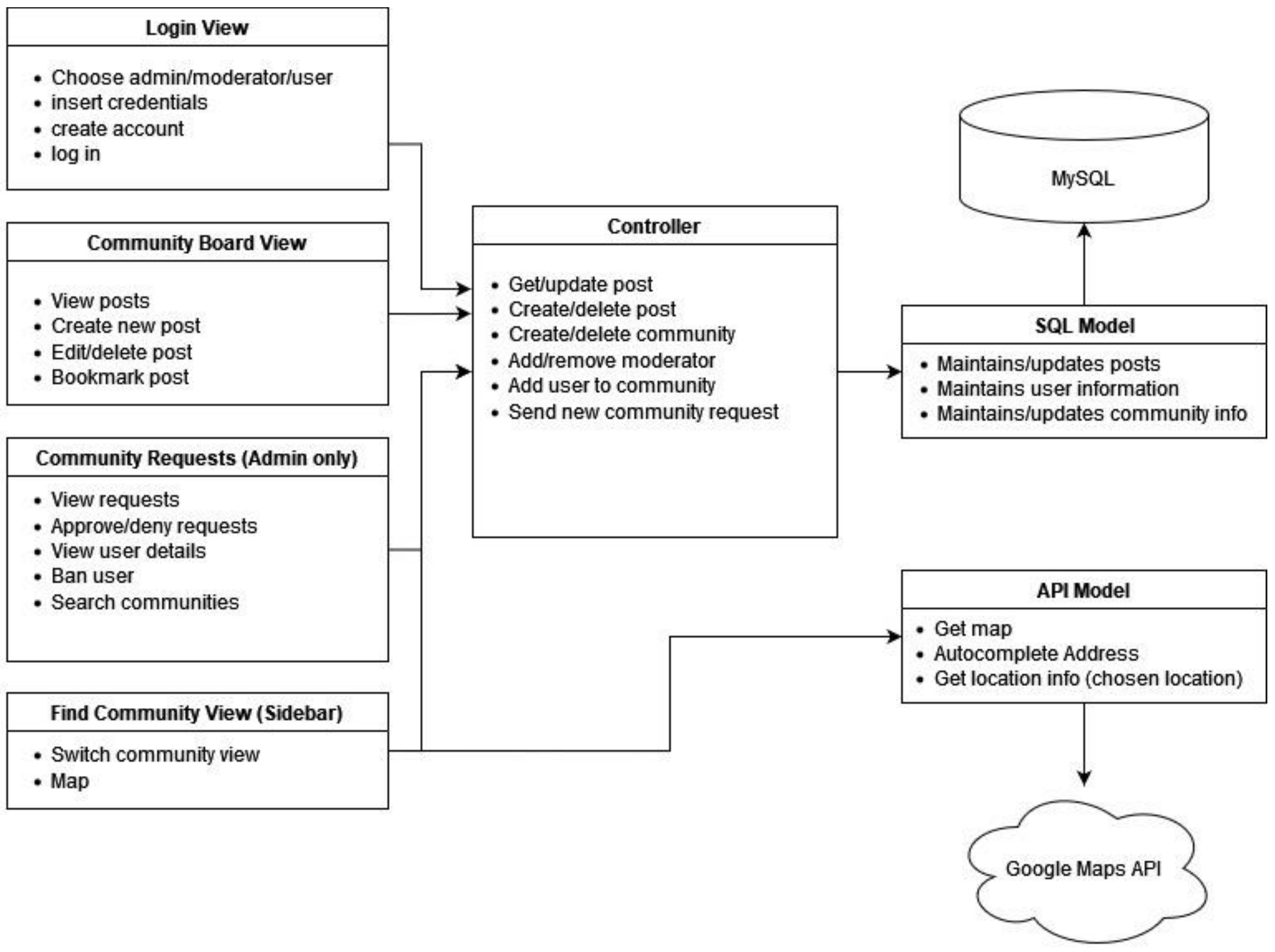
- **Use-Case Name**: Make Announcement
    - **Initial Assumption**: Announcement page exists
    - **Normal**: Post is made in announcements section of the app
    - **What Can Go Wrong**: Announcement section is unavailable/unviewable
    - **Other Activities**: N/A
    - **System State on Completion**: Announcement is in its section
- **Use-Case Name**: Ban User
    - **Initial Assumption**: User is in the community
    - **Normal**: User is removed from community and blacklisted from it on the server
    - **What Can Go Wrong**: User not in the community – cannot be banned
    - **Other Activities**: N/A
    - **System State on Completion**: Blacklist updated and backed up

**5.9.3. Actor: Basic User (Kol)**

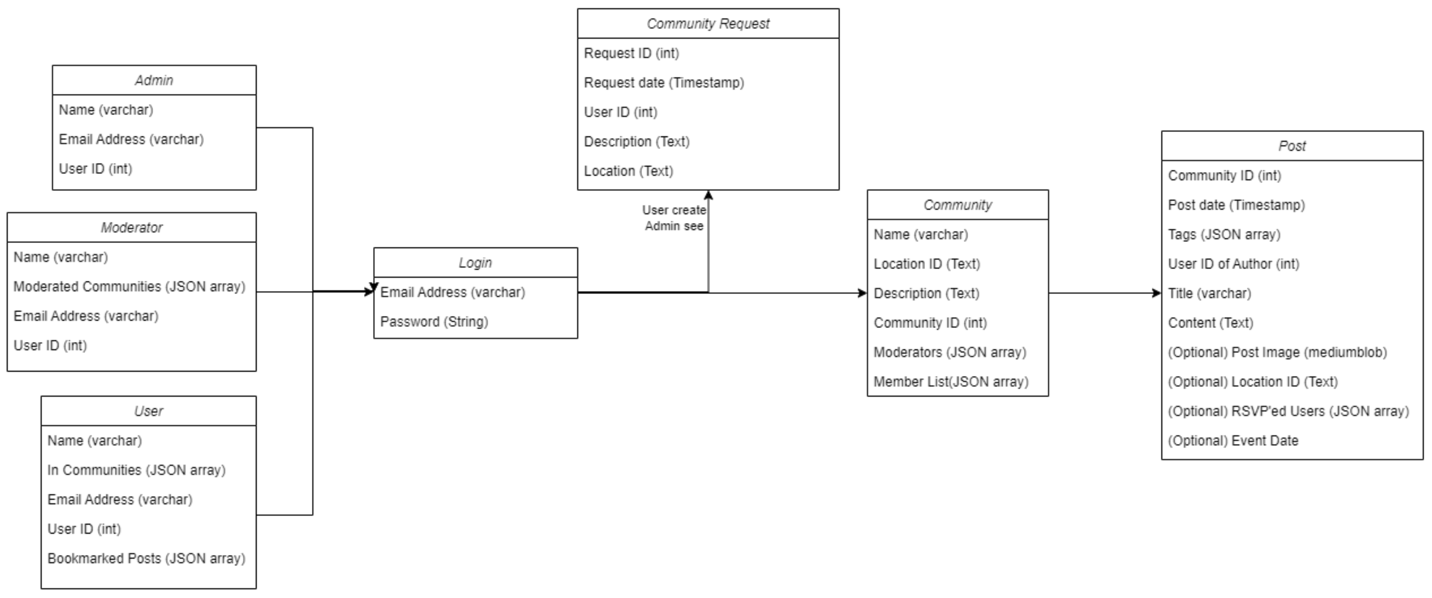
- **Use-Case Name**: RSVP Event
    - **Initial Assumption**: Button for 'yes' and 'no'
    - **Normal**: Button is clicked, attendance information is saved
    - **What Can Go Wrong**: RSVP List at capacity
    - **Other Activities**: N/A
    - **System State on Completion**: User is RSVP'd to the event
- **Use-Case Name**: Request to Create Community
    - **Initial Assumption**: Community not already created in requested location
    - **Normal**: Request is sent to admin, admin does use-case "Create Community"
    - **What Can Go Wrong**: request unable to send – blocked servers. Community already established
    - **Other Activities**: N/A
    - **System State on Completion**: Admin receives request

# 6. Design Documents

## 6.1. Software Architecture

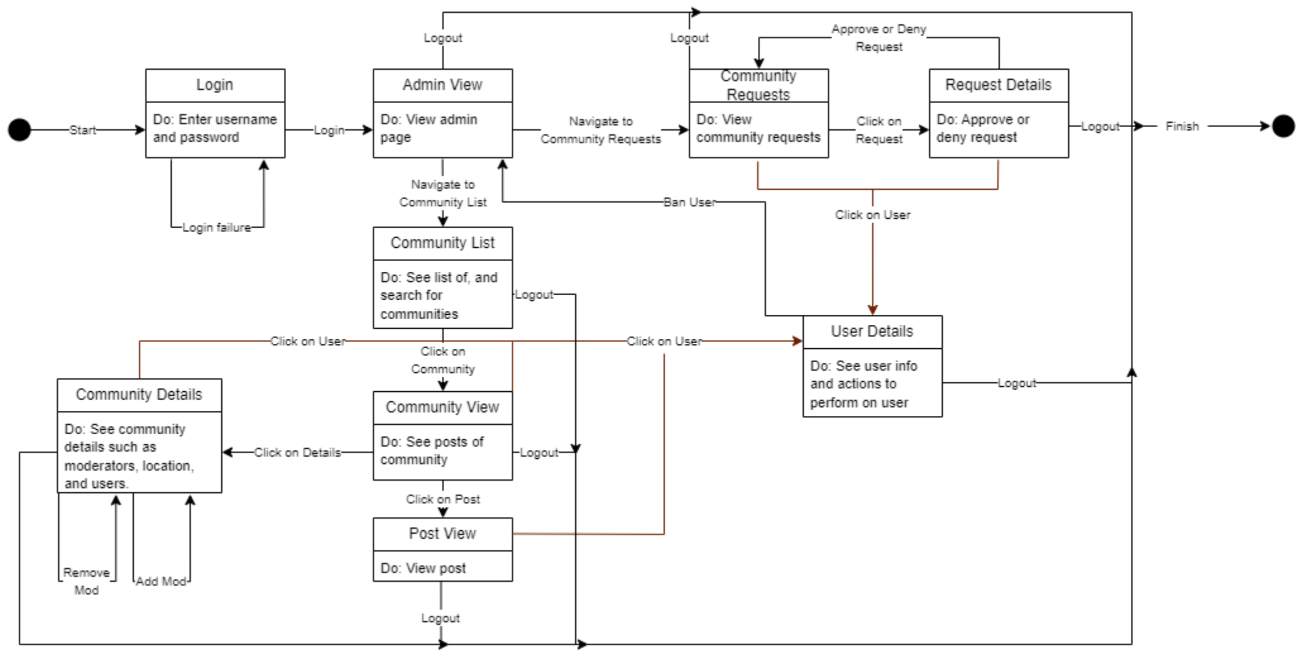


## 6.2. High-Level Database Schema

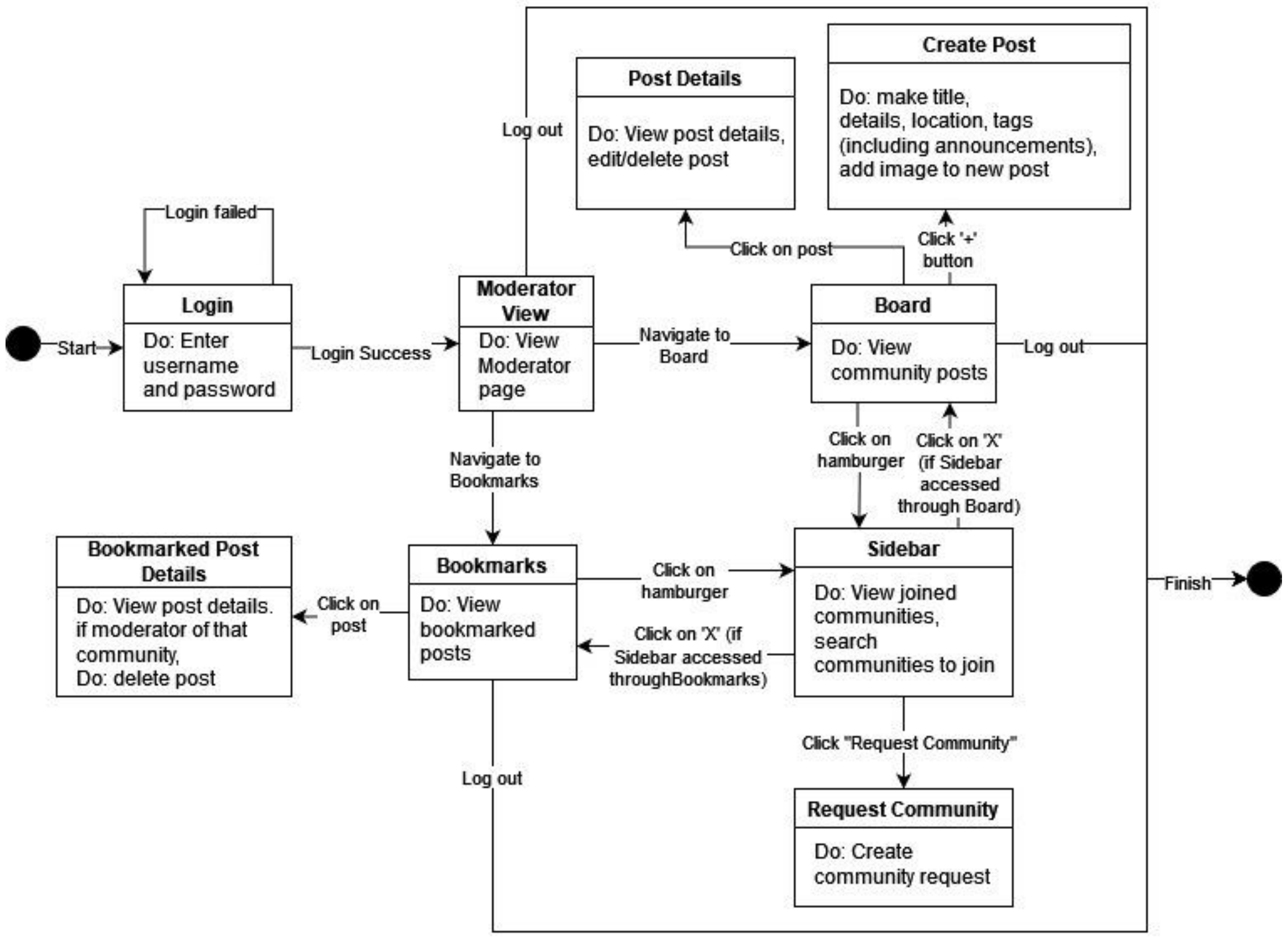


## 6.3. Software Design

### 6.3.1. State Machine Diagram: Admin (Logan Roberts)

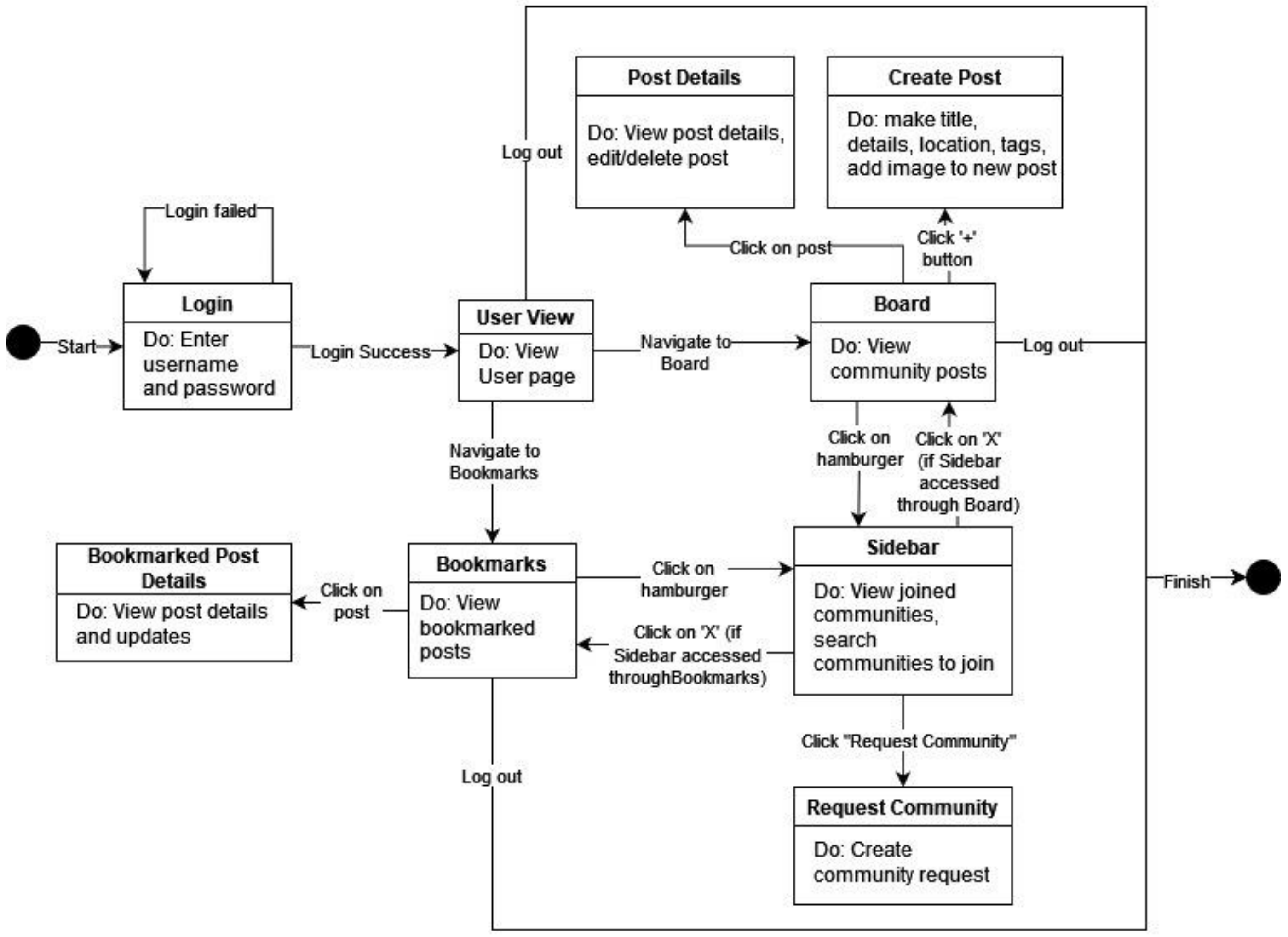


### 6.3.2. State Machine Diagram: Moderator (Ethan Mongelli)

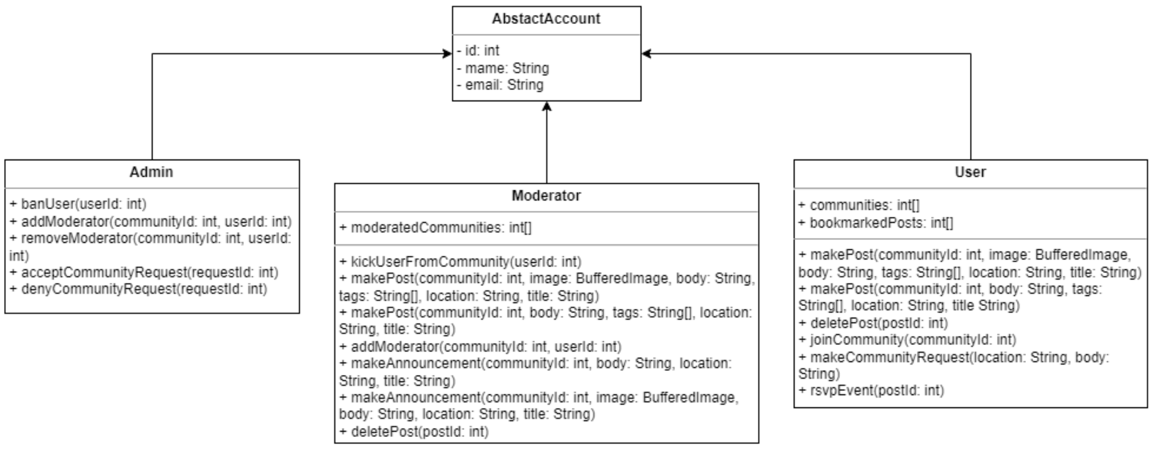


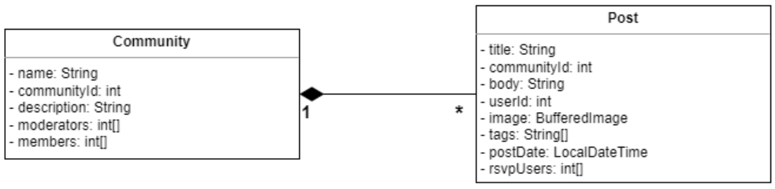
Start
Login
Do: Enter username and password
Login failed
Login Success
Moderator View
Do: View Moderator page
Log out
Navigate to Board
Navigate to Bookmarks
Post Details
Do: View post details, edit/delete post
Create Post
Do: make title, details, location, tags (including announcements), add image to new post
Click on post
Click '+' button
Board
Do: View community posts
Log out
Click on hamburger
Click on 'X' (if Sidebar accessed through Board)
Sidebar
Do: View joined communities, search communities to join
Finish
Bookmarked Post Details
Do: View post details. if moderator of that community,
Do: delete post
Click on post
Bookmarks
Do: View bookmarked posts
Click on hamburger
Click on 'X' (if Sidebar accessed throughBookmarks)
Log out
Click "Request Community"
Request Community
Do: Create community request

### 6.3.3. State Machine Diagram: User (Kol Herget)


Post Details
Do: View post details, edit/delete post
Create Post
Do: make title, details, location, tags, add image to new post
Log out
Login failed
Click on post
Click '+' button
Login
Do: Enter username and password
Start
Login Success
User View
Do: View User page
Navigate to Board
Board
Do: View community posts
Log out
Navigate to Bookmarks
Click on hamburger
Click on 'X' (if Sidebar accessed through Board)
Bookmarked Post Details
Do: View post details and updates
Click on post
Bookmarks
Do: View bookmarked posts
Click on hamburger
Click on 'X' (if Sidebar accessed throughBookmarks)
Sidebar
Do: View joined communities, search communities to join
Finish
Click "Request Community"
Log out
Request Community
Do: Create community request

## 6.4. UML Class Diagram


AbstactAccount
- id: int
- mame: String
- email: String
Admin
+ banUser(userId: int)
+ addModerator(communityId: int, userId: int)
+ removeModerator(communityId: int, userId: int)
+ acceptCommunityRequest(requestId: int)
+ denyCommunityRequest(requestId: int)
Moderator
+ moderatedCommunities: int[]
+ kickUserFromCommunity(userId: int)
+ makePost(communityId: int, image: BufferedImage, body: String, tags: String[], location: String, title: String)
+ makePost(communityId: int, body: String, tags: String[], location: String, title: String)
+ addModerator(communityId: int, userId: int)
+ makeAnnouncement(communityId: int, body: String, location: String, title: String)
+ makeAnnouncement(communityId: int, image: BufferedImage, body: String, location: String, title: String)
+ deletePost(postId: int)
User
+ communities: int[]
+ bookmarkedPosts: int[]
+ makePost(communityId: int, image: BufferedImage, body: String, tags: String[], location: String, title: String)
+ makePost(communityId: int, body: String, tags: String[], location: String, title String)
+ deletePost(postId: int)
+ joinCommunity(communityId: int)
+ makeCommunityRequest(location: String, body: String)
+ rsvpEvent(postId: int)

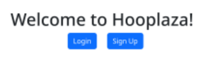


# 7. Scenario

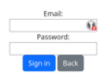
## 7.1. Brief Written Scenario with Screenshots

### 7.1.1 Admin (Scenario and development by Logan Roberts):

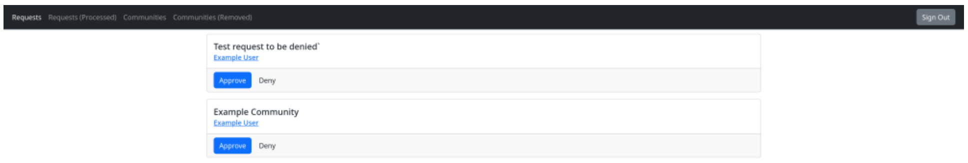
The admin scenario will walk through the actions an admin user can take starting from the situation where there is are 2 community requests.
Whenever a user initially navigates to the page they will be greeted with the home page:


Welcome to Hooplaza!

Admins will already have accounts and will go to the login page:



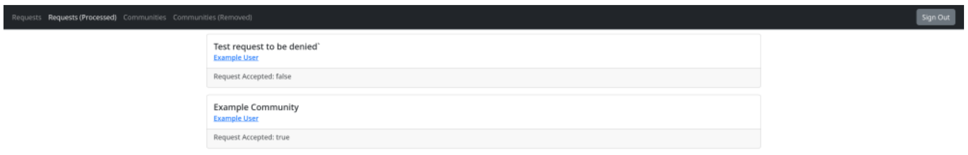
After the admin logs in this is the page they see:



Test request to be denied`
Example User

Example Community
Example User

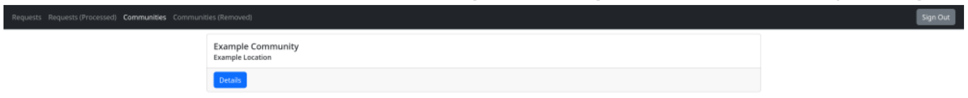
From here the admin can accept or deny requests and they will be removed from the requests page. They will still be viewale under the “Requests (Processed)” page. Which will look like so after some requests have been processed:



Test request to be denied`
Example User
Request Accepted: false

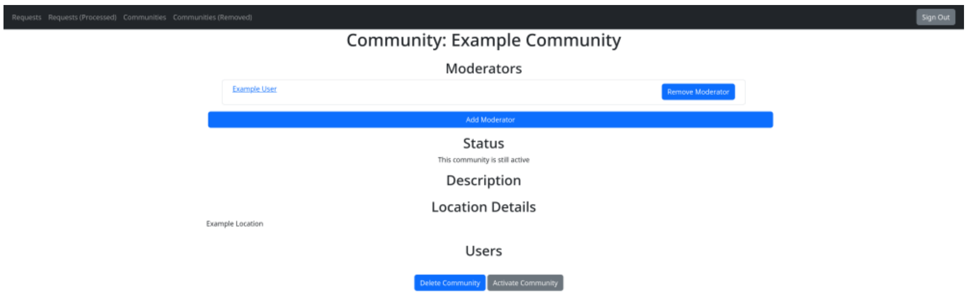
Example Community
Example User
Request Accepted: true

If an admin wants to view and manage existing communities, they can go to the communities page.



Example Community
Example Location

From here they can view community details where they can see all moderators/users as well as add more moderators:



Community: Example Community

Moderators

Example User

Status

This community is still active

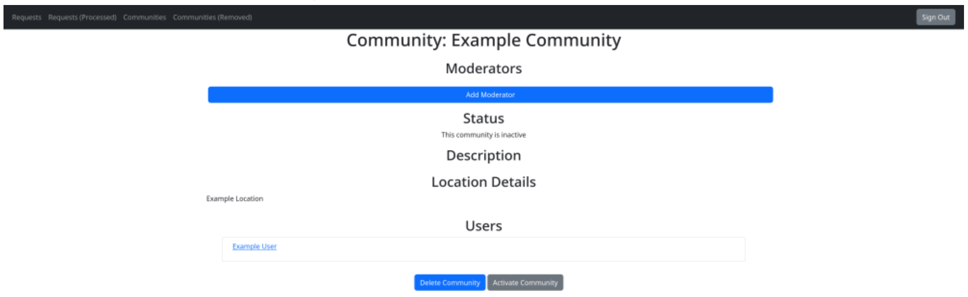
Description

Location Details

Example Location

Users

The description is able to be editited by the moderators. If the admin were to remove the current moderator and make the community inactive it would look like this:



Community: Example Community

Moderators

Status

This community is inactive

Description

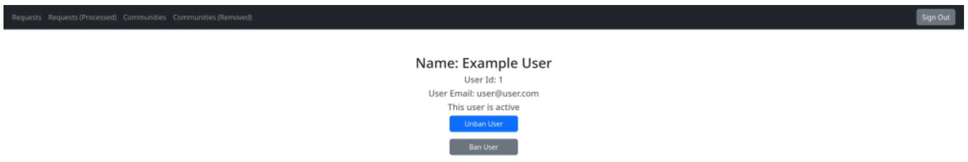
Location Details

Example Location

Users

Example User

Admins can also click on users to see details about them. On this page the admin can ban or unban the user.



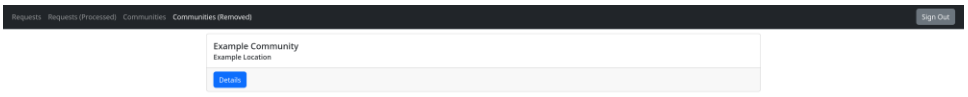
Name: Example User

User Id: 1

User Email: user@user.com

This user is active

The admin can also see removed communities by going to the “Communities (Removed)” tab.



Example Community

Example Location

From here the actions that can be performed are the same. Now however the community is not visible under the Communities tab. When the admin is done they can click the “Sign Out” button and they are returned to the home screen.