Rooted

Design Document

September 6, 2019

Team 2

Julien San Diego, asandieg@purdue.edu

Cyrus San Santiago, santiac@purdue.edu

Ken Sodetz, sodetz@purdue.edu

Ben Malone, malone74@purdue.edu

Abigehl Slattery, aslatter@purdue.edu

Purpose

Though it is very common for college students to find their home within groups or communities while they attend school, they often completely fall out of the loop and lose relevance once they graduate. Simultaneously, new members who join miss out on the opportunity to network and seek wisdom from the decade-long history of the group's alumni.

Our app, 'Rooted', seeks to provide a networking platform where communities of people can keep a clean and well-documented history of all of their past members. The page will be designed to follow the structure of a tree, flowing with branches down the page that contain pictures of all alumni and links to their contact information. Such a platform provides two major benefits: the preservation of stories and accomplishments of all former members, and the creation of an expansive network for any member throughout time to reach out to for opportunities.

Design Outline

High Level Overview

Our application will be a web app that allows users to post links to different kinds of media, and post text which other users will critique. The application will run on a client server model, with the server simultaneously handling information from a large number of clients. The client will send a request to the server, which will then send a query to our database in order to access the correct data. The database will return the data to the server, and the server will send a response back to the client.

- Client
    - Provides user with an interface
    - Sends request to the server
    - Receives the response from the server, and parses the response to display the correct information on the interface
- Server
    - Receives and handles request from the client
    - Server parses request and forwards a query to the database
    - Receives data from the database, and forms a response to send to the client
    - Sends the response to the client
- Database
    - Stores all user info, along with communities, messages, notifications, submissions, feedback, comments, etc
    - Returns the correct information to the server

Design Issues

Functional Issue

1. What information do we need for a user to sign up for an account?
    - Option 1: Username and password only
    - Option 2: Username, password, email address
    - Option 3: Username, password, email address, security question

    Choice: Option 3

    Justification: Username and password are necessary for accounts in order to create a layer of protection and identification for each user. Obtaining the user's email address will assist in implementing verification for the user accounts. Furthermore, it can be used for notifications, news updates on the application and password reset information. A security question we feel would add a good layer of security for our users and would not be very difficult to add. So, we added it to the information needed. Overall, option 3 should provide the necessary information for signing up for a users.

2. What kind of personal messaging system do we want to use?
    - Option 1: Instant messaging
    - Option 2: Non real time messaging system (like emails or private messages)

    Choice: Option 2

    Justification: When thinking about how users will interact with the application, we don't believe users would be frequently chatting with other users. Thus, having an instant messaging feature would not be used as much. Also, our backend framework isn't realtime as well which makes implementing instant messaging very difficult. So, going with a non real time messaging system is ideal as we still need a messaging esque feature for users to interact with each other.

3. How many levels of administration should our application have?
    - Option 1: One level (general admin)
    - Option 2: Two levels (sitewide admin, group admin)
    - Option 3: Three levels (sitewide admin, group admin, group creator admin)
    - Option 4: Four levels (sitewide admin, group admin, group creator admin, group moderator)

    Choice: Option 4

    Justification: When thinking about the different permissions available in the application, especially when a big number is active in a community, we want to create an environment that is safe and where there is structure, thus we decided to bring in a site admin. Site admins are brought in for the purpose to ensure Rooted members are not causing issues and the environment is safe. If any member is found to be a threat, site admins can ban members. In addition to that, Rooted has different groups, and we wanted to ensure that there is a member who is in charge of ensuring that the correct information is displayed and only specific members are in the group. This issue brought in group creator admins. However, as groups get bigger, there are many different responsibilities. For example, a size of sororities grow as a new rush season begins. As more members join sororities, members also leave as they graduate. In such case, members who are in charge of the Rooted group page will need to pass on their permissions to other members who are staying. But not everyone can have the permission. Thus, we decided to create a group admin and group moderator - which embodies more specific permissions.

4. Should non moderators be able to add members to a group?
    - Option 1: Yes
    - Option 2: No

    Choice: Option 2

Justification: In groups, only members involved in a group is allowed to participate. In this case, if a group is big, there should be a few members in a group who can control who are added in a group. We decided that if we gave all members the capability to add members, any member can be added, and sometimes may go unnoticed if the group is so big as not everyone may not know that a member is actually not a part of the group. Therefore, we decided that only moderator of a group can add members to a group.

5. How does a member join a group?
    - Option 1: A member can join any group
    - Option 2: A member must request to join
    - Option 3: A member must get an invite
    - Option 4: A member must either request or get an invite to join

    Choice: Option 4

    Justification: When thinking about groups in the application, we wanted to ensure that only members involved in a group is allowed to be a part of the group. We first decided that a member can join invite, which would allow selected members to join. However, an issue arises that if invites are sent for a specific batch, and if a user doesn't create a Rooted account until after invites are sent, the user may not be able to join. The user may ask to get an invite, but sometimes it can take a while or may be forgotten. We then decided that a user should be able to have another option, which is to request to join. Group admins and group moderators will then be able to see who requested to join and has the permission to reject or approve a users request.

Non Functional Issue

1. Which frontend framework should we use?
    - Option 1: React
    - Option 2: Angular
    - Option 3: Vue

Choice: Option 2
Justification: We chose Angular as the majority of our team has some kind of experience with Angular which will make more time for building the application and less time learning how to use the actual framework. Furthermore, it's in Javascript which lends well to our backend of node/express which also uses Javascript. Lastly, it's component architecture will allow us to save time during development through code reuse and help keep our code organized.

2. Which backend framework should we use?
    - Option 1: Django
    - Option 2: Node
    - Option 3: Spring

Choice: Option 2
Justification: We chose Node for the backend as it is extremely lightweight and like our frontend framework uses Javascript which will help ease the learning curve for the developers on our team not well versed in the backend as most know Javascript and allows for the teams to be cross functional. Furthermore, Node has a lot of modules accessible through it's package manager that we can leverage to decrease development time.

3. Which database should we use?
    - Option 1: MongoDB
    - Option 2: MySQL

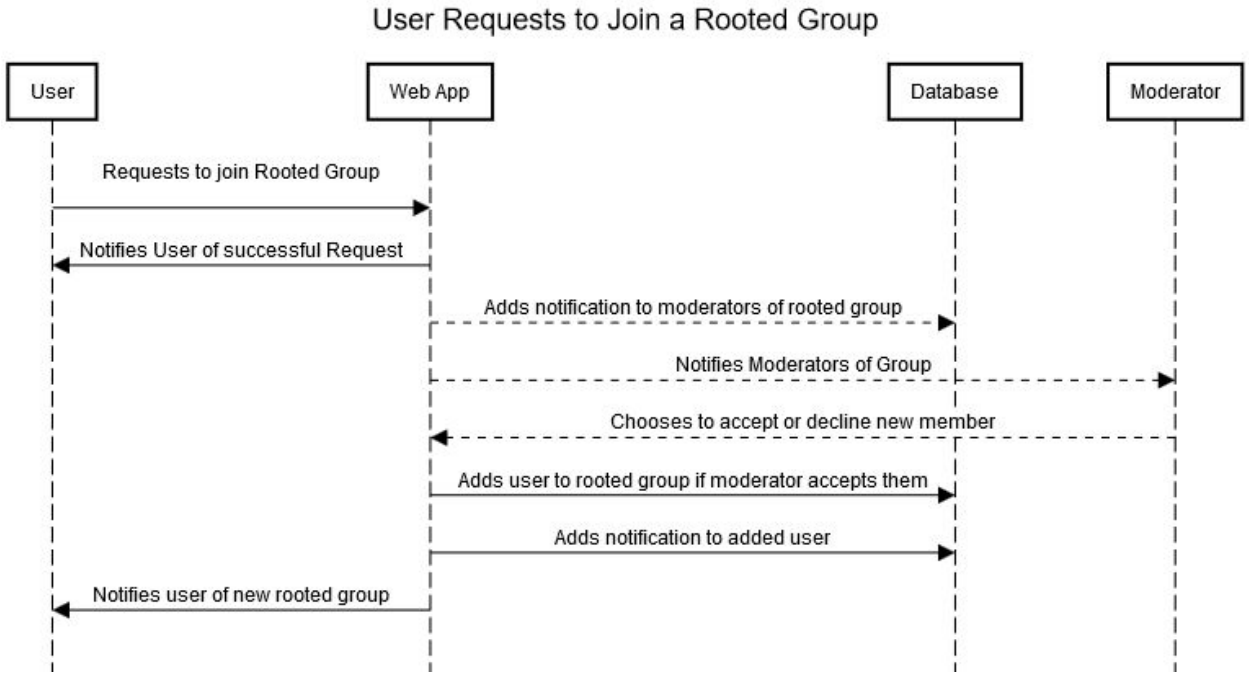
Choice: Option 1
Justification: Since we are going to have submissions for different mediums with a variety of different types of attributes for each, having a schemaless database such as MongoDB is very useful. Also, MongoDB is very easy to scale and is frequently used with the frontend framework we have chosen in the form of the MERN stack. Furthermore, many of our members have experience using this database and thus it would require the least amount of setup time.

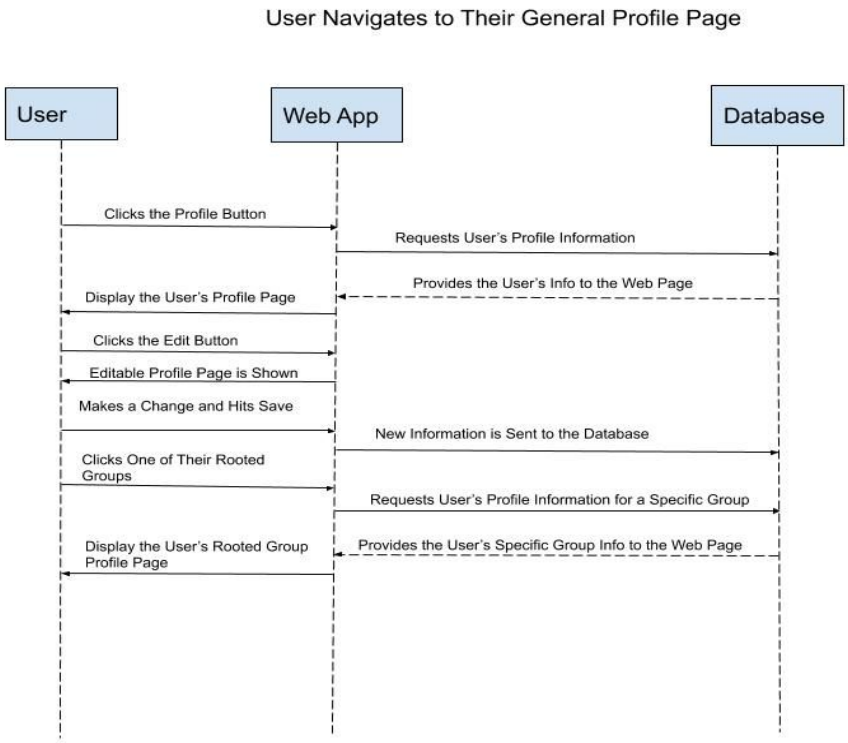
4. Which web service should we use?
    - Option 1: Azure
    - Option 2: AWS
    - Option 3: Heroku
    - Option 4: Google Cloud

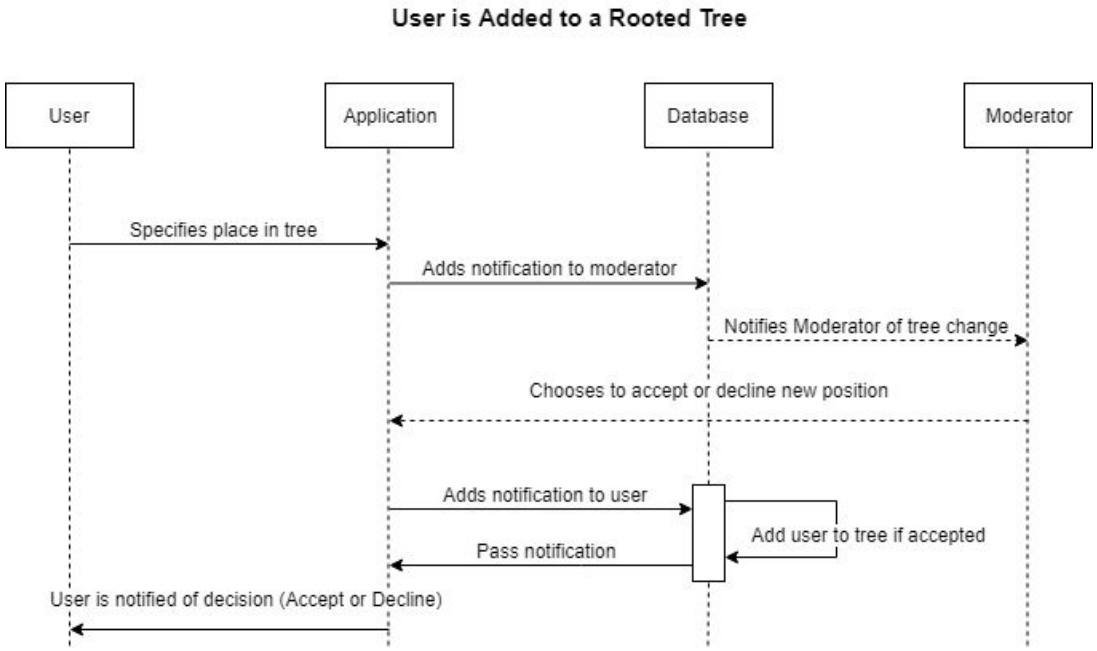
Choice: Option 3
Justification: Heroku out of the options listed allows for the simplest setup as it is a PaaS which will allow us to focus on building the actual application and less time worrying about the infrastructure. Furthermore, it is free and includes continuous deployment with our version control of choice (GitHub)

Design Details

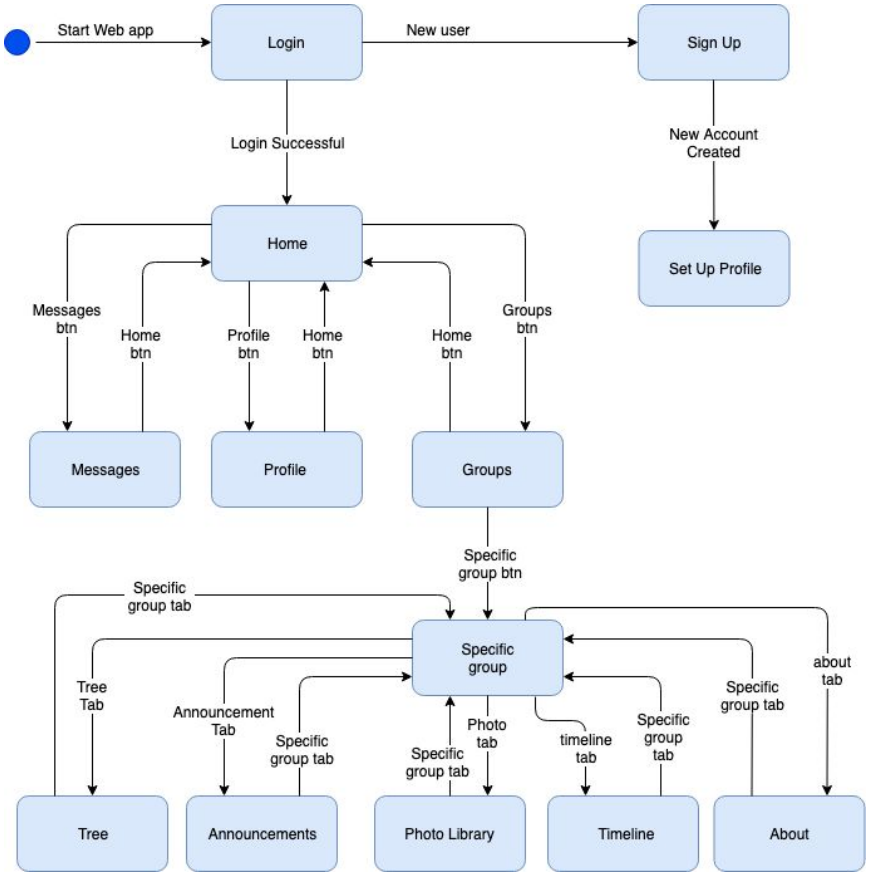
Sequence Diagrams





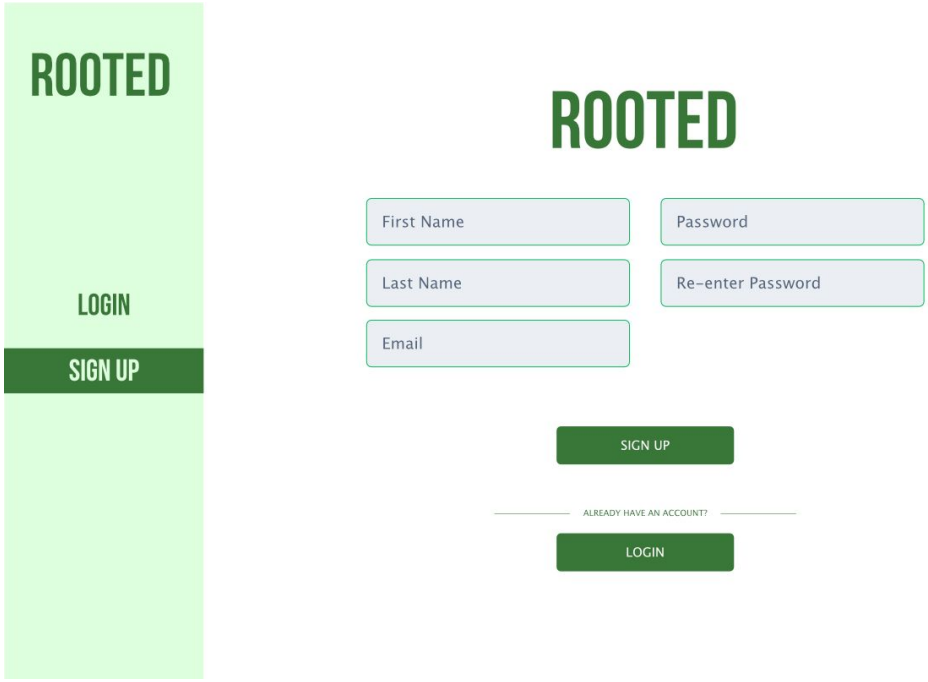


Navigation Flow Map

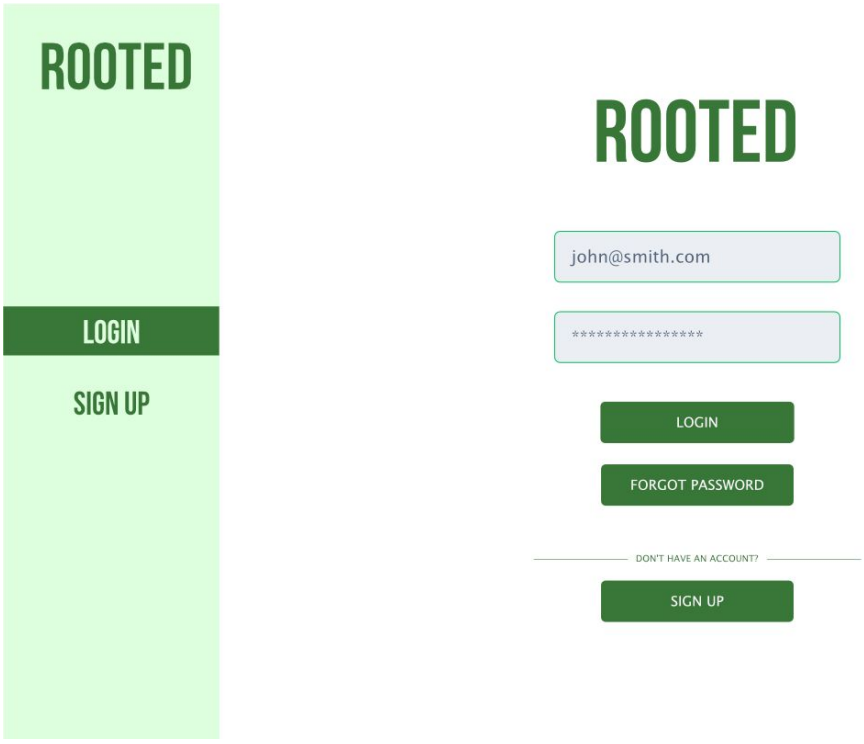


UI

Register Page\

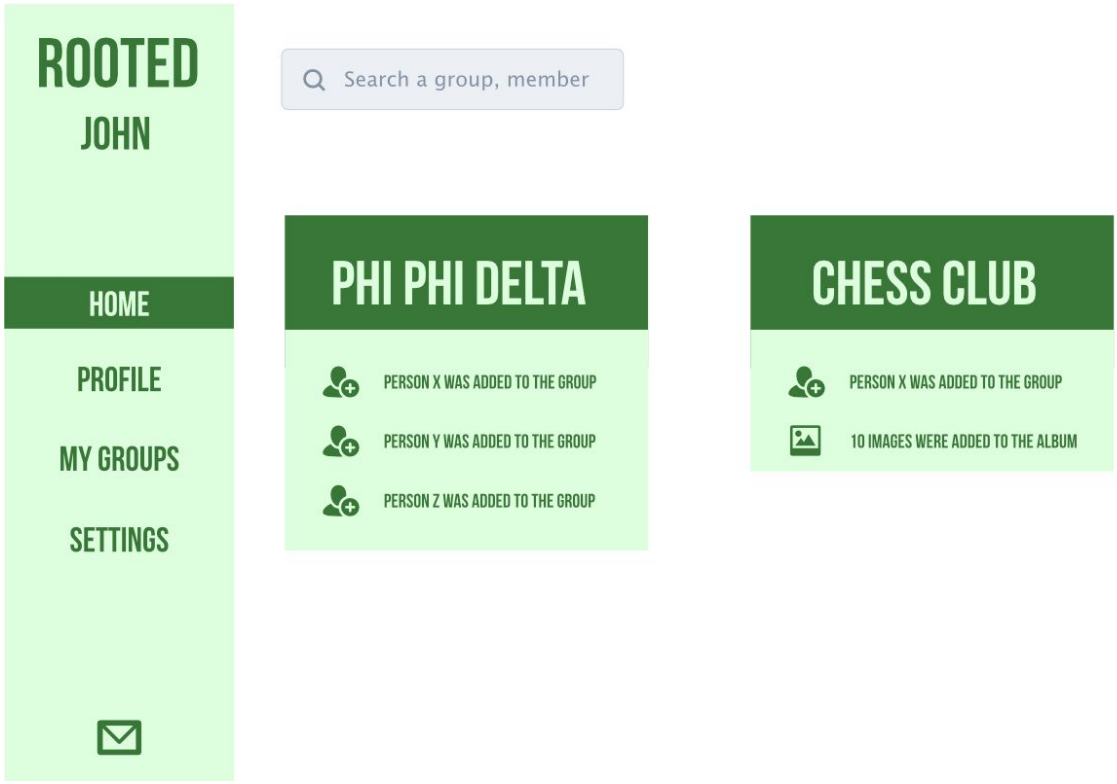


Login Page

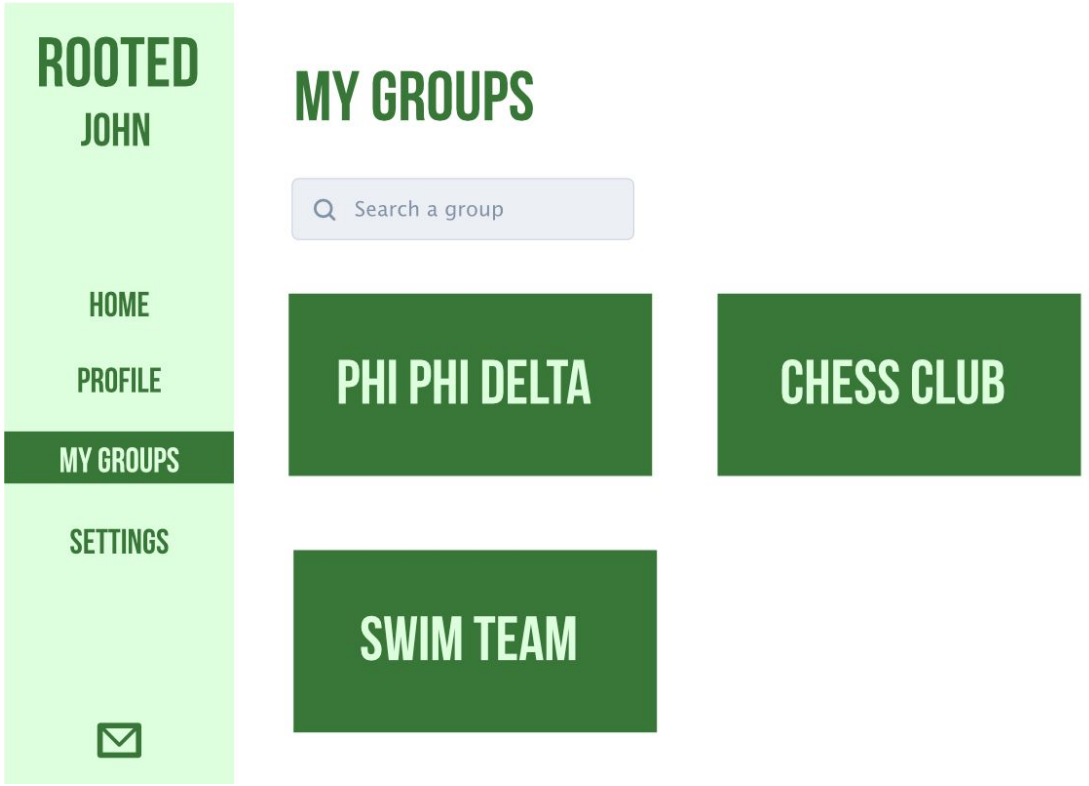


Profile Page

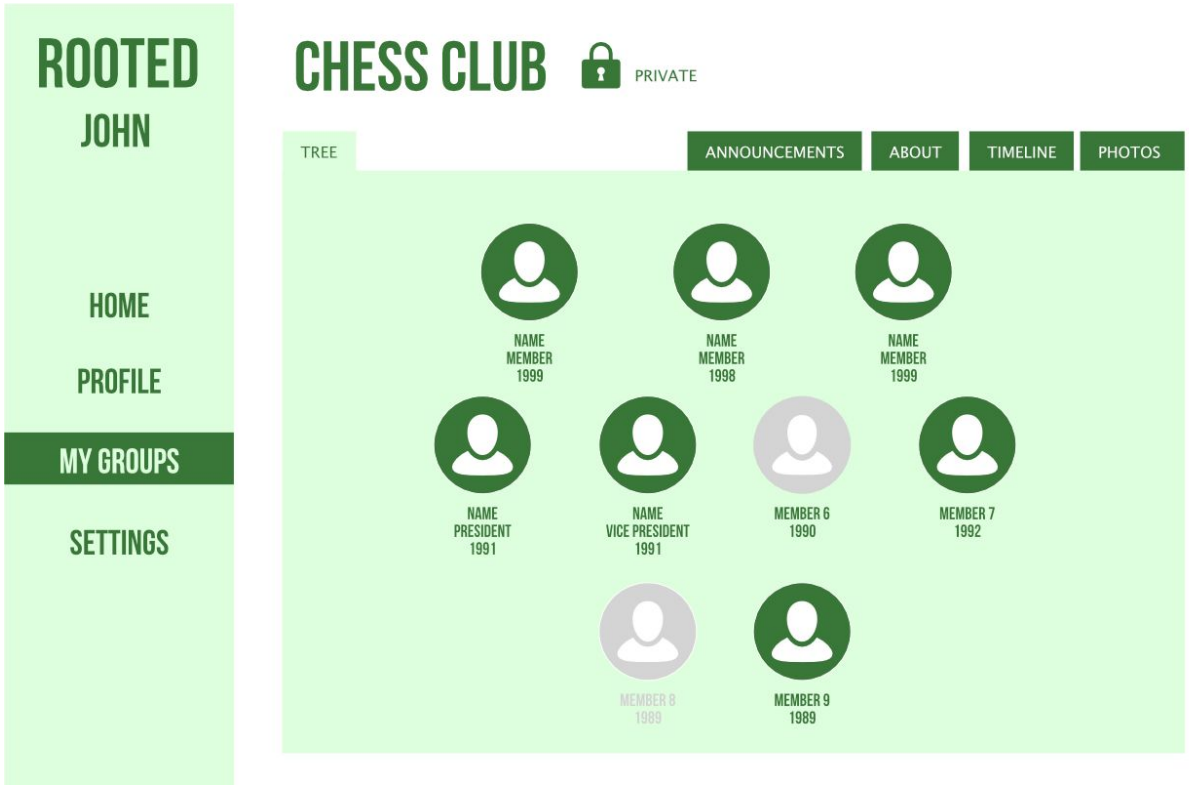
Home Page



My Groups Page



Specific Group Page



Functional Requirements

Stories for Logging In

Backlog ID	Functional Requirement	Hours
1	As a new user, I would like to be able to register for an account	2
2	As an existing user, I would like to be able to login	4
3	As an existing user, I would like to be able to logout	2
4	As a user, I would like to be able to edit my email address	1
5	As a user, I would like my password to be reset if I forget it	1
6	As a user, I would like to be able to change my password	1

Stories for General Users

7	As a user, I would like to be able to create and edit a general profile page, with info such as my name, birth year, bio, links to social media, etc	3
8	As a user, I would like to be able to upload a profile image	4
9	As a user, I would like to be able to selectively hide or display specific info on my profile page	2
10	As a user, I would like to be able to upload photos to my profile page's photo library	4

11	As a user, I would like to be able to view all groups that I am a part of from my general profile page	1
12	As a user, I would like to be able to view other user's profile pages	1
13	As a user, I would like to be able to directly message other users	2
14	As a user, I would like to be able to receive notifications	2
15	As a user, I would like to be able to search for groups	2
16	As a user, I would like to be able to request to join a group	2
17	As a user, I would like to be able to respond to invites to groups	2
18	As a user, I would like to be able to create a group and be granted group creator privileges	2
19	As a general user, I would like to be able to report users of a group to the group's moderation team	1
20	As a general user, I would like to be able to report a group to the sitewide admins	1

Stories for Sitewide Admins

21	As a sitewide admin, I would like to be able to close any group	1
22	As a sitewide admin, I would like to be able to ban any user	1
23	As a sitewide admin, I would like to be able to unban any user	1

Stories for Group Moderation Team

Backlog ID	Functional Requirement	Hours
24	As a group creator or group admin, I would like to be able to change the role statuses of other members within my group (e.g. admin, moderator, member)	2
25	As a group admin or moderator, I would like to be able to ban users from my group	1
26	As a group admin or moderator, I would like to be able to unban users from my group	1
27	As a group admin, I would like to be able to close my own group	1
28	As a group admin or moderator, I would like to be able to invite non-members to join my group	1
29	As a group admin or moderator, I would like to be able to approve non-member requests to join my group	1
30	As a group admin or moderator, I would like to be able to customize the color scheme of my group page	2
31	As a group admin or moderator, I would like to be able to upload a custom image as a banner to the top of my group page	4
32	As a group admin or moderator, I would like to be able to adjust the tree structure of my group	4
33	As a group admin or moderator, I would like to be able to add/edit information for non-rooted members (profile picture, name, title, involvement)	2
34	As a group admin or moderator, I would like to be able to approve photo upload requests for the group photo library	2

35	As a group admin or moderator, I would like to be able to make text posts to the group announcements page	2
36	As a group admin or moderator, I would like to be able to edit our about page	1
37	As a group admin or moderator, I would like to have a user interface for adding/editing items on our group timeline page	3
38	As a group admin, I would like to be able to make my group public or private	1
39	As a group admin or moderator, I would like to be able to visibly establish a distinction non-rooted members from rooted members	1

Stories for Group Members

40	As a rooted member of a group, I would like to be able to add/edit my personal info (name, profile picture, title, involvement)	1
41	As a member of a group, I would like to be able to upload photos to my page within the group	3
42	As a member of a group, I would like to be able to request to the group moderation team that certain non-members be invited to the group	2
43	As a group member, I would like to be able to submit photo upload requests for the group photo library	1
44	As a group member, I would like to be able to anonymously submit messages to the group moderation team	1

Nonfunctional Requirements

Architecture and Performance

We will be developing the application with a separate frontend and backend. This will allow us to effectively divide our work between the members in our group, decrease the compatibility issues between the two sides, and also make version control a lot easier. We plan on using the MEAN stack (MongoDB, Express, Angular, and Node) as it is a popular and proven software stack with lots of resources and documentation for when we need assistance. The frontend will be constructed using React and will pull data from the backend via requests to the API. Separating the backend from the frontend will also ease the burden of expanding to more platforms such as iOS or Android. Since MongoDB is not real-time, our application will not have real-time responsiveness. However, all requests and responses will be very quick.

Security

Since we will be storing a lot of information sensitive to our users, we want to take the necessary measures to make sure our application is secure. There are many existing modules that can be easily integrated into Express/Node that includes several security features to fight against common exploits. Furthermore, we will be limiting access to features on the website through the use of user roles and requiring authentication for all API requests in order to prevent exploitation of backend routes. We will also be utilizing hashing to make sure sensitive information in our database is secure such as with passwords.

Usability

The interface will be easy to navigate for users, with an intuitive and simple user interface. Organization will be important, with lots of features, to make sure the home page and screen is not too cluttered. Any user should be able to navigate the website on their own accord, and without needing a demonstration or tutorial. As our target audience will be variable (a wide range of ages) we want to make sure people with different levels of technological fluency will all be able to use our website without issue.

Hosting/Deployment

For hosting, many of our team members have existing experience deploying using Heroku and it is well integrated with GitHub, which is the platform we will be using to handle version control. Having continuous integration and deployment with GitHub through Heroku will allow our development process to move much more efficiently.