

# MARIST COLLEGE

## Memorandum

### Helping Hands

To: Pablo Rivas, Christopher Algozzine, Juan Arias  
Professors of the School of Computer Science and Mathematics

Class: CMPT 475/477 CS/IT/IS Project I & II

From: Liam Harwood  
Cristian Hernandez  
Daren Pagan  
Juan Vasquez  
Maxim Vitkin

Date: December 6, 2017

Subject: Final Paper

# Table of Contents

I.	Executive Summary .....	2
II.	Analysis .....	2
	User Requirements.....	2-3
	Use Case Diagram.....	4
	Activity Diagrams.....	5-6
III.	Plan.....	6-8
IV.	Cost/ROI.....	8-9
V.	Application Design.....	9
	Database Design.....	9
	Installation Instructions.....	9-11
	User Interface Design.....	12-24
VI.	Infrastructure Design.....	25
	IT Requirements.....	25-27
	Linux Configuration Instructions.....	27-28
	Linux Server Instructions.....	28-29

## **I. Executive Summary**

Helping Hands is a social networking site, where multiple communities come together and share their thoughts and prayers for one another. People come together, align themselves with their religions or communities, and they can post statuses in which people can like, dislike, and comment on these statuses or also known as prayers. When it comes to these types of networking sites, not many come around. Because of that, we want to provide a site where people feel comfortable being in their communities, sharing thoughts and beliefs with other people that share the same interest and avoid any inappropriate content where moderators and admins make sure the wrong posts are taken care of.

## **II. Analysis**

### **USER REQUIREMENTS**

#### **Basic User**

- A basic user can create an account / profile page.
  - This account contains the user's full name, birthday, location, picture, and religions.
  - This account is not meant to be anonymous; users will use their real names in addition to a handle / username.
- A basic user can manage their account to change any information as needed.
- A basic user can create Posts (Prayer Requests) of up to 140 characters.
  - They can also add a description of up to 5000 characters, post updates of up to 140 characters, upload pictures associated with the prayer request, use hashtags to associate it with similar posts, and mark their own posts as complete (i.e. "Prayer Answered").
- A basic user can manage their settings, religions, and other preferences.
- A basic user can switch between different views such as least responses, most popular, or followed user's timelines.
- A basic user can pray (like Facebook "like" or Reddit "upvote"), comment, etc. on posts.
- A basic user can downvote posts (posts are auto-reported with enough downvotes).

- A basic user can receive notifications for community activity or activity on their posts.
- A basic user can view profiles of any other users.
- A basic user can petition for a new “unverified” religion/community which is later approved.
- A basic user can join verified religions / communities.
- Users can earn points that contribute to their reputation rating on the site:
  - “Praying hand” points for praying for someone
  - “Halo” points for liking completed posts (answered prayers)
  - “Hammer” points for flagging reported posts
  - “Knee” points for upvotes on their own posts
- Users with high enough reputation gain Power User status.

#### **Power User (includes Basic User Requirements)**

- A power user can report problematic posts for moderators to examine.

#### **Moderators can ban users (in their communities).**

- Moderator (Creators of communities/religions, includes Basic User Requirements)
- Moderators can delete posts (in their communities).

#### **Administrator (Developers / Site Owners)**

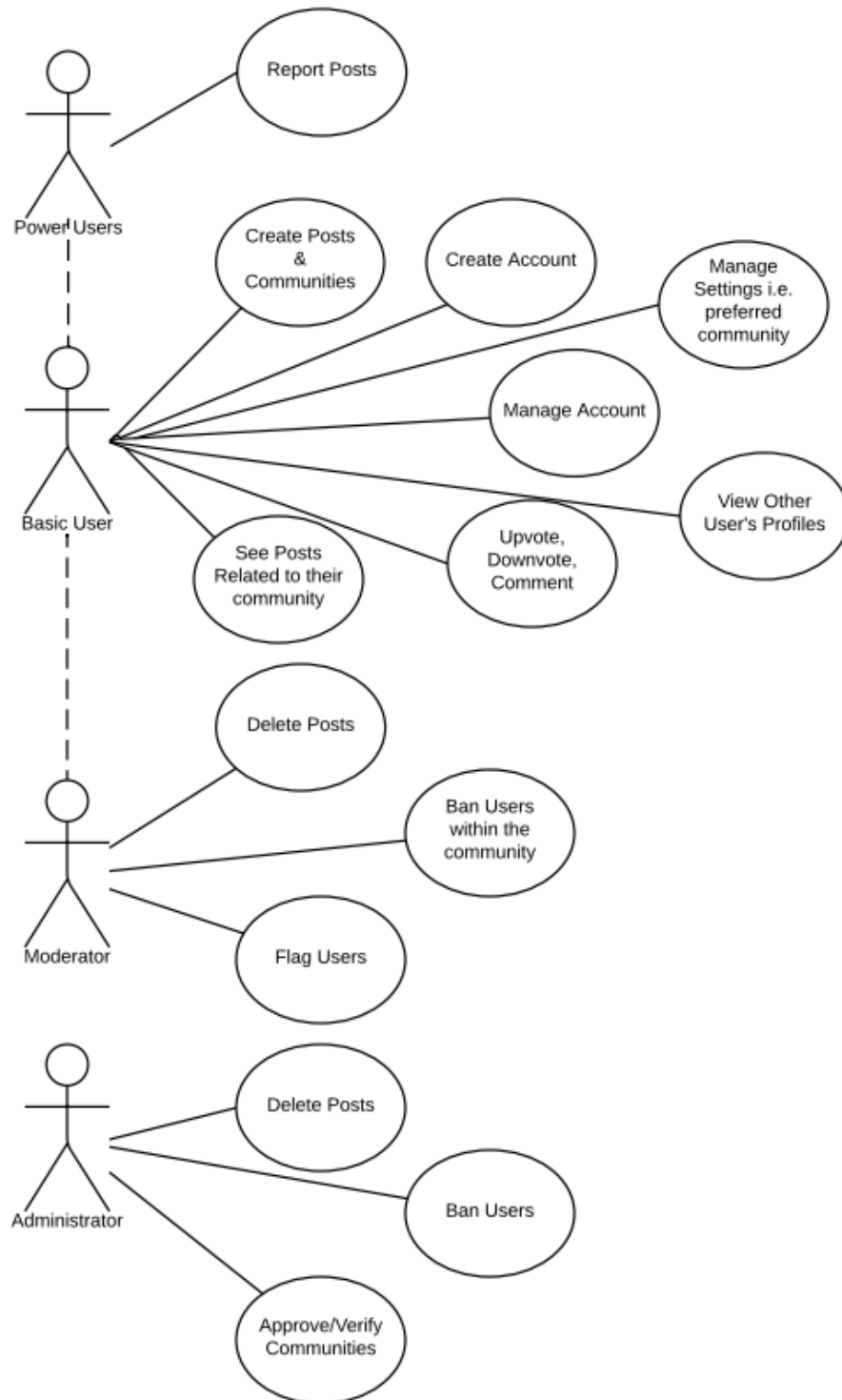
- An administrator can delete any post.
- An administrator can ban users that have been flagged by **THE SYSTEM**.
- An administrator can approve and verify religions / communities.

#### **Other Requirements**

- Preferred technologies for the implementation are Linux and PostgreSQL.
- The web app should work in both Chrome and Safari browsers.
- A mobile app is necessary, and the website should be mobile-ready.
- Each user should have a single encrypted password for authentication.
- Each user can connect account to Facebook, Twitter, or email.

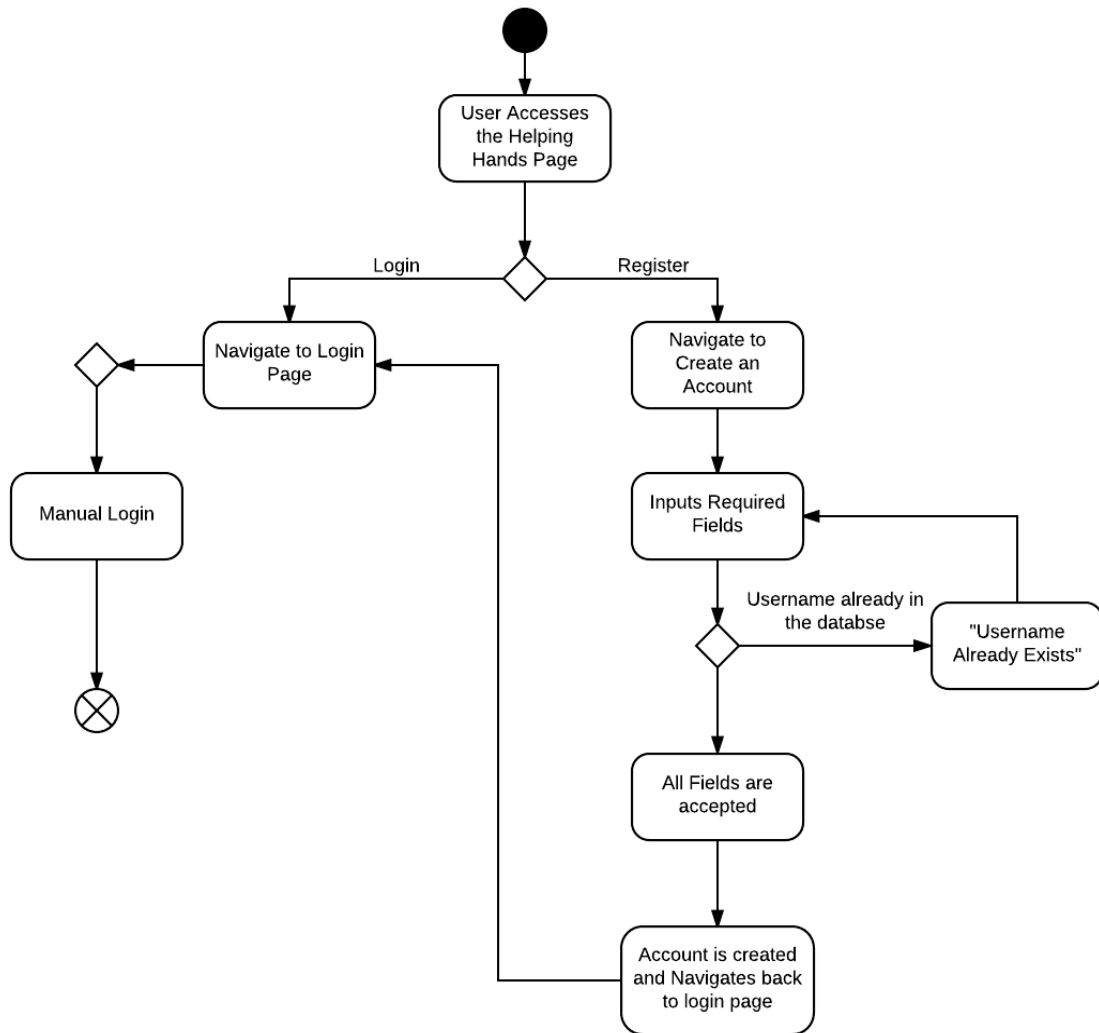
#### **Use Case Diagram**

PSN Use Case Diagram

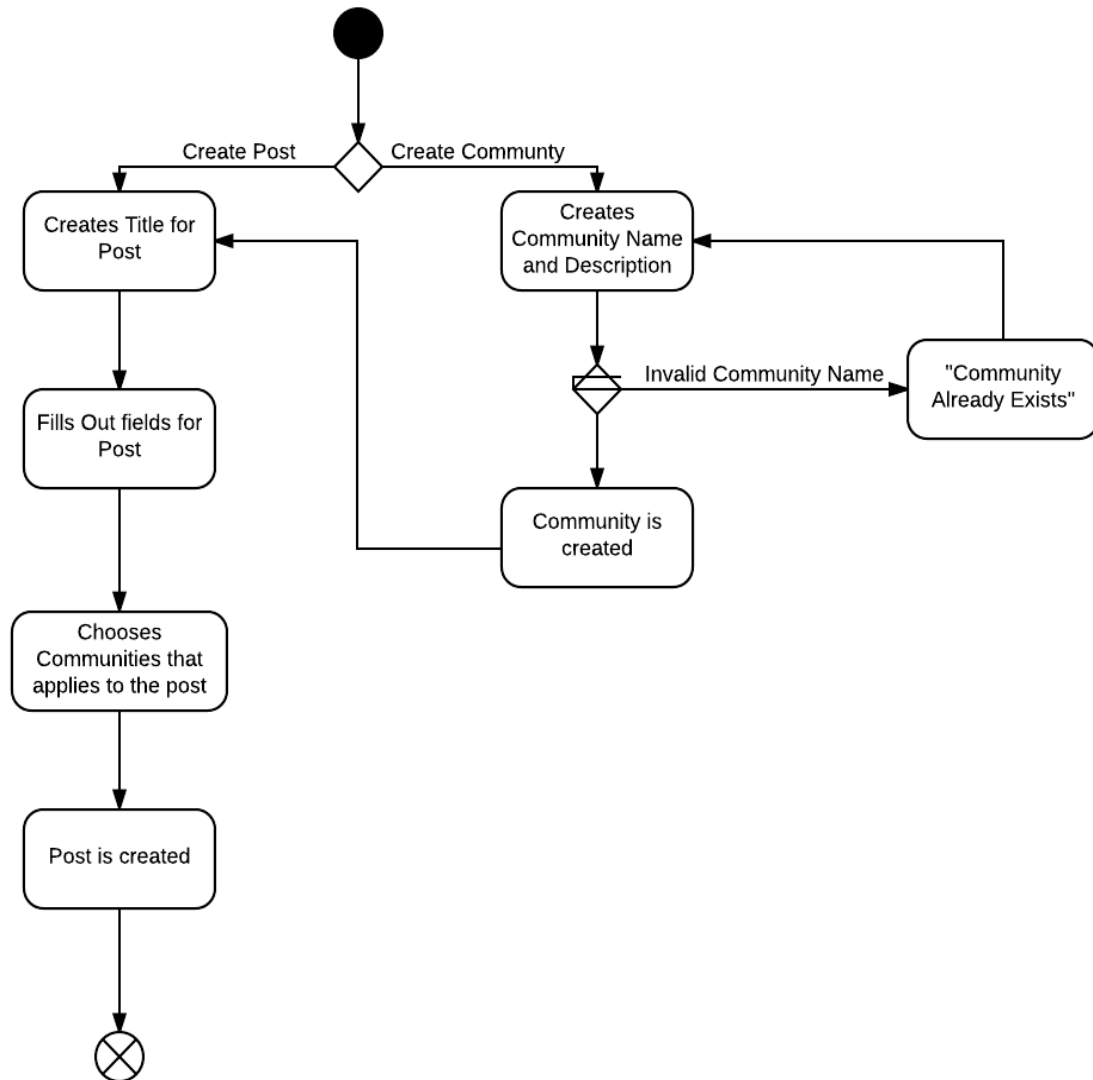


## Activity Diagrams

### Login



## Creating Posts



## III. Plan

### Week One: August 30, 2017

Came together for our first meeting. We assigned the roles of the project and who oversaw what in this project. Max and Juan were going to oversee front-end development, while Liam was responsible for back-end development. Daren was responsible for IT related work, such as putting up servers, allowing permissions for the users, etc. and Cristian oversaw the IS work:

preparing weekly reports, making sure everyone had something to do throughout the sixteen weeks, etc.

### **Week Two: September 6, 2017**

We decided to go with the Prayer Social Network project overview and gathered requirements for the project. We came up with a list of questions that ranged from anything basic to specific requirements that the client desired when coming up with this site. We wanted to make sure that we got everything the client wanted.

### **Week Three: September 13, 2017**

Created the basic plan layout for the remaining parts of the project including the project plan. We laid out, a week by week basis of what we wanted to accomplish before our weekly reports with the class. We also included user requirements, and use case diagrams.

### **Week Five: September 27, 2017**

ER diagrams are completed by now and back end development has begun. UI mockups are completed and ready to go then. Front end development kicks off as well. Our server is created and is up and running throughout the time. However, we lost the server somehow and was rebuilt later throughout the week.

### **Week Seven: October 11, 2017**

The database is implemented at this point meeting all the requirements for the to succeed on the three normal forms test. Basic front-end implication is placed. Things such as json, bootstrap, and configuration files.

### **Week Nine: October 25, 2017**

Security is strengthened through the back-end database system by implementing CORS, and support is added for password encryption. Endpoints are added for adding posts, removing posts, and following posts. On the front-end side, main dashboard components, post retrieval, and profile cards were added onto the webpage.

### **Week Eleven: November 8, 2017**



Registration, post creation, community, votes and authentication endpoints were added onto the back-end process. A lot of cleanup such as making sure that the functions worked properly that weren't working before. On the front-end side, the dashboard was cleaned up, laid out, and added basic authentication. Client-side routing was also placed. The base login page, community pages, and user pages, and post voting were created.

### **Week Thirteen: November 22, 2017**

Basic post submissions, buttons to follow and unfollow, authentication were created and fully functional within front end development. Within backend development, date columns, getting comments on a post, creating a post, unfollowing communities and users were implemented. Servers were also down, which delayed us from making any other progress for a while.

### **Week Fifteen: December 6, 2017**

The settings and registration pages were created and work now. Session storage was implemented so people can stay logged into their accounts and don't have to sign every time they access a different page. Users now have score cards in their accounts showing their reputation throughout the site. There is search bar functionality now where users can look up other users, posts, or communities. In the backend development, reputation systems were implemented where it is based upon a user's score on their profile. Endpoints were created for the rest of the necessary tools on the webpage. Image uploading is fully functional.

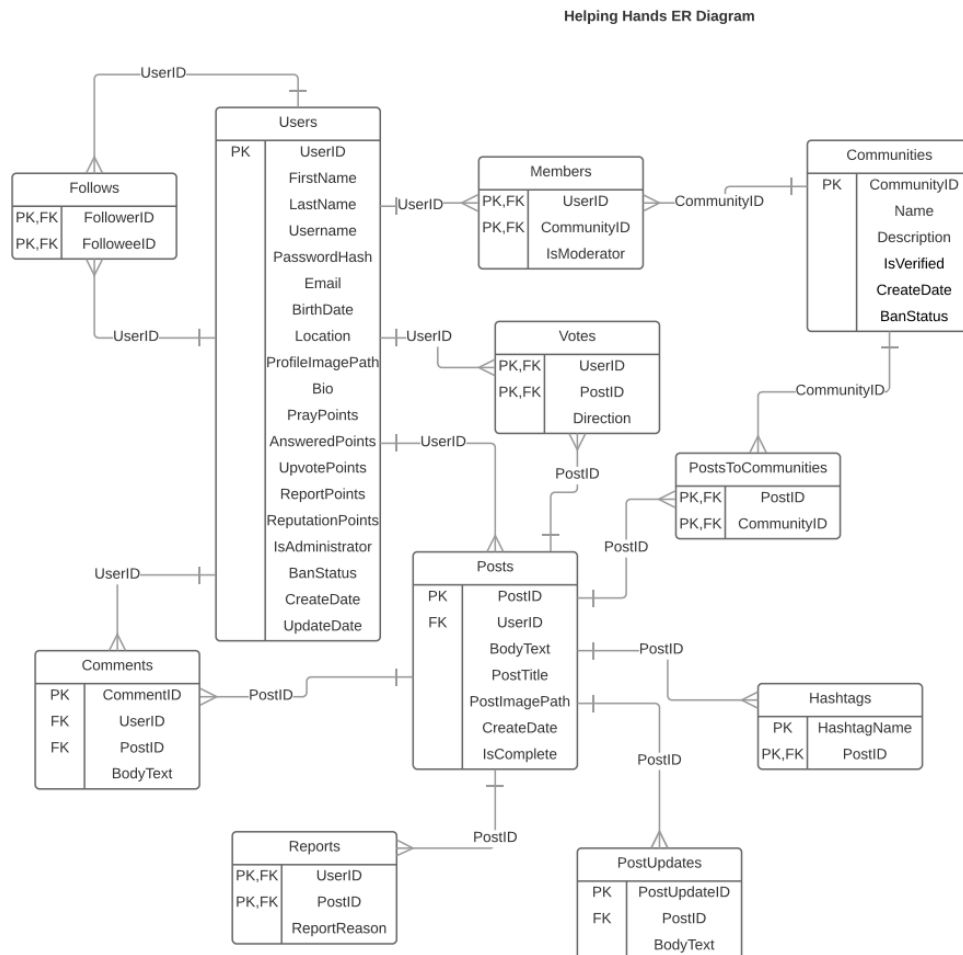
## **IV. Cost/ROI**

We decided that we would take the approach of Amazon Cloud Services and rent out servers from there site. We would rent three large servers: one for a database, one for development, and one for production. Since in the IT requirements we are going to want our system up 90 percent of the time we want to use 23.4 hours of the server active throughout the day. The cost of Amazon servers are 40 cents an hour for Unix which would be used for our database and 50 cents for Windows use which will be used for development and front-end production. Our cost yearly for the Unix server would be about \$3407.04. Our production side will be only up and running for about 23.3 hours so that would be around \$4277 for each other server. The yearly cost for all servers would be around \$11,961.04. We have decided that we would have nine

employees working throughout the servers, assigning three to each server and keeping everything up and running. The cost of the employees all depends how successful the site is, and how much work they will be providing in the near future. That is all up to the manager to decide the cost for the employees.

## V. Application Design

### Database Design:



### Helping Hands Installation Instructions:

#### System Requirements:

- Linux (Ubuntu 16.04 preferred)
- Apache Web Server

- PostgreSQL database
  - Java 8
1. In Postgres, create a database with a name such as **helpinghandsdb**.
  2. Create three roles named **admin**, **app**, and **reporting**. Give each of them a password and login ability, and edit the **pg\_hba.conf** so that all roles use md5 authentication. After that, restart Postgres.
  3. Using git, clone **<https://github.com/liamrharwood/Capping2017>** into a location of your choosing.
  4. You will find an SQL file in the top level of the repository called **create\_db.sql**. Run the following command: **psql -U admin -d [database] -f create\_db.sql**. This will populate the database with the initial tables.
  5. Before proceeding, install Maven and npm and update node.js: **sudo apt-get install maven; sudo apt-get install npm; sudo npm install n -g; sudo n stable;**
  6. Navigate to the **Capping2017/backend/HelpingHands** directory and run **mvn package** to build the backend jar file.
  7. Now run the following command to migrate the database to account for all changes: **sudo java -Ddw.database.url=jdbc://localhost:5432/[database] -Ddw.database.user=admin -Ddw.database.password=[admin password] -jar target/HelpingHands-1.0-SNAPSHOT.jar db migrate helping-hands.yml**
  8. Navigate back to **Capping2017**. Run **npm install** followed by **npm run build**.
  9. Copy the contents of **Capping2017/src/client** to **/var/www/html** to host the files on the Apache Web Server. You may need to change your Apache's configuration file in **/etc/apache2/sites-available** to rewrite rules, since this is a single page application. Use

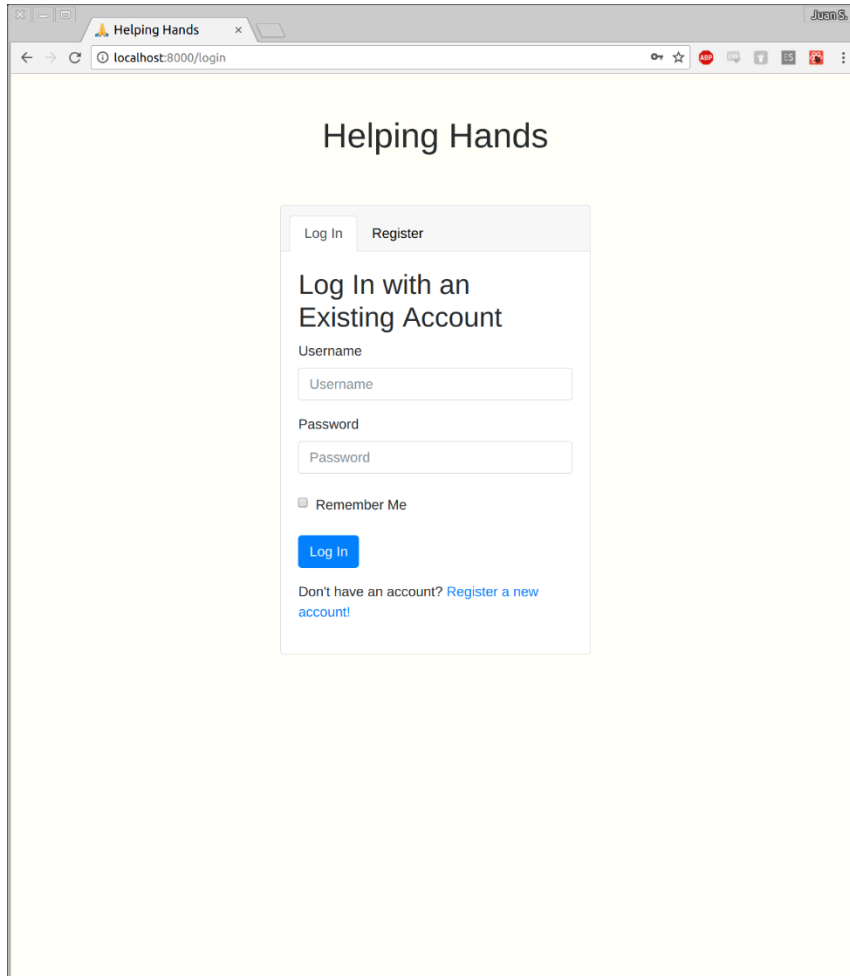
the following configuration:

```
<Directory "/var/www/html">
    RewriteEngine on
    RewriteCond %{REQUEST_FILENAME} -f [OR]
    RewriteCond %{REQUEST_FILENAME} -d
    RewriteRule ^ - [L]
    RewriteRule ^ index.html [L]
</Directory>
```

10. Navigate back to **Capping2017/backend/HelpingHands** and run the following: **sudo java -Ddw.database.url=jdbc://localhost:5432/[database] -Ddw.database.user=app -Ddw.database.password=[app password] -jar target/HelpingHands-1.0-SNAPSHOT.jar server helping-hands.yml**
11. Congratulations! Helping Hands is now running on your server! To view valid API endpoints, read the **endpoint-doc.txt** file in **Capping2017**.

## User Interface Design:

### 1.) Login View - Log In Tab



The screenshot shows a web browser window with the title 'Helping Hands' and the URL 'localhost:8000/login'. The page has a light yellow background. In the center, there is a white login form with a grey header containing two tabs: 'Log In' (selected) and 'Register'. The form is titled 'Log In with an Existing Account'. It contains a 'Username' label and a text input field, a 'Password' label and a password input field, and a 'Remember Me' checkbox. Below these fields is a blue 'Log In' button. At the bottom of the form, there is a link that says 'Don't have an account? [Register a new account!](#)'.

The Login view is the first one the user sees, and has two “sections” separated within two tabs.

The first tab is ‘Log In’ which is where the user enters an existing username and password to log in. Before they log in, the user can check the remember me box so that they stay logged in, or click the link below the blue button, which will take them to the ‘Register’ tab. If they do not have the correct username/password combination for an existing password, they will get an error.

## 1.) Login View - Register Tab

Helping Hands

Login Register

### Register a New Account

First Name\*

Last Name\*

Your Personal Username\*

Email address\*

We'll never share your email with anyone else.

Password\*

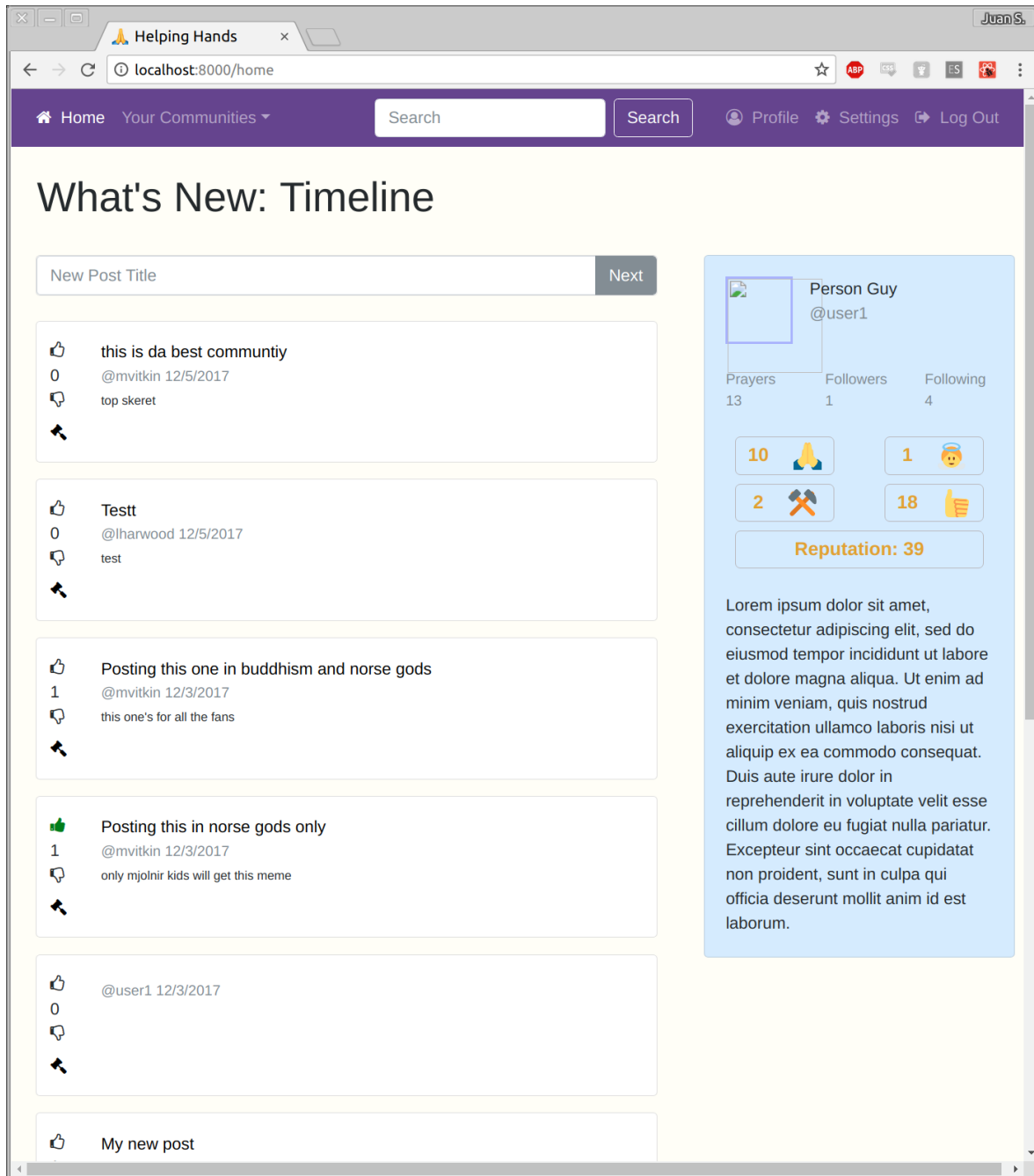
Confirm Password\*

Birth Date\*

Register

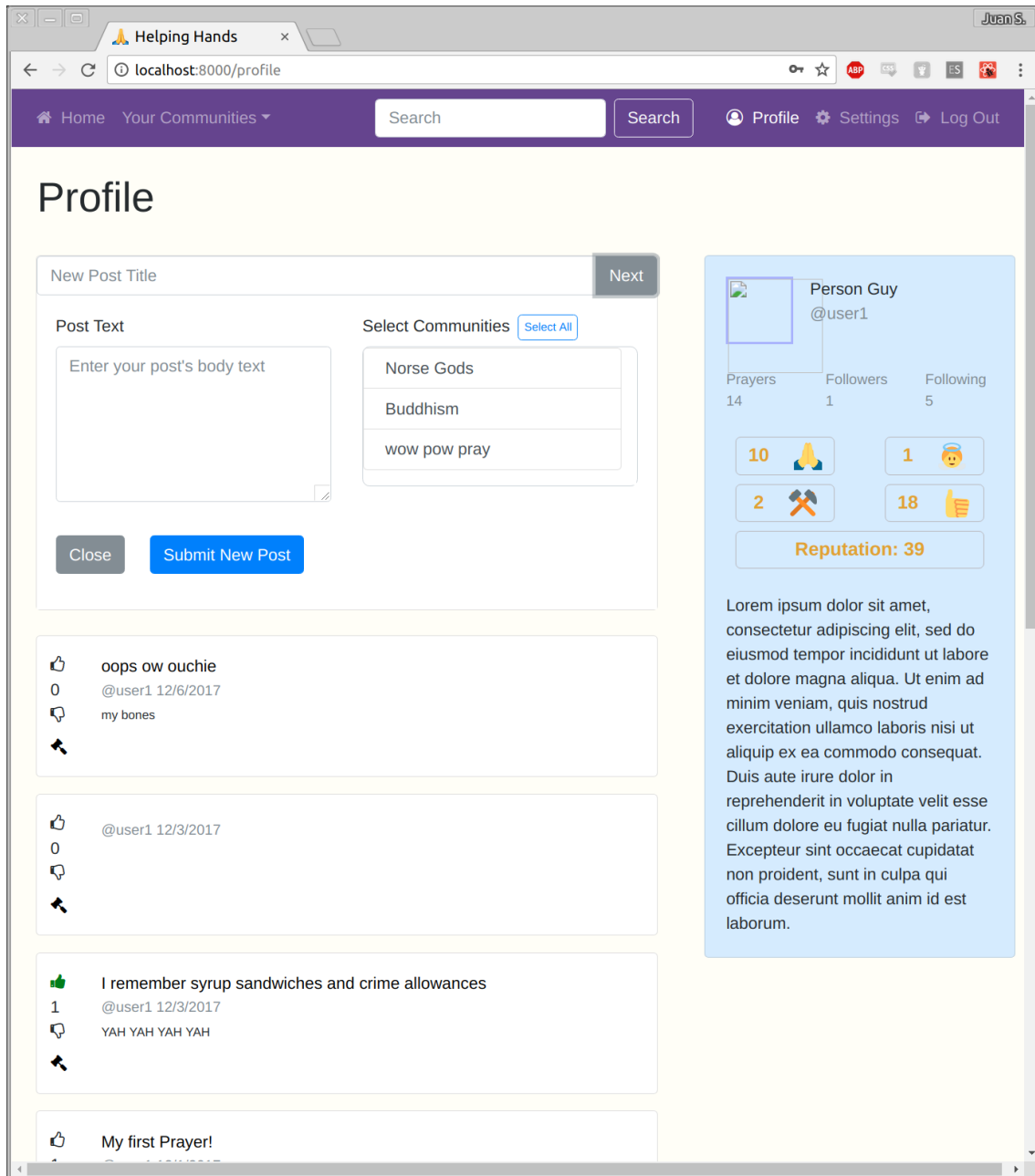
The second tab on the Login view, this tab is where a user goes if they want to create a new account. They enter all their information, and click on the register button. If they are missing any of the required fields, enter an already-existing-username, or their passwords don't match, then they are given an error and must fix it before they can continue. Once registered, they can login using the Log In tab.

## 2.) Home View



Once the user has logged in, they are greeted by the Home view, with both the timeline as well as their profile card on the dashboard. Here, they can create posts, view posts from all of their communities, and see what their profile looks like to anyone else. They can use the navigation bar at the top to go to other views as well. If they click on a post they are taken to the individual Post view, and if they click on a user they are taken to that user's Profile view.

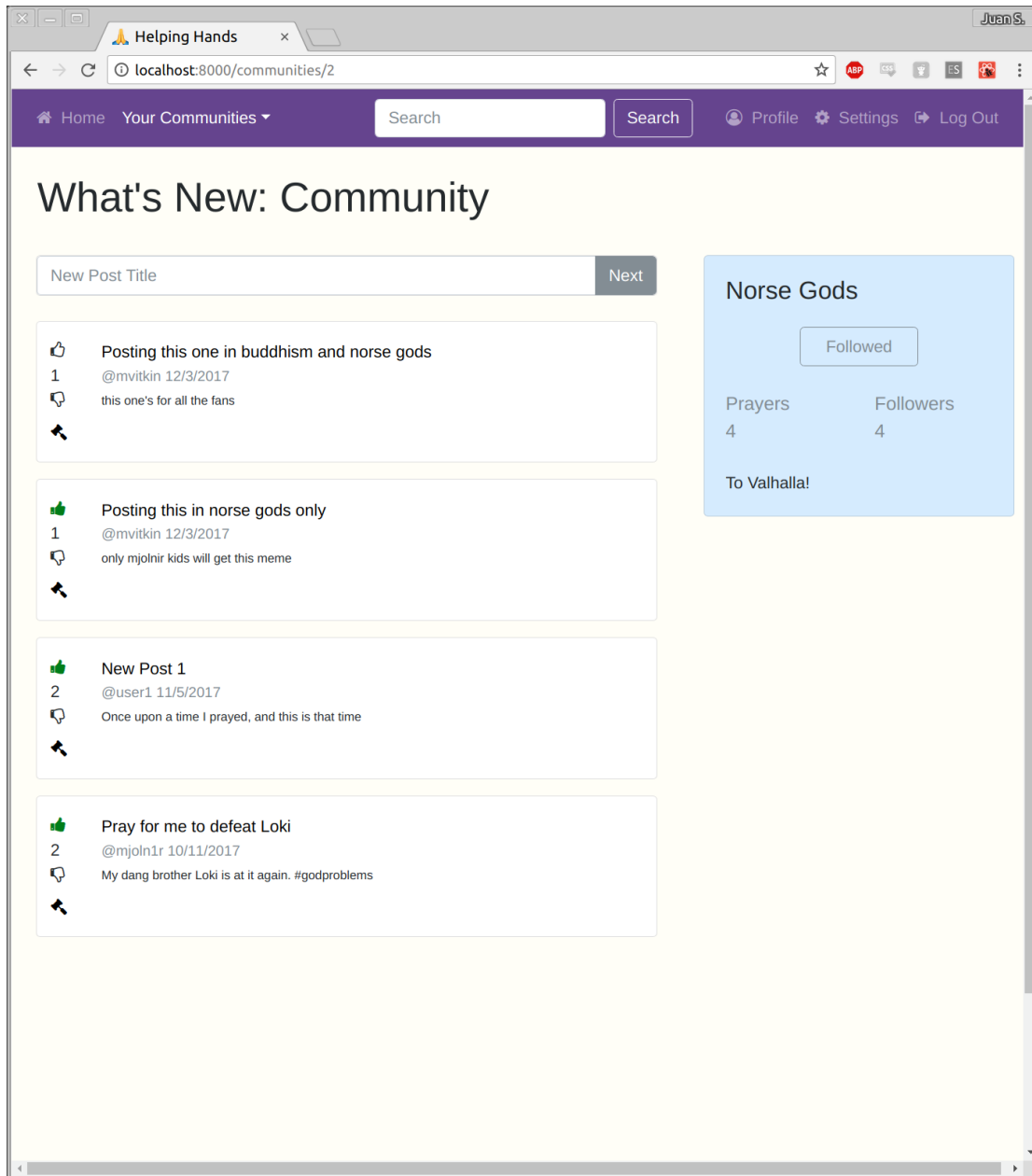
### 3.) Profile View w/ Post Creator



On their own profile, the user can view all of their posts, as well as their profile card. Like with the Home view, they can create a post using the post creator, shown here. In the post creator, they can assign the post a title, body text, what communities they want the post to go to, and close the post creator if they change their mind.

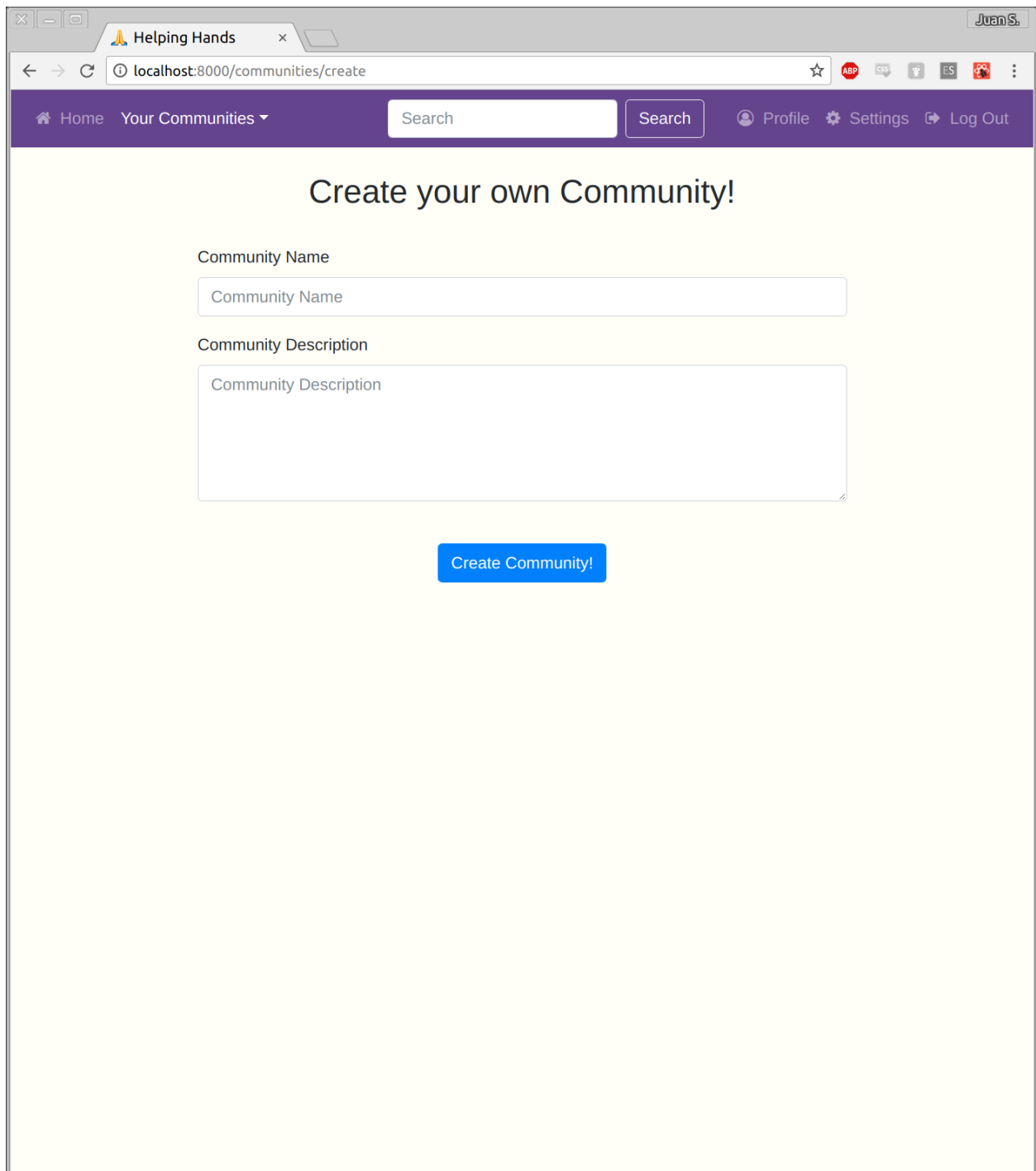


## 4.) Community View



On the Community view, the user can see all of the posts made to that community (shared with other communities or otherwise), as well as the community card, which contains the info for that community. The user can also create a new post with the post creator, exactly like they could in the Home and Profile views. As with any view that can look at posts, the user can vote on each post, as well as report it.

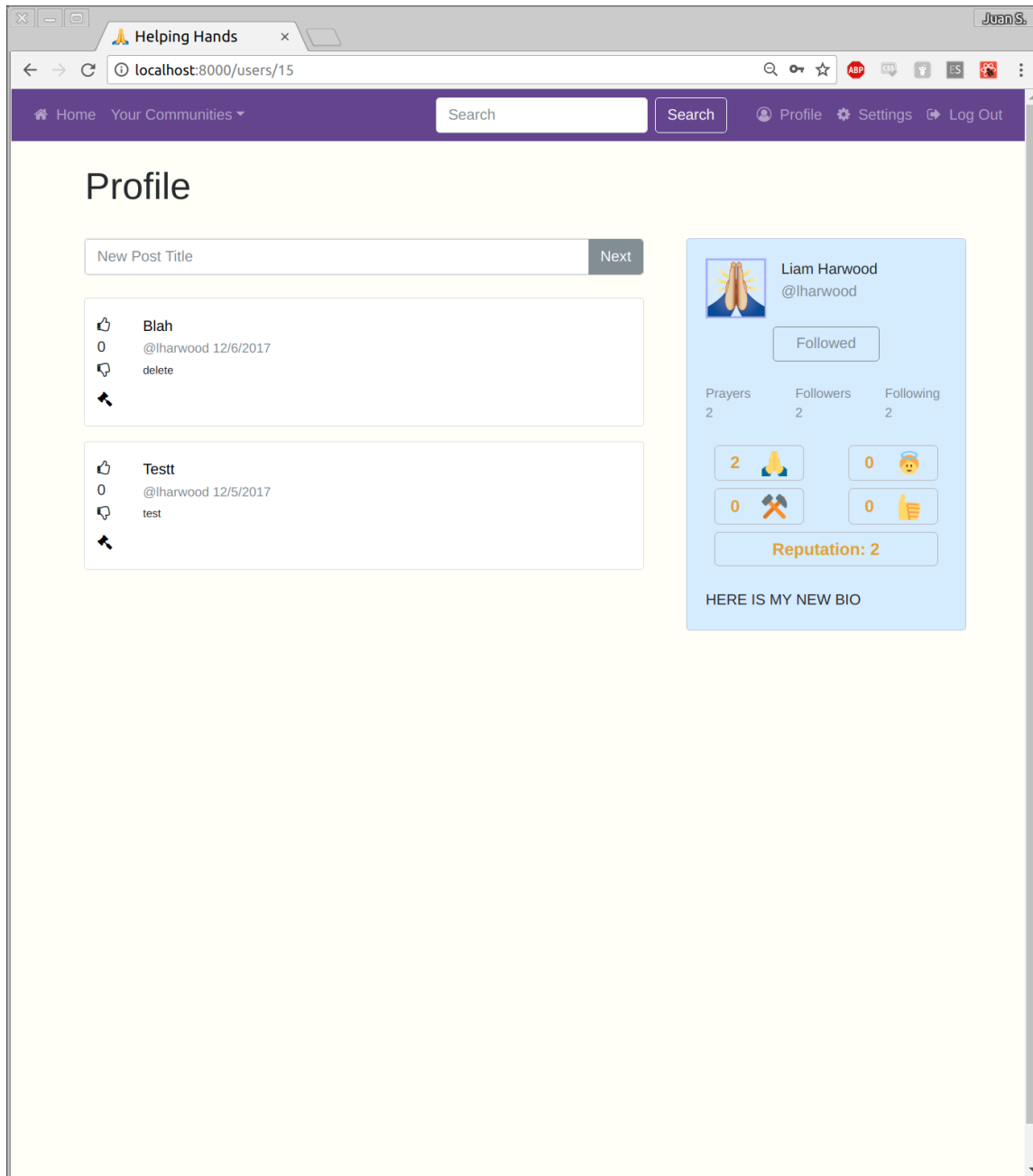
## 5.) Community Creator View



The screenshot shows a web browser window with the title 'Helping Hands' and the URL 'localhost:8000/communities/create'. The browser's address bar shows the URL. The page has a purple navigation bar with links for 'Home', 'Your Communities' (with a dropdown arrow), a search bar, and 'Search'. On the right side of the navigation bar are links for 'Profile', 'Settings', and 'Log Out'. The main content area has a light yellow background and is titled 'Create your own Community!'. Below the title are two input fields: 'Community Name' and 'Community Description'. The 'Community Name' field is a single-line text input, and the 'Community Description' field is a multi-line text area. Below these fields is a blue button labeled 'Create Community!'. The browser's status bar at the bottom shows the user's name 'Juan S.'.

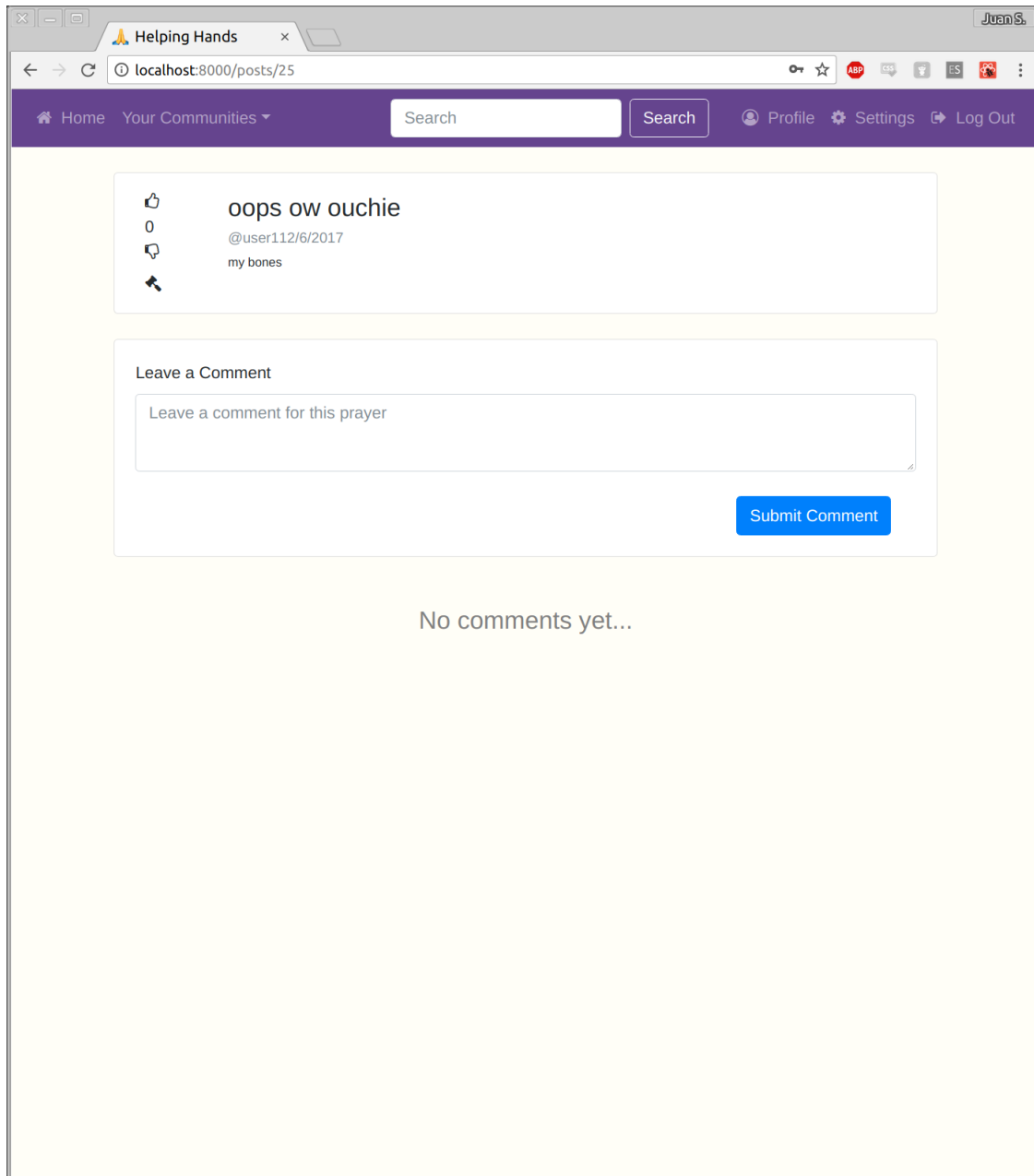
If the user cannot find the community they wish to join, they can make their own, with a name and a description. This view is accessed by clicking the 'Your Communities' tab in the navigation bar and pressing the community creation button

## 6.) Other Users



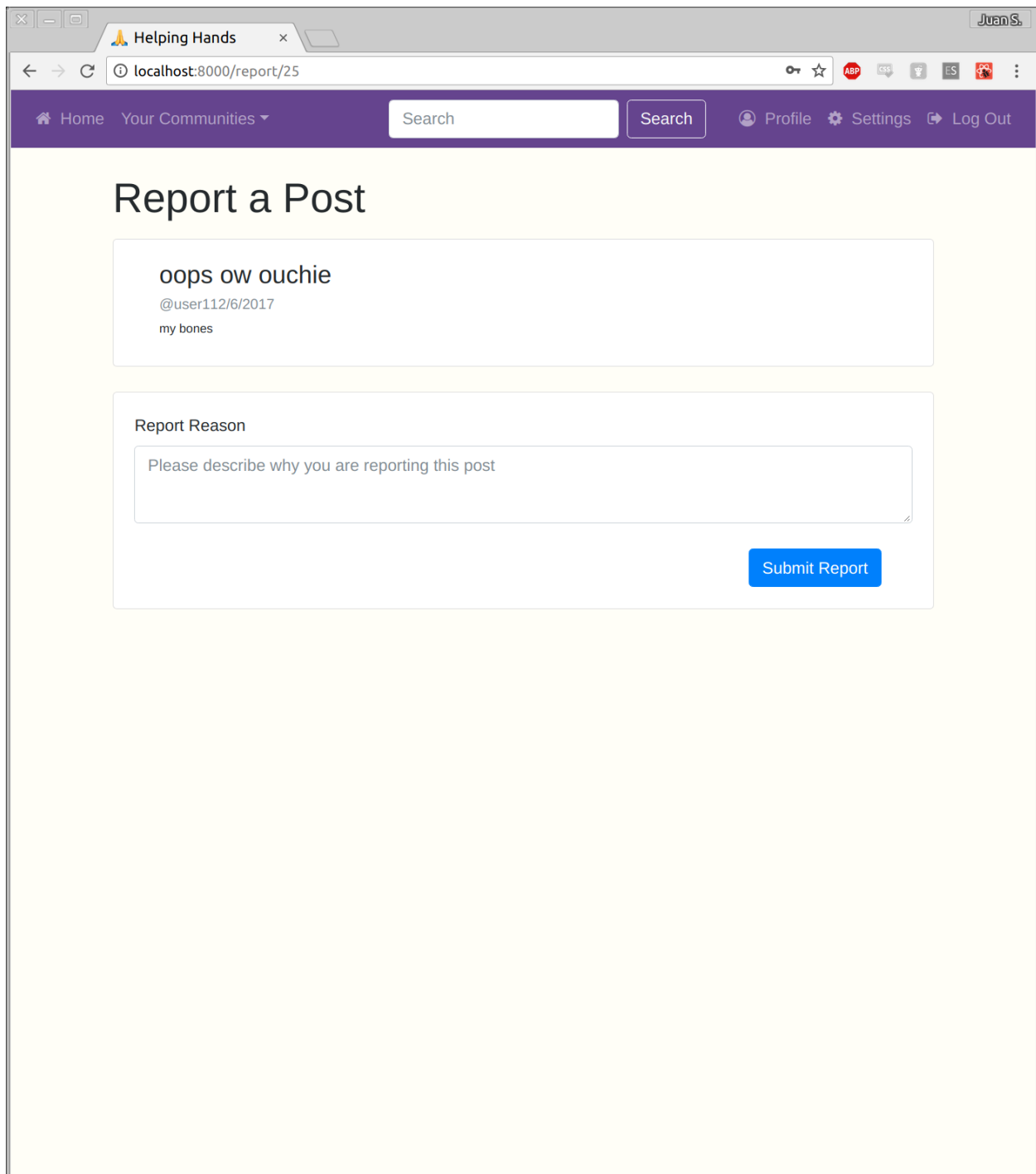
If the user ever wishes to go to a specific user's page, they can click on their name inside of a post, as well as search for them on the search bar at the top. Once on a user's profile, they can see all their posts as well as information about them on their profile card. They can also create a new post, but the post will be posted to whatever communities they decide, NOT on the user they are viewing's profile.

## 7.) Individual Post View



On the individual Post view, the user can vote, report, or leave a comment. This view can be reached by clicking on any post title, at any time, on any other view. If it is the user's own post, they can mark as answered or submit updates.

## 8.) Report View



The screenshot shows a web browser window with the title 'Helping Hands' and the URL 'localhost:8000/report/25'. The browser's address bar shows the URL. The page has a purple header with navigation links: 'Home', 'Your Communities', a search bar, and 'Search'. On the right side of the header are links for 'Profile', 'Settings', and 'Log Out'. The main content area is titled 'Report a Post' and contains a form for reporting a post. The form includes a text input field with the text 'oops ow ouchie', a timestamp '@user112/6/2017', and a label 'my bones'. Below this is a section titled 'Report Reason' with a text input field containing the placeholder text 'Please describe why you are reporting this post'. A blue button labeled 'Submit Report' is located at the bottom right of the form.

Helping Hands

localhost:8000/report/25

Home Your Communities Search Search Profile Settings Log Out

### Report a Post

oops ow ouchie  
@user112/6/2017  
my bones

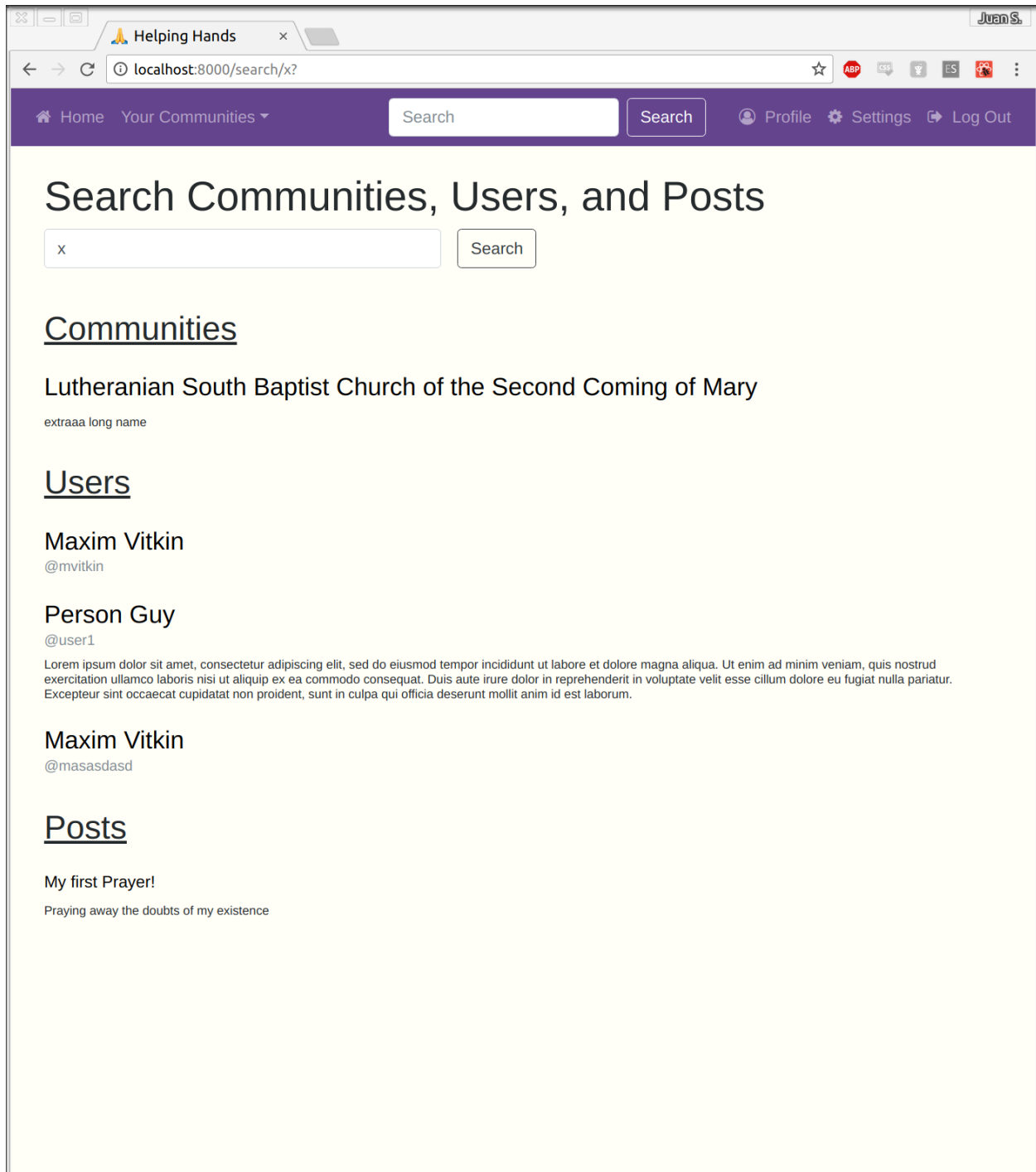
Report Reason

Please describe why you are reporting this post

Submit Report

On the Report view, the user can report the post by leaving a comment. This view can be reached by clicking on the report button on any post, at any time, on any other view.

## 9.) Search View



At any time, the user can enter a query into the navigation bar search field. Once they press the search button, they are taken to the Search view. Here, they can access communities, users, and

posts that match their query. Clicking on any of these will take them to the corresponding page. They can also search for a new query if they wish.

## 10.) Settings View

Helping Hands

localhost:8000/settings

Home Your Communities Search Profile Settings Log Out

### Customize your Profile

First Name: Person

Username: user1

Last Name: Guy

Email Address: Email

Bio: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Save Changes

### Change Your Profile Picture

Image must be square and be less than 5MB.

Choose File No file chosen Upload

### Change Your Password

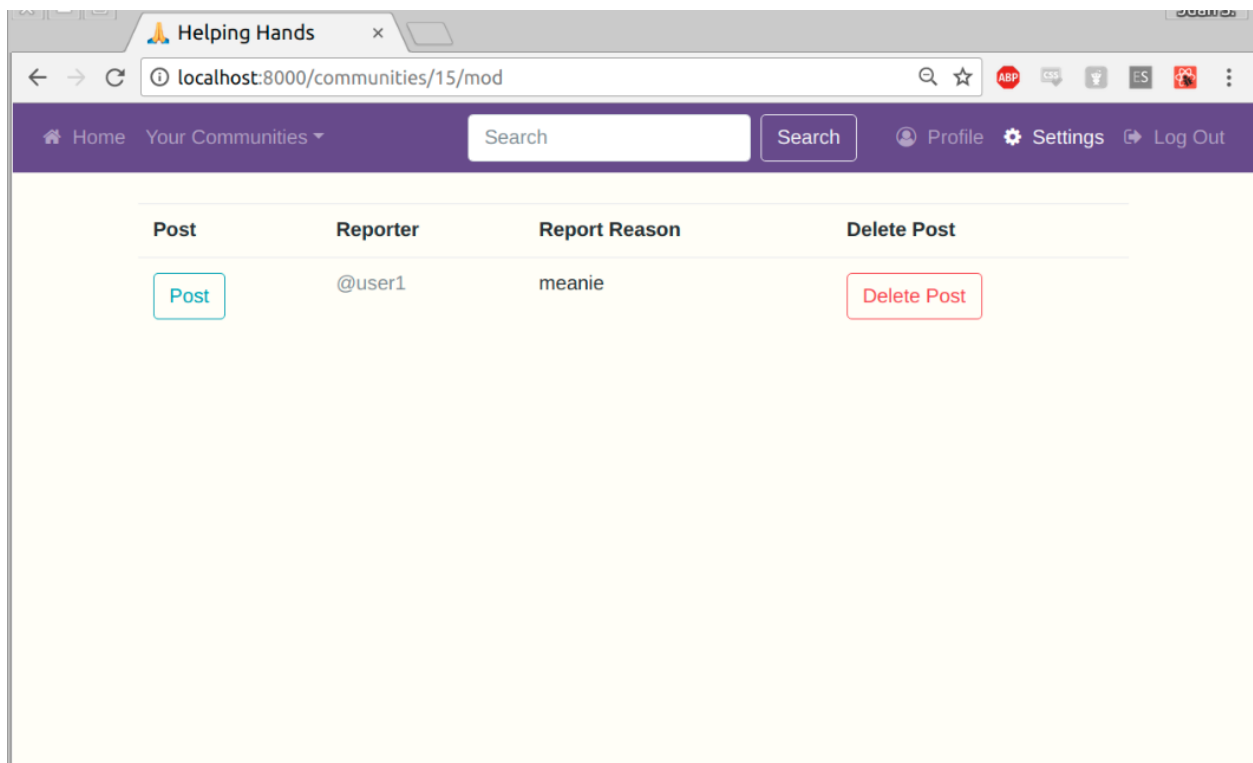
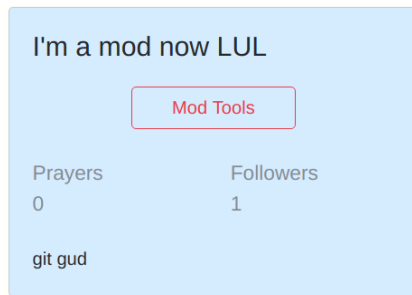
Password: Password

Confirm Password: Re-type Password

Change Password

Once accessed (through the settings view in the navigation bar), the user can change their first/last names, username, email address, bio, profile picture or password by editing the fields. Password change is confined to its own area, as well as profile picture change. Any changes to the password must be made with both passwords matching.

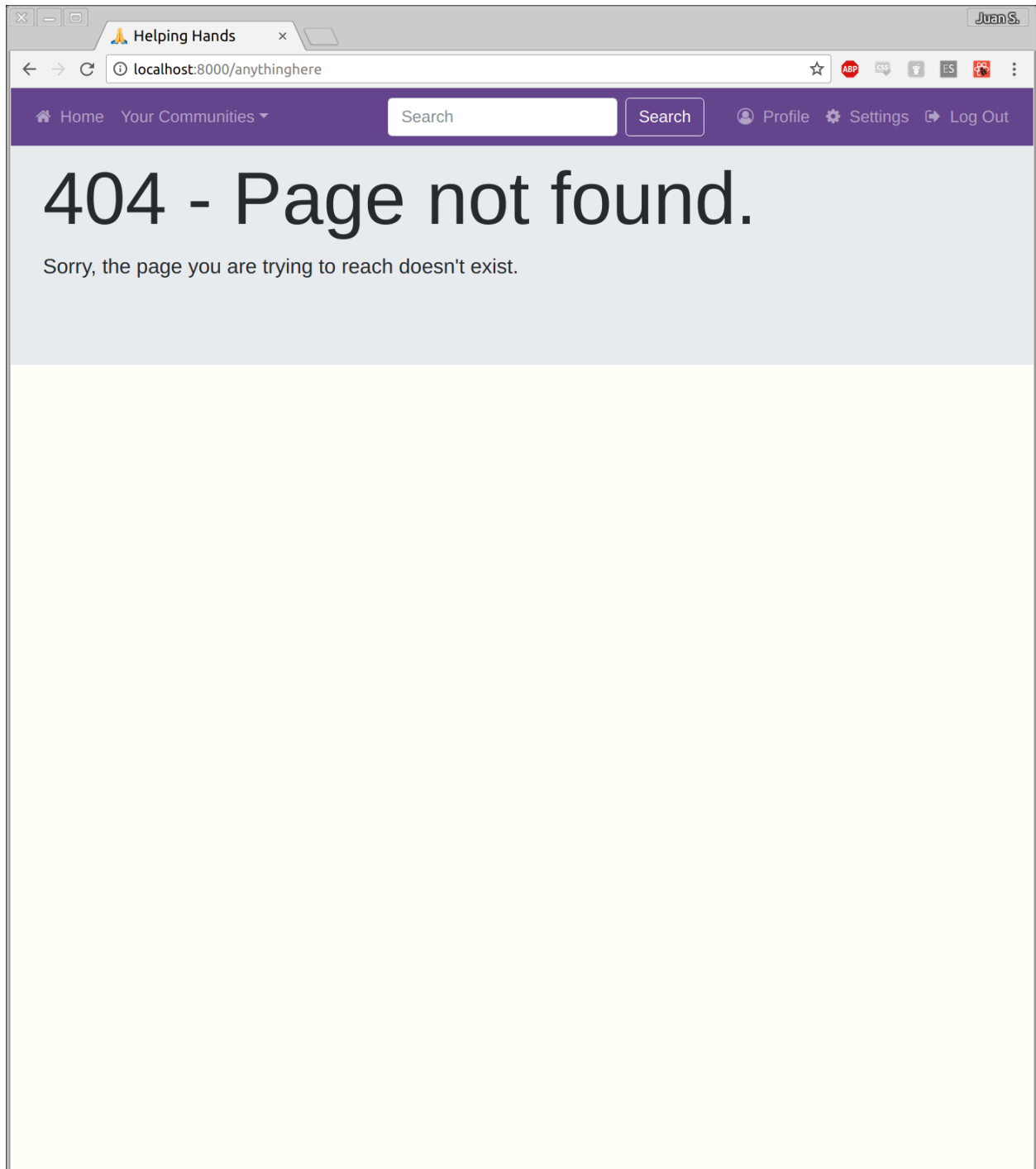
## 11.) Mod Tools



Only accessible when a user has created a community, Mod Tools is accessed by going to the community the user has created as said user and clicking on the Mod Tools button in the community card. Once in the Mod Tools view, the user can view reports on posts made in that community, and delete them if they see fit.



## 11.) 404 Page



If at any time the user enters a nonsense URL or tries to access non-existing content, they are taken to this 404 page, which they can navigate away from by using the navigation bar.

## **VI. Infrastructure Design**

### **IT Requirements**

#### **1. Server Platform (for each “server” required)**

##### **1.1. Physical system requirements**

###### **1.1.1. Storage capacity**

Due to limitations set by Marist College, we have been allotted a maximum of 50 gigs storage capacity.

###### **1.1.2. Speed requirements / response time parameters**

To ensure optimal user experience and satisfaction, the server response time must be no greater than a 2 second delay.

###### **1.1.3. Scalability plans**

Scalability is high in terms of adding more servers to support more users, more space for backups, and a more efficient maintenance period.

##### **1.2. Virtual system requirements**

###### **1.2.1. Ubuntu Linux 16.04**

###### **1.2.2. Number of images expected**

Currently we will have 1 image of the standard settings of our servers. As time progresses and we have more users and must tweak settings to accommodate more server load, we will have perhaps 2 or 3 separate images for servers having to deal with increased load, connections, rollback servers that take the last image of the most updated server

##### **1.3. Connectivity**

###### **1.3.1. Network considerations**

Traffic will be handled by our database.

###### **1.3.2. Interconnection to what other systems**

Servers will be able to connect to each other.

#### **2. Reliability**

##### **2.1. Service Level Agreements**

###### **2.1.1. Uptime Requirements**

100% uptime with back up servers for to use while the main servers are in maintenance.

#### 2.1.2. Response Time Requirements

To ensure positive user experience response time must be no greater than 2 seconds.

### 3. Recoverability

3.1. Where are things backed up? How often? We're backing up to other servers.

Backups are initiated by the hour, and as time progresses and the servers grow, the time between backups will halve until it is within the seconds. One hour -> 30minutes -> 15minutes -> 7.5minutes -> 3minutes -> 1minute -> as a person changes content.

3.2. Access to backups?

Only the admin can remote access into the server, anyone else needs physical access to the server with the correct permissions.

3.3. What data is transient and doesn't need to be stored longer term?

To preserve server space, posts older than 1 year will be archived, anything older than 5 years will be sent to heaven.

### 4. Security and Privacy

4.1. Database

4.1.1. Access controls by userid / roles

DB Admin: Full database access to users, roles, any form of database required information. Can edit entire tables.

DB Editor: Able to read files, but can only write to some files. Cannot edit entire tables.

DB Helping Hand (Someone who works for Helping Hand): View/Read Only access to database lists. NO EDIT POWERS.

4.1.2. Update vs. Access

DB Admin has update power only.

DB Helping Hand only has access to view the data.

4.2. Account information

4.2.1. User data

4.2.1.1. Username and Password are stored on a database and encrypted via database side encryption.

#### 4.3. Admin access controls

- 4.3.1. Admins have the power to create new users and delete users. Remove old/inappropriate prayers (shared between admin & moderator).

### 5. Maintenance

#### 5.1. Planned down time requirements

##### 5.1.1. Database maintenance

Database is maintained at a 99% uptime and done so with the use of backup servers. The other 1% will be used as downtime for maintenance, updates, etc.

##### 5.1.2. Times of year when IT does maintenance

January, June for now, after more servers are established, then the maintenance will be most likely weekly with no downtime.

##### 5.1.3. Times of year when the systems are not available?

99% uptime.

### Server Setup Instructions

1. Download vmware tools and access server IP addressed at 10.10.7.41. User and temp password should be given to you by Joint Study people.
2. Create VMs by clicking the Create New Virtual Machine button.
  - a. Configuration: Select Typical
  - b. Name and Location: Name your VM to be relevant to the server, if it's a database make sure to label it so.
  - c. Host and Cluster: Select HC PureFlex and under it the IP 10.10.7.30.
  - d. Storage: Select Capping-VM
  - e. Guest Operating System: Depending on your project, select Windows, Linux, or Other. In our case, Ubuntu Linux 64 bit.
  - f. Network: Leave as is.
  - g. Create a disk: However, many gigs you feel like is required for the server. Thin Provisioning.
  - h. You can assign more memory to the VM is you need it.
3. VM Setup Complete.

### Linux Configuration Instructions:

1. Open Console on a VM and mount Ubuntu 16.04 64 bit .iso file onto it.

## 2. Boot OS.

- a. Language: Preferred Language
- b. Ubuntu Menu: Install Ubuntu Server
- c. Language: Preferred Language
- d. Location: Your Location
- e. Keyboard Configuration: No, it detects on its own.
- f. Configure Your Keyboard: What is your keyboard?
- g. Keyboard Layout: Your Keyboard layout.
- h. Auto Configurations (Hardware, Network, etc)
- i. Name server addresses: You may leave blank.
- j. Hostname: Pick a hostname.
- k. Full Name for the new User: Put a name down.
- l. Username for your account: Put a username down (This is permanent)
- m. Choose a Password for the new user: Choose a password.
- n. Re-enter Password: Re-enter Password.
- o. Encrypt your home directory: No
- p. Time zone: Your Time zone
- q. Partition Disks: Guided - Use Entire Disk
- r. Disk Partition: One option, choose it.
- s. Write Changes to disk: Yes
- t. HTTP Proxy Information: Blank
- u. Managing Upgrades: No Automatic Updates
- v. Software to install: Dependent on the server.
- w. GRUB boot loader: Yes
- x. Finish Installation: Yes

### **Linux Server Instructions:**

Resources given: Assigned IP Range 10.10.7.191-193, 2 DNS nameservers 10.12.1.11-12, one gateway 10.10.7.1

1. Login to server
2. Run the command `sudo apt-get update` since this is the first time using the new server.
3. Setup connectivity using `ifconfig -a` to see which network you are configuring.

4. Enter interfaces using `sudo nano/etc/network/interfaces`
  - a. Change auto to static.
  - b. Enter your IP information using enters to separate them.
    - i. `iface interface inet static`
    - ii. `address 10.10.7.191~3` depending on which is available.
    - iii. `netmask /24`
    - iv. `gateway 10.10.7.1`
    - v. `dns-nameservers 10.12.1.11 10.12.1.12` (You may also use googles 8.8.8.8 if you want.)
    - vi. Save config and restart.
5. Use the command `sudo apt-get install [software]` to install most software (lamp, java8, etc)
6. Setup relevant information on new software, (accounts, passwords on apache or sql for example)
7. Check for connectivity via ping.
8. Do this for however many times you need you create servers (most likely three)
9. Configuration Complete