Analysis Me

**Fall**

2015

Technical Report

Service Oriented Computing (18-655)

**Team 14**

Xunrong Li

Teng Fu

Dacheng Wen

Xiaofen Liu

# Table of Contents

[Table of Contents](#hgjdgxs)

[Introduction](#hjxlccyambl5d)

[Motivation](#h8mqist5402tr)

[Related work](#huhih2n9ol9z)

[System design](#hfdv7lf9z56w)

[Frontend](#hdbuyr185kim2)

[Backend](#hf61f1vh28hxf)

[System implementation](#h9zz5zbxi0e8o)

[Frontend](#h97ajpw9tw4b2)

[Backend](#hhhh786md7mm3)

[Experiments and analysis](#hk46hcqg1o9wi)

[Upload and store photo](#h3xoyy3euwh5l)

[Stack overflow when converting object to JSON](#hiais1ib6v8bh)

[Conclusions and future work](#h8530n2o2v5le)

[Contribution of each team member](#hgkzpuh1tkjxi)

[Tutorial](#hvzsi82qud0ai)

[Appendix:](#hufiloynwvlmo)

[Github Repo](#hy7gc8wks02u)

[Readme](#hep2kqxo36kds)

[Run Backend](#hs1z4o368j4sj)

[Run Frontend](#hpw5x0qk2rxpb)

[API-Backend](#hodm6vxtxjmjh)

# Introduction

This project is going to deliver a social network exclusively used by NASA researchers to connect themselves with others.

With this social network, researchers in NASA are able to follow their interested researchers, and can be followed by others as well. So that the social network can help to enhance the connection among researches.

Besides, registered researchers are able to add their own new posts, share others’ posts, and comment and like others’ posts. In this case, researchers can use this social network to share their research progress, result, and problems, and know the research related things of other researchers. And they might be able to inspire/be inspired by others, and help/ be helped by others.

Therefore, this social network can help to facilitate the scientific researches in NASA.

# Motivation

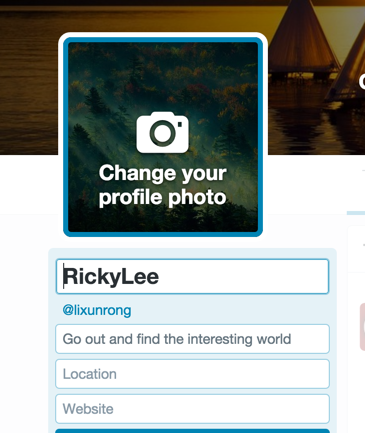
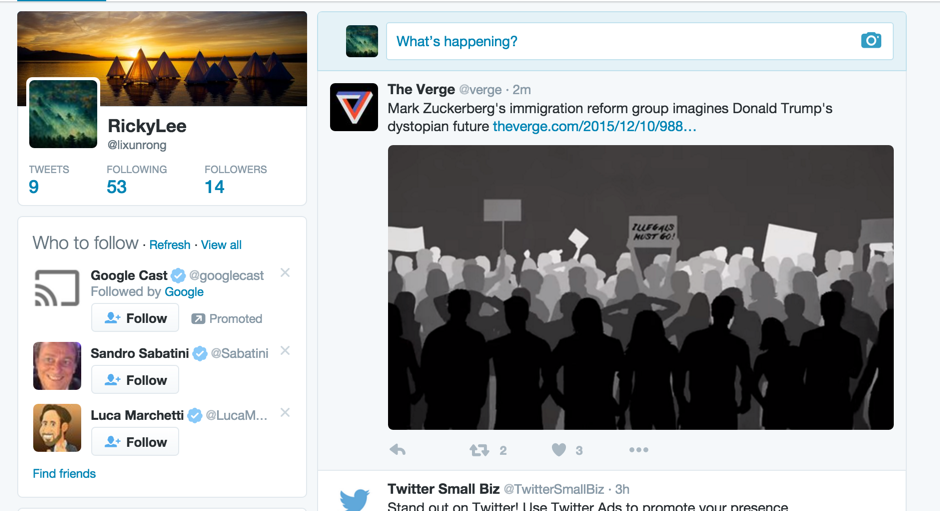
Currently, NASA doesn’t have a internal system that can be used by the internal researchers to communicate with each other and share their researches to others. In order to enhance the cooperation among researchers and facilitate the scientific researches, NASA wants to develop an internal social network which dedicate to provide a platform for the researchers to use. So that help to improve the cooperation of researchers and furthermore facilitate the researches.

# Related work

Before working on this project, we studied the 2 most popular social networks in the world: Facebook and Twitter, for two reasons. One is for the functionalities, another is for the frontend design and layout.

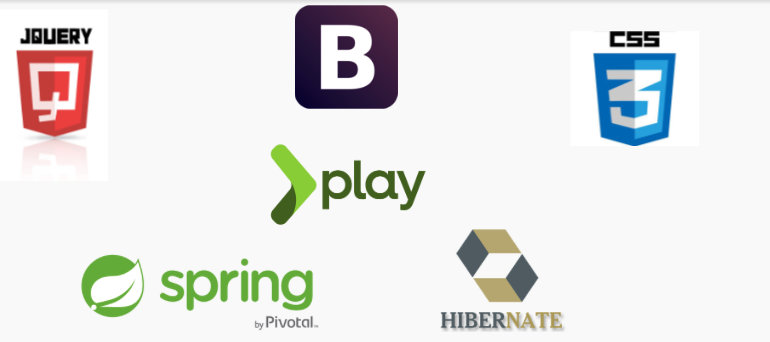
Firstly, we checked the 2 social networks for the functionalities including home page, main page, add/edit a post, share a post, following user, comment a post, like a post, search functions, share location, extra feature (add a flag layer in front of the profile photo), etc. This study gave us a sense about how to design the behavior of the functionalities that will be included in our social network.

In addition, we studied the frontend design of Facebook and Twitter. Since the UX design of Facebook and Twitter must be applied after very restricted and complex test in order to make the UI more user-friendly and efficient, studying their design helped us to know what a good and industry-style UI should be like, so studying these 2 social networks inspired us a lot about our frontend design.

# System design

We used Play framework, JQuery, CSS, Bootstrap, spring framework and hibernate in this project.



## Frontend

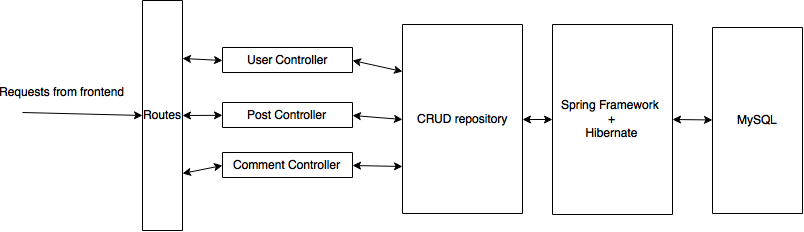
## Untitled Diagram (1).png

For the frontend, the flow image above is an overall design flow for the front-end part, which follows MVC design principle.

When user start to interact with the page or go the certain page and the client-side JavaScript will handle the user input and change the views correspondingly to give user feedback about his/her input. Meanwhile, the JavaScript also sends ajax requests to the front-end controllers. The routes defines the mappings from these request locations to controllers. These controllers will use the models that we designed to manipulate data and interact with back-end via sending HTTP requests and receiving response, mostly Json data. Based on the response and data that return from backend, controllers will parser the data and populate the views with the corresponding data using Scala template engine.

In addition to Play framework, we use jQuery to manipulate DOM element and sending ajax request, use Bootstrap and CSS3 to style the views and user interface. For the location feature, we also use Google Map APIs and HTML5 methods to get the latitude, longitude and location.

## Backend



For the backend part, our design is that the HTTP requests from the frontend will be routed by the play framework to the corresponding controller, in our case, the controllers are user controller, post controller and the comment controller. The controllers uses the CRUD repository to perform CRUD operations on the models (user, post and comment). THe CRUD repositories use spring framework and hibernate to interact with MySQL to persist the changes.

The reasons for this design is that we can leverage the advantages of play framework, spring framework and hibernate. By using these frameworks, we could mainly focus on the logic of the changes on the models, instead of how to realize the tedious database and server operations. The routing can be completed by play framework, which can associate the incoming requests to the corresponding method in the controller. The controller can use the CRUD repository to perform operation on the model and let the spring and hibernate framework to translate the changes to the database.

# System implementation

## Frontend

Models:

* User - Includes fields like userName, password, firstName, lastName, affiliation, research interests, email, hasFrontLayerPhoto, etc and their getters and setters, as well as user-related methods to send APICalls to backend includes getUserById, getUserByUserName, getFollowingUsers, getFollowers, etc.
* Post - Includes fields like id, content, user, comments, time, the number of likes/shares/comments, location, etc and their getters and setters, as well as post-related methods to send APICalls to backend includes getFollowingPosts, getUserPosts, getSharedPosts, getTopTenPosts, etc
* Comment - Includes fields like id, post, user, content, time and their getters and setters.

Controllers:

* Application - Handle the login and signup request
* UserController - Handle user-related request, e.g update user profile, etc
* PostController - Handle post-related request, e.g new post, delete post, etc.
* MainController - Handle the request to render main page, home page, search result page, etc

Views:

* home - user personalized home page
* main - main page that shows all the posts from his/her following users
* topPosts - show top ten popular posts
* other - other users’ personalized home page
* search - search result page
* login, signup and createSuccess - login, signup, create successfully page

## Backend

Model:

There are three models there are added / modified to completed this project: User, Post and Comment.

* User - Includes fields like userName, password, firstName, lastName, affiliation, research interests, email, photo content type, hasFrontLayerPhoto, etc and their getters and setters
* Post - Includes fields like id, content, user, comments, time, the number of likes/shares, the users who like/shares the post, and the comments of the post, location, etc and their getters and setters
* Comment - Includes fields like id, post, user, content, time and their getters and setters.

Repository:

In order to leverage spring framework and hibernate to access the database, there are three CRUD repository for the three models: UserRepository, PostRepository and CommentRepository.

* UserRepository - handle user-related database operation, e.g get a user by user id.
* PostRepository - handle post-related database operation, e.g get all the posts created by a user.
* CommentRepository - handle comment-related database operation, e.g get all the comment of a post ordered by timestamp.

Controller:

There are three controller to control the datas and fulfill the request from the frontend.

* UserController - Handle user-related request, e.g add, get and update user profile, etc
* PostController - Handle post-related request, e.g new post, delete post, share post, like post etc.
* CommentControlller - Handle the comment-related request, e.g new comment, delete comment etc.

View:

There is not view in the backend.

Relation between models:

In order to utilize the spring framework and hibernate, we need to define the relation between the models.

User - User:

For the follow feature: many-to-many relation, because a user can follow many users, and can be followed by many users.

User - Post:

For the like feature: many -to-many relation. A user can like many post, and a post can be liked by many users.

For the share feature: many -to-many relation. A user can share many post, and a post can be shared by many users.

Post - User:

For the ownership feature: one-to-many, because a post can have only one owner, but a user can have many posts.

Post - Comment:

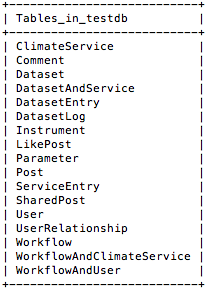
One-to-many, because a post can have many comment, but a comment can only be associated with only one post.

Comment - User:

For the ownership feature: one-to-many, because a comment can have only one owner, but a user can have many comments.

According to the relation, we add annotations to the variables of the models and let the frameworks to handle the MySQL data.

The following features show the tables created by the framework to fulfill the requirement of the backend.



For the user's’ profile image, we store the image at the backend under the folder ‘image’, we assume each user will have only one image, and therefore, the filename of the photos are the id of the user (the id of the user is unique).

# Experiments and analysis

## Upload and store photo

We attempted to store profile photo of user user in the backend by using “user id + file extension” as the filename, assuming each user will has at most one profile photo. In order to do so, we need to get the extension of the image file. However, we found the we the backend play framework receive image on the HTTP multibody part, the filename and the extension of that file will be lost, and changed to something like “multipartBody1975347307109665184asTemporaryFile”, that means we cannot get the type of the image form the filename.

Solution:

We found that we can get the type of the image directly using the method getContentType, and the return value, for example, is “image/png”, which can be used to display the image on the response. Therefore problem solved.

## Stack overflow when converting object to JSON

We encountered stack overflow error when we tried to convert post and user object to JSON object. The error message shown in the terminal seems to be a infinite loop. We found that because of the many-to-many relation between user-user and user-post, every time a user object is being converted to JSON object, the users he/she following, and the users who following him/her, and the posts he/she created and shared will be included in the JSON object. Therefore, in some cases, for example, two users, user 1 and user 2, follow each other, if we want to convert user 1 to JSON object, user 2 will also be included, and because user 2 need to be converted, user 1 will be converted again … and thus, an infinite loop is formed.

Solution:

We add some annotation to the variables of the models and use customized exclusion strategy during conversion , to ensure that there would not be any infinite during conversion process.

**Upload profile image from front-end directly**

We tried to show the image on the front-end directly as soon as user upload a image, so we have to use a tricky way instead of following MVC pattern that sending a request to the front-end controller and then communicate with backend. Instead, we try to send the ajax request from client-side JavaScript directly skipping the controllers.

In order to do this, we are actually sending a cross-domain request, which is not supported by default in the Play framework. So I have to find a way to enable to CORS filter by setting Globals configuration to the back-end server.

**Image filter and sync state at both front-end and back-end**

Borrowing an idea from facebook rainbow filter and France flag filter, We tried to implement a similar feature. In order to display it correctly, we utilized css absolute layout to take the element out of the normal flow to put the filter image on top of the original image to accomplish this feature.

In addition, we also need to sync the state of filter between front-end and back-end, so we design an extra field at User model to store the state and as soon as user choose add the filter, front-end will display the filter layer and sending the request to backend to update the field at the same time.

# Conclusions and future work

We finished all the project requirements and also implemented an additional feature that let user put a flag layer in front of their profile photo.

Beyond those features in the requirements, we list some future works below and we think the social network can be better by completing all these items.

1. Adding more advanced feature
   1. Post photo, video, external link, and other resources
   2. Support @ function
   3. Support notification
2. Polishing user interface
   1. Enable user to customize theme for the home page
3. Unit test and integration test

To increase the usability of this social network, we can add more advanced feature. For example, now we only support pure text post, but in the future, we can add support for photo, video, external link (articles), and other resources to make user be able to enrich the content of their posts. Also, it’s better to let user be able to @ others in their posts. And new post notification is another good idea to improve the usability.

Continuously polishing the UI is the second way to increase this social network. Based on our current UI design, we can support more fancy UI features such as customized home page theme to make the social network more beautiful.

In addition, doing more unit test and integration test can help us to make the APIs more stable, so it’s a good way to improve our project.

# Contribution of each team member

Team leader : Xunrong Li

Responsibility :

1. Divide and assign team work to other teammates.
2. Frontend development for social network
3. Writing technical reports and preparing presentations for sprints as well as finals.

Team member 1 : Xiaofen Liu

Responsibility :

1.Frontend development for social network

2.Writing technical reports and preparing presentations for sprints as well as finals.

Team member 2 : Teng Fu

Responsibility :

1.Backend development for social network

2.Writing technical reports and preparing presentations for sprints as well as finals.

Team member 3 : Dacheng Wen

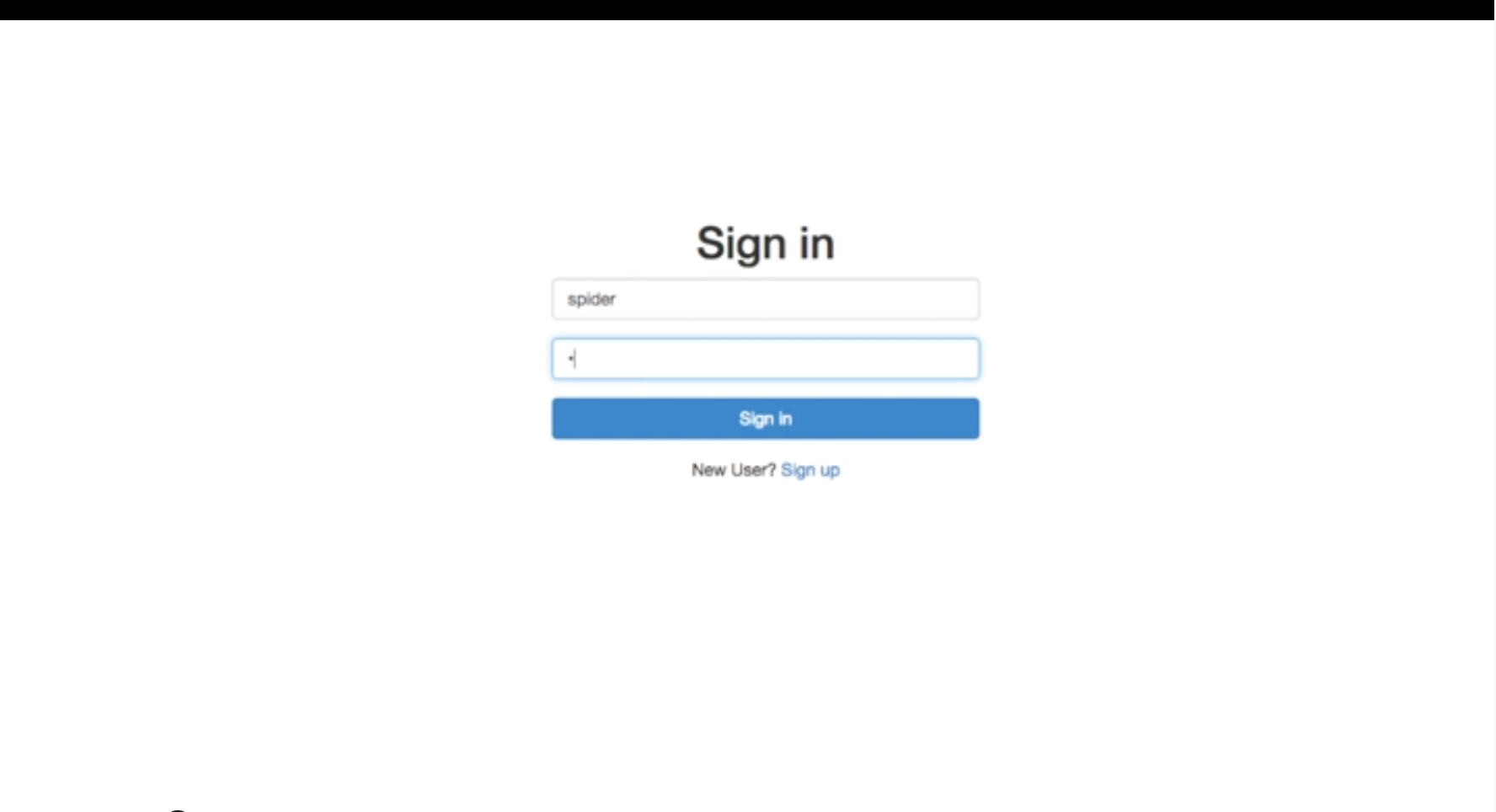
Responsibility :

1.Backend development for social network

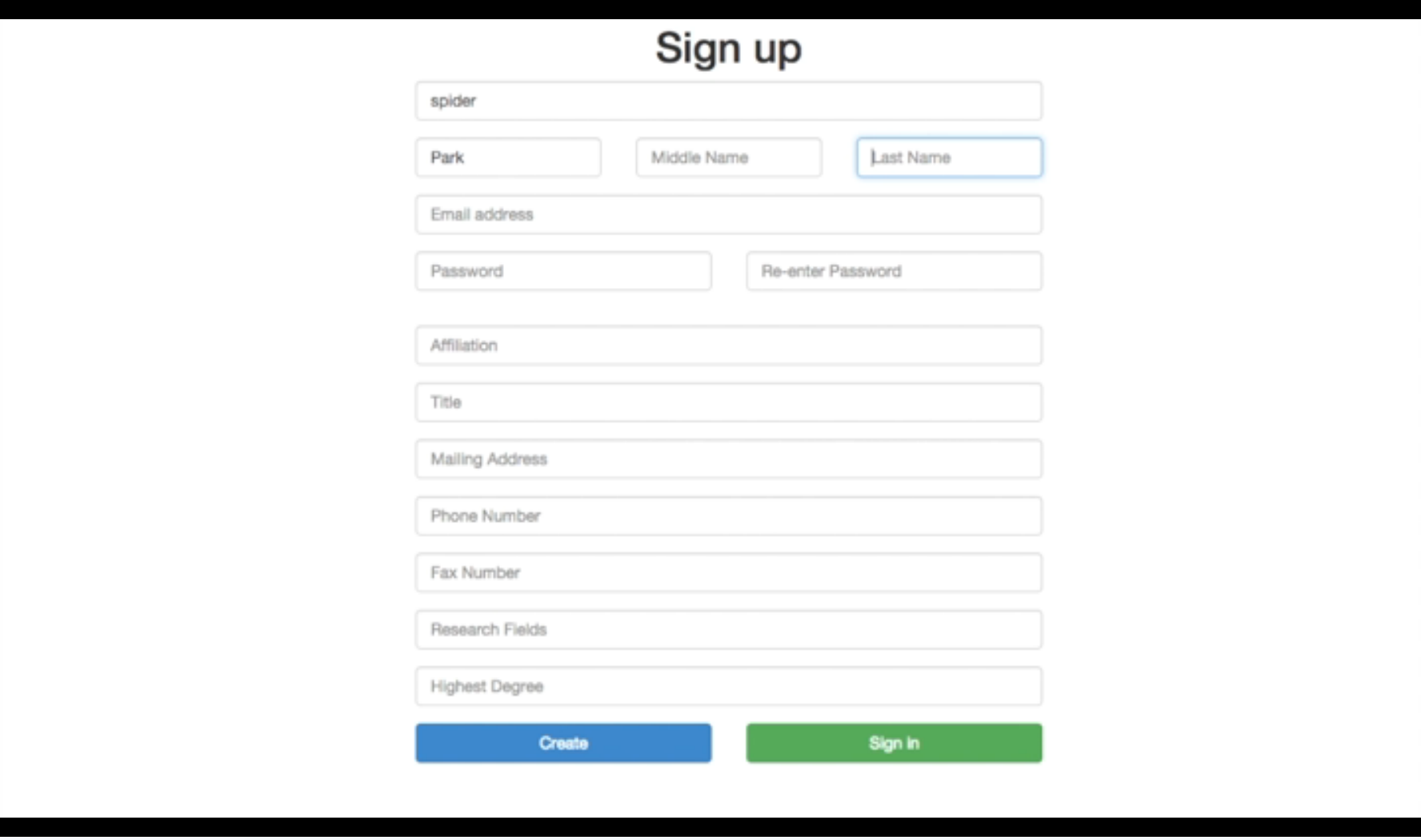
2.Writing technical reports and preparing presentations for sprints as well as finals.

# Tutorial

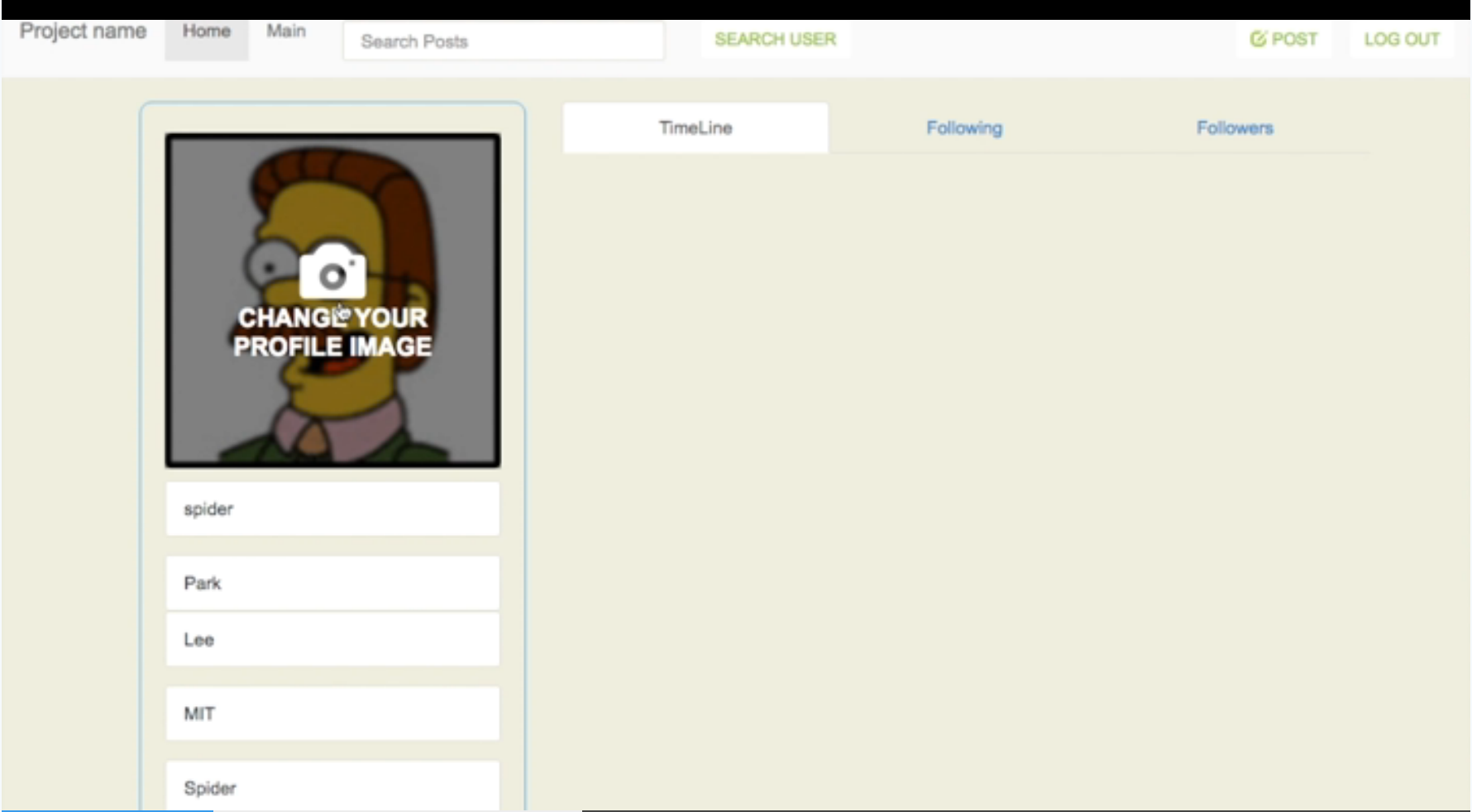
1. In the login page, login if you already have an account or register one if you don’t.



Register an account



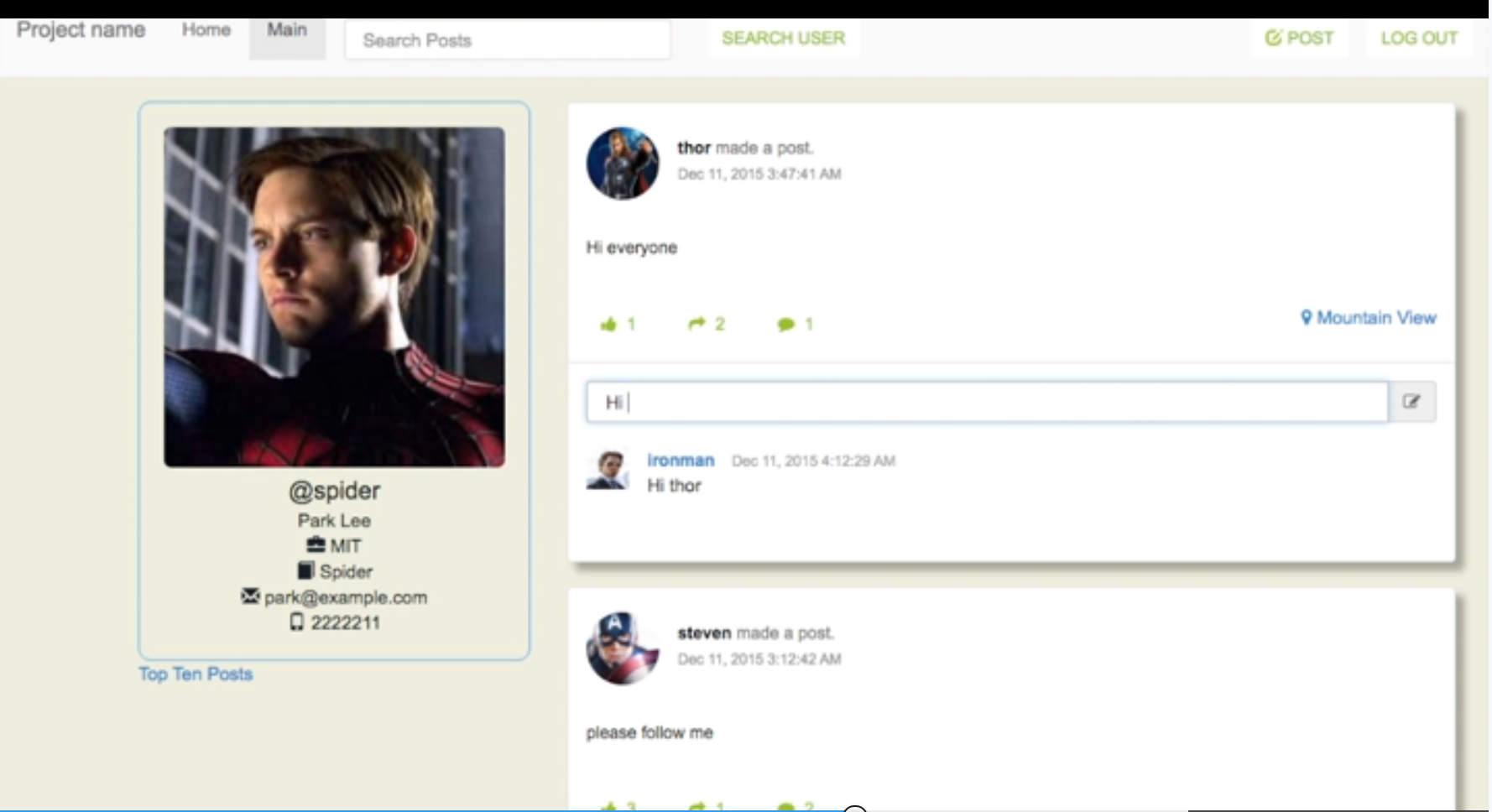
2. Once log into the site, you are automatically located to home page. You can choose to change your profile logo by clicking on the camera button on the left side.



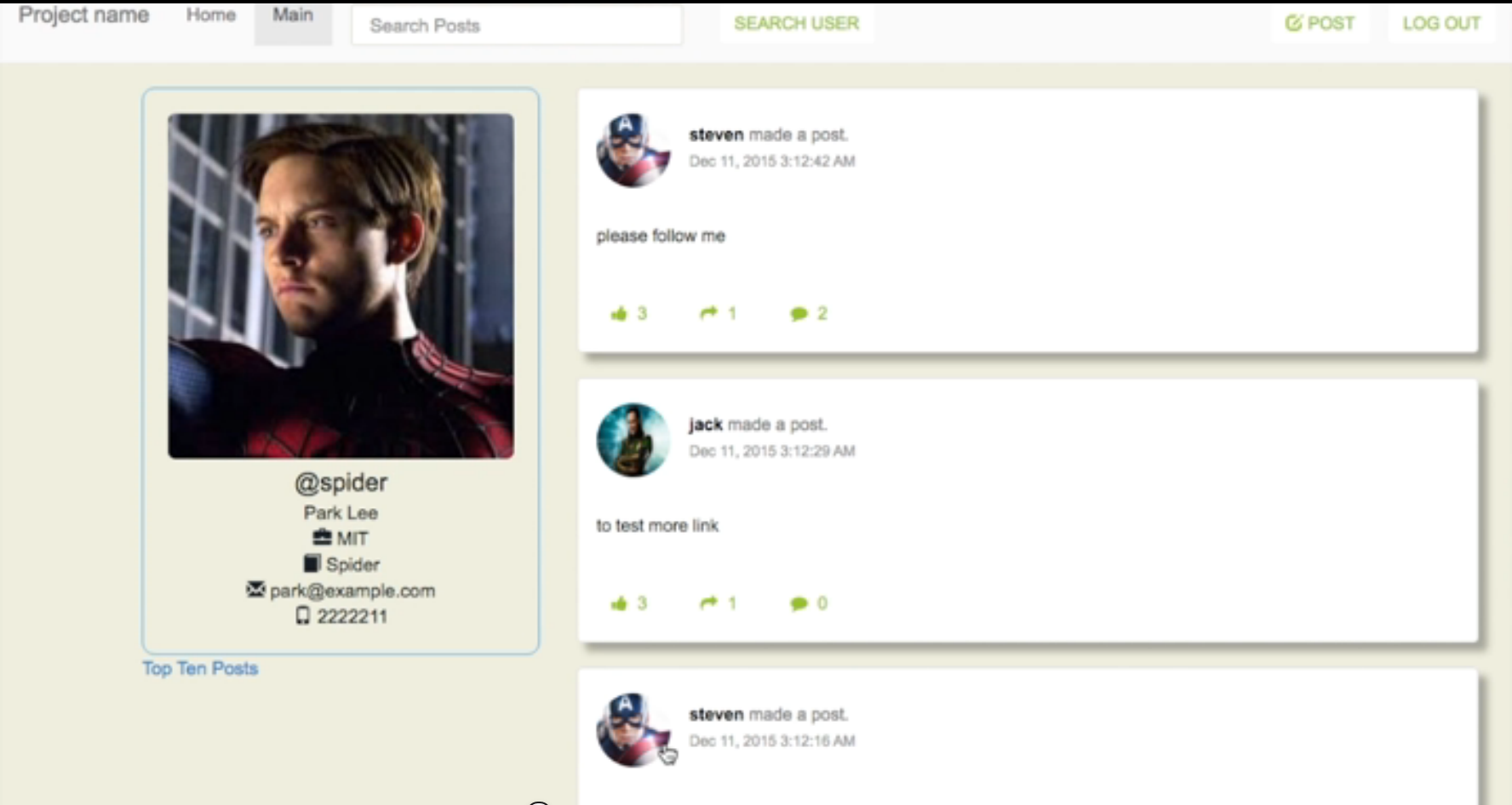
3. In the home page, you can view all of the posts that are posted by either you or people you follow. You can also check people you follow and people following you.

# 

You can leave a comment or like other people’s posts.



4. You can also switch to main tab, where you can view all posts from all users who registered in this site.

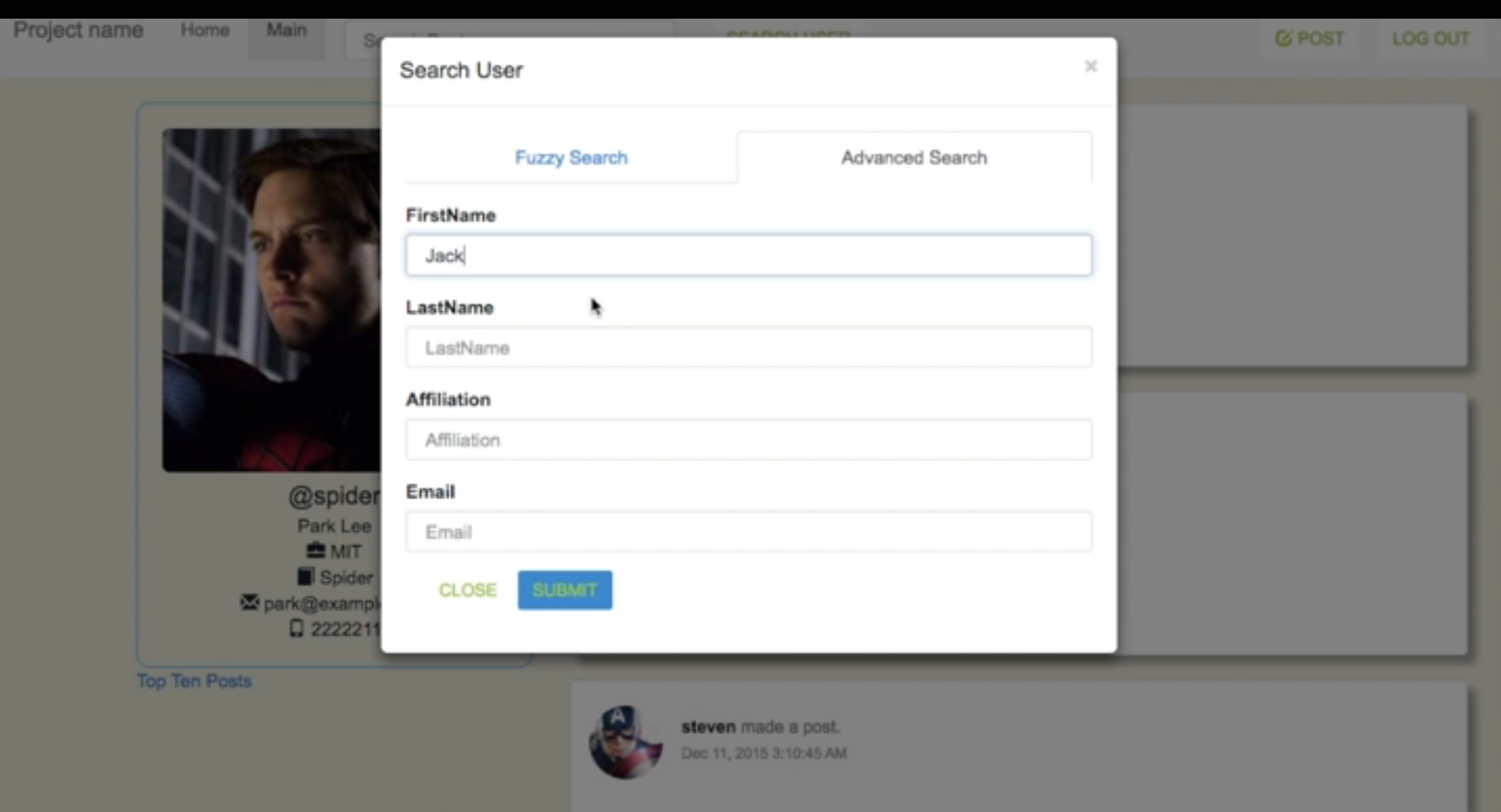


5. In the navigation bar, you can also do search. There are two different kinds of search. One is the Fuzzy search, another is the advanced search.

In fuzzy search, you can just search people based on their name.

# 

In advanced search, you can search based on first name, last name, email and affiliation, which enables user to find results more accurate.

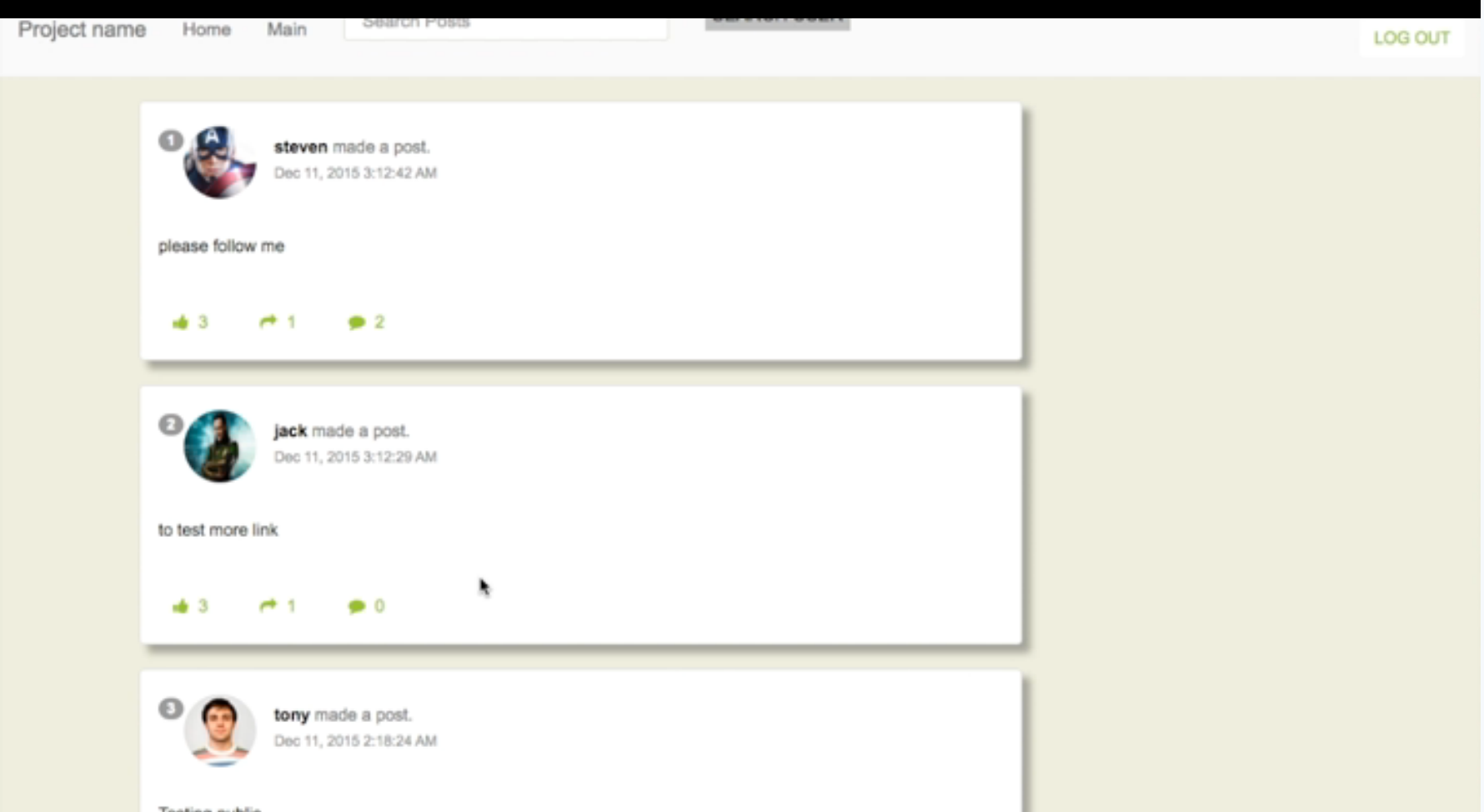


6. Of course you can publish your own post. In the navigation bar, click on new post and you will be navigated to a input box.

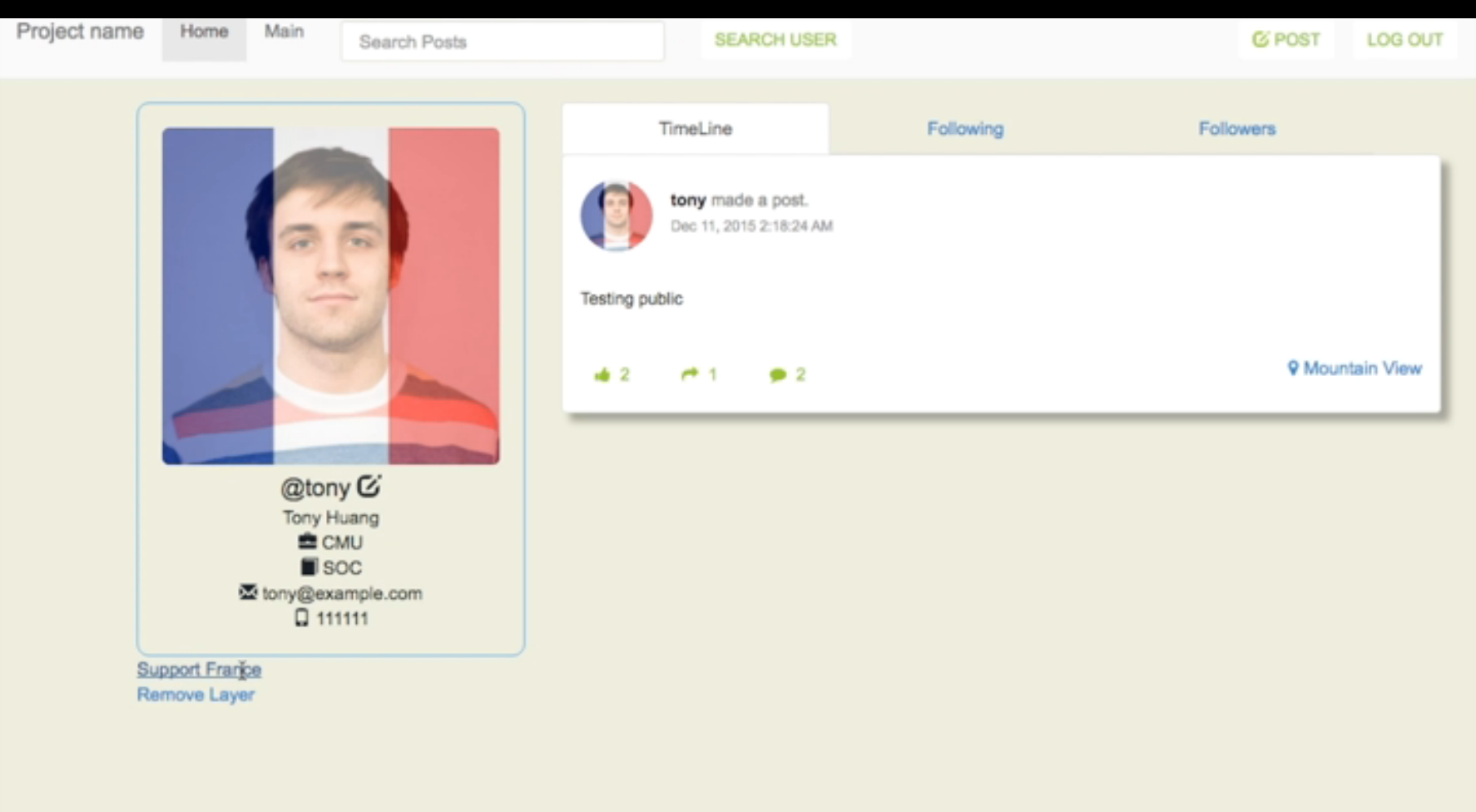
You can write down some text in the input box. If you choose to expose your location, you can click on get location button. If you want your post to be a private one, you can toggle on the private button. Then you can choose to post it out or just cancel it.

# 

7. You can also view the top ten posts on this site by clicking the link under the profile pane in main page.



8. Finally, you can add a layer to your profile page to support France. Just click on the support France button under the profile pane in your home page. You can remove it by clicking on remove layer button if you don’t want the layer anymore.



# 

# Appendix:

## Github Repo

<https://github.com/cmusv-sc/SOC-Fall-2015-Team14-Lead-Xunrong-Li>

## Readme

## Run Backend

Go to the backend folder, run './activator "run 9034"'

## Run Frontend

Go to the frontend folder, run './activator run'

## API-Backend

