CSC309

CSC309 2015 Summer

GoodTeam: Looking for partners

Group Member

Baige Liu `g5liubai`

Zhiyu Kang `g5garyk`

Xiaoyu Yan `c4yanxia`

Git Url

https://github.com/ericxyan/csc309

Demo video Url

http://youtu.be/mISL8h5mf4c

Description

GoodTeam is a platform to gather students specializing in different fields to work together on some cool ideas. University students always come up with some fantastic ideas in their study, but in many cases they do not have enough knowledge, skills or time to achieve them. Even though some of the knowledges are elementary levels, it's not necessary for a student to spend too much time to study all of the required skills, because someone in the relevant majors may have learned it and they also want to do something to practice. If we can gather these students together, it would be wonderful, because everyone can just focus on their major-related knowledge to make a project. In this way, we can not only bring many ideas to real but every member in the group can practice what they have learned, gain real professional work experiences, and develop their teamwork abilities. When they are applying for a real job, companies can take their records on GoodTeam as a good reference.

How it works

Users introduce themselves in their profile and set up their relevant majors and mastered skills. A initiator can be a student who has a cool idea and want to find someone to work together on it or a student who just want to do a classic project to practice what they have learned. A initiator publishes a description of the project on the platform with requirements of skills. This post will notify all students who have set up the relevant skills in their profiles. Then if someone is interested in the project, he or she can apply for it. The initiator chooses partners according to the applicant's major, project experiences and comments from the applicant's earlier partners. When the project finished, group members rate each other and leave some comments.

Challenges for GoodTeam

Ideas protection might be a challenge for GoodTeam, since someone may not want to publish their ideas on the public platform or someone may steal one's idea to profit. We may allow initiators to just publish a topic or skill requirements, but this would result in losses of applicants.

Features

Our project is to built a web application to implement the GoodTeam platform. The following is a list of the features:

- **Single Page Application**: We design our website as a single page application that we use `ajax` to change the content of page based on user's requests.
- **User Authentication**: Verifying user's account using passport module. We will justify this in the authenation section below.
- **Login credential encrption**: We encrypted both password and the cookie session to make sure login infomation is secured: 1) express-session is used to encrypt the session so that user cannot modify it. Also, the passwords are all encrypted by bcrypt-nodejs module.
- **User Profile**: Each user has a profile including privacy informations, emails, majors,

skills, experiences, current projects and rating history.
- **User Interactions**: A user can post a project and becomes a project initiator. Users can read the description and requirement of projects and submit applications to initiators. Initiators can accept any number of applications to form a group. Only project's authorized user can modify the project.
- **Implicit Social Networking**: Users in a same project are automatically considered as friends. They can check each other's information that only friends have the privilege.
- **Reputation System**: Users can comment a posted project. Also they could rate each other. In this case, both project and users has reputations that can be viewed by others.
- **Search and Recommendation System:** Projects could be searched based on their name and tags, the result would be displayed correspondingly. Also we can recommend users having different skills.
- **User Administrative View**: The project administrative view allows initiators to check the applicant list of a project and add or remove group members.
- **Administrative view**: The admin user has access to Admin pages that has full authorization to all datas in the website.

Structure

We are using MEAN stack to build this project.

- `Design Pattern` : MVC.
- `MongoDB` : Database
- `Express.js` : Controller, Driver
- `Angular.js` : Controller, Template
- `Node.js` : Server
- `Bootstrap` : Style
- `Library` : mongoose, passport, valdr, bcrypt-nodejs
- `Unittest` : supertest, should, mocha

Database Schema

We are going to use mongodb with mongoose module in our project. The database saves all informationsabout our website, and provide functionality of all database operations. Whenever the pages need to interact with database, this module should be used. The followings are our database schemas:

- `User` : { UserID, PassWord, NickName, Email, Ceil, Skills(list of skills), Rating(list of ratingID))}
- `Project` : { ProjectName, Description, Subjects(list of string), Start_time , Status, Admin, Member(list of ObjectID ref to User), Member(list of ObjectID ref to User), Comments(list of commentID) }
- `Rating` : { RatingID, RaterID, Stars, Comments}

Module Design

Authentation module. (implemented)

We use passport-local and passport-google-Oauth2 strategies as our login strategies. We serialize and store the encrypted login session using passport-session. We build isAuthenticated middleware in express to control the access. This would be detailed discussed later.

Main page search module (implemented)

The Main page search module is responsible for the display on main page. At first, it would display the current 10 projects, and you can use the tags at left to show users with given skills. Also if you want to do other searches, you can use the up-side panel.

Panel module (implemented)

The panel module provides functionality of upper side panel for all pages. This module takes the session information from authenation module, provide and display the corresponding functionalities that the user now can choose on the panel. Also, there is a search bar that can search the name of user and projects, it would redirect user to the result page that contains user and projects with keywords.

Profile Module (implemented)

The profile module is aimed to display the user's infomation. It would show the user's basic infomation, user's projects, and the ratings for this user. Based on user's access authorization, the displayed content would be different. Only the owner of this profile has access to private content and can modify the content of his infomation.

Project Module (implemented)

This module is responsible for managing project informations. The content would be varied depended on user's access. Only admin and initiators can modify the content of the project, other user can view and apply for the projects. After the project is finished, all members can rate each other in this project using rating module.

Rating Module (implemented)

This module is used to rate and comment a user or project. Project members can use this module to rate each other when the project is finished.

Recruitment Module (implemented)

The project page will use this module. If the request comes from a publisher, this module displays all the applications so that the publisher can accept or refuse an application. Publishers can also delete a member before the project finished. If the request comes from a normal user and the project's state is still in recruiting, this module will display apply options, users can apply for the relevant positions as the publisher's requirments.

SignUp Module (implemented)

SignUp module provide the functionality that new users can signup via it. Validator is used here to control the input of user's infomation.

Administrative Page (implemented)

There is an administrative page that logged in admin user can access using `'/admin'` . He has premission to delete any projects or users here, and he can view the total number of projects and users. Other users would be redirect to main page if they are trying to request this page.

REST API Design

Search

- `GET /api/search/skill/:skill` : Take one parameter skill, would send all user json with that skill.

- `GET /api/search/Admin/:userID` : Take one parameter userId, would send all project json that admin is that user.
- `GET /api/search/Memeber/:userID` : Take one parameter userId, would send all project json that the user is the member of it.
- `GET /api/search/Candidate/:userID` : Take one parameter userId, would send all project json that the user is the candidate of it.

Users

- `GET /api/users/:userid` : Get the user json with UserId equals to the parameter userId.
- `GET /api/users/name/:nickname` : Get the user jsons array that users' name contains paramerter nickname.
- `GET /api/users/:id/:pwd` : Get the user json with parameter id and pwd. Verifying if the id and pwd are valid.
- `PUT /api/user` : Update user's infomation to database, should pass the user's json in the body of request.
- `POST /api/user` : Post a new user to database, should pass the user's json in the body of request.
- `DELETE /api/user/:id` : Delete the user with given id in the database, send the prompt string back.
- `GET /api/users/valid/nickname/:nickname` : Check wether the nickname is valid.CSC309

Projects

- `GET /api/projects/` : Get all projects in db, send them as an array of json.
- `GET /api/projects/:id` : Get the project json with given _id.
- `GET /api/projects/name/:projectName` : Get all projects json in array where their name contains given projectNmae parameter.
- `PUT /api/projects/:id` : Update the project with given _id, should pass the project json by request's body. Will send the updated project json back.
- `POST /api/projects` : Post the new project that's in the request's body to database, return the json of the project after saved.
- `DELETE /api/projects/:id` : Delete the project with given id, return the result of deletion.

Rating

- `GET /api/rating/:id` : Query the user with given id, and get all that user's ratings.

- `POST /api/rating/:userId` : Add a new rating to the user with userId, the rating json is passed by request's body.
- `DELETE /api/rating/:userid/:ratingId` : Delete the specific rating by given userId, ratingId.

Authentication

- `GET /auth/success` : Send the json of the success state and the user json.
- `GET /auth/failure` : Send the json of failure state and the message.
- `POST /auth/login` : Post the login infomation, redirect user to different pages based on the authentication middleware.
- `GET /auth/google` : Get the google authentication middleware.
- `GET /auth/google/callback` : The callback api used by google after authenticated.
- `POST /auth/signup` : Useing singup middleware, and redirect user based on user's request content.
- `GET /auth/loggedin` : Check if the user is logged in, send the user json if logged in. Otherwise send '0'
- `GET /auth/signout` : Sign out and destory the session.

Secure Security vulnerabilities

Local Strategy and Passport-session

The local strategy provide use the basic authentication middleware that can check user's login infomation. By different conditions, it can redirect users to different pages. Also it build up user's authentication cookie using passport-session if successed. The cookie session is encypted by the given key in our backend so that it is hard for user to mock a fake session to access to unauthorized content. Also, the middleware is used before the api requests so that whenever the user has a request, they are checked by authentication middleware at first.

Google Strategy (third-party)

The google-Oauth2 Stragety is our third-party login strategy. We obtain the api and tokens from google, and whenever users are trying to login with this strategy, they are going to be redirected to google at first. After they loggin as google+ user, the callback function would redirect the user back, and send the profile json of the user back to us. We grab the infomation that we need and update them into database. For these users, their google

OpenID is the UserId in our databse that is used to identify them.

Encyping user credentials

The cookie is encrypted by passport-session so that the cookie session is secured. Also, even the attacker decrypt the cookie session or they get user's login information in some ways. The passport of users are also encrypted by bcrypt-nodejs as the second protection here. As a result, the credentials are scured

Authorization

We have the authorization fields for all projects and user infomation. We check whether users could have permission in controllers so that some actions are not provided (like modification) if the user is not premitted.

NoSql Injection

Since we are using mongoDB, there are no SQL statement in it. Also, we avoid to use any syntax that accept any javascript expression. In this case, we prevent injecting scripts in to database queries.

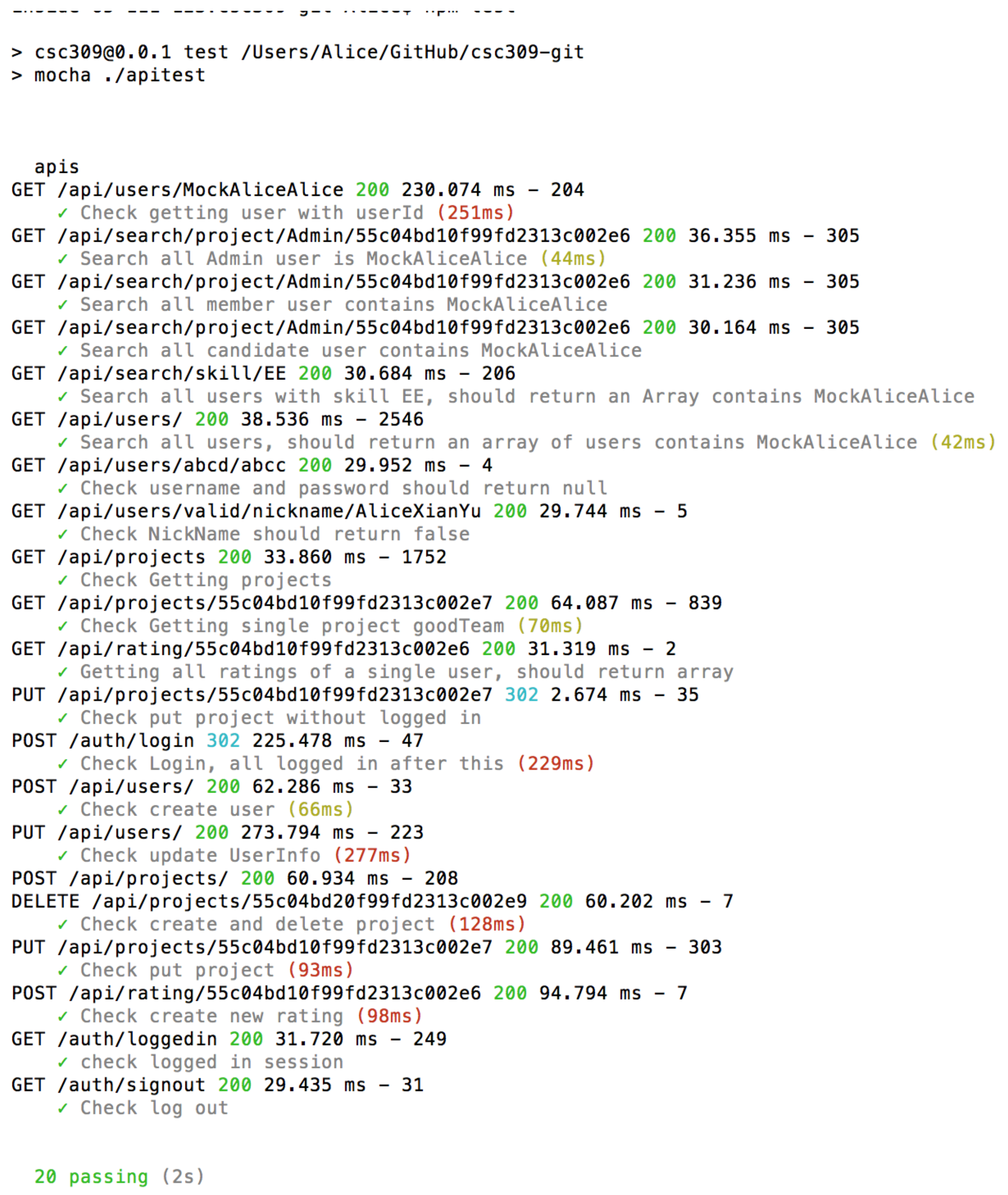
Other Tests

Unittests

We use mocha, supertest and should modules to do the unittest of our apis and authentication. There are 19 unittest total, and they are all passed. The unittest file is `apitest.js` . In order to run the unittest, run `npm test` .

performance

The performance of apis could be checked in mocha unittest result. You can get the mocha test result by `npm test` . Below is the result we have.



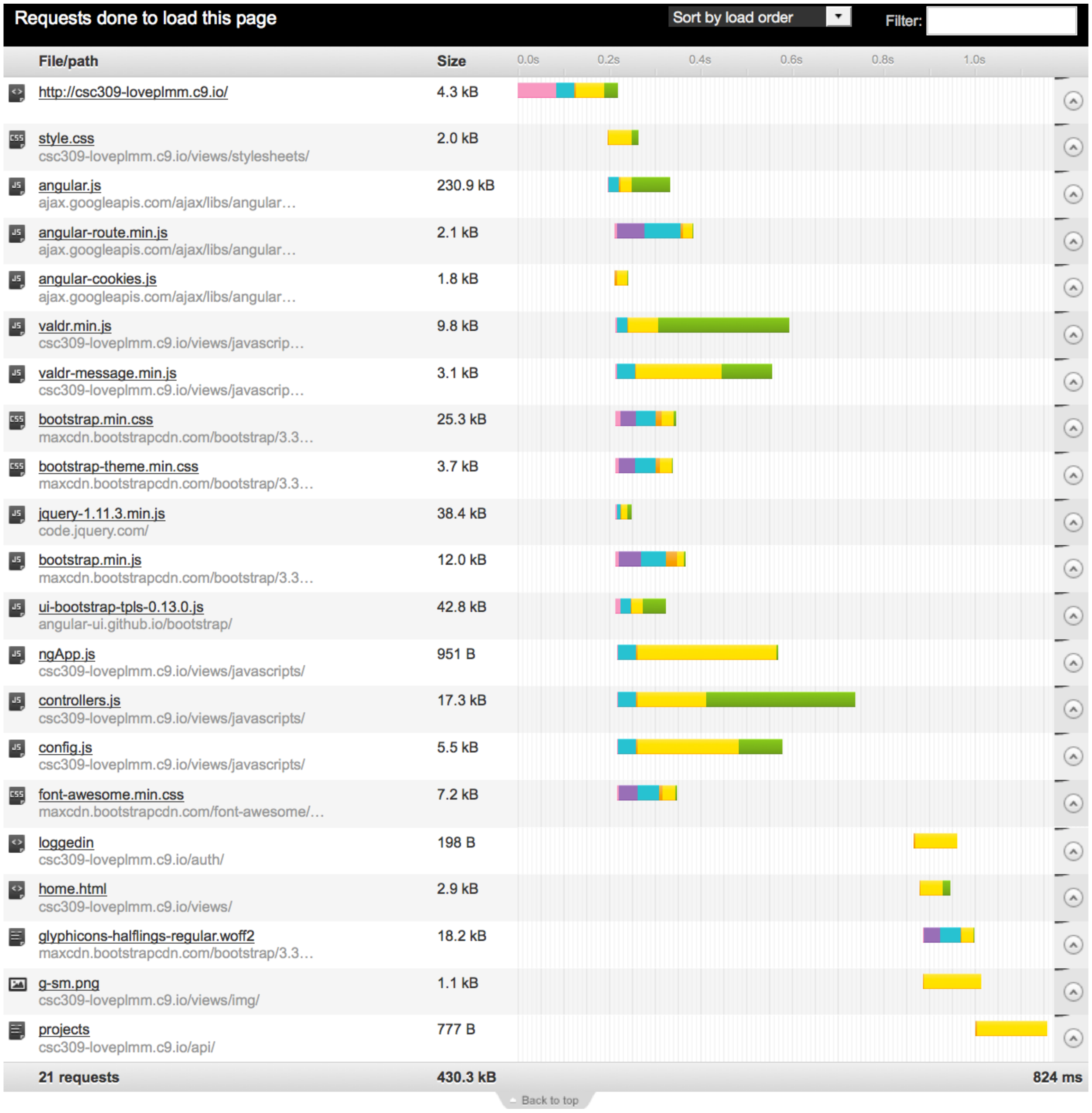
```
> csc309@0.0.1 test /Users/Alice/GitHub/csc309-git
> mocha ./apitest



  apis
GET /api/users/MockAliceAlice 200 230.074 ms - 204
    ✓ Check getting user with userId (251ms)
GET /api/search/project/Admin/55c04bd10f99fd2313c002e6 200 36.355 ms - 305
    ✓ Search all Admin user is MockAliceAlice (44ms)
GET /api/search/project/Admin/55c04bd10f99fd2313c002e6 200 31.236 ms - 305
    ✓ Search all member user contains MockAliceAlice
GET /api/search/project/Admin/55c04bd10f99fd2313c002e6 200 30.164 ms - 305
    ✓ Search all candidate user contains MockAliceAlice
GET /api/search/skill/EE 200 30.684 ms - 206
    ✓ Search all users with skill EE, should return an Array contains MockAliceAlice
GET /api/users/ 200 38.536 ms - 2546
    ✓ Search all users, should return an array of users contains MockAliceAlice (42ms)
GET /api/users/abcd/abcc 200 29.952 ms - 4
    ✓ Check username and password should return null
GET /api/users/valid/nickname/AliceXianYu 200 29.744 ms - 5
    ✓ Check NickName should return false
GET /api/projects 200 33.860 ms - 1752
    ✓ Check Getting projects
GET /api/projects/55c04bd10f99fd2313c002e7 200 64.087 ms - 839
    ✓ Check Getting single project goodTeam (70ms)
GET /api/rating/55c04bd10f99fd2313c002e6 200 31.319 ms - 2
    ✓ Getting all ratings of a single user, should return array
PUT /api/projects/55c04bd10f99fd2313c002e7 302 2.674 ms - 35
    ✓ Check put project without logged in
POST /auth/login 302 225.478 ms - 47
    ✓ Check Login, all logged in after this (229ms)
POST /api/users/ 200 62.286 ms - 33
    ✓ Check create user (66ms)
PUT /api/users/ 200 273.794 ms - 223
    ✓ Check update UserInfo (277ms)
POST /api/projects/ 200 60.934 ms - 208
DELETE /api/projects/55c04bd20f99fd2313c002e9 200 60.202 ms - 7
    ✓ Check create and delete project (128ms)
PUT /api/projects/55c04bd10f99fd2313c002e7 200 89.461 ms - 303
    ✓ Check put project (93ms)
POST /api/rating/55c04bd10f99fd2313c002e6 200 94.794 ms - 7
    ✓ Check create new rating (98ms)
GET /auth/loggedin 200 31.720 ms - 249
    ✓ check logged in session
GET /auth/signout 200 29.435 ms - 31
    ✓ Check log out


  20 passing (2s)
```

Performance test for main page:



File/path	Size
http://csc309-loveplmm.c9.io/	4.3 kB
style.css csc309-loveplmm.c9.io/views/stylesheets/	2.0 kB
angular.js ajax.googleapis.com/ajax/libs/angular…	230.9 kB
angular-route.min.js ajax.googleapis.com/ajax/libs/angular…	2.1 kB
angular-cookies.js ajax.googleapis.com/ajax/libs/angular…	1.8 kB
valdr.min.js csc309-loveplmm.c9.io/views/javascrip…	9.8 kB
valdr-message.min.js csc309-loveplmm.c9.io/views/javascrip…	3.1 kB
bootstrap.min.css maxcdn.bootstrapcdn.com/bootstrap/3.3…	25.3 kB
bootstrap-theme.min.css maxcdn.bootstrapcdn.com/bootstrap/3.3…	3.7 kB
jquery-1.11.3.min.js code.jquery.com/	38.4 kB
bootstrap.min.js maxcdn.bootstrapcdn.com/bootstrap/3.3…	12.0 kB
ui-bootstrap-tpls-0.13.0.js angular-ui.github.io/bootstrap/	42.8 kB
ngApp.js csc309-loveplmm.c9.io/views/javascripts/	951 B
controllers.js csc309-loveplmm.c9.io/views/javascripts/	17.3 kB
config.js csc309-loveplmm.c9.io/views/javascripts/	5.5 kB
font-awesome.min.css maxcdn.bootstrapcdn.com/font-awesome/…	7.2 kB
loggedin csc309-loveplmm.c9.io/auth/	198 B
home.html csc309-loveplmm.c9.io/views/	2.9 kB
glyphicons-halflings-regular.woff2 maxcdn.bootstrapcdn.com/bootstrap/3.3…	18.2 kB
g-sm.png csc309-loveplmm.c9.io/views/img/	1.1 kB
projects csc309-loveplmm.c9.io/api/	777 B
21 requests	**430.3 kB**

824 ms

How did we improve

- Our site is single page application so that there are several controllers that can be shared across different contents, and they don't need to be load several time.
- We use service global variable provided by angularJS to store those data that we are going to re-use. In this case, we avoid those unnecessary database queries to enhance the speed of our website.

Security

Testing the XSS with Create a new project with description "", and try to view the content of this project. The result is the XSS javascript is not working. The reason of this is our website didn't have any API requests that returns javascript, everything there are just plain text. The javascript cannot be executed for sure.