CSC309 Community Fund Phase IV

g3aishih, c2laitim

Introduction

Install:

npm install - Get the dependencies
node server.js – Start the server

The website is hosted on localhost:8080
I had trouble with the AWS servers and was told by the professor to run on localhost

Changes:

In phase IV of the project, we've decided to drastically change our methods and architecture as well as some features. The two most noticeable changes are:

MySQL → MongoDB
React.js, Socket.io → Angular.js

We've replaced MySQL with MongoDB and React.js, Sockiet.io with Angular.js
MongoDB was a lot easier to use for this project because every query is responded with data in JSON format and Angular.js was easier to work with in our opinion.

Features and functionality

General

- Error checking is done on all forms
- Site is user friendly and does not require any explicit knowledge of websites and computers to use

Administrative

- Sign up
- Log in
- Log out

User

- Add/Edit location and interest preferences
- Edit account information
- See other users in your community and their reputation
- Like/Dislike a user in your community
- Leave chat messages on user profiles
- Delete chat messages on user profiles if you are the one who wrote the message or if you are the owner of the profile
- Keep track of the date a user joined

Projects

- Create a project
- Find projects with that meet your preferences
- Fund another user's project
- Like/Dislike another user's project
- View a project in detail
- Leave comments on projects
- Delete commends on projects if you are the one who wrote the message or if you are the initiator of the project
- Remove your own project
- Description of a project is not shown on main page as the projects are displayed as table rows and long descriptions will result in poor formatting
- Keep track of the date the project was created
- View total number of projects on the server
- View total amount of funding across all projects on the server

Features missing due to time and manpower restrictions

- Keep track of the days to reach a goal
- Display the total number of funds a user has given
- Display the total number of projects that a user has initiated
- Keep track of the total number of projects that have reached their funding goals
- Keep track of the total number of projects that have no reached their funding goals
- Keep track of the total number of projects in the user's community
- Sorting projects by title, date, likes/dislikes
- Like/dislike a comment

Remove feature from phase III

- Retrieve lost password

Software Architecture and high level design

This website was made using the MEAN stack.
M – MongoDB
E – Express
A – AngularJS
N – NodeJS

MongoDB – Serves as our database. Keeps information by following a schema that we designed
Express – Our routing component, handles the different requests sent to the server
AngularJS – The component in charge of the front end
Node JS – Backend RESTful framework

System hierarchy

- csc309-community-fund
    - /node_modules – Dependencies used by the website
    - /src
        - /controllers – Javascript that controls the behaviour of specific web pages
        - /html – Front end of all the web pages
        - /models – Defines the schema for the User and Project objects in our application
        - /packages – Additional dependencies (e.g Bootstrap)
        - /routes – RESTful API router
    - server.js – Server file

How it all works:

When the user first enters, the server sends the root page. It checks if there exists a cookie in the user's browser and tries to identify the user based on the cookie. If they can be identified, they are automatically logged into the main page, else they are given the choice to login or sign up.

Many buttons in the application open modals for the user to interact with. This was a simple design choice we made as it seemed cleaner than to make a separate html page for each component of the application.

When on the page, the page controller is responsible to handle all click and events that occur from the main page, such as opening a modal. When opening a modal, we require a modal controller that controls the behaviour of that specific modal. When data is entered in the modal, it is checked for error before it is submitted. After it is submitted, it will be passed as a JSON object into a function that sends a GET, POST, PUT, or DELETE request to the server with the JSON object. The server responds to the request and does whatever it's supposed to do and responds with another JSON object. At this point the server has done its job and we have a JSON object at our disposal. We can do whatever we need to do with the JSON object and refresh the page to see any changes that should have occurred.

Information representation

There are only two objects in our application, Project objects and User objects

Project Schema:

```
title: {
        type: String,
        required: true
},
description: {
        type: String,
        required: false
},
funds: {
        raised: {
                type: Number,
                required: true,
                default: 0
        },
        goal: {
                type: Number,
                required: true
        }
},
category: {
        type: Array,
        default: []
},
location: {
        type: Array,
        default: []
},
comments: {
        type: Array,
        required: false,
        default: []
},
date: {
        dateObj: {
                type: Date,
                default: Date.now
        },
        parsedDate: {
                type: String
        }
},
rating: {
        likes: {
                type: Number,
                default: 0
        },
        dislikes: {
                type: Number,
                default: 0
```

```
        }
    },
    author: {
        id: String,
        name: String,
        email: String
    }
```

User Schema

```
    name: {
        type: String,
        required: true
    },
    login: {
        email: {
            type: String,
            required: true,
            index: {
                unique: true
            }
        },
        password: {
            type: String,
            required: true
        }
    },
    date: {
        dateObj: {
            type: Date,
            default: Date.now
        },
        parsedDate: {
            type: String
        }
    },
    comments: {
        type: Array,
        required: false,
        default: []
    },
    preferences: {
        interests: {
            type: Array,
            default: []
        },
        location: {
            type: Array,
            default: []
        }
    },
    projects: {
        funding: {
            type: Array,
```

```
                default: []
        },
        initated: {
                type: Array,
                default: []
        }
},
rated: {
        likes: [{
                type: String,
                index: {
                        unique: true
                },
                default: []
        }],
        dislikes: [{
                type: String,
                index: {
                        unique: true
                },
                default: []
        }]
},
rating: {
        likes: {
                type: Number,
                default: 0
        },
        dislikes: {
                type: Number,
                default: 0
        }
}
```

Examples of schemas

User Schema


```
[
  {
    "_id": "5521e70dd6fbf5fc1d8c1878",
    "name": "b",
    "rating": {
      "dislikes": 0,
      "likes": 0
    },
    "rated": {
      "dislikes": [],
      "likes": []
    },
    "projects": {
      "initated": [],
      "funding": []
    },
    "preferences": {
      "location": [
        "Vancouver"
      ],
      "interests": [
        "Technology"
      ]
    },
    "comments": [
      [
        "user a",
        "fuck off",
        "a@a.ca",
        "5521e6ffd6fbf5fc1d8c1877"
      ]
    ],
    "date": {
      "parsedDate": "Apr 5, 2015",
      "dateObj": "2015-04-06T01:53:17.740Z"
    },
    "login": {
      "email": "b@b.ca",
      "password": "$2a$10$i7CFhhT.mjTSmA5QKUO.I.BHI2RaLeGVBwOADrYgoHFFnediOJjwm"
    }
  },
  {
    "_id": "5521e6ffd6fbf5fc1d8c1877",
    "name": "user a",
    "rating": {
      "dislikes": 0,
      "likes": 0
    },
    "rated": {
      "dislikes": [],
      "likes": []
```

```
  },
  "projects": {
    "initated": [],
    "funding": []
  },
  "preferences": {
    "location": [
      "Palo Alto"
    ],
    "interests": [
      "Art"
    ]
  },
  "comments": [],
  "date": {
    "parsedDate": "Apr 5, 2015",
    "dateObj": "2015-04-06T01:53:03.738Z"
  },
  "login": {
    "email": "a@a.ca",
    "password": "$2a$10$KRK2vkn4IHuqri.Z4ozPSO5bWzxAotopjIxihvSvfefgz3pKwIuHK"
  }
},
{
  "_id": "5521ec77d0d4d4081feed097",
  "name": "Shihan Ai",
  "rating": {
    "dislikes": 0,
    "likes": 0
  },
  "rated": {
    "dislikes": [],
    "likes": []
  },
  "projects": {
    "initated": [],
    "funding": []
  },
  "preferences": {
    "location": [
      "San Jose"
    ],
    "interests": [
      "Fashion"
    ]
  },
  "comments": [],
  "date": {
    "parsedDate": "Apr 5, 2015",
    "dateObj": "2015-04-06T02:16:23.121Z"
  },
  "login": {
    "email": "shihan.ai@mail.utoronto.ca",
```

```
      "password": "$2a$10$VXkbWuQAvMWtziM4kcL45uPc4FqcYQQ.PCL7ZEz8JSA5MZvzrTiLu"
    }
  }
]
```


Project Schema


```
[
  {
    "_id": "5521e743d6fbf5fc1d8c1879",
    "title": "asdf",
    "description": "asdfg",
    "numFunders": 0,
    "author": {
      "id": "5521e6ffd6fbf5fc1d8c1877",
      "name": "user a",
      "email": "a@a.ca"
    },
    "rating": {
      "dislikes": 0,
      "likes": 0
    },
    "date": {
      "parsedDate": "Apr 5, 2015",
      "dateObj": "2015-04-06T01:54:11.864Z"
    },
    "comments": [],
    "location": [
      "San Jose"
    ],
    "category": [
      "Food"
    ],
    "funds": {
      "goal": 1212,
      "raised": 0
    }
  },
  {
    "_id": "5521e751d6fbf5fc1d8c187a",
    "title": "ggg",
    "description": "gggg",
    "numFunders": 0,
    "author": {
      "id": "5521e70dd6fbf5fc1d8c1878",
      "name": "b",
      "email": "b@b.ca"
    },
    "rating": {
      "dislikes": 0,
      "likes": 0
    },
    "date": {
      "parsedDate": "Apr 5, 2015",
```

```
      "dateObj": "2015-04-06T01:54:25.585Z"
    },
    "comments": [
      [
        "user a",
        "lolol",
        "a@a.ca",
        "5521e6ffd6fbf5fc1d8c1877"
      ],
      [
        "user a",
        "did this break too",
        "a@a.ca",
        "5521e6ffd6fbf5fc1d8c1877"
      ],
      [
        "user a",
        "asdf",
        "a@a.ca",
        "5521e6ffd6fbf5fc1d8c1877"
      ],
      [
        "5521e6ffd6fbf5fc1d8c1877",
        "user a",
        "oo",
        "a@a.ca"
      ],
      [
        "user a",
        "rr",
        "a@a.ca",
        "5521e6ffd6fbf5fc1d8c1877"
      ],
      [
        "user a",
        "gg",
        "a@a.ca",
        "5521e6ffd6fbf5fc1d8c1877"
      ]
    ],
    "location": [
      "Palo Alto"
    ],
    "category": [
      "Art"
    ],
    "funds": {
      "goal": 322,
      "raised": 321
    }
  },
  {
    "_id": "5521ed5399efb4601587f080",
```

```
    "title": "asdfsdfa",
    "description": "323",
    "numFunders": 0,
    "author": {
      "id": "5521ec77d0d4d4081feed097",
      "name": "Shihan Ai",
      "email": "shihan.ai@mail.utoronto.ca"
    },
    "rating": {
      "dislikes": 0,
      "likes": 0
    },
    "date": {
      "parsedDate": "Apr 5, 2015",
      "dateObj": "2015-04-06T02:20:03.605Z"
    },
    "comments": [],
    "location": [
      "Palo Alto"
    ],
    "category": [
      "Art"
    ],
    "funds": {
      "goal": 2342,
      "raised": 0
    }
  }
]
```

Testing

Error checking was done on all forms to ensure smooth user experience
Functionality testing was done with numerous sets of data and user profiles

Testing that routes were working was the first priority because without correct routes, the rest of the application will crumble

Testing that appropriate responses were in the proper JSON format was done to ensure smooth passage of information throughout the application

Testing that data was properly queried from the MongoDB database was another top priority

Testing the proper usage of AngularJS to ensure only the intended material to be displayed to the user will be displayed (e.g Only admin can see Admin button to display aggregate information)

Extends what was tested in Phase III