A PLATFORM FOR YOUNG KIDS TO SHARE PROGRAMMING PROJECTS

A Project

Presented to the

Faculty of

California State Polytechnic University, Pomona

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

In

Computer Science

By

Pavan Soundara

2018

SIGNATURE PAGE

PROJECT: A PLATFORM FOR YOUNG KIDS TO SHARE PROGRAMMING PROJECTS

AUTHOR: Pavan Soundara

DATE SUBMITTED: Spring 2018

Computer Science Department

Dr. Yu Sun _____
Project Committee Chair
Assistant Professor of Computer Science

Dr. Sampath Jayarathna _____
Assistant Professor of Computer Science

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to Dr. Yu Sun, my project advisor, for his patient guidance, enthusiastic encouragement, and useful critiques of this project work. I would also like to thank Dr. Sampath Jayarathna, for his advice and assistance as project committee member. My grateful thanks are also extended to Dr. Gilbert Young, for his help throughout my master's journey. I am grateful to all of those with whom I have had the pleasure to work during this and other related projects. Finally, I want to thank my parents and family for their support and encouragement throughout my master's journey. Without their constant encouragement and support this would have never been possible.

ABSTRACT

Everybody loves to be recognized and appreciated for the work and effort they put in a project. It could be a painting or a music video or a paper art or anything. Kids, of all the people, are to be recognized and appreciated more for the effort they put in. This would not only encourage them to work on more projects in future but also will spur and motivate them to take certain paths in future that could help them shape their future better. There are several platforms available to share various projects online. But, one particular platform that isn't readily available or accessible for kids is a platform for kids to share their programming projects. There are a lot of programs that introduce school kids to programming and provide them hands-on experience with building projects. Students often work on their own personal projects with the knowledge acquired from the programming classes. But, they don't have any platform where they can post the projects they completed and make a portfolio of all the projects that they worked on. ShareMyWorks tries to solve this problem by providing a platform to young kids to proudly display their programming projects and share it with their parents as well as the fellow students. Students can also attach files related to projects so that interested individuals can download them and contribute by further enhancing the project. This would not only help as a platform to display projects, but also a platform to promote and recognize creative talent at a young age.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1 - INTRODUCTION

In this era, most kids are accustomed to using electronic devices such as mobile phones, laptop, desktop computers, gaming consoles and smart televisions. Given the devices' familiarity among the kids, various programs and organizations are trying to further bring them closer to these by spurring their interests in the lower level working and functionalities of the devices. Various organizations already have programs that teach school kids programming by employing techniques that make it easy for them to understand the basics and then use the same knowledge in developing various creative applications mostly games [2]. In kids, programming has proven to teach them to experiment, strengthen their thinking, improve their creativity, problem solving skills, and give them confidence [1]. Often kids make projects with knowledge acquired from the teaching programs. But, they don't have a centralized platform that can be used to post the projects that they created. We built ShareMyWorks to solve this problem by providing a platform for sharing the projects that kids make. ShareMyWorks is divided into two main parts. First part lets students to register onto the platform, join courses and post projects related to that courses or post projects that are not related to any of the courses available. Students can also upload their screenshots and zip files of the project. Students can provide the mobile number and email address of their parents and an email notification is sent to them with a link to the project every time the student posts a project. Second part of the application lets the organizations or course instructors to add or remove students from a course and manage the overall content being posted in relation to the course. We also have an administrator account option that manages the instructors as well as the students.

Chapter 2 - REQUIREMENTS

In this section, the types of users and their privileges and responsibilities has been defined.

2.1 Users

Table 1 Type of Users

Type of user	Description
Student	Registers as a normal user with information such as name, email address, parents name, mobile number, and email address.
Instructor	Registers as a normal user and then gets added as an instructor. Can view information about students as well as courses. Can create users account, courses, projects, and delete the same.
Admin	Single admin is created at the time of making of application and has all the privileges available such as adding or removing students, instructors, courses, projects, and admins.

2.2 User Requirements

2.2.1 Functional Requirements

a. Register, login, logout, change password and accept invite:

- User creates account by registering with their information.
- Inputs the username and password access the user specific homepage.
- Error is displayed incase username or password is wrong.

- User can request for new password incase he forgets his old password.
- User can accept invite by going to his email and clicking on the link provided.

b. Student:

- Has the privileges to register, add projects, and remove own projects.
- Can view own as well other users' projects, edit own projects, and accept invitation to a course.
- Can update or edit their own profile.

c. Instructor:

- Has more privileges than students and can see all the information related to students and courses.
- Can add and remove courses.
- Can add or remove students from a course.
- Can remove projects from a course.
- Can send or withdraw course invitation for a student.

d. Admin

- Has all the ultimate privileges available on the application.
- Can add, remove, and edit a student, instructor, and course.
- Can add or remove admins.
- Can add or remove projects.

2.2.2 Non-functional Requirements

- The user interface is intuitive and easy to use.
- All the options are available easily.
- All the information provided to the students is consistent and not confusing.
- Allowing users to upload files and check those files for any malicious content.

- Actively check for mischievous content being posted as projects.
- Allowing users based on their privileges to add or edit data.

Chapter 3 - SYSTEM ARCHITECTURE

ShareMyWorks as an application has two parts that power the web application. The front end and the back end. The front end of the application is powered by ReactJs, Redux, Bootstrap, HTML and CSS. The REST based backend of the application has been developed with Loopback framework which is connected to a MongoDB database server.

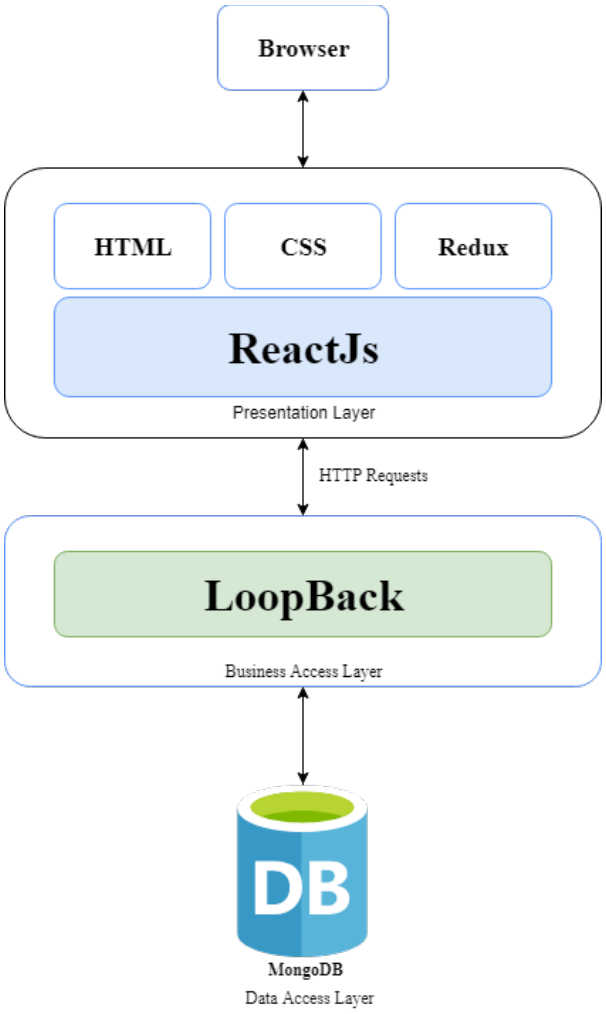

Figure 1 System Architecture

We employ a Model-View-Controller approach in our architecture. After initially loading the application in the web browser, only data that is exchanged between the front end and the back end is in JSON format and it is requested through HTTP requests [3].

3.1 Front-end

The front-end of the application primarily comprises of two parts. The view layer and the control layer. The control layer manages the presentation of the view layers by actively manipulating the document object model (DOM) based on the user activity. We use ReactJs along with Redux [5] for controlling the front-end application by taking the user input and communicating with the backend network. React is one of the most popular front-end frameworks available today. Apart from React, we have frameworks such as Vue.js and Angular available for developing the front-end of an application. I chose React for the ease of implementing various features in the application as well as the control that react provides in terms of managing the application. We also use redux which helps in maintain a central state of user. This helps in managing the user state regardless of which page the user is on. Redux also helps in managing a role-based view which is one of the key features in our application. The view layer is then rendered with the option provided by react in HTML and CSS. Bootstrap framework helped in maintaining a responsive and consistent front-end across various browsers, platforms, and devices [8].

3.2 Back-end

Back-end of the application is built using Loopback [3] which is Node.js [4] based backend framework that helps in rapidly developing backend by building APIs on the fly. Loopback is a pure API framework. With Loopback setting up models is possible in minutes and Loopback automatically generates API endpoints for the models. Loopback provides a user-friendly API explorer which lists all the available endpoints, their sample response and option to make a request [9]. Loopback seamlessly connects to any type of database server [7].

We use a NoSQL MongoDB database server which is hosted at mlabs [6]. Loopback also has built-in role-based controls which was used in our application to setup initial user model. Loopback provides oAuth user and registration model without requiring any additional modules and also has custom policies for users based on their role. We also use amazon S3 bucket for storing the images and files that students upload as part of their project. Loopback provides a connector to upload files via the API and then store the link to those files in the database. Later, while retrieving the projects, the files and images are also retrieved by requesting appropriate links.

3.3 Security

Security is top most priority when building a web-based application. Since data is being exchanged between the front-end on the client side to the back-end on the server, it can be easily captured or modified by a third-party threat. To prevent this, we force HTTPS on all communications which is secured by TLS that provides an end-to-end encryption so that even if some third-party threat gets hold of the information, it will be completely encrypted. To prevent unauthorized access to information, we require registration and logging in to the web app before the user can post or edit anything. Apart from the registration and login requirement, only instructors and admins have access to information regarding all the registered students. Only users with admin credentials are allowed to add more instructors or removing existing instructors. This sanctions in place help in maintaining the security of the application [9].

3.4 Data Connection

As part of the Loopback framework, connecting with datastores such as SQL and MongoDB are made easy by the inbuilt connector that connects MongoDB to Loopback [9]. We use a remote MongoDB service mLab for ShareMyWorks. If there are existing models in the database, loopback automatically discovers all the models and makes API endpoints.

A snippet of the code used for connecting Loopback with MongoB is as follows.

```
"shareworks-ds": {
   "host": "*****.mlab.com",
   "port": 1234,
   "url": "mongodb://******:******@*****.mlab.com:1234/
   "database": "dbname",
   "password": "******",
   "name": "db",
   "user": "shareworks ",
   "connector": "mongodb"
},
```

3.5 API

API endpoints play a key role in the application architecture and it is further discussed in detailed in the following sections [10]. In our application, API endpoints are divided into 4 major categories. Account, activity, course, and organization are the categories.

3.5.1 Account

An account manages the registration, login, and role of the users. Users are assigned a default role on registration which is student. Based on the user role, users will be directed to different pages upon logging in. Account is directly related to Activities to identify the projects posted by the users.

Registration

Sending post request to the below API from the front-end persists the provided data onto the database and then create an access token for the current session and stores it as a cookie. User is automatically logged in and redirected to the homepage [10].

Registration API Endpoint:

Request Type: Post URL: https://dev-sharemyworks-backend.herokuapp.com/api/Account

Request from Front-end:

```
axios.post(SIGNIN_API, user).then(response => {
      if (response.status === 200) {
        window.sessionStorage.setItem('accessToken', response.data.id);
        window.sessionStorage.setItem('userId', response.data.userId);
        if (response.data.role === 'admin') {
          this.setState({ isAdmin: true });
        }
        this.setState({
          redirect: true,
          success: true,
          accessToken: response.data.id,
          userId: response.data.userId,
          loading: false
        });
      }
    })
```

Login

Takes username and password as input, validates the input and on success returns an access token that is stored as cookie in the browser session. User is then redirected to different homepages depending on their role.

Login API Endpoint:

```
Request Type: Post
URL: https://dev-sharemyworks-backend.herokuapp.com/api/Account/login
```

Request from Front-end:

```
axios.post(SIGNIN_API, user).then(response => {
      if (response.status === 200) {
        window.sessionStorage.setItem('accessToken', response.data.id);
        window.sessionStorage.setItem('userId', response.data.userId);
        if (response.data.role === 'admin') {
          this.setState({ isAdmin: true });
        }
        this.setState({
          redirect: true,
          success: true,
          accessToken: response.data.id,
          userId: response.data.userId,
          loading: false
        });
      }
    })
```

Logout

Invalidates user session by removing the cookie from the users' browser session and by invalidating the access token created and stored in the backend.

Logout Endpoint:

Request Type: Post
URL: https://dev-sharemyworks-backend.herokuapp.com/api/Account/logout

Request from Front-end:

```
axios.post(postUrl).then(response => {
        if (response.status === 204) {
            window.sessionStorage.setItem('accessToken', null);
            window.sessionStorage.setItem('userId', null);
            this.setState({ logout: true });
        }
    }).catch(error => {
        console.log(error);
    });
```

3.5.2 Activity

Activity model manages all the user projects as well as the files. It uploads the files on the S3 bucket and then retrieves them when requested. It also persists the links of images and files uploaded as a property of activity.

Add Project

Allows user to add new project. It has various inputs such as title, description, link, images, and files.

Add Project Endpoint:

Request Type: Post
URL: https://dev-sharemyworks-backend.herokuapp.com/api/Account/{id}/activities

Request from Front-end:

```
axios({
    method: 'post',
    url: postUrl,
    params: {
      access_token: this.state.accessToken
    },
    data: activity
  }).then(response => {
    if (response.status === 200) {
      const activityId = response.data.id;
      this.filterFiles(this.state.files, activityId, false).then(filteredFiles => {
        this.filterFiles(this.state.images, activityId, true).then(filteredImages => {
          this.setState({ redirect: true });
        });
      });
    }
  })
```

Add Files

Allows user to upload images and files related to a project. Gets uploaded to S3 bucket and then the link is stored in the database.

Add Files Endpoint:

Request Type: Post
URL: https://dev-sharemyworks-backend.herokuapp.com/api/Account/{id}/upload

Request from Front-end:

```
axios({
        method: 'post',
        url: ACTIVITY_API + '/' + activityId + '/upload',
        params: {
          access_token: this.state.accessToken,
          isImage: isImg
        },
        data: bodyFormData
      }).then(response => {
        resolve({
          name: response.data.result.files.file[0].originalFilename,
          link: response.data.result.files.file[0].providerResponse.location
        });
      });
```

Delete Project

Allows user to delete their post.

Delete Project Endpoint:

```
Request Type: Delete
URL: https://dev-sharemyworks-backend.herokuapp.com/api/Activity/{id}
```

Request from Front-end:

```
axios({
    method: 'delete',
    url: deleteUrl,
    params: {
      access_token: this.props.accessToken
    }
  })
```

Delete Files

This gets requested automatically when a user request for their project to be deleted. It deletes all the images and files related to the project from the S3 bucket.

Delete Files Endpoint:

```
Request Type: Delete
URL: https://dev-sharemyworks-backend.herokuapp.com/api/Activity/{id}/removeAllFiles
```

Request from Front-end:

```
axios({
    method: 'delete',
    url: deleteFiles,
    params: {
      access_token: this.props.accessToken
    }
  })
```

Get Projects

This retrieves all the projects posted so far by the user.

Delete Files Endpoint:

```
Request Type: Get
URL: https://dev-sharemyworks-backend.herokuapp.com/api/Account/{id}/Activity
```

Request from Front-end:

```
axios.get(getLink, {
      params: {
        access_token: this.state.accessToken
      }
    }).then(response => {
      this.setState({
        activities: response.data,
        activitiesCount: response.data.length,
        projectsLoading: false
      });
    });
```

3.5.3 Course

Course API is the base for adding new courses, adding students to course, sending invites to the students for enrollment into the course, remove course and remove students from course.

Add Course

Allows instructor or admin to add new courses.

```
Request Type: Post
URL: https://dev-sharemyworks-backend.herokuapp.com/api/Course
```

Request from Front-end:

```
axios({
        method: 'post',
        url: COURSE_API,
        params: {
          access_token: this.state.accessToken,
          data: courseInfo
        },
        data: bodyFormData
      })
```

Get Courses

Fetches all the available courses.

Delete Files Endpoint:

```
Request Type: Get
URL: https://dev-sharemyworks-backend.herokuapp.com/api/Course
```

Request from Front-end:

```
axios.get(courseLink, {
      params: {
        access_token: this.state.accessToken
      }
    }).then(response => {
      this.setState({
        courses: response.data,
      });
    });
```

Add Student to Course

Adds student with a given id to a course of given id. This API is only accessible to instructors or admin.

Student-course Endpoint:

Request Type: Put URL: https://dev-sharemyworks-backend.herokuapp.com/api/Account/{id}/courses/rel/{fk}

Invite Student to Course

Invite a student with the given id to the course. Student gets an email with the invite link and upon opening the link, user gets enrolled into the course.

Invite Endpoint:

Request Type: Post URL: https://dev-sharemyworks-backend.herokuapp.com/api/Course/{id}/invite

Remove Course

Allows instructor or admin to remove new courses.

Request Type: Delete URL: https://dev-sharemyworks-backend.herokuapp.com/api/Course/{id}

Remove Student from Course

Removes student with a given id from a course of given id. This API is only accessible to instructors or admin.

Student-course Endpoint:

Request Type: Delete URL: https://dev-sharemyworks-backend.herokuapp.com/api/Account/{id}/courses/rel/{fk}

Withdraw Invite of Course

Withdraws the invite that was issued for a student with given id for a course.

Invite Endpoint:

Request Type: Delete URL: https://dev-sharemyworks-backend.herokuapp.com/api/Course/{id}/remInvite

Get Course Instructor

Fetches the instructor for the course with given id.

Course Endpoint:

Request Type: Get URL: https://dev-sharemyworks-backend.herokuapp.com/api/Course/{id}/instructor

3.5.4 Organization

Organization acts as base for managing instructors as well as courses. Organization can be created or removed only by the admins.

Add Organization

Organization Endpoint:

Request Type: Post URL: https://dev-sharemyworks-backend.herokuapp.com/api/Organization

Add Courses in Organization

Creates new courses within the organization.

Organization Endpoint:

Request Type: Post URL: https://dev-sharemyworks-backend.herokuapp.com/api/Organization/{id}/Courses

Remove Organization

Organization Endpoint:

Request Type: Delete URL: https://dev-sharemyworks-backend.herokuapp.com/api/Organization/{id}

Remove Courses from Organization

Organization Endpoint:

Request Type: Delete URL: https://dev-sharemyworks-backend.herokuapp.com/api/Organization/{id}/Courses

Get Organizations

Organization Endpoint:

Request Type: Get URL: https://dev-sharemyworks-backend.herokuapp.com/api/Organization

Get Courses in Organization

Fetches courses within the organization.

Organization Endpoint:

Request Type: Get

URL: https://dev-sharemyworks-backend.herokuapp.com/api/Organization/{id}/Courses

Chapter 4 - USER INTERFACES

4.1 Signup Page

Full page view of the signup page. Allows new users to register on the platform.

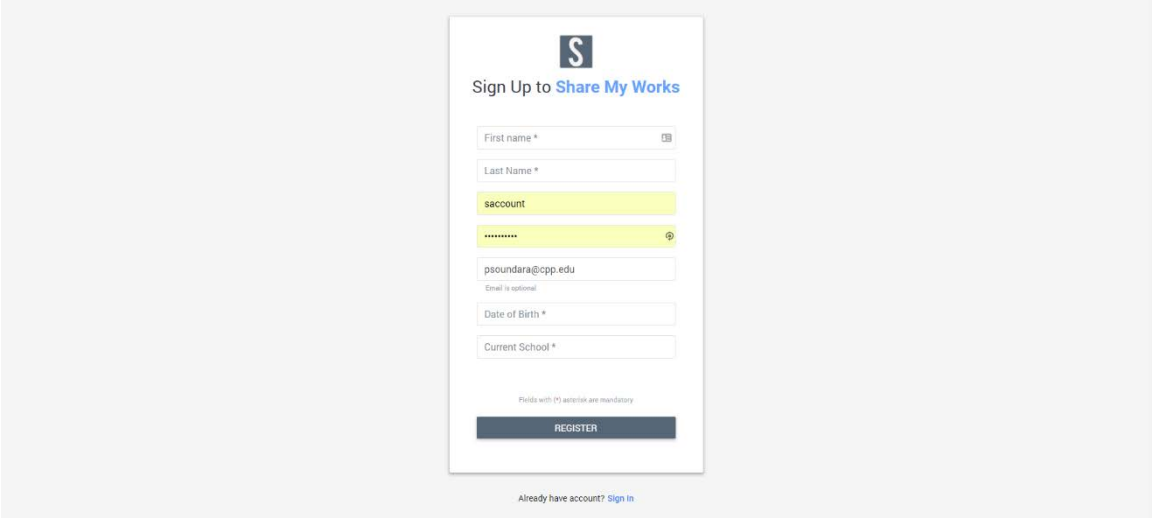

Figure 2 Signup Page

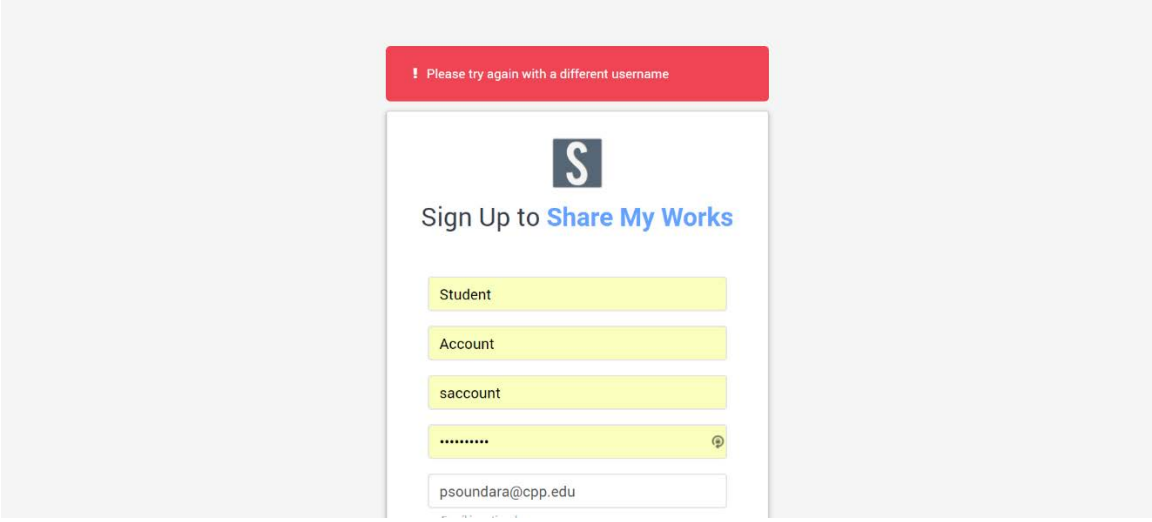

Figure 3 Signup Page Error Feedback

4.2 Sign In View

Sign in view which is the default view when the application is open in a web browser.

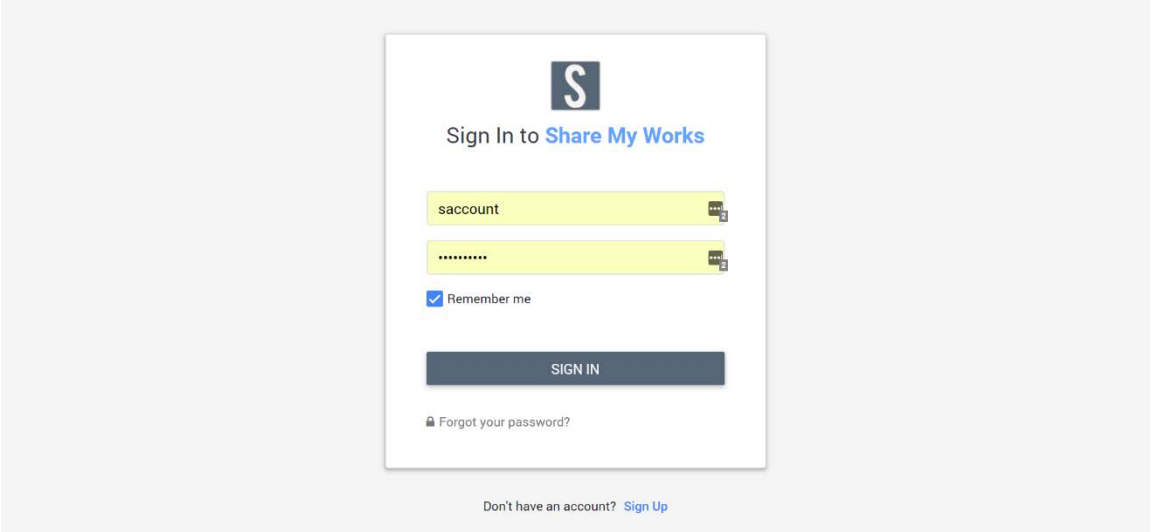

Figure 4 Sign In View

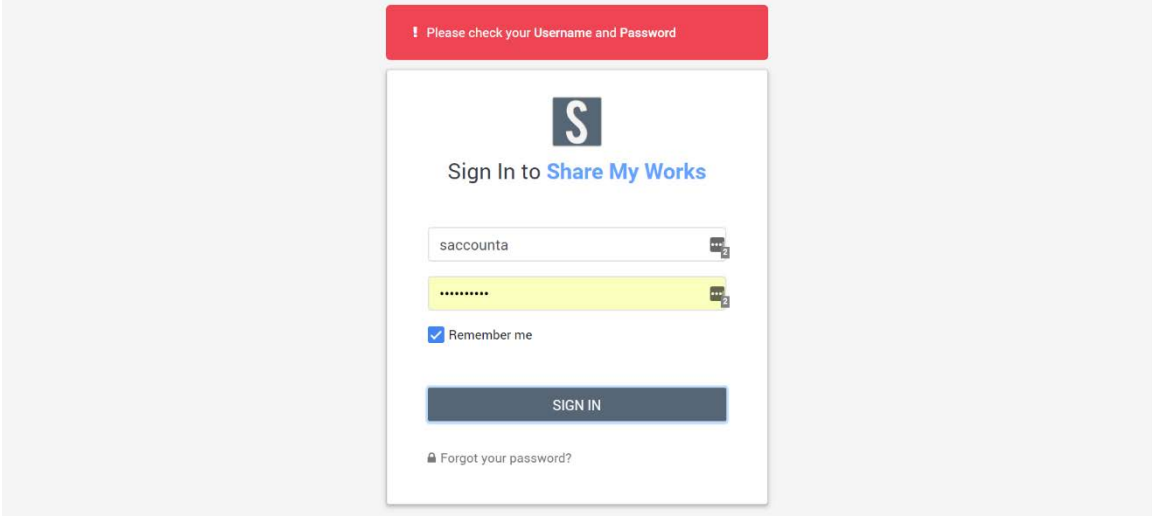

Figure 5 Sign In Error View

4.3 Profile View

Displays all the information about the user including the projects posted by the user. User can navigate using the navbar to different pages.

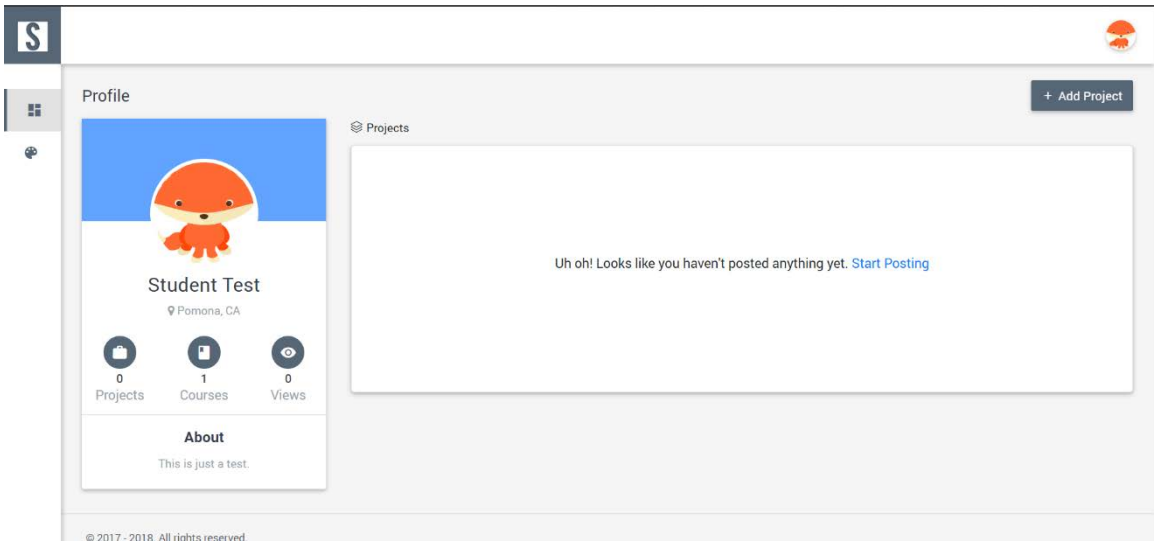

Figure 6 New User Profile View

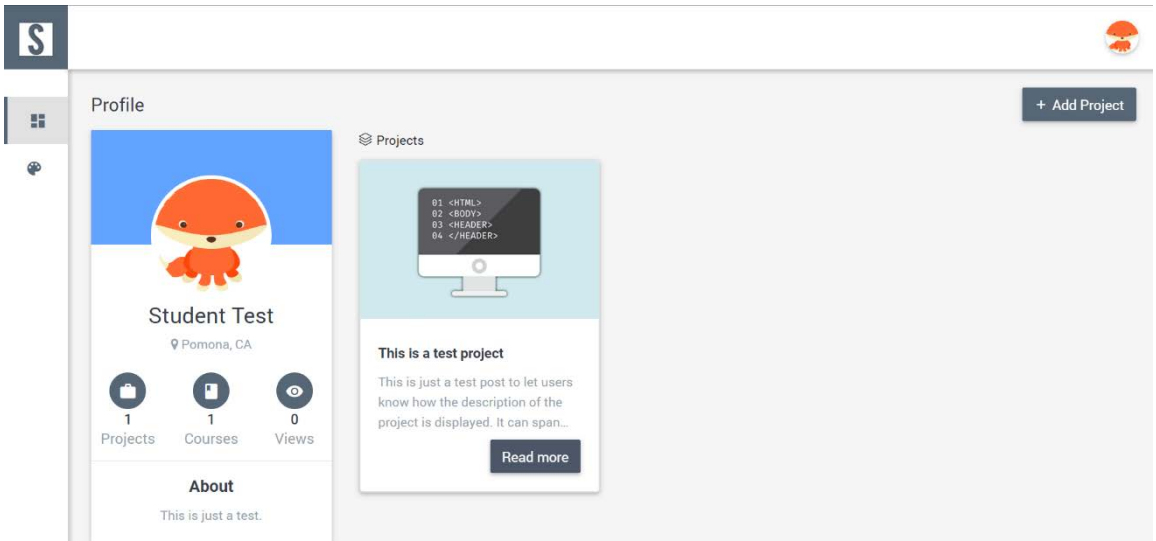

Figure 7 User Profile View with a Project

4.4 Add Project Page

Allows user to add projects to his profile.

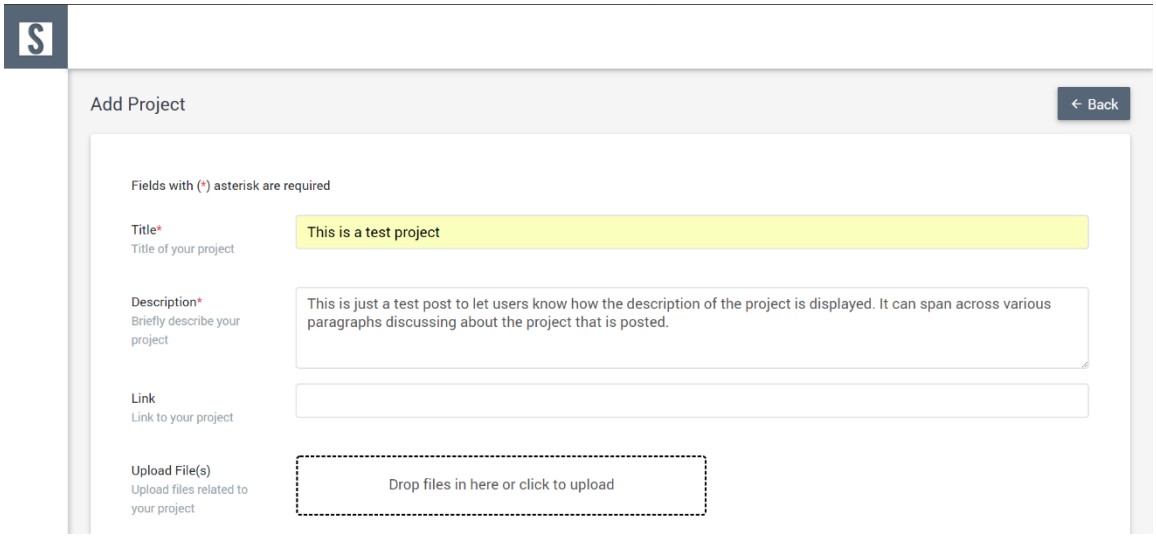

Figure 8 Add Project Page Part 1

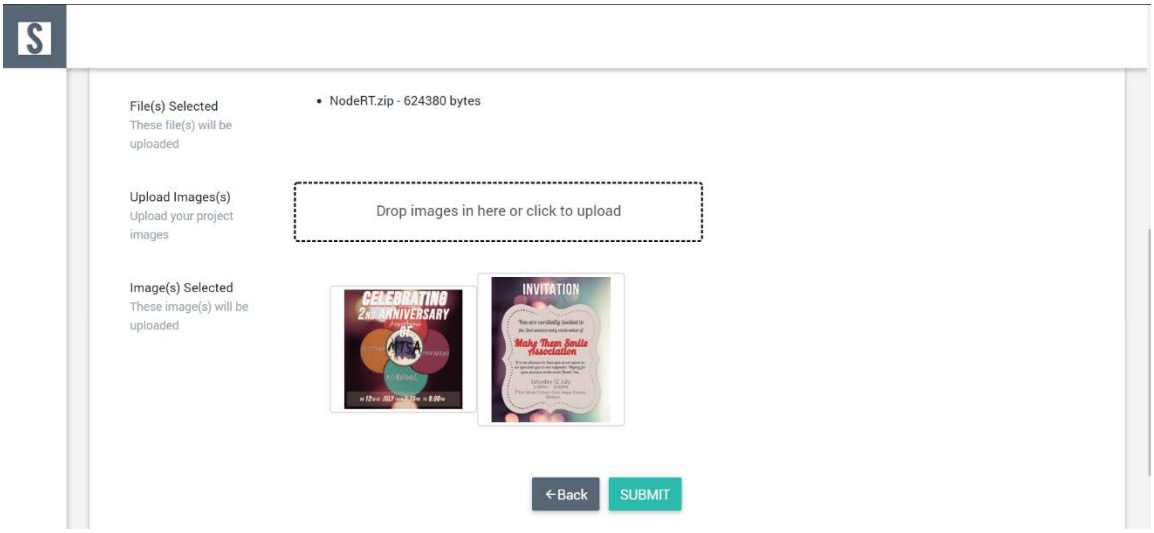

Figure 9 Add Project Page Part 2

4.5 Profile Settings Page

Allows user to edit their personal information.

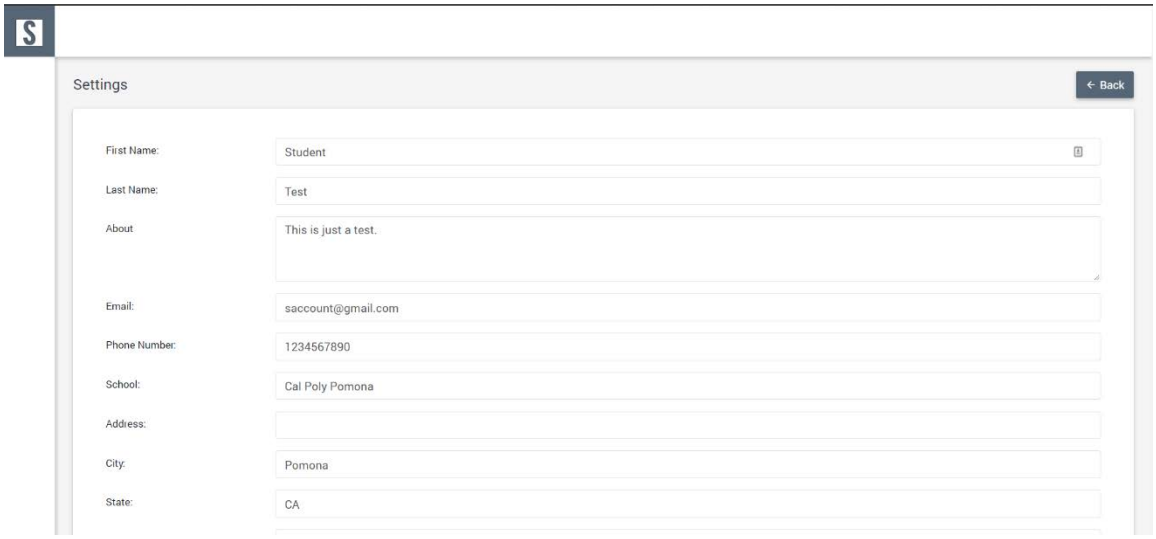

Figure 10 Profile Settings Page

4.6 View Project Page

Gives user a full-page view of the project, its description, screenshots or images and files attached.

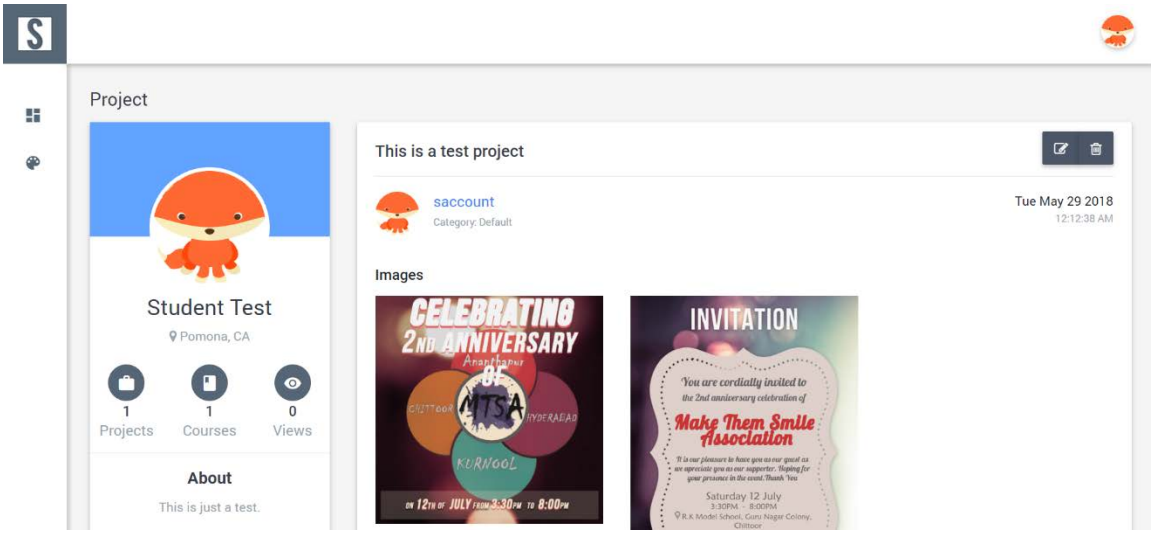

Figure 11 View Project Page Part 1

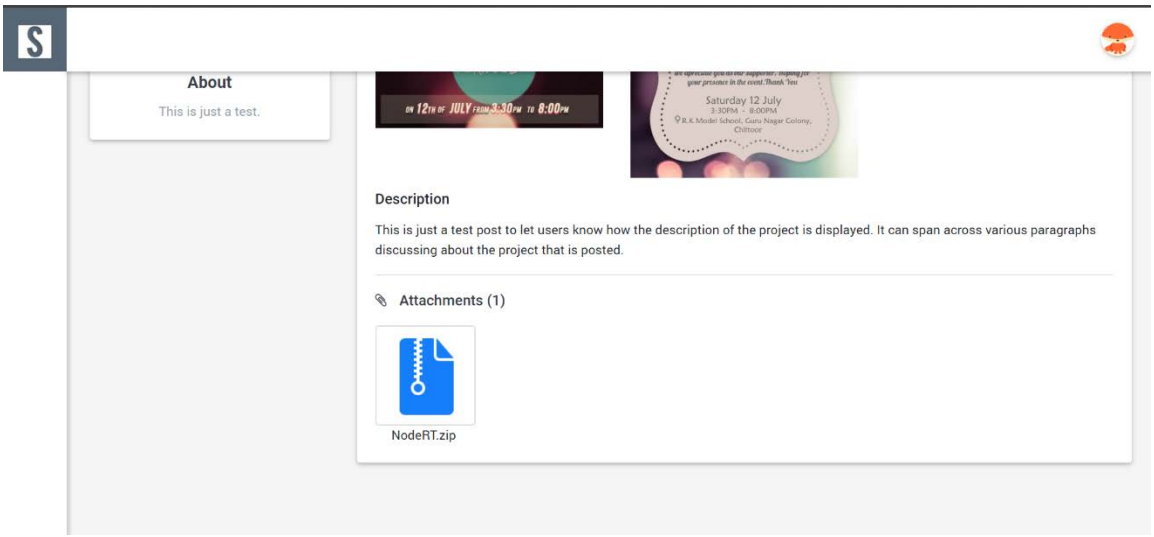

Figure 12 View Project Page Part 2

4.7 Admin Page

Admin page shows all the students enrolled in a certain course. It allows admin to add student to courses as well as remove students from courses.

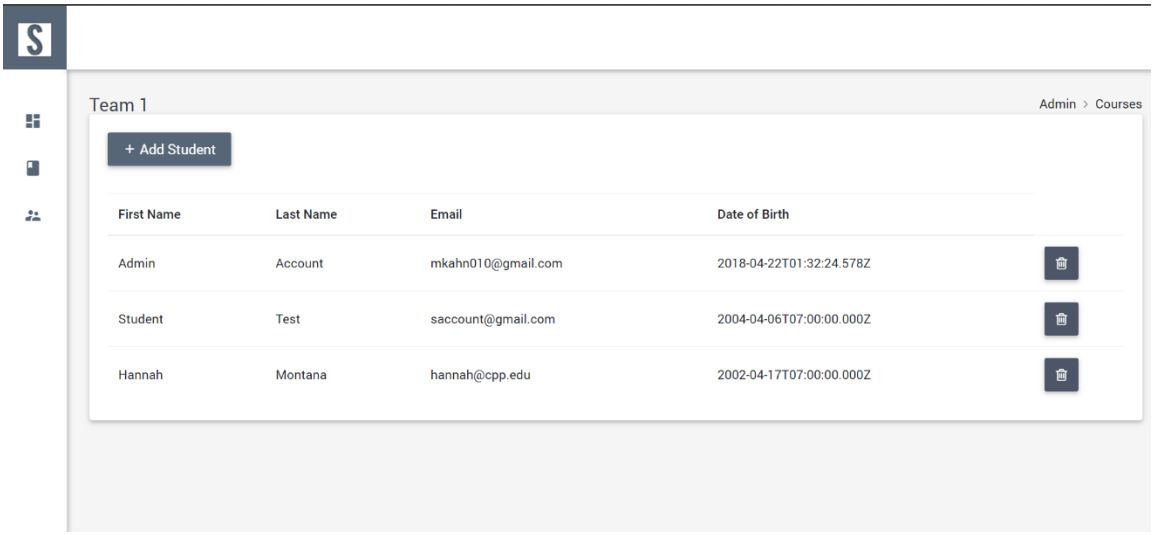

Figure 13 Admin Page

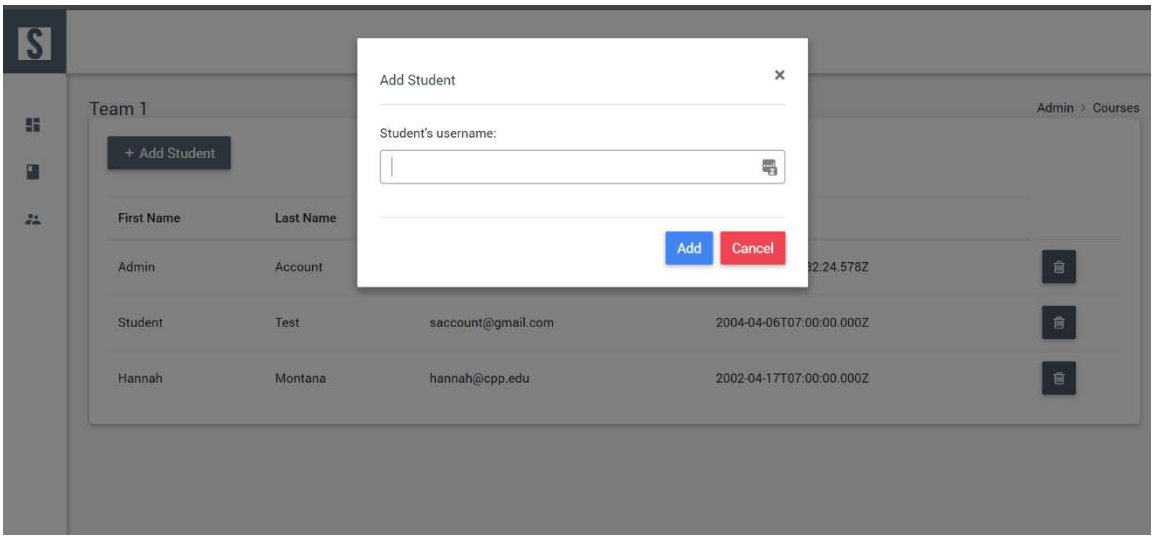

Figure 14 Add Student to Course Page

Chapter 5 - CONCLUSION AND FUTURE WORK

Unavailability of platform for young kids to post and share their programming projects has inspired me in building this application. This application has the potential to help thousands of kids proudly display their programming projects and also be recognized and appreciated for the same. This, as discussed earlier will play a key part in deciding which path to pursue and shape their future. Parents of the kids can also recognize and encourage them at a very young age. By this project, we provide an easy to use platform that can be used both by students as well as by instructors of various teaching programs. We developed a project that is stable, consistent, and intuitive. Various organizations can also take advantage of this product to manage various courses taught by their instructors. This project provides all the basic necessary features needed for an online portfolio and class management system.

Despite having all the basic features, we believe there is still a huge room for improvement for the application in terms of performance, features, metrics, and usability. We believe that implementing features like point system, user views, sharing on social media options, live code editing capabilities, free instant submission to Google Play store or Apple Store, email updates, user engagement metrics, desktop and mobile versions of the application could make this one of the top competitors in the segment. We plan to integrate all these features and constantly add more new features in the future.

REFERENCES

[1] “How Learning To Code Develops Kids’ Creativity | Tynker Blog,” How Learning To Code Develops Kids’ Creativity. [Online]. Available: https://www.tynker.com/blog/articles/ideas-and-tips/learning-to-code-develops-creativity-in-kids/. [Accessed: 29-Mar-2018].

[2] L. B, “Coding for Kids: Free Websites That Teach Kids Programming,” Mommy Poppins - Things To Do in New York City with Kids, 01-Jan-2018. [Online]. Available: https://mommypoppins.com/coding-kids-free-websites-teach-learn-programming. [Accessed: 29-Mar-2018].

[3] N. Duffy, “Getting Started with Node.js and LoopBack,” Semaphore - Continuous Integration, Deployment, TDD, DevOps tutorials. [Online]. Available: https://semaphoreci.com/community/tutorials/getting-started-with-node-js-and-loopback. [Accessed: 29-Mar-2018].

[4] K. Sandoval, “13 Node.js Frameworks to Build Web APIs | Nordic APIs |,” Nordic APIs, 27-Oct-2017. [Online]. Available: https://nordicapis.com/13-node-js-frameworks-to-build-web-apis/. [Accessed: 29-Mar-2018].

[5] A. Fedosejev and A. Bush, React.js essentials: a fast-paced guide to designing and building scalable and maintainable web apps with React.js. Birmingham: Packt Publishing, 2015.

[6] K. Chodorow, MongoDB: the definitive guide. Beijing: OReilly, 2014.

[7] M. Satheesh, Web development with MongoDB and NodeJS. Packt Publishing, 2015.

[8] S. Lauesen, User interface design: a software engineering perspective. Harlow: Pearson Addison-Wesley, 2007.

[9] “Getting started with LoopBack,” LoopBack 2.x | LoopBack Documentation. [Online]. Available: http://loopback.io/doc/en/lb3/Getting-started-with-LoopBack.html. [Accessed: 29-Mar-2018].

[10] D. Prado, “Loopback and create-react-app Developer Tutorial - Gorilla ...,” Loopback and create-react-app Developer Tutorial, 01-Feb-2018. [Online]. Available: https://gorillalogic.com/blog/loopback-and-create-react-app-developer-tutorial/. [Accessed: 29-Mar-2018].