

12-2018

SOCIAL NETWORK FOR SOFTWARE DEVELOPERS

Sanket Prabhakar Jadhav
California State University - San Bernardino

Follow this and additional works at: <https://scholarworks.lib.csusb.edu/etd>



Part of the [Other Computer Engineering Commons](#)

Recommended Citation

Jadhav, Sanket Prabhakar, "SOCIAL NETWORK FOR SOFTWARE DEVELOPERS" (2018). *Electronic Theses, Projects, and Dissertations*. 782.

<https://scholarworks.lib.csusb.edu/etd/782>

This Project is brought to you for free and open access by the Office of Graduate Studies at CSUSB ScholarWorks. It has been accepted for inclusion in Electronic Theses, Projects, and Dissertations by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.

SOCIAL NETWORK FOR SOFTWARE DEVELOPERS

A Project
Presented to the
Faculty of
California State University,
San Bernardino

In Partial Fulfilment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Sanket Prabhakar Jadhav
December 2018

SOCIAL NETWORK FOR SOFTWARE DEVELOPERS

A Project
Presented to the
Faculty of
California State University,
San Bernardino

by
Sanket Prabhakar Jadhav

December 2018

Approved by:

Dr. Owen Murphy, Advisor, Computer Science and Engineering

Dr. Kerstin Voigt, Committee Member

Dr. Yunfei Hou, Committee Member

© 2018 Sanket Prabhakar Jadhav

ABSTRACT

This project is the design and implementation of a web-based message board for software developers. The purpose of “Social Network for Software Developers” is to connect inexperienced software developers with experienced software developers.

ACKNOWLEDGEMENTS

I would like to express my special thanks of gratitude to my advisor, mentor and supporter Dr. Owen Murphy for giving me guidance and knowledge throughout this project. I would also like to thank Dr. Kerstin Voigt and Dr. Yunfei Hou for being the committee members and for their valuable advice and support.

Lastly, this master's degree has been made possible not only through the financial support but most importantly the moral support of my parents --- thank you!

TABLE OF CONTENTS

ABSTRACT	iii
ACKNOWLEDGEMENTS.....	iv
LIST OF FIGURES	vii
CHAPTER ONE: INTRODUCTION	
Background.....	1
Existing System	2
CHAPTER TWO: SYSTEM ANALYSIS	
Proposed System	3
System Requirement Specification	4
Server Requirements	4
Software Requirements.....	4
Software Used	5
Reactjs.....	5
Nodejs Framework.....	5
Axios	5
Mongodb and Mlab	5
Authentication	7
CHAPTER THREE: SYSTEM DESIGN	
Uml Diagrams	8
Use Case Diagram.....	8
Data Flow Diagram	9
Sequence Diagram	11

CHAPTER FOUR: SYSTEM TESTING

Testing Documentation	12
Unit Testing	12
User Acceptance Testing	12
Test Result	13

CHAPTER FIVE: SYSTEM SCREENSHOTS

System Working	14
Landing Page	14
Developers Options	16
View Profile	17
Login	18
Sign-Up	19
Form Section	20
Delete Information	22
Post Section	23

CHAPTER SIX: FUTURE ENHANCEMENTS

Future Project Development	24
Better Password Management	24
Online Tutoring	24
Job Openings	24

APPENDIX: CODE FOR THE LOGIN SECTION	25
--	----

REFERENCES	44
------------------	----

LIST OF FIGURES

Figure 1. Mongodb connection to Mlab using Mongoose to build tables.	6
Figure 2. Mongodb connection to Mlab	6
Figure 3. Demonstration of Passport working mechanism	7
Figure 4. Use case diagram for new users	8
Figure 5. Data flow diagram for login and registration	9
Figure 6. Data flow for the post feed.....	10
Figure 7. Sequence diagram for update user profile.....	11
Figure 8. Landing page for the web application.....	14
Figure 9. Screenshot displayed when a user clicks on the developers option....	16
Figure 10. Display profile picture of the user	17
Figure 11. Display the login screen.	18
Figure 12. Sign up screen for new users	19
Figure 13. Add experience form	20
Figure 14. Add education form	21
Figure 15. Information about deleting account.....	22
Figure 16. Illustration of the post section	23

CHAPTER ONE

INTRODUCTION

Background

This project is the design and implementation of a web-based message board for software developers. The purpose is to connect inexperienced developers in computer science with experienced developers to learn coding in a short amount of time.

- In “Social Network for Software Developers”, beginners in Computer Science who are struggling with computer programming can get help from senior developers.
- Authenticated users can express their views on each message with the help of comments. Every post has like and dislike options. So, without reviewing all the comments, the user can reach a solution based on the number of “likes on comment”.
- This application is protected with functions like login facilities and authentication routes. (Authentication routes refer to the routes that only certain users can access based on their authentication status)
- The project will help users create a portfolio. With the help of this portfolio, users may obtain future internship and job opportunities.

- With the use of all the features, “Social Network for Software Developers” can help aspiring developers to save time in computer programming.

Existing System

Many new software developers who are struggling with problems and questions have few alternatives for help. Stackoverflow is a site to connect developers, but before posting questions, new users must first answer questions. Codementor and Chegg are alternatives, but they require a paid subscription.

CHAPTER TWO

SYSTEM ANALYSIS

Proposed System

The application begins at the home screen. If the user doesn't have an account on this website, the user can view the summary profile of the existing developers. From the home page, the user will get two options either sign-up or login.

- If any developer is signed up at the website, he can add a profile picture with his status, his company and his location.
- The user can also link all his social profiles. Furthermore, he can add his summary of a profile and he can add his programming skills sets of programming. Also, the user can add work experience.
- There is also a section for education. If the user is taking any classes from any school or any boot camp, he can add this to his profile.
- If a valid user is logged in to his profile, he can edit his profile and add or delete experience and education.
- In the post section all the developers can post their comments. Any developers who have any questions can add comments in this section and then other developers can help to solve their issue.

- In the post sections, developers can like or dislike the comments of other developers' posts. Developers can also delete their personal comments from the post section.

System Requirement Specification

“The System” needs to have the following hardware and software requirements to run the application.

Server Requirements

- Heroku – Website server.
- Mlab – Database server

Software Requirements

- Heroku account
- Reactjs
- Nodejs
- Axios
- MongoDB and Mlab

The user only needs a workstation with a browser to access the website.

Software Used

Reactjs

Reactjs is a Javascript library for building web interfaces. It is used to develop single page applications. When the user visits the website, it downloads Javascript code into the browser which then runs in the browser. The user doesn't need to go a server for every page.

Nodejs framework

Nodejs is an open source development platform for executing Javascript code server-side. Nodejs is useful for developing applications that require a persistent connection from the browser to the server and is often used for real-time applications such as chat, news feeds and web push notifications.

Axios

Axios is a promise-based HTTP client that works both in the browser and in a Nodejs environment. It basically provides a single API for dealing with XMLHttpRequests and a node's http interface.

Mongodb and Mlab.

Mongodb is the frontend database program used in this project. It connects to the database Mlab. The program mongoose is used to create a table schema at Mlab. Mongodb is a database system that allows you to store

documents with a dynamic structure. These documents are saved inside a collection.

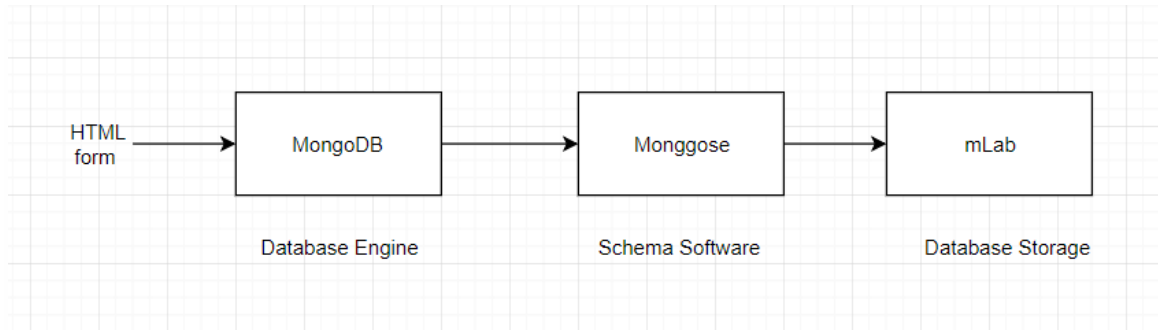


Figure 1. Mongodb connection to Mlab using Mongoose to build tables.

When the user fills an HTML form, Mongodb saves all the information with the help of Mongoose. This data is stored in JSON format (JSON stands for Javascript Object Notation and it's a data format. JSON is a way to hold bits of information, like a database.). Once the schema is created, Mongodb then connects directly to the Mlab storage engine for queries and updates.

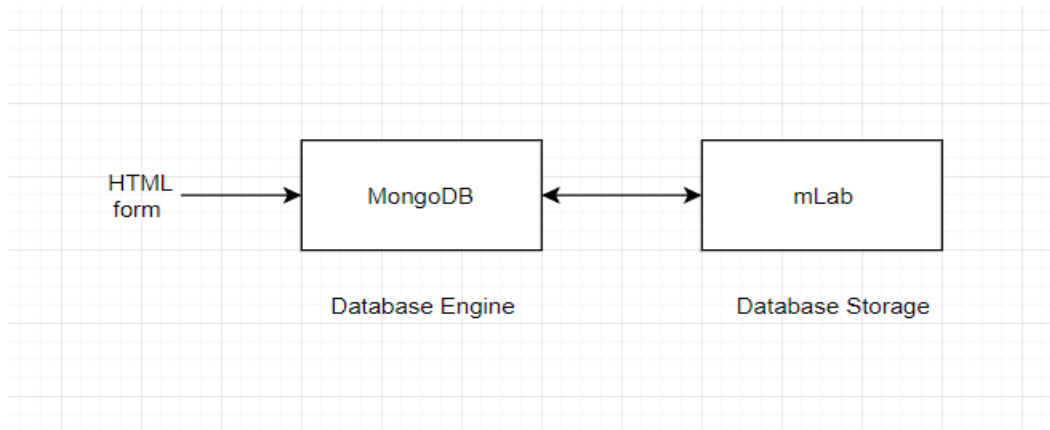


Figure 2. Mongodb connection to Mlab.

Authentication

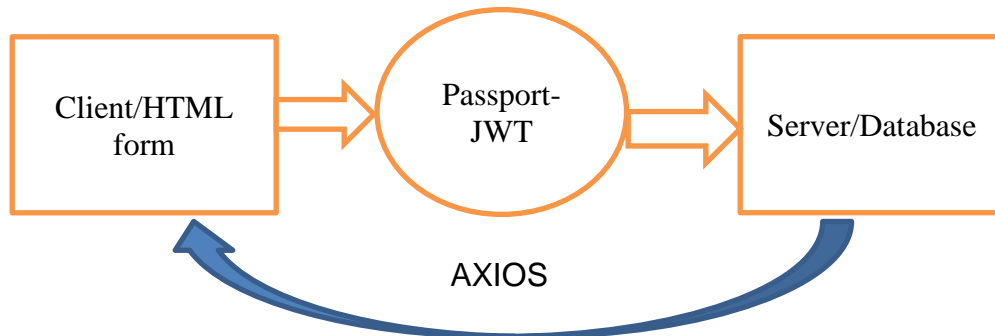


Figure 3. Demonstration of Passport working mechanism.

- Passport is a program that accepts login credentials from an HTML form and accesses the database.
- When the user filled out any form, it generates a token (which contains user information) and sends it to the database.
- After receiving the token from passport package, the database verifies that information and redirects the user to the login page. In any case, if the token information does not match, then the user will be redirected to the home screen.

CHAPTER THREE

SYSTEM DESIGN

Uml Diagrams

Use Case Diagram

A use case diagram help to understand what will happen on each event. With the help of the use case, we can find out the series of operation. In the use case, developer can create the overview of project working in the form of diagrams.

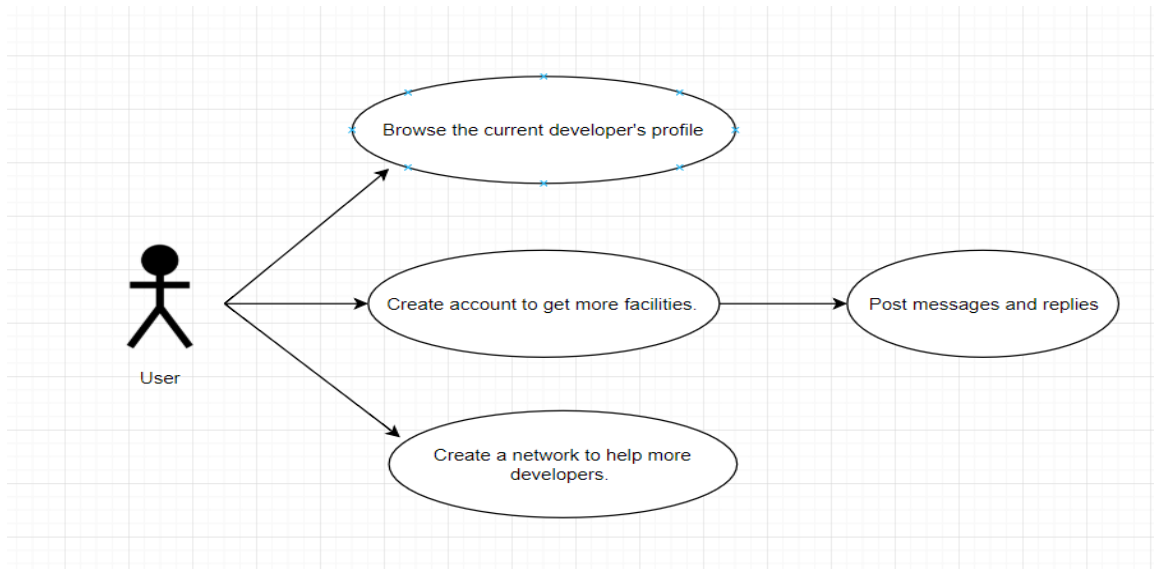


Figure 4. Use case diagram for new users.

Data Flow Diagram

The data flow diagram represents how the data flows in the system. It also provides a detailed description of each system component. The following are the data flow diagrams for the Login and Registration.

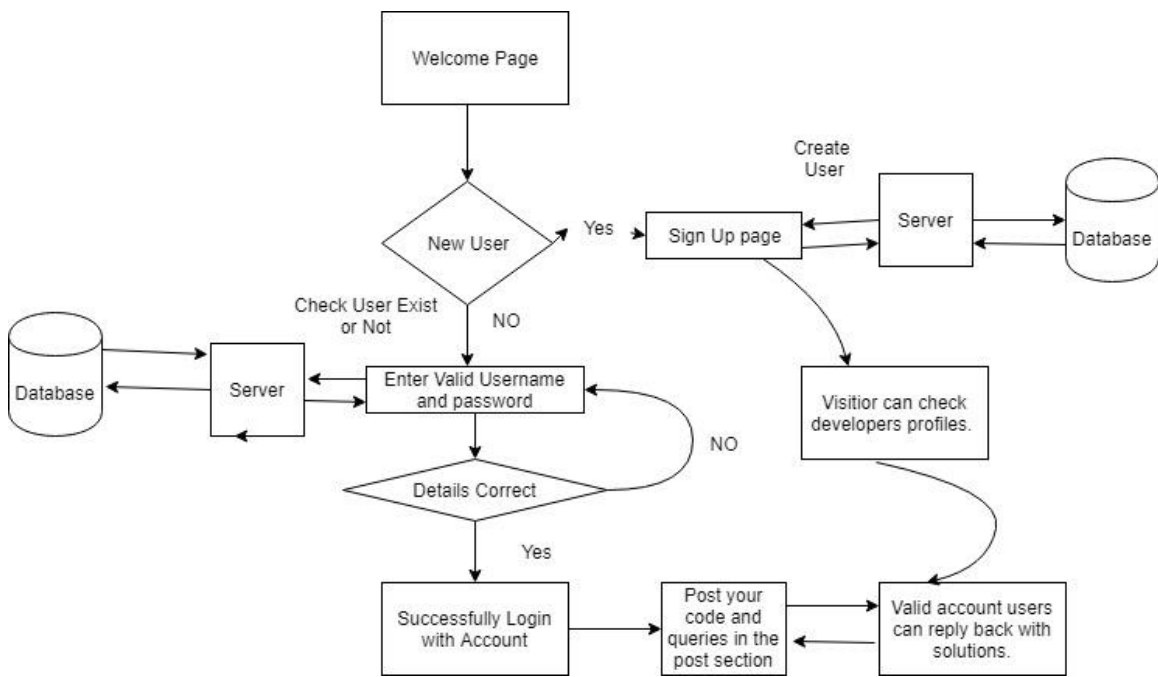


Figure 5. Data flow diagram for login and registration.

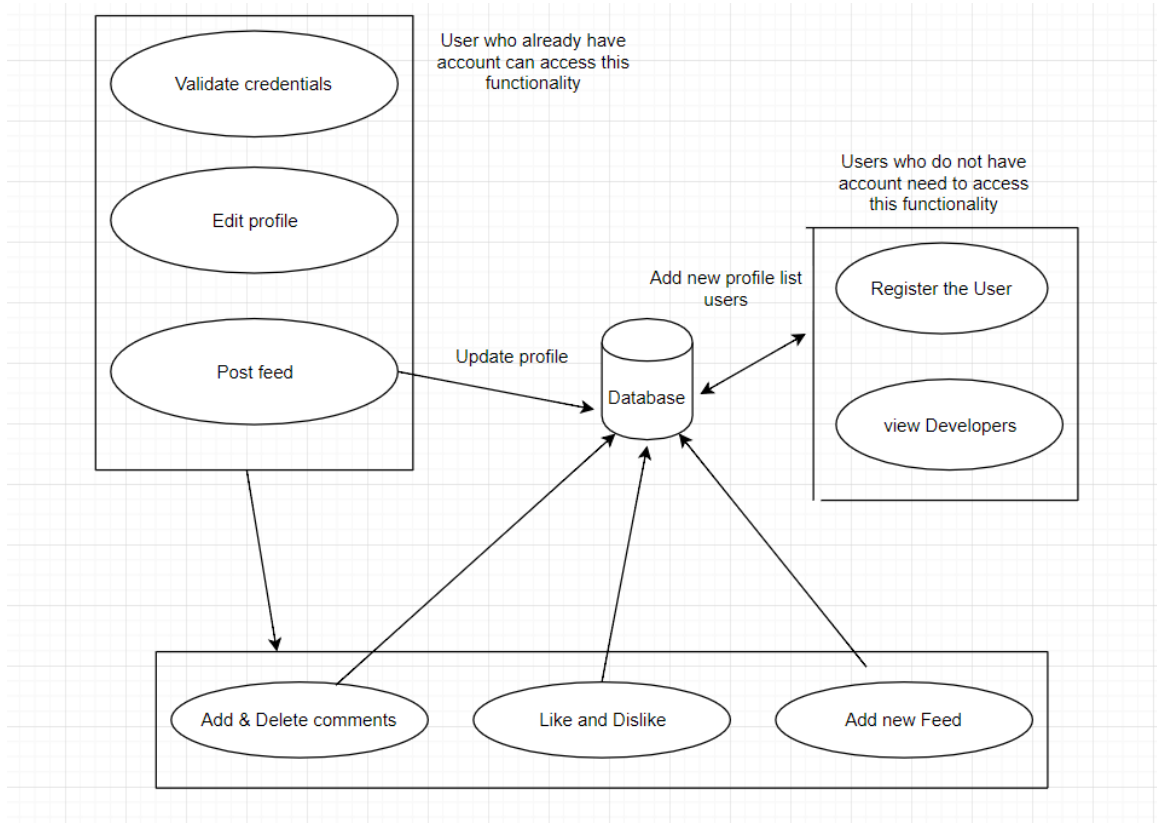


Figure 6. Data flow for the post feed

Sequence Diagram

In Sequence Diagram, object interactions are arranged in time sequence. Here in this case, objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are sometimes called event diagrams or event scenarios. A sequence diagram shows with a parallel vertical lines, different processes and objects that live simultaneously in those lines. In the horizontal arrows all the messages are exchanged.

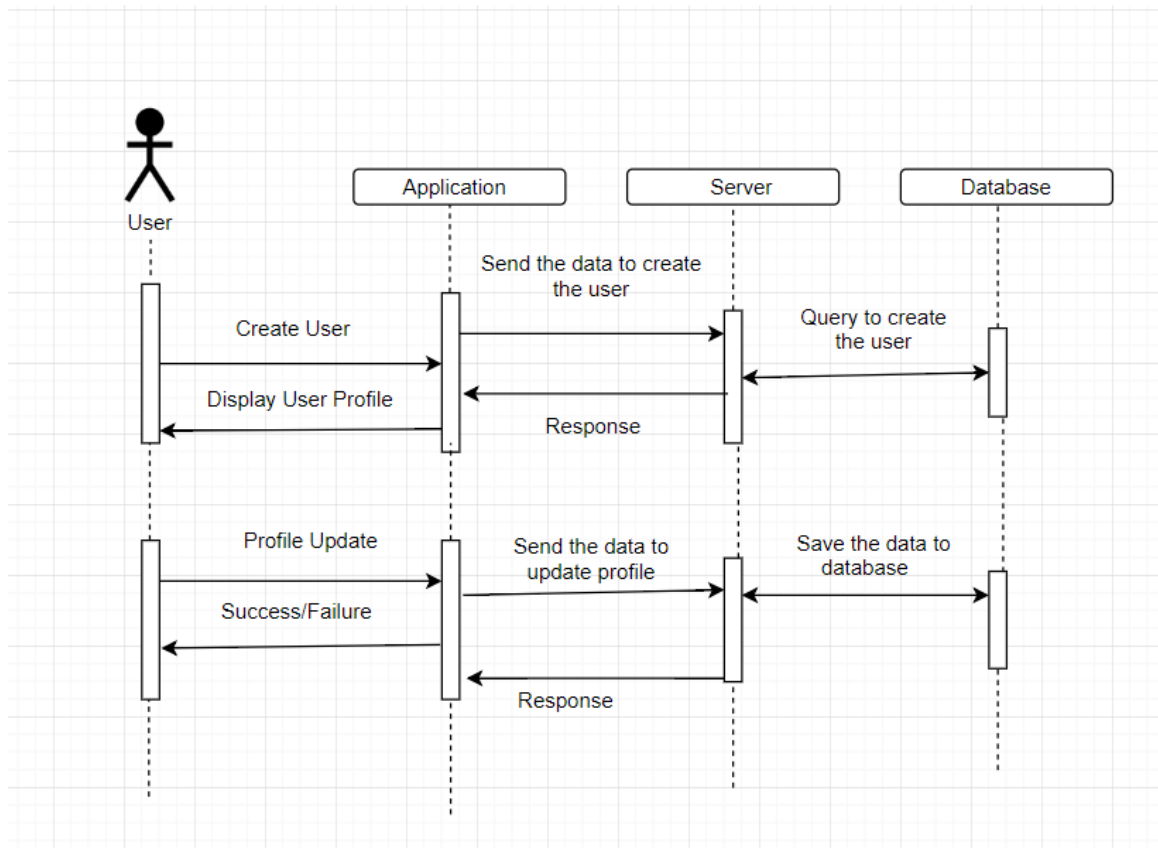


Figure 7. Sequence diagram for update user profile.

CHAPTER FOUR

SYSTEM TESTING

Testing Documentation

In software testing, we can find how computer software will operate. In testing, the developer checks the entire software for errors and corrections. Sometimes, the developer checks the individual part and sometimes the developer checks the whole system at once. Git bash is a software uses for testing a part of the computer. There are different routes are available in Software Testing, but I only used a couple of routes here.

Unit Testing

Software Testing can be performed in different ways. Each part is checked several times for errors and corrections. Software developers use Unit Testing every day. Different automation tools used at the end of the day to check whether things are working as mentioned in the scripts. Some organization brings a couple of parts of the software's together for testing. This can be done in several times. That is another type of testing which is called Module Testing. Module Testing is not included in this project.

User Acceptance Testing

In User Acceptance Testing, all the parts of the software are tested together. Before delivering this application to a client, developers check all the part whether they are performing as mentioned in the script. If a client checks the application, he can check all the parts are working properly as demanded. User Acceptance is the most widely use testing in the organization. User Acceptance is mandatory in all the organization. This is a good practice for successful application deployment.

Test Results

This project has been tested with the tests mentioned above in the section. During developing the software, I have tested this application for different errors and corrections. There are no problems are found in this application.

CHAPTER FIVE

SYSTEM SCREENSHOTS

System Working

Landing Page

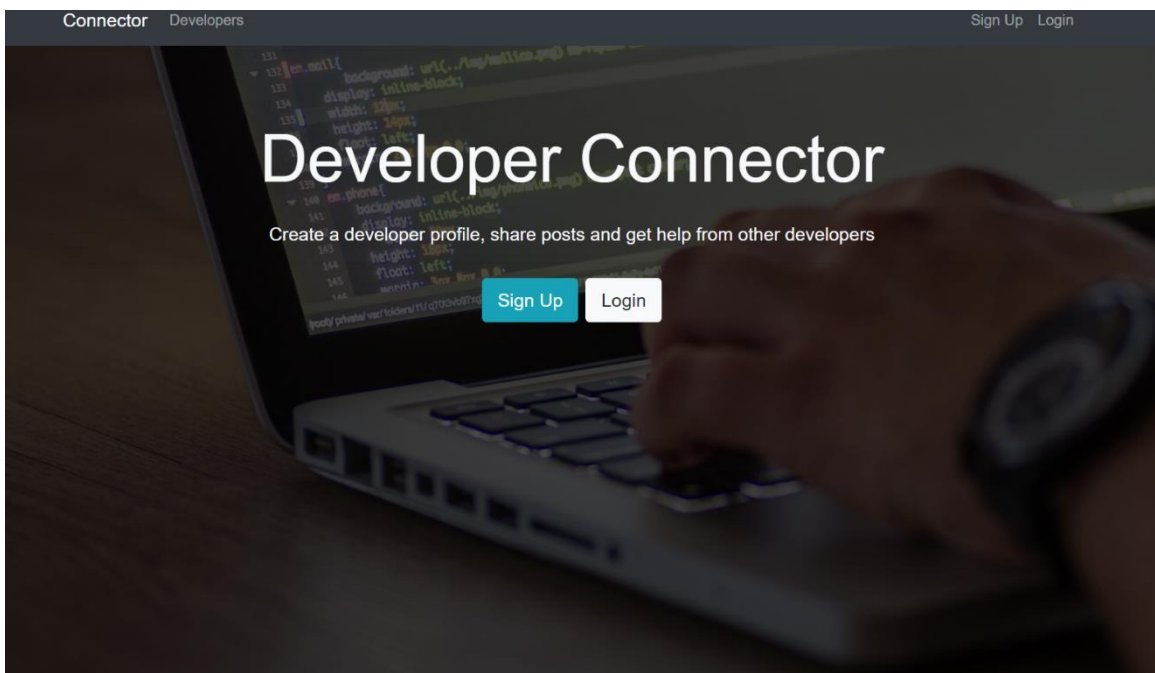


Figure 8. Landing page for the web application.

In the above picture, the user can see the landing page of the “Social Network for Software Developers”. Every user, whether signed up or not, will land on this web page.

As shown, on the right side, there are two options Sign Up and Login. On the left side, there is a home button (connector) and a Developers button.

If the user clicks on 'Developers' button, then all the developers who are signed up will be listed.

Developers Option

Developer Profiles

Browse and connect with developers

The screenshot displays two developer profiles in a list. Each profile includes a circular profile picture, the developer's name, their current role and location, a 'View Profile' button, and a 'Skill Set' section with a list of skills.

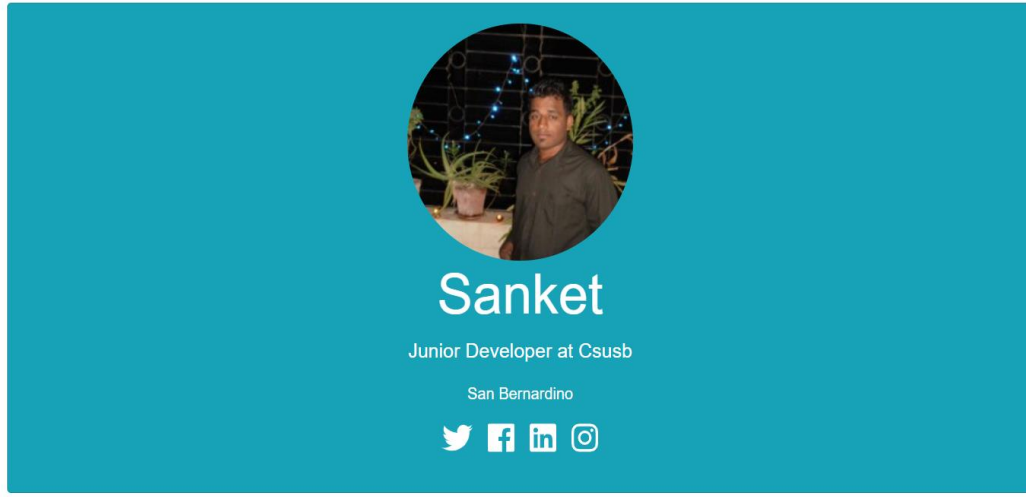
Developer Name	Role & Location	Skills
Sanket	Junior Developer at Csub San Bernardino	React, Angular, HTML, CSS
prasad	Senior Developer at List Group India	Java

Figure 9. Screenshot displayed when a user clicks on the developers' option.

As shown, any user who has not signed up can access this option. This is the only option non-users can access. The developers who are already signed will be listed here.

View Profile:

[Back To Profiles](#)



A user profile card for Sanket. It features a circular profile picture of a man in a dark shirt standing in front of a window with plants. The background of the card is a solid teal color. Below the picture, the name 'Sanket' is written in a large white font, followed by 'Junior Developer at Csub' and 'San Bernardino' in smaller white text. At the bottom of the card are icons for Twitter, Facebook, LinkedIn, and Instagram.

Sanket's Bio

I am a student of Computer Science in California state university san Bernardino. I am looking to pursue my career in the field of programming and development.

Figure 10. Display profile picture of the user.

Login

Connector Developers Sign Up Login

Log In

Sign in to your DevConnector account

[Submit](#)

Copyright © 2018 Connector

Figure 11. Display the login screen.

On the right-hand side, if the user clicks on the Login option, they will be redirected to this webpage. Only users who have signed up will be able to login.

Sign-Up

The screenshot shows a web page with a dark header. On the left, it says 'Connector Developers'. On the right, it says 'Sign Up Login'. The main content area is white and features the title 'Sign Up' in a large font, followed by the subtitle 'Create your DevConnector account'. Below this are four input fields: 'Name', 'Email', 'Password', and 'Confirm Password'. A small note below the 'Email' field reads 'This site uses Gravatar so if you want a profile image, use a Gravatar email'. At the bottom of the form is a teal 'Submit' button. The footer is a dark bar with the text 'Copyright © 2018 Connector'.

Figure 12. Sign up screen for new users.

When a user wants to join, he can choose the sign-up option. The user needs to fill in all the information. After that, the user will be redirected to another form which will ask about their personal information and professional information which will be saved as the user's profile.

Form Section

After signing up, the user will be required to complete forms for Experience and Education

Add Experience

Add any job or position that you have had in the past or current

* = required fields

* Company

* Job Title

Location

From Date

mm/dd/yyyy

To Date

mm/dd/yyyy

Current Job

Job Description

Tell us about the the position

Submit

Figure 13. Add experience form.

Add Education

Add any school, bootcamp, etc that you have attended

* = required fields

From Date

To Date

Current Job

Tell us about the program that you were in

Submit

Figure 14. Add education form.

Delete Information

Dashboard

Welcome [Sanket](#)

[Edit Profile](#) [Add Experience](#) [Add Education](#)

Experience Credentials

Company	Title	Years	
Reliance Communication	Technical Support Executive	2013/10/31 -2014/04/20	Delete
E-centric solution	Database Administrator	2015/02/28 -2015/05/31	Delete

Education Credentials

School	Degree	Years	
SCOE	Bachelor of Engineering	2009/08/31 -2013/05/31	Delete
California State University San Bernardino	Master of Science	2015/09/23 - Now	Delete

[Delete My Account](#)

Figure 15. Information about deleting account.

As shown in the above picture, if the user is signed up, in the dashboard section the user can see all his information which he has added to this application. The user can also delete the information which he filled out. Furthermore, at any point of time the user can go ahead and delete his account by clicking 'Delete my account'.

Post Section.

After logging into the application, the user will be seen one more option 'Post Feed'.

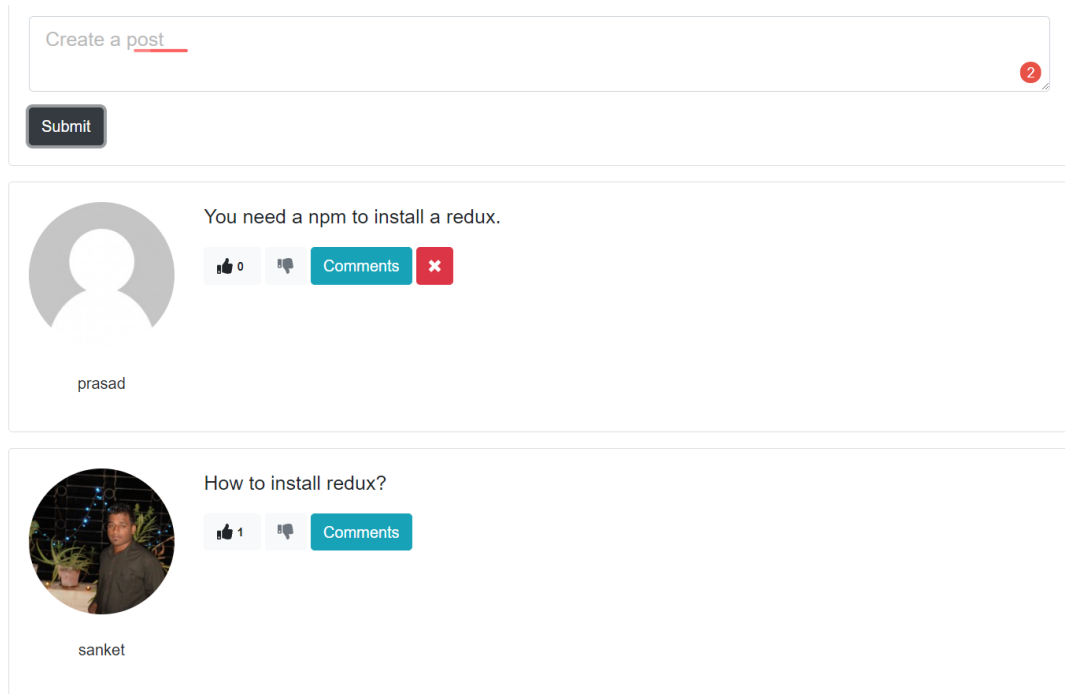


Figure 16. Illustration of the post section.

An authenticated user can post his questions and the other users can reply to the post. There is an option for like and dislikes. With the help of this option, other users can find out which is the best answer. Moreover, if any user posts a question, then he can go back and delete that post.

CHAPTER SIX

FUTURE ENHANCEMENTS

Future Project Development

Better Password Management

- Passwords will expire after a certain period of time.
- Old passwords cannot be re-used.
- Passwords must have a specific “Strength” to be used

Online Tutoring

A feature where senior developers can tutor other users on-line will enhance this project.

Job Openings

With this feature, aspiring developers can explore job opportunities.

APPENDIX
CODE FOR THE LOGIN SECTION.

CODE FOR THE LOGIN SECTION.

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';
import { connect } from 'react-redux';
import { loginUser } from '../actions/authActions';
import TextFieldGroup from '../common/TextFieldGroup';

class Login extends Component {
  constructor() {
    super();
    this.state = {
      email: "",
      password: "",
      errors: {}
    };

    this.onChange = this.onChange.bind(this);
    this.onSubmit = this.onSubmit.bind(this);
  }

  componentDidMount() {
    if (this.props.auth.isAuthenticated) {
```

```
        this.props.history.push('/dashboard');
    }
}

componentWillReceiveProps(nextProps) {
    if (nextProps.auth.isAuthenticated) {
        this.props.history.push('/dashboard');
    }

    if (nextProps.errors) {
        this.setState({ errors: nextProps.errors });
    }
}

onSubmit(e) {
    e.preventDefault();

    const userData = {
        email: this.state.email,
        password: this.state.password
    };
};
```

```

    this.props.loginUser(userData);
  }

  onChange(e) {
    this.setState({ [e.target.name]: e.target.value });
  }

  render() {
    const { errors } = this.state;

    return (
      <div className="login">
        <div className="container">
          <div className="row">
            <div className="col-md-8 m-auto">
              <h1 className="display-4 text-center">Log
In</h1>
              <p className="lead text-center">
                Sign in to your DevConnector account
              </p>
              <form onSubmit={this.onSubmit}>
                <TextFieldGroup

```

```
placeholder="Email Address"  
name="email"  
type="email"  
value={this.state.email}  
onChange={this.onChange}  
error={errors.email}  
/>
```

```
<TextFieldGroup  
  placeholder="Password"  
  name="password"  
  type="password"  
  value={this.state.password}  
  onChange={this.onChange}  
  error={errors.password}  
/>
```

```
<input type="submit" className="btn btn-info btn-  
block mt-4" />
```

```
</form>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
    </div>
  );
}
}
```

```
Login.propTypes = {
  loginUser: PropTypes.func.isRequired,
  auth: PropTypes.object.isRequired,
  errors: PropTypes.object.isRequired
};
```

```
const mapStateToProps = state => ({
  auth: state.auth,
  errors: state.errors
});
```

```
export default connect(mapStateToProps, { loginUser
})(Login);
```

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';
import { withRouter } from 'react-router-dom';
import { connect } from 'react-redux';
```

```
import { registerUser } from '../actions/authActions';
import TextFieldGroup from '../common/TextFieldGroup';

class Register extends Component {
  constructor() {
    super();
    this.state = {
      name: "",
      email: "",
      password: "",
      password2: "",
      errors: {}
    };

    this.onChange = this.onChange.bind(this);
    this.onSubmit = this.onSubmit.bind(this);
  }

  componentDidMount() {
    if (this.props.auth.isAuthenticated) {
      this.props.history.push('/dashboard');
    }
  }
}
```



```
}
```

```
componentWillReceiveProps(nextProps) {  
  if (nextProps.errors) {  
    this.setState({ errors: nextProps.errors });  
  }  
}
```

```
onChange(e) {  
  this.setState({ [e.target.name]: e.target.value });  
}
```

```
onSubmit(e) {  
  e.preventDefault();
```

```
  const newUser = {  
    name: this.state.name,  
    email: this.state.email,  
    password: this.state.password,  
    password2: this.state.password2  
  };
```

```

    this.props.registerUser(newUser, this.props.history);
  }

  render() {
    const { errors } = this.state;

    return (
      <div className="register">
        <div className="container">
          <div className="row">
            <div className="col-md-8 m-auto">
              <h1 className="display-4 text-center">Sign Up</h1>
              <p className="lead text-center">
                Create your DevConnector account
              </p>
              <form noValidate onSubmit={this.onSubmit}>
                <TextFieldGroup
                  placeholder="Name"
                  name="name"
                  value={this.state.name}
                  onChange={this.onChange}
                  error={errors.name}
                />
              </form>
            </div>
          </div>
        </div>
      </div>
    );
  }
}

```

```
/>
```

```
<TextFieldGroup
```

```
  placeholder="Email"
```

```
  name="email"
```

```
  type="email"
```

```
  value={this.state.email}
```

```
  onChange={this.onChange}
```

```
  error={errors.email}
```

```
  info="This site uses Gravatar so if you want a profile
```

```
image, use a Gravatar email"
```

```
/>
```

```
<TextFieldGroup
```

```
  placeholder="Password"
```

```
  name="password"
```

```
  type="password"
```

```
  value={this.state.password}
```

```
  onChange={this.onChange}
```

```
  error={errors.password}
```

```
/>
```

```
<TextFieldGroup
```

```
  placeholder="Confirm Password"
```

```
  name="password2"
```

```

        type="password"
        value={this.state.password2}
        onChange={this.onChange}
        error={errors.password2}
    />
    <input type="submit" className="btn btn-info btn-block
mt-4" />
    </form>
  </div>
</div>
</div>
</div>
);
}
}

```

```

Register.propTypes = {
  registerUser: PropTypes.func.isRequired,
  auth: PropTypes.object.isRequired,
  errors: PropTypes.object.isRequired
};

```

```

const mapStateToProps = state => ({
  auth: state.auth,
  errors: state.errors
});

export default connect(mapStateToProps, { registerUser
})(withRouter(Register));

import React, { Component } from 'react';
import PropTypes from 'prop-types';
import { connect } from 'react-redux';
import TextAreaFieldGroup from '../common/TextAreaFieldGroup';
import { addComment } from '../actions/postActions';

class CommentForm extends Component {
  constructor(props) {
    super(props);
    this.state = {
      text: "",
      errors: {}
    };

    this.onChange = this.onChange.bind(this);
  }

```

```
    this.onSubmit = this.onSubmit.bind(this);
  }

  componentWillReceiveProps(newProps) {
    if (newProps.errors) {
      this.setState({ errors: newProps.errors });
    }
  }
}

onSubmit(e) {
  e.preventDefault();

  const { user } = this.props.auth;
  const { postId } = this.props;

  const newComment = {
    text: this.state.text,
    name: user.name,
    avatar: user.avatar
  };

  this.props.addComment(postId, newComment);
}
```

```

    this.setState({ text: " " });
  }

  onChange(e) {
    this.setState({ [e.target.name]: e.target.value });
  }

  render() {
    const { errors } = this.state;

    return (
      <div className="post-form mb-3">
        <div className="card card-info">
          <div className="card-header bg-info text-white">
            Make a comment...
          </div>
          <div className="card-body">
            <form onSubmit={this.onSubmit}>
              <div className="form-group">
                <TextAreaFieldGroup
                  placeholder="Reply to post"
                  name="text"

```

```

        value={this.state.text}
        onChange={this.onChange}
        error={errors.text}
    />
</div>
<button type="submit" className="btn btn-dark">
    Submit
</button>
</form>
</div>
</div>
</div>
);
}
}

```

```

CommentForm.propTypes = {
    addPost: PropTypes.func.isRequired,
    auth: PropTypes.object.isRequired,
    postId: PropTypes.string.isRequired,
    errors: PropTypes.object.isRequired
};

```



```

const mapStateToProps = state => ({
  auth: state.auth,
  errors: state.errors
});

export default connect(mapStateToProps, { addComment
})(CommentForm);

import React, { Component } from 'react';
import { connect } from 'react-redux';
import PropTypes from 'prop-types';
import { deleteComment } from '../actions/postActions';

class CommentItem extends Component {
  onDeleteClick(postId, commentId) {
    this.props.deleteComment(postId, commentId);
  }

  render() {
    const { comment, postId, auth } = this.props;

    return (

```

```

<div className="card card-body mb-3">
  <div className="row">
    <div className="col-md-2">
      <a href="profile.html">
        <img
          className="rounded-circle d-none d-md-block"
          src={comment.avatar}
          alt=""
        />
      </a>
      <br />
      <p className="text-center">{comment.name}</p>
    </div>
    <div className="col-md-10">
      <p className="lead">{comment.text}</p>
      {comment.user === auth.user.id ? (
        <button
          onClick={this.onDeleteClick.bind(this, postId,
comment._id)}
          type="button"
          className="btn btn-danger mr-1"
        >

```

```

        <i className="fas fa-times" />
      </button>
    ) : null}
  </div>
</div>
</div>
);
}
}

CommentItem.propTypes = {
  deleteComment: PropTypes.func.isRequired,
  comment: PropTypes.object.isRequired,
  postId: PropTypes.string.isRequired,
  auth: PropTypes.object.isRequired
};

const mapStateToProps = state => ({
  auth: state.auth
});

export default connect(mapStateToProps, { deleteComment

```

```
})(CommentItem);
```

REFERENCES

- [1]. Reactjs (2013). Reactjs Introduction [Online].
Available: <https://Reactjs.org/docs/getting-started.html>
- [2]. Node.js (2017). Node.js Guides [Online]
Available: <https://Nodejs.org/en/docs/guides>
- [3]. Node package manager (2016). Npm services [Online]
Available: <https://docs.npmjs.com/>
- [4]. Mongodb (2018). Document-Oriented Database Program. [Online]
Available: <https://docs.Mongodb.com/manual/>
- [5]. Mlab (2011) Online database server [online]
Available: <https://docs.Mlab.com/>
- [6]. Postman (2011) Application to interact with HTTP api. [Online]
Available: <https://www.getpostman.com/docs/v6/>
- [7]. Bootstrap (2018) Javascript Library to design front-end [Online]
Available: <https://getbootstrap.com/docs/4.1/getting-started/introduction/>
- [8]. Javascript (2018). Core programming language to design everything,
Available: https://developer.mozilla.org/en-US/docs/Web/Javascript/Reference/Methods_Index
- [9]. Redux (2015). Redux Documentation [Online]
Available: <https://redux.js.org/>