**DISCUSSION BOARD**

*A*

*Project Report*

*Submitted in partial fulfilment of the*

*Requirements for the award of the Degree of*

**BACHELOR OF ENGINEERING**

IN

**INFORMATION TECHNOLOGY**

By

**T.AJAY(1602-17-737-006)**

**P.MANEESH(1602-17-737-018)**

**M.PRUDHVI RAJU(1602-17-737-026)**

*Under the guidance of*

**SRINIVASA CHAKRAVARTHY**

**Assistant Professor**



**Department of Information Technology**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University) Ibrahimbagh, Hyderabad-31**

**2019-2020**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University) Hyderabad-500 031**

**Department of Information Technology**



**DECLARATION BY THE CANDIDATE**

I, **T.AJAY, P.MANEESH, M.PRUDHVI RAJU** bearing hall ticket number **1602-17-737-006, 1602-17-737-018, 1602-17-737-026** hereby declare that the project report entitled **“DISCUSSION BOARD”** under the guidance of **SRINIVASA CHAKRAVARTHY**, Assistant Professor, Department of Information Technology, Vasavi College of Engineering, Hyderabad, is submitted in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Information Technology.**

This is a record of bonafide work carried out by me and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

**T.AJAY (1602-17-737-006)**

**P.MANEESH (1602-17-737-018)**

**M.PRUDHVI RAJU (1602-17-737-026)**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University) Hyderabad-500 031**

**Department of Information Technology**



**BONAFIDE CERTIFICATE**

Thisis to certify that the project entitled “**DISCUSSION BOARD”** being submitted by **T.AJAY,P.MANEESH,M.PRUDHVI RAJU** bearing **1602-17-737-006, 1602-17-737-018, 1602-17-737-026** in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Information Technology is a record of bonafide work carried out by him/her under my guidance.

**Srinivasa Chakravarthy Dr. K. Ram Mohan Rao**

**Assistant Professor HOD , IT**

**Internal Guide**

**ACKNOWLEDGEMENT**

The satisfaction that accompanies that the successful completion of the project would not have been possible without the kind support and help of many individuals. We would like to extend my sincere thanks to all of them.

We would like to take the opportunity to express our humble gratitude to Chakravarthy Sir under whom we executed this project.

We would also use this opportunity to thank our senior Manideep(1602-16-737-022). We are grateful to his guidance, and constructive suggestions that helped us in the preparation of this project. His constant guidance and willingness to share his vast knowledge made us understand this project and its manifestations in great depths and helped us to complete the assigned tasks. We would like to thank all faculty members and staff of the Department of Information Technology for their generous help in various ways for the completion of this project.

Finally, yet importantly, We would like to express our heartfelt thanks to our HOD Dr. K. Ram Mohan Rao Sir and classmates for their help and wishes for the successful completion of this project.

# 

# ABSTRACT

A lot of students lack behind the class, hesitate to ask the doubts in the class because they might think their doubts are silly and they might loose the interest in that subject . So our

aim is to find a solution for those students and we have come up with an idea of implementing a discussion board where the students can post their questions and they can get their solutions.

Discussion Board is a website where people can gather to have discussions about specific topic. All they have to do is register and login to get started. Once logged in, they can find the different modules like creation of query, searching a query and many more. So firstly, user can search for his/her doubts if the user doesn’t find any then he can create the query and wait for the respective solution till the admin or any other user posts it. So in this way students who are holding themselves back from asking their doubts in the classroom can get a platform to ask their doubts and can find a good solution.

**Table of Contents**

1.Introduction……………………………………………………………………………………….8

1.1 App title and Description……………………………………………………….8

1.2 Motivation……………………………………………………………………………..8

1.3 Objectives……………………………………………………………………………….8

1.4 Scope……………………………………………………………………………………..8

2.Software Analysis………………………………………………………………………………..9

2.1 General Description………………………………………………………………..9

2.1.1 Product Overview……………………………………….………......9

2.1.2 User Characteristics…….…………………………..……………….9

2.1.3 General Characteristics……………………………………………..9

2.2.System Requirements…………………….……………………………………….9

2.2.1 Hardware Requirements…….………………………………………9

2.2.2 Software Requirements………..…………………………………….10

2.3 System Features………………………………………………………………………10

2.4 Design Features……………………………………………………………………….10

3.Related Work……………………………………………………………………………………….10

4.System Design.…………………………………………………………………………………....11

4.1Architecture and Technology used………………………………………..…11

4.2 Design……………………………………………………………………………………..11

1.Use-Case Diagram……………………………………………………………..11

2.Class Diagram……………………………………………………………………12

3.Sequence Diagram…………………………………………………………….12

5. Implementation…………………………………………………………………………………..13

6.Results…………………………………………………………………………………………………82

7.Testing………………………….……………………………………………………………………..85

8.Conclusions and Future Scope………………………………….….……………………..85

9.References…………………………………………………………………………………………..86

1. **INTRODUCTION**
   1. **App Title and Description**

Discussion Board is an area where people can have conversations through posting their ideas ,responding to the other posts and also posting questions. For example ,we have Quora application where users will post their doubts related to any kind and then either another user will or a Quora representative will reply with the suggestions. So ,In an educational environment ,a Discussion board can be created such that students are able to gather their ideas and thoughts before posting and sharing their reflective responses ,which leads to indepth learning.

**1.2 Motivation**

Discussion Board builds a community with the peers ,it creates opportunities for students to practice and also many respond to the other queries by posting their own solution. So here we use questions that will encourage the students to explore ,compare ,discuss and reflect their responses based on their personal experience and we have some instructors who engage with the students by interacting with them within the discussion board.

**1.3 Objectives**

Design a application that creates a communication medium for students who wants to clarify their doubts and can join people who are already a part of this board .

1. Students can get the solutions for their queries.
2. To give all students a voice ,especially those who are normally quiet and hesitate to ask their doubts in the classroom.

**1.4 Scope**

We designed this web application mainly for the engineering students who hesitate to ask their doubts in the classroom irrespective of the subject.e he

**2.SOFTWARE ANALYSIS**

**2.1 GENERAL CHARACTERISTICS**

* + 1. **Product Overview:** The discussion Board consists of different modules depending upon the user role he/she will be able to access only specific modules like the admin can remove some kind of queries which he/she thinks irrelevant and it is userfriendly.
    2. **User Characteristics :** User should have certain qualifications to log in to our website and to enter his/her specific queries regarding subject like he must know some basic English and atleast matriculation. The user registers himself and logs in through his registered account.Once logged in, the user can see the homepage which displays all the queries that are present in the system.
    3. **General Characteristics :** We can run this web application on any web browser.

**2.2 SYSTEM REQUIREMENTS**

**2.2.1 Hardware Requirements**

* 1 GB Ram
* 1 GB Core

**2.2.1 Software Requirements**

* Node Js
* MongoDB

**2.3 SYSTEM FEATURES**

This Web Application consists of two different modules they are :

1. Administrator
2. Student

Administrator :

So he/she is the one who maintains all the database which contains each and every data regarding the user details ,queries and their solutions.

Student :

Here the student is the user for our application.He/she should login through their respective logins if they don’t have the account they must signup .Once they login they can either search for a query or create a query.

**2.4 DESIGN CONSTRAINTS**

Software Constraints : User can run this application either in Linux or windows but need to install the nodeJS and mongoDB.

Hardware Constraints :The system will run on a core processor with minimum 512MB ram.

Acceptance Criteria : Before accepting ,the developer must check whether the application is running properly or not and should also check whether the data is stored correctly.

**3.RELATED WORK**

MARKET REASEARCH :

QUORA : Quora is an question and answer website where questions are asked answers, followed and edited by the internet users .Quora also provide the interesting questions that are relevant to the users preferences .

Stanford |teaching commons :This a online Forum ,where students can have discussions with each other on course related topics .Students contribute to the discussion by posting messages to an online message board.

Limitations :

This system doesn’t gives us the best answer for the query that the user searched. So it might be difficult for the user to select the best answer among all the solutions that other users posted .

**4.SYSTEM DESIGN**

**4.1 ARCHITECTURE AND TECHNOLOGY USE**

Front End :

1) React

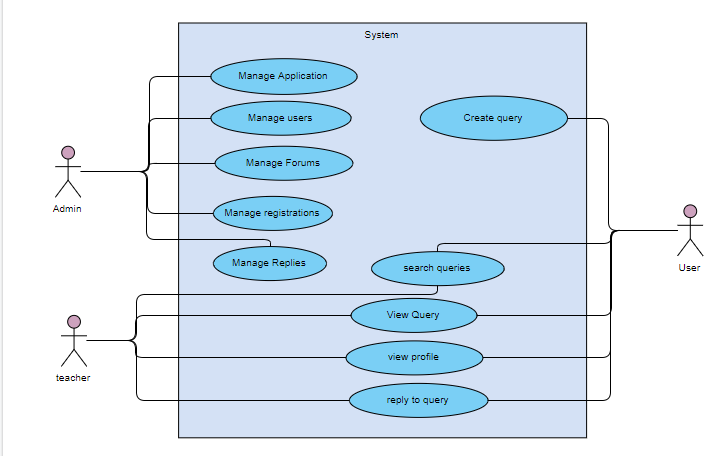
Back end :

1)NodeJS

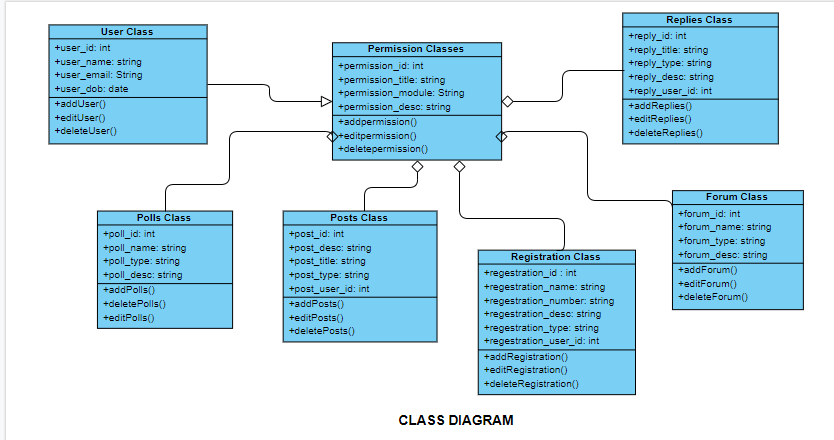
2)Express

**4.2 UML DIAGRAMS**

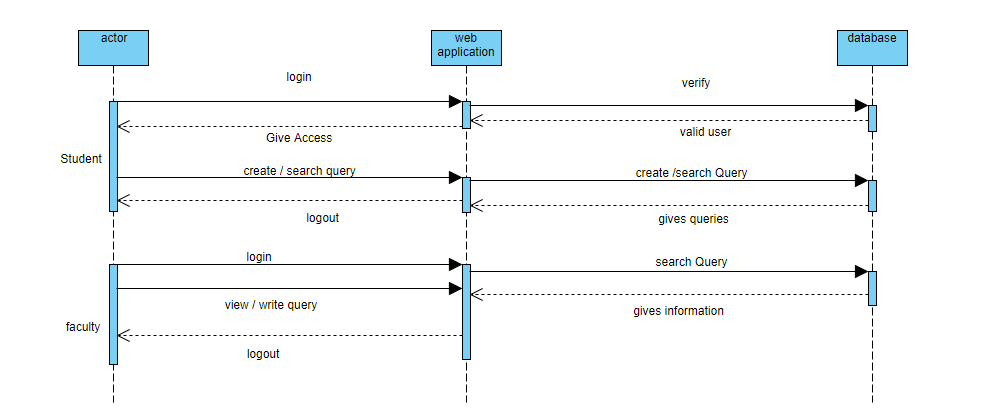
1) **Use-Case Diagram** :

****

1. **Class Diagram** :



1. **Sequence Diagram** :



**5.IMPLEMENTATION CODE**

**FRONT END**

Index.Js File

import React from "react";

import ReactDOM from "react-dom";

import "./index.css";

import App from "./App";

import logo from "./vlogo.gif";

import axios from "axios";

import \* as serviceWorker from "./serviceWorker";

import Createquery from "./createquery";

import Viewqueries from "./viewing/viewqueries";

import ViewSolutions from "./viewing/viewsolutions";

function logindata() {

  let userd;

}

class View extends React.Component {

  constructor(props) {

    super(props);

    this.state = {

      navbar: true,

      loginpage: true,

      userdata: undefined,

      createquery: false,

      viewquery: false,

      queryId: "",

      render: "login",

      viewsolutions: false,

      solutionId: "",

      question: undefined,

    };

    this.changehandler = this.changehandler.bind(this);

    this.forrendering = this.forrendering.bind(this);

  }

  changehandler(param) {

    //  e.preventDefault()

    this.setState(param);

  }

  forrendering(param) {

    switch (param) {

      case "login":

        return <Login data="anudeep" c={this.changehandler} />;

      case "viewquery":

        return <Viewqueries c={this.changehandler} />;

      case "viewsolutions":

        return (

          <ViewSolutions

            queryId={this.state.queryId}

            userdata={this.state.userdata}

            c={this.changehandler}

            question={this.state.question}

          />

        );

      case "createquery":

        console.log("this is userdata");

        console.log(this.state.userdata);

        return (

          <Createquery userdata={this.state.userdata} c={this.changehandler} />

        );

      case "about":

        return <About c={this.changehandler} />;

      case "logout":

        return <Login data="anudeep" c={this.changehandler} />;

    }

  }

  render() {

    return (

      <div>

        <Navbar c={this.changehandler} userdata={this.state.userdata} />

        {this.forrendering(this.state.render)}

      </div>

    );

  }

}

export default logindata;

class About extends React.Component {

  constructor(props) {

    super(props);

    this.state = {};

    this.movetologin = this.movetologin.bind(this);

  }

  movetologin() {

    this.props.c({ render: "login" });

  }

  render() {

    return (

      <html>

        <head>

          <link

            rel="stylesheet"

            href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"

            integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"

            crossorigin="anonymous"

          />

          <link

            rel="stylesheet"

            href="/css/app.component.css"

            type="text/css"

          />

          <script

            type="text/js"

            src="node\_modules/react/dist/react.min.js"

          ></script>

          <script

            type="text/js"

            src="node\_modules/react-dom/dist/react-dom.min.js"

          ></script>

        </head>

        <body>

          <div>

            <h5

              style={{

                marginLeft: "20%",

                marginRight: "20%",

                marginTop: "5% ",

              }}

            >

              This project is aimed at developing online form for the group

              discussion This is a web-based tool. Any user can post the doubts

              topics and can reply for the other user doubts. The user can

              invites others for Discussion and submit query. This is useful for

              a small office, school or a department or for that matter any

              group who is interested to organize it effectively. Facility to

              share the resource and post articles that can be viewed by

              registered user.

            </h5>

            <h5

              style={{

                fontStyle: "serif",

                marginLeft: "32%",

                marginRight: "20%",

                marginTop: "5%",

                fontSize: "36px",

              }}

            >

              Your search for the solution ends here...!!{" "}

            </h5>

          </div>

        </body>

      </html>

    );

  }

}

class Navbar extends React.Component {

  constructor(props) {

    super(props);

    this.state = {};

    this.forrendercreatequery = this.forrendercreatequery.bind(this);

    this.forenderviewquery = this.forenderviewquery.bind(this);

    this.forrenderabout = this.forrenderabout.bind(this);

    this.forlogout = this.forlogout.bind(this);

    this.forlogin = this.forlogin.bind(this);

  }

  forrendercreatequery() {

    if (this.props.userdata != undefined) {

      this.props.c({ render: "createquery" });

    }

  }

  forenderviewquery() {

    if (this.props.userdata != undefined) {

      this.props.c({ render: "viewquery" });

    }

  }

  forrenderabout() {

    {

      this.props.c({ render: "about" });

    }

  }

  forlogout() {

    this.props.c({ render: "logout" });

  }

  forlogin() {

    this.props.c({ render: "login" });

  }

  render() {

    return (

      <html>

        <head>

          <link

            rel="stylesheet"

            href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"

            integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"

            crossorigin="anonymous"

          />

          <link

            rel="stylesheet"

            href="/css/app.component.css"

            type="text/css"

          />

          <script

            type="text/js"

            src="node\_modules/react/dist/react.min.js"

          ></script>

          <script

            type="text/js"

            src="node\_modules/react-dom/dist/react-dom.min.js"

          ></script>

        </head>

        <body>

          <div>

            <div className="header ">

              <img

                src={logo}

                style={{ position: "absolute" }}

                align="center"

                height="140"

                width="140"

                alt="logo"

              />

              <h1 style={{ textAlign: "center", color: "white" }}>

                VCE Discussion-Board

              </h1>

              <p style={{ marginLeft: "55%", color: "white" }}>

                {" "}

                your search for the solution ends here...!!!

              </p>

              <ul className="nav justify-content-center">

                <li className="nav-item">

                  <a

                    className="nav-link active"

                    href="#"

                    tabindex="-1"

                    style={{ color: "yellow" }}

                    onClick={this.forlogin}

                  >

                    Log In

                  </a>

                </li>

                <li className="nav-item">

                  <a

                    className="nav-link active"

                    href="#"

                    style={{ color: "yellow" }}

                    onClick={this.forrendercreatequery}

                  >

                    Createquery

                  </a>

                </li>

                <li className="nav-item">

                  <a

                    className="nav-link active"

                    href="#"

                    style={{ color: "yellow" }}

                    onClick={this.forrenderabout}

                  >

                    About

                  </a>

                </li>

                <li className="nav-item">

                  <a

                    className="nav-link active"

                    style={{ color: "yellow" }}

                    href="#"

                    tabindex="-1"

                    onClick={this.forenderviewquery}

                  >

                    Viewqueries

                  </a>

                </li>

                <li className="nav-item">

                  <a

                    className="nav-link active"

                    style={{ color: "yellow" }}

                    href="#"

                    tabindex="-1"

                    onClick={this.forlogout}

                  >

                    Logout

                  </a>

                </li>

              </ul>

            </div>

          </div>

        </body>

      </html>

    );

  }

}

class Login extends React.Component {

  constructor(props) {

    super(props);

    this.state = {

      firstName: "",

      lastName: "",

      signupemail: "",

      signuppassword: "",

      mobilenumber: "",

      email: "ajay@gmail.com",

      password: "123456",

      nav: props.data,

    };

    this.handlechange = this.handlechange.bind(this);

    this.signup = this.signup.bind(this);

    this.login = this.login.bind(this);

  }

  handlechange(event) {

    this.setState({

      [event.target.name]: event.target.value,

    });

  }

  login(e) {

    // console.log("clicked on login", Date.now());

    e.preventDefault();

    // let userDetails = {

    //   userId: "aj",

    //   firstName: "aj",

    //   lastName: "aj",

    //   password: "aj",

    //   email: "aj@gmail.com",

    //   mobileNumber: 2525252525,

    //   createdOn: 1587663197939,

    // };

    // logindata.userd = userDetails;

    // this.props.c({ userdata: userDetails });

    // // alert("success")

    // this.props.c({ render: "viewquery" });

    axios

      .post("http://127.0.0.1:3001/api/v1/users/login", this.state)

      .then((response) => {

        logindata.userd = response.data.data.userDetails;

        this.props.c({ userdata: response.data.data.userDetails });

        // alert("success")

        this.props.c({ render: "viewquery" });

      })

      .catch((error) => {

        alert("Error");

        console.log(error);

      });

  }

  signup(e) {

    let signupdetails = {

      firstName: this.state.firstName,

      lastName: this.state.lastName,

      mobileNumber: this.state.mobilenumber,

      email: this.state.signupemail,

      password: this.state.signuppassword,

    };

    e.preventDefault();

    axios

      .post("http://127.0.0.1:3001/api/v1/users/signup", signupdetails)

      .then((response) => {

        // alert("success");

      })

      .catch((error) => {

        alert("Error");

        console.log(error);

      });

  }

  render() {

    return (

      <html>

        <head>

          <link

            rel="stylesheet"

            href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"

            integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"

            crossorigin="anonymous"

          />

          <link

            rel="stylesheet"

            href="/css/app.component.css"

            type="text/css"

          />

          <script

            type="text/js"

            src="node\_modules/react/dist/react.min.js"

          ></script>

          <script

            type="text/js"

            src="node\_modules/react-dom/dist/react-dom.min.js"

          ></script>

          <style></style>

        </head>

        <body>

          <div className="col-md-3 one">

            <div className="form-group ">

              <form onSubmit={this.login}>

                <input

                  type="email"

                  className="form-control"

                  name="email"

                  value={this.state.email}

                  placeholder="Email"

                  onChange={this.handlechange}

                />

                <br />

                <input

                  type="password"

                  className="form-control"

                  name="password"

                  value={this.state.password}

                  placeholder="Password"

                  onChange={this.handlechange}

                />

                <br />

                <button

                  type="submit"

                  className="btn btn-dark"

                  style={{ marginLeft: "41%" }}

                >

                  LogIn

                </button>

              </form>

            </div>

          </div>

          <br />

          <hr />

          <h3 style={{ textAlign: "center" }}>Not Registered?</h3>

          <div

            style={{ marginLeft: "30%", marginRight: "30%", marginTop: "1%" }}

          >

            <form onSubmit={this.signup}>

              <div className="form-group">

                <input

                  type="text"

                  className="form-control"

                  name="firstName"

                  value={this.state.firstName}

                  aria-describedby="emailHelp"

                  onChange={this.handlechange}

                  placeholder="Enter First Name"

                  required

                />

                <br />

                <input

                  type="text"

                  className="form-control"

                  name="lastName"

                  value={this.state.lastName}

                  onChange={this.handlechange}

                  aria-describedby="emailHelp"

                  placeholder="Enter Last Name"

                  required

                />

                <br />

                <input

                  type="email"

                  className="form-control"

                  name="signupemail"

                  value={this.state.signupemail}

                  onChange={this.handlechange}

                  aria-describedby="emailHelp"

                  placeholder="Enter EmailId"

                  required

                />

                <br />

                <input

                  type="password"

                  className="form-control"

                  name="signuppassword"

                  onChange={this.handlechange}

                  value={this.state.signuppassword}

                  aria-describedby="emailHelp"

                  placeholder="Enter Password"

                  required

                />

                <br />

                <input

                  type="text"

                  className="form-control"

                  name="mobilenumber"

                  value={this.state.mobilenumber}

                  onChange={this.handlechange}

                  aria-describedby="emailHelp"

                  placeholder="Enter Mobile number"

                  required

                />

              </div>

              <button type="submit" className="btn btn-success">

                Submit

              </button>

            </form>

          </div>

          <footer className="page-footer font-small blue">

            <div className="footer-copyright text-center py-3">

              about us:

              <a href="https://w3.com"> Ajay, Maneesh, Prudhvi</a>

            </div>

            <div className="footer-copyright text-center py-3">

              <p>Studying at Vasavi College of Engineering, Department of IT</p>

            </div>

          </footer>

        </body>

      </html>

    );

  }

}

ReactDOM.render(<View />, document.getElementById("root"));

// If you want your app to work offline and load faster, you can change

// unregister() to register() below. Note this comes with some pitfalls.

// Learn more about service workers: https://bit.ly/CRA-PWA

serviceWorker.unregister();

**CreateQuery.js File**

import React from "react";

import ReactDOM from "react-dom";

import "./index.css";

import axios from "axios";

class Createquery extends React.Component {

  constructor(props) {

    super(props);

    this.state = {

      query: "",

      tag: "",

    };

    this.handlequery = this.handlequery.bind(this);

    this.handlebutton = this.handlebutton.bind(this);

    this.handleform = this.handleform.bind(this);

  }

  handlequery(e) {

    this.setState({

      [e.target.name]: e.target.value,

    });

  }

  handlebutton(e) {

    this.setState({

      [e.target.name]: e.target.value,

    });

  }

  handleform(e) {

    e.preventDefault();

    let querydata = {

      query: this.state.query,

      tag: this.state.tag,

      username:

        this.props.userdata.firstName + " " + this.props.userdata.lastName,

      userId: this.props.userdata.userId,

    };

    console.log("This is query data");

    console.log(querydata);

    axios

      .post("http://127.0.0.1:3001/api/v1/query/create/", querydata)

      .then((response) => {

        // alert("success");

        this.props.c({ render: "viewquery" });

      })

      .catch((error) => {

        alert("Error");

        console.log(error);

      });

  }

  render() {

    return (

      <html>

        <head>

          <link

            rel="stylesheet"

            href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"

            integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"

            crossorigin="anonymous"

          />

          <link

            rel="stylesheet"

            href="/css/app.component.css"

            type="text/css"

          />

          <script

            type="text/js"

            src="node\_modules/react/dist/react.min.js"

          ></script>

          <script

            type="text/js"

            src="node\_modules/react-dom/dist/react-dom.min.js"

          ></script>

        </head>

        <body>

          <div className="container">

            <h2 align="center">ADD QUERY</h2>

            <div

              className="row"

              style={{

                textAlign: "left",

                marginLeft: "15%",

                marginRight: "15%",

              }}

            >

              <div className="col-md-12">

                <form onSubmit={this.handleform}>

                  <div className="form-group"></div>

                  <div className="form-group">

                    <textarea

                      name="query"

                      className="form-control"

                      rows="3"

                      placeholder="write your query.."

                      onChange={this.handlequery}

                      required

                    ></textarea>

                  </div>

                  <button

                    type="button"

                    className="btn btn-dark"

                    style={{ marginLeft: "15%", marginRight: "3%" }}

                    name="tag"

                    onClick={this.handlebutton}

                    value="OperatingSystem"

                  >

                    OperatingSystem

                  </button>

                  <button

                    type="button"

                    className="btn btn-dark"

                    style={{ marginRight: "3%" }}

                    name="tag"

                    onClick={this.handlebutton}

                    value="ComputerNetworks"

                  >

                    ComputerNetworks

                  </button>

                  <button

                    type="button"

                    className="btn btn-dark"

                    style={{ marginRight: "3%" }}

                    name="tag"

                    onClick={this.handlebutton}

                    value="ReactJs"

                  >

                    ReactJs

                  </button>

                  <button

                    type="button"

                    className="btn btn-dark"

                    style={{ marginRight: "3%" }}

                    name="tag"

                    onClick={this.handlebutton}

                    value="NodeJs"

                  >

                    NodeJs

                  </button>

                  <h1 style={{ padding: 10, marginLeft: "25%" }}>

                    tag:{this.state.tag}

                  </h1>

                  <br />

                  <br />

                  <br />

                  <button

                    className="btn btn-success"

                    style={{ marginLeft: "35%" }}

                  >

                    Post the Query

                  </button>

                </form>

              </div>

            </div>

          </div>

        </body>

      </html>

    );

  }

}

export default Createquery;

**ViewQueries.Js File**

import React from "react";

import ReactDOM from "react-dom";

import "../index.css";

import axios from "axios";

import Querycard from "../card/querycard";

import logindata from "../index";

class ViewQueries extends React.Component {

  constructor(props) {

    super(props);

    this.state = {

      querydata: undefined,

      tag: "",

    };

    this.buttonclicked = this.buttonclicked.bind(this);

    this.rerender = this.rerender.bind(this);

    this.myqueries = this.myqueries.bind(this);

  }

  componentDidMount() {

    console.log("this in component did mount");

    axios

      .get("http://127.0.0.1:3001/api/v1/query/all/" + this.state.tag)

      .then((response) => {

        console.log(response.data);

        const queriesdata = response.data.map((obj) => (

          <Querycard key={obj.queryId} data={obj} c={this.props.c} />

        ));

        console.log(queriesdata);

        this.setState({ querydata: queriesdata });

        console.log(this.state.querydata);

      });

  }

  rerender(val) {

    console.log("this in 2component did mount");

    axios

      .get("http://127.0.0.1:3001/api/v1/query/all/" + val)

      .then((response) => {

        console.log(response);

        if (response.data != "no queries found") {

          const queriesdata = response.data.map((obj) => (

            <Querycard key={obj.queryId} data={obj} c={this.props.c} />

          ));

          console.log(queriesdata);

          this.setState({ querydata: queriesdata });

          console.log(this.state.querydata);

        } else {

          this.setState({ querydata: "no queries" });

        }

      });

  }

  myqueries() {

    axios

      .get(

        "http://127.0.0.1:3001/api/v1/query/all/mine/" + logindata.userd.userId

      )

      .then((response) => {

        console.log(response);

        if (response.data != "no queries found") {

          const queriesdata = response.data.map((obj) => (

            <Querycard key={obj.queryId} data={obj} c={this.props.c} />

          ));

          console.log(queriesdata);

          this.setState({ querydata: queriesdata });

          console.log(this.state.querydata);

        } else {

          this.setState({ querydata: "no queries" });

        }

      });

  }

  buttonclicked(e) {

    this.setState({

      [e.target.name]: e.target.value,

    });

    this.rerender(e.target.value);

  }

  render() {

    return (

      <html>

        <head>

          <link

            rel="stylesheet"

            href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"

            integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"

            crossorigin="anonymous"

          />

          <link

            rel="stylesheet"

            href="/css/app.component.css"

            type="text/css"

          />

          <script

            type="text/js"

            src="node\_modules/react/dist/react.min.js"

          ></script>

          <script

            type="text/js"

            src="node\_modules/react-dom/dist/react-dom.min.js"

          ></script>

        </head>

        <body>

          <div style={{ marginTop: "3%" }}>

            <button

              className="btn  btn-success"

              style={{ marginLeft: "30%" }}

              name="tag"

              value="OperatingSystem"

              onClick={this.buttonclicked}

            >

              OperatingSystem

            </button>

            <button

              className="btn btn-success"

              style={{ marginLeft: "2%" }}

              name="tag"

              value="ComputerNetworks"

              onClick={this.buttonclicked}

            >

              ComputerNetworks

            </button>

            <button

              className="btn btn-success"

              style={{ marginLeft: "2%" }}

              name="tag"

              value="ReactJs"

              onClick={this.buttonclicked}

            >

              ReactJs

            </button>

            <button

              className="btn btn-success"

              style={{ marginLeft: "2%" }}

              name="tag"

              value="NodeJs"

              onClick={this.buttonclicked}

            >

              NodeJs

            </button>

            <button

              className="btn btn-success"

              style={{ marginLeft: "2%" }}

              name="tag"

              value=""

              onClick={this.buttonclicked}

            >

              all

            </button>

            {this.state.querydata == "no queries" && (

              <h3 style={{ marginLeft: "35%", marginTop: "10%" }}>

                No Queries to Display{" "}

              </h3>

            )}

            {this.state.querydata != "no queries" && (

              <div>{this.state.querydata}</div>

            )}

          </div>

        </body>

      </html>

    );

  }

}

export default ViewQueries;

**ViewSolutions.js File**

import React from "react";

import ReactDOM from "react-dom";

import "../index.css";

import axios from "axios";

import Solutioncard from "../card/solutioncard";

import logindata from "../index";

class ViewSolutions extends React.Component {

  constructor(props) {

    super(props);

    this.state = {

      solution: "",

      solutions: undefined,

      queryid: this.props.queryId,

      username:

        this.props.userdata.firstName + " " + this.props.userdata.lastName,

      userId: this.props.userdata.userId,

      voted: [1],

      upvotes: 0,

    };

    this.solutionhandler = this.solutionhandler.bind(this);

    this.formaftersubmitting = this.formaftersubmitting.bind(this);

    this.componentforrerender = this.componentforrerender.bind(this);

    this.forceUpdate();

  }

  componentDidMount() {

    axios

      .get("http://127.0.0.1:3001/api/v1/solution/all/" + this.state.queryid)

      .then((response) => {

        //alert("success")

        console.log(response.data);

        const solutiondata = response.data.map((obj) => (

          <Solutioncard

            key={obj.solutionId}

            data={obj}

            c={this.props.c}

            queryId={this.state.queryid}

            click={this.componentforrerender}

          />

        ));

        console.log(solutiondata);

        this.setState({ solutions: solutiondata });

        this.props.c({ createquery: false, viewquery: true, loginpage: false });

        console.log(logindata.userd);

      })

      .catch((error) => {

        //alert("Error")

        console.log(error);

      });

    this.forceUpdate();

  }

  componentforrerender() {

    axios

      .get("http://127.0.0.1:3001/api/v1/solution/all/" + this.state.queryid)

      .then((response) => {

        //alert("success")

        console.log(response.data);

        const solutiondata = response.data.map((obj) => (

          <Solutioncard

            key={obj.solutionId}

            data={obj}

            c={this.props.c}

            queryId={this.state.queryid}

          />

        ));

        console.log(solutiondata);

        this.setState({ solutions: solutiondata });

        this.props.c({ createquery: false, viewquery: true, loginpage: false });

      })

      .catch((error) => {

        //alert("Error")

        console.log(error);

      });

  }

  solutionhandler(e) {

    this.setState({

      [e.target.name]: e.target.value,

    });

  }

  formaftersubmitting(e) {

    e.preventDefault();

    // alert("entered form after submitting method " + this.state.queryid);

    axios

      .post(

        "http://127.0.0.1:3001/api/v1/solution/create/" + this.state.queryid,

        this.state

      )

      .then((response) => {

        //  alert("success");

        this.componentforrerender();

      })

      .catch((error) => {

        alert("Error");

        console.log(error);

      });

  }

  render() {

    return (

      <html>

        <head>

          <link

            rel="stylesheet"

            href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"

            integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"

            crossorigin="anonymous"

          />

          <link

            rel="stylesheet"

            href="/css/app.component.css"

            type="text/css"

          />

          <script

            type="text/js"

            src="node\_modules/react/dist/react.min.js"

          ></script>

          <script

            type="text/js"

            src="node\_modules/react-dom/dist/react-dom.min.js"

          ></script>

        </head>

        <body style={{ marginTop: "2%" }}>

          <h2 style={{ textAlign: "center" }}>{this.props.question}</h2>

          <div>{this.state.solutions}</div>

          <form>

            <input

              type="text"

              name="solution"

              value={this.state.solution}

              style={{

                width: "75%",

                marginTop: "2%",

                marginRight: "10%",

                marginLeft: "10%",

                marginBottom: "2%",

                maxLength: "35",

              }}

              onChange={this.solutionhandler}

            />

            <br />

            <button

              className="btn btn-success"

              style={{ marginLeft: "40%", marginBottom: "2%" }}

              onClick={this.formaftersubmitting}

            >

              submit

            </button>

          </form>

        </body>

      </html>

    );

  }

}

export default ViewSolutions;

**BACKEND CODE**

**USER.JS FILE**

const express = require('express');

//const router = express.Router();

const userController = require("./../controllers/userController");

const controllers=require("./../controllers/appController")

const appConfig = require("./../config/appConfig")

const mid1=require("./../middlewares/auth")

var cors=require('cors');

var app=express()

let setRouter = (app) => {

    let baseUrl = `${appConfig.apiVersion}`;

    app.use(cors())

    app.post(`${baseUrl}/users/signup`,userController.signUpFunction);

    app.post(`${baseUrl}/users/login`,userController.loginFunction);

    app.post(`${baseUrl}/logout/:authToken`,mid1.isAuthorized, userController.logout);

    app.post(baseUrl+'/query/create/',controllers.createquery);

    app.post(baseUrl+'/solution/create/:queryId',controllers.createsolution)

    app.get(baseUrl+'/query/all/',controllers.getallqueries);

    app.get(baseUrl+'/query/all/mine/:userId',controllers.getuserqueries);

    app.get(baseUrl+'/solution/all/:queryId',controllers.getallsolutions);

    app.get(baseUrl+'/solution/:queryId/:solutionId',controllers.getsinglesolution);

    app.get(baseUrl+'/query/all/:tag',controllers.gettagqueries);

    app.put(baseUrl+'/solution/:queryId/:solutionId',controllers.editvotes);

}

module.exports={

    setRouter:setRouter

}

**Appconfig.js File**

let appConfig = {

  port: 3001,

  allowedCorsOrigin: "\*",

  env: "dev",

  apiVersion: "/api/v1",

  db: {

    uri: "mongodb://127.0.0.1:27017/DiscussionAppDB",

  },

};

module.exports = {

  port: appConfig.port,

  allowedCorsOrigin: appConfig.allowedCorsOrigin,

  env: appConfig.env,

  db: appConfig.db,

  apiVersion: appConfig.apiVersion,

};

**AppController.js File**

const mongoose=require("mongoose");

const query=require('../models/query');

const querySchema=mongoose.model('query');

const solutions=require('../models/solution');

const solutionSchema=mongoose.model('solutions');

const response=require('../libs/responseLib');

const express=require('express');

const shortid=require('shortid');

let getallqueries=(req,res)=>{

    console.log("came into get all queries at backend")

    querySchema.find().sort({"created":-1})

    .select('-\_\_v-\_id')

    .lean()

    .exec((err,result)=>{

        if(err)

        {

            console.log(err)

            res.send(err)

        }

        else if(result == undefined || result == null || result ==''){

            console.log("no queries found")

            res.send("no queries found")

        }

        else{

            let apiresponse=response.generate("false","all queries fetched successfully","200",res)

            res.send(result)

        }

    })

}

let getuserqueries=(req,res)=>{

    console.log("came into get all queries at backend")

    querySchema.find({'userId':req.params.userId})

    .select('-\_\_v-\_id')

    .lean()

    .exec((err,result)=>{

        if(err)

        {

            console.log(err)

            res.send(err)

        }

        else if(result == undefined || result == null || result ==''){

            console.log("no queries found")

            res.send("no queries found")

        }

        else{

            let apiresponse=response.generate("false","all queries fetched successfully","200",res)

            res.send(result)

        }

    })

}

let gettagqueries=(req,res)=>{

    console.log("came into get all queries at backend")

    querySchema.find({'tag':req.params.tag})

    .select('-\_\_v-\_id')

    .lean()

    .exec((err,result)=>{

        if(err)

        {

            console.log(err)

            res.send(err)

        }

        else if(result == undefined || result == null || result ==''){

            console.log("no queries found")

            res.send("no queries found")

        }

        else{

            let apiresponse=response.generate("false","all queries fetched successfully","200",res)

            res.send(result)

        }

    })

}

let getsinglesolution=(req,res)=>{

    console.log("came into get all solutions at backend")

    solutionSchema.find({'queryId':req.params.queryId,'solutionId':req.params.solutionId})

    .select('-\_\_v-\_id')

    .lean()

    .exec((err,result)=>{

        if(err)

        {

            console.log(err)

            res.send(err)

        }

        else if(result == undefined || result == null || result ==''){

            console.log("no queries found")

            res.send("no queries found")

        }

        else{

            let apiresponse=response.generate("false","all queries fetched successfully","200",res)

            console.log(result)

            res.send(result)

        }

    })

}

let getallsolutions=(req,res)=>{

    console.log("came into get all solutions at backend")

    solutionSchema.find({'queryId':req.params.queryId}).sort({"upvotes":-1})

    .select('-\_\_v-\_id')

    .lean()

    .exec((err,result)=>{

        if(err)

        {

            console.log(err)

            res.send(err)

        }

        else if(result == undefined || result == null || result ==''){

            console.log("no queries found")

            res.send("no queries found")

        }

        else{

            let apiresponse=response.generate("false","all queries fetched successfully","200",res)

            console.log(result)

            res.send(result)

        }

    })

}

let createquery=(req,res)=>{

    let todays=Date.now()

    let localdate=new Date(todays)

    let queryIds=shortid.generate()

    console.log("this is userid :" +req.body.userId);

    let newquery=new querySchema({

        userId:req.body.userId,

        username:req.body.username,

        queryId:queryIds,

        query:req.body.query,

        tag:req.body.tag,

        created:localdate

    })

    newquery.save((err,result)=>{

        if(err)

        {

            console.log(err)

            res.send(err)

        }

        else{

            res.send(result)

        }

    })

}

let createsolution=(req,res)=>{

    let today=Date.now()

    let solutionIds=shortid.generate()

    let newsolution=new solutionSchema({

       userId:req.body.userId,

       username:req.body.username,

       queryId:req.params.queryId,

       solutionId:solutionIds,

       solution:req.body.solution,

       upvotes:req.body.upvotes,

       created:today

    })

    newsolution.save((err,result)=>{

        if(err)

        {

            console.log(err)

            res.send(err)

        }

        else{

            console.log(result)

            res.send(result)

        }

    })

}

let editvotes=(req,res)=>{

    let options=req.body;

    console.log(options);

    solutionSchema.findOneAndUpdate({'queryId':req.params.queryId,'solutionId':req.params.solutionId},options,{multi:true}).exec((err,result)=>{

     if(err)

     {

         console.log(err)

         res.send(err)

     }

     else if(result== undefined||result==null||result==''){

         console.log("no list found")

         res.send("no list found")

     }

     else{

         console.log("result sent from editlist")

         res.send(result)

     }

    })

 }

module.exports={

    getallqueries:getallqueries,

    getallsolutions:getallsolutions,

    createquery:createquery,

    createsolution:createsolution,

    getsinglesolution:getsinglesolution,

    editvotes:editvotes,

    gettagqueries:gettagqueries,

    getuserqueries:getuserqueries

}

**UserContoller.js File**

const mongoose = require('mongoose');

var User=require('../models/user')

const UserModel = mongoose.model('User')

var Auth=require('../models/Auth')

const AuthModel=mongoose.model('Auth')

var cookies=require('cookies');

const shortid = require('shortid');

const time = require('./../libs/timeLib');

const response = require('./../libs/responseLib')

const logger = require('./../libs/loggerLib');

const validateInput = require('../libs/paramsValidationLib')

const check = require('../libs/checkLib')

const passwordLib = require('./../libs/generatePasswordLib');

const token = require('../libs/tokenLib')

/\* Models \*/

// start user signup function

let useridvalue;

let signUpFunction = (req, res) => {

    let validateUserInput=()=>{

        return new Promise((resolve, reject)=>{

            if(req.body.email)

            {

                if(!validateInput.Email(req.body.email))

                {

                    let apiResponse=response.generate(true,'Email Does not met the requirement',400,null)

                    reject(apiResponse)

                }

                else if(check.isEmpty(req.body.password)){

                    let apiResponse=response.generate(true,'"password" is missing')

                    reject(apiResponse)

                }

                else{

                    resolve(req)

                }

            }

            else{

                logger.error('Field Missing During User Creation','userController:createUser()',5)

                let apiResponse=response.generate(true,'one or More Parameter is missing',400,null)

                reject(apiResponse)

            }

        })

    }

    let createUser=()=>{

        return new Promise((resolve,reject)=>{

            UserModel.findOne({email:req.body.email})

            .exec((err,retrievedUserDetails)=>{

                if(err){

                    logger.error(err.message,'userController:createUser',10)

                    let apiResponse=response.generate(true,'Failed To Create User',400,null)

                     reject(apiResponse)

                }

                else if(check.isEmpty(retrievedUserDetails)){

                    console.log(req.body)

                    let newUser=new UserModel({

                     userId:shortid.generate(),

                     firstName:req.body.firstName,

                     lastName:req.body.lastName||'',

                     email:req.body.email.toLowerCase(),

                     mobileNumber:req.body.mobileNumber,

                     password:passwordLib.hashpassword(req.body.password),

                     apiKey:req.params.apiKey||req.query.apiKey||req.body.apiKey,

                     createdOn:time.now()

                    })

                    newUser.save((err,newUser)=>{

                        if(err){

                            console.log(err)

                            logger.error(err.message,"userController:CreateUser",10)

                            let apiResponse=response.generate(true,'Failed To Create User',400,null)

                            reject(apiResponse)

                        }

                        else{

                            let newUserObj=newUser.toObject();

                            resolve(newUserObj)

                        }

                    })

                }

                else{

                    logger.error('User Already Present',"userController:CreateUser",4);

                    let apiResponse=response.generate(true,'Failed To Create User',403,null)

                    reject(apiResponse)

                }

            })

        })

    }

    validateUserInput(req,res)

    .then(createUser)

    .then((resolve)=>{

        delete resolve.password

        res.header("Access-Control-Allow-Origin", "\*");

        let apiResponse=response.generate(false,"user created",200,resolve)

        res.send(apiResponse)

    })

    .catch((err)=>{

        console.log(err);

        res.send(err);

    })

}// end user signup function

// start of login function

let loginFunction = (req, res) => {

    let findUser=()=>{

        console.log("find user");

        return new Promise((resolve,reject)=>{

                 console.log("entered promise "+req.body.email +' '+req.body.password)

            if(req.body.email){

                console.log("entered if loop")

                UserModel.findOne({email:req.body.email},(err,userDetails)=>{

                   if(err)

                   {

                       console.log("finding usermodel")

                       console.log(err)

                       logger.error('Failed to retrieve User Data',"userController: findUser()",10)

                       let apiResponse =response.generate(true,"Failed To Find User Details",500,null)

                       reject(apiResponse)

                   }

                   else if(check.isEmpty(userDetails)){

                       console.log("entered empty")

                    logger.error('No User Found', 'userController: findUser()', 7)

                        let apiResponse = response.generate(true, 'No User Details Found', 404, null)

                        reject(apiResponse)

                   }

                   else{

                    console.log("entered resolve")

                    logger.info('User FOund','userController: findUser()',10)

                       resolve(userDetails)

                   }

                })

            }

            else {

                let apiResponse = response.generate(true, '"email" parameter is missing', 400, null)

                reject(apiResponse)

            }

        })

    }

    let validatePassword = (retrievedUserDetails) => {

        console.log("validatePassword");

        console.log(retrievedUserDetails);

        return new Promise((resolve, reject) => {

            passwordLib.comparePassword(req.body.password,retrievedUserDetails.password, (err, isMatch) => {

                if (err) {

                    console.log(err)

                    logger.error(err.message, 'userController: validatePassword()', 10)

                    let apiResponse = response.generate(true, 'Login Failed', 500, null)

                    reject(apiResponse)

                } else if (isMatch) {

                    var Cookies = new cookies(req, res)

                    Cookies.set('userid',retrievedUserDetails.userId);

                    let retrievedUserDetailsObj = retrievedUserDetails.toObject()

                    delete retrievedUserDetailsObj.password

                    delete retrievedUserDetailsObj.\_id

                    delete retrievedUserDetailsObj.\_\_v

                    delete retrievedUserDetailsObj.createdOn

                    delete retrievedUserDetailsObj.modifiedOn

                    resolve(retrievedUserDetailsObj)

                } else {

                    logger.info('Login Failed Due To Invalid Password', 'userController: validatePassword()', 10)

                    let apiResponse = response.generate(true, 'Wrong Password.Login Failed', 400, null)

                    reject(apiResponse)

                }

            })

        })

    }

    let generateToken=(userDetails)=>{

        console.log('generate token');

        return new Promise((resolve,reject)=>{

            token.generateToken(userDetails,(err,tokenDetails)=>{

                if(err){

                    console.log(err)

                    let apiResponse=response.generate(true,'failed to generate token',500,null)

                    reject(apiResponse)

                }

                else{

                   tokenDetails.userId=userDetails.userId

                   tokenDetails.userDetails=userDetails

                   resolve(tokenDetails)

                }

            })

        })

    }

let saveToken=(tokenDetails)=>{

     console.log("save token")

    return new Promise((resolve,reject)=>{

        AuthModel.findOne({userId:tokenDetails.userId},(err,retrievedTokenDetails)=>{

           if(err)

           {

               console.log(err.message,'userController:saveToken',10)

               let apiResponse=response.generate(true,'Failed To Generate Token',500,null)

               reject(apiResponse)

            }

            else if(check.isEmpty(retrievedTokenDetails)){

                let newAuthToken=new AuthModel({

                    userId:tokenDetails.userId,

                    authToken:tokenDetails.token,

                    tokenSecret:tokenDetails.tokenSecret,

                    tokenGenerationTime:time.now()

                })

                newAuthToken.save((err,newTokenDetails)=>{

                    if(err){

                        console.log(err)

                        logger.error(err.message,'userController:saveToken', 10)

                        let apiResponse = response.generate(true, 'Failed To Generate Token', 500, null)

                        reject(apiResponse)

                    } else {

                        console.log("this is "+newAuthToken.userId);

                        let responseBody = {

                            authToken: newTokenDetails.authToken,

                            userDetails: tokenDetails.userDetails

                        }

                        resolve(responseBody)

                    }

                })

            }

            else{

                retrievedTokenDetails.authToken=tokenDetails.token

                retrievedTokenDetails.tokenSecret=tokenDetails.tokenSecret

                retrievedTokenDetails.tokenGenerationTime=time.now()

                retrievedTokenDetails.save((err, newTokenDetails) => {

                    if (err) {

                        console.log(err)

                        logger.error(err.message, 'userController: saveToken', 10)

                        let apiResponse = response.generate(true, 'Failed To Generate Token', 500, null)

                        reject(apiResponse)

                    } else {

                        let responseBody = {

                            authToken: newTokenDetails.authToken,

                            userDetails: tokenDetails.userDetails

                        }

                        resolve(responseBody)

                    }

                })

            }

        })

    })

}

findUser(req,res)

.then(validatePassword)

.then(generateToken)

.then(saveToken)

.then((resolve)=>{

    console.log("hi");

    let apiResponse=response.generate(false,'Login Successfull',200,resolve)

    res.status(200)

    res.send(apiResponse)

})

.catch((err)=>{

    console.log("hello");

    console.log("error handler");

    console.log(err);

    res.status(err.status);

    res.send(err)

})

}

// end of the login function

let logout = (req, res) => {

console.log("entered logout " )

    AuthModel.findOneAndRemove({userId:req.user.userId},(err,result)=>{

        if(err){

            console.log(err)

            let apiResponse = response.generate(true, `error occurred: ${err.message}`, 500, null)

            res.send(apiResponse)

        } else if (check.isEmpty(result)) {

            let apiResponse = response.generate(true, 'Already Logged Out or Invalid UserId', 404, null)

            res.send(apiResponse)

        } else {

            let apiResponse = response.generate(false, 'Logged Out Successfully', 200, null)

            res.send(apiResponse)

        }

      })

} // end of the logout function.

module.exports = {

    signUpFunction: signUpFunction,

    loginFunction: loginFunction,

    logout: logout

}// end exports

**Auth.js File**

const mongoose = require('mongoose')

const jwt = require('jsonwebtoken')

const request = require("request")

const Auth = mongoose.model('Auth')

const logger = require('./../libs/loggerLib')

const responseLib = require('./../libs/responseLib')

const token = require('./../libs/tokenLib')

const check = require('./../libs/checkLib')

let isAuthorized = (req, res, next) => {

    console.log("In is authorized method");

  if (req.params.authToken || req.query.authToken || req.body.authToken || req.header('authToken')) {

    Auth.findOne({authToken: req.header('authToken') || req.params.authToken || req.body.authToken || req.query.authToken}, (err, authDetails) => {

      if (err) {

        console.log("this is an error:"+err)

        logger.error(err.message, 'AuthorizationMiddleware', 10)

        let apiResponse = responseLib.generate(true, 'Failed To Authorized', 500, null)

        res.send(apiResponse)

      } else if (check.isEmpty(authDetails))

      {

        console.log("no authorization key")

        logger.error('No AuthorizationKey ', 'AuthorizationMiddleware', 10)

        let apiResponse = responseLib.generate(true, 'Invalid Or Expired AuthorizationKey', 404, null)

        res.send(apiResponse)

      } else {

        token.verifyToken(authDetails.authToken,"someveryrandomstringthatnobodycanguess",(err,decoded)=>{

            if(err){

              console.log("this is secret "+authDetails.tokenSecret)

                logger.error(err.message, 'Authorization Middleware', 10)

                let apiResponse = responseLib.generate(true, 'Failed To Authorized', 500, null)

                res.send(apiResponse)

            }

            else{

                req.user = {userId: decoded.data.userId}

                next()

            }

        });// end verify token

      }

    })

  } else {

    logger.error('AuthorizationToken Missing', 'AuthorizationMiddleware', 5)

    let apiResponse = responseLib.generate(true, 'AuthorizationToken Is Missing In Request', 400, null)

    res.send(apiResponse)

  }

}

module.exports = {

  isAuthorized: isAuthorized

}

**AppErrorHandler.js File**

const response = require('./../libs/responseLib')

let errorHandler = (err,req, res, next) => {

    console.log("application error handler called");

    console.log(err);

    let apiResponse = response.generate(true, 'Some error occured at global level',500, null)

    res.send(apiResponse)

}// end request ip logger function

let notFoundHandler = (req,res,next)=>{

    console.log("Global not found handler called");

    let apiResponse = response.generate(true, 'Route not found in the application',404, null)

    res.status(404).send(apiResponse)

}// end not found handler

module.exports = {

    globalErrorHandler : errorHandler,

    globalNotFoundHandler : notFoundHandler

}

**RouteLogger.js File**

const appConfig = require('./../../config/appConfig');

let requestIpLogger = (req, res, next) => {

    let remoteIp = req.connection.remoteAddress + '://' + req.connection.remotePort;

    let realIp = req.headers['X-REAL-IP'];

    console.log(req.method+" Request Made from " + remoteIp + ' for route' + req.originalUrl);

    if (req.method === 'OPTIONS') {

        console.log('!OPTIONS');

        var headers = {};

        // IE8 does not allow domains to be specified, just the \*

        // headers["Access-Control-Allow-Origin"] = req.headers.origin;

        headers["Access-Control-Allow-Origin"] = "\*";

        headers["Access-Control-Allow-Methods"] = "POST, GET, PUT, DELETE, OPTIONS";

        headers["Access-Control-Allow-Credentials"] = false;

        headers["Access-Control-Max-Age"] = '86400'; // 24 hours

        headers["Access-Control-Allow-Headers"] = "X-Requested-With, X-HTTP-Method-Override, Content-Type, Accept";

        res.writeHead(200, headers);

        res.end();

  }

  else{

     // enable or disable cors here

     res.header("Access-Control-Allow-Origin", appConfig.allowedCorsOrigin);

     res.header('Access-Control-Allow-Methods', 'GET, PUT, POST, DELETE, OPTIONS');

     res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");

     //console.log(res.header)

     // end cors config

     next();

  }

}// end request ip logger function

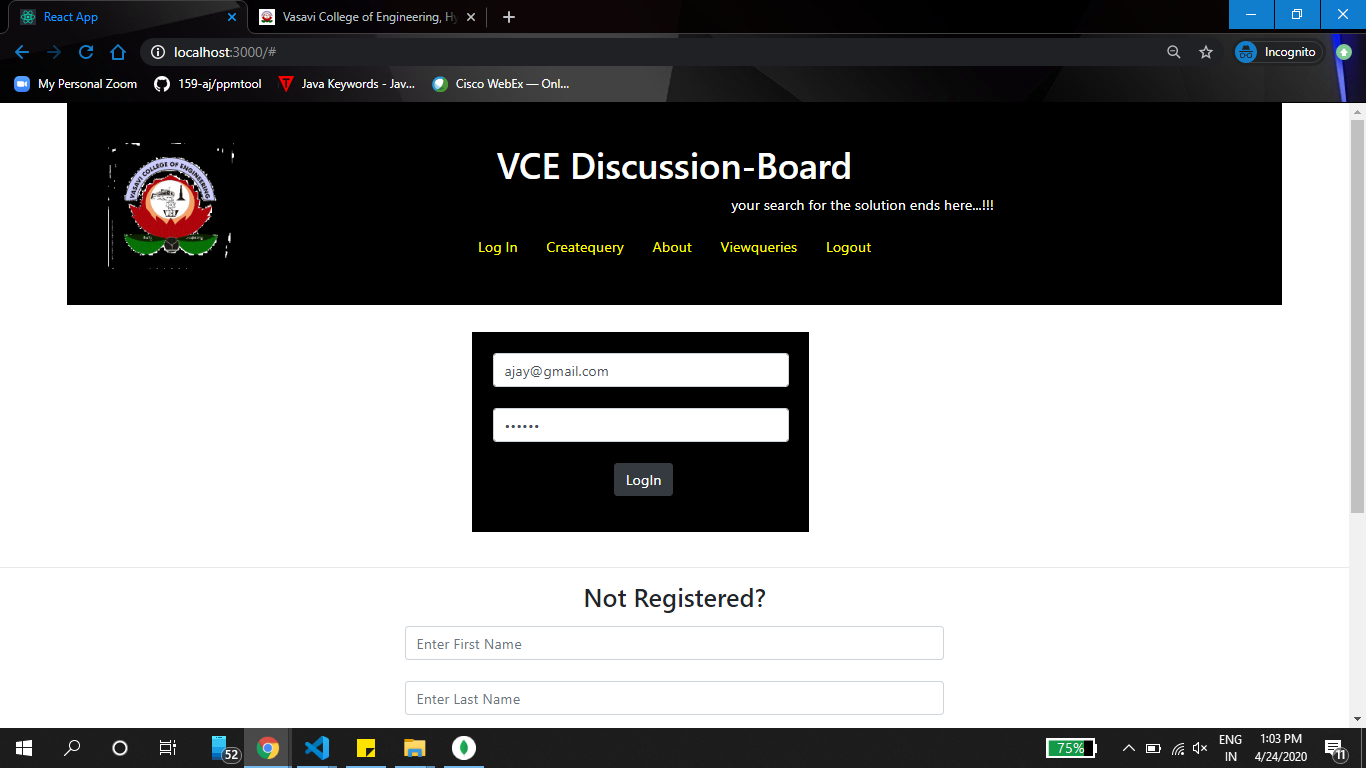
module.exports = {

    logIp: requestIpLogger

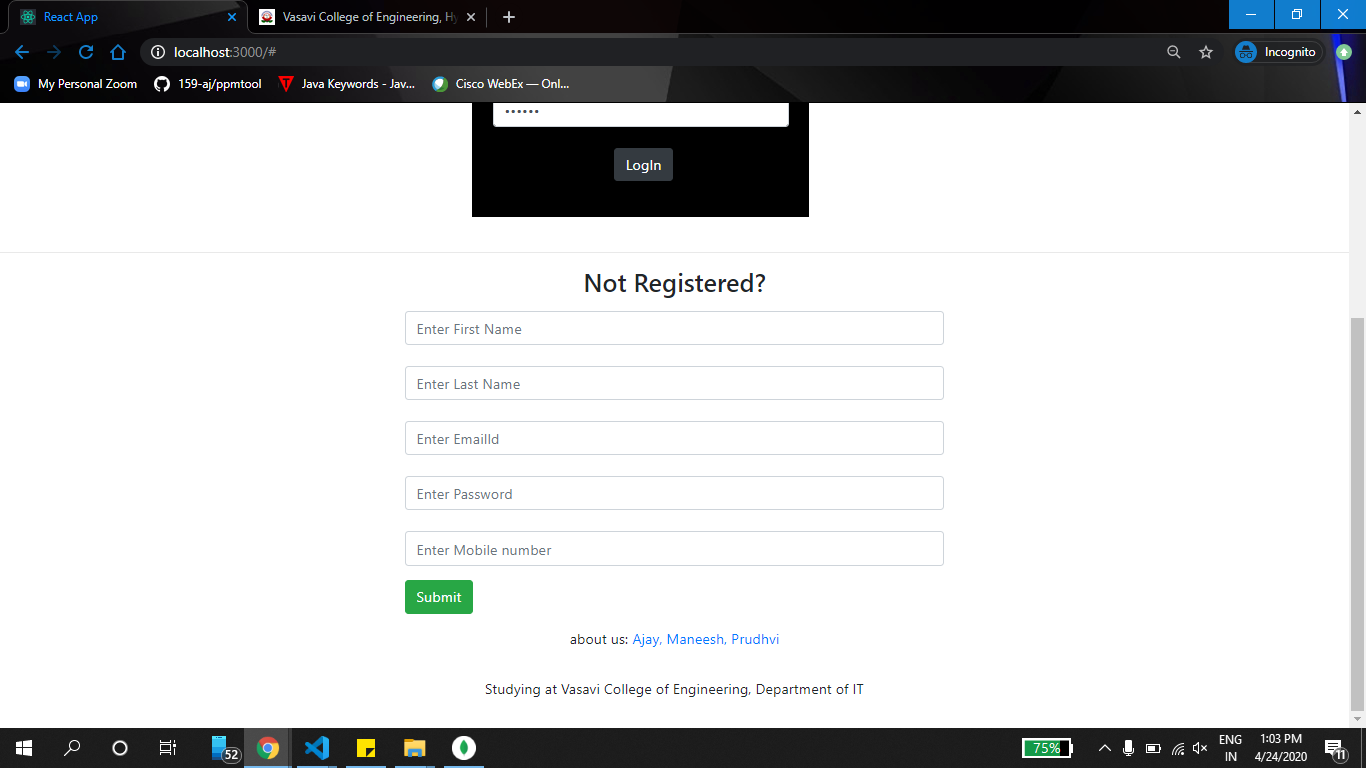
}

**6.RESULTS**

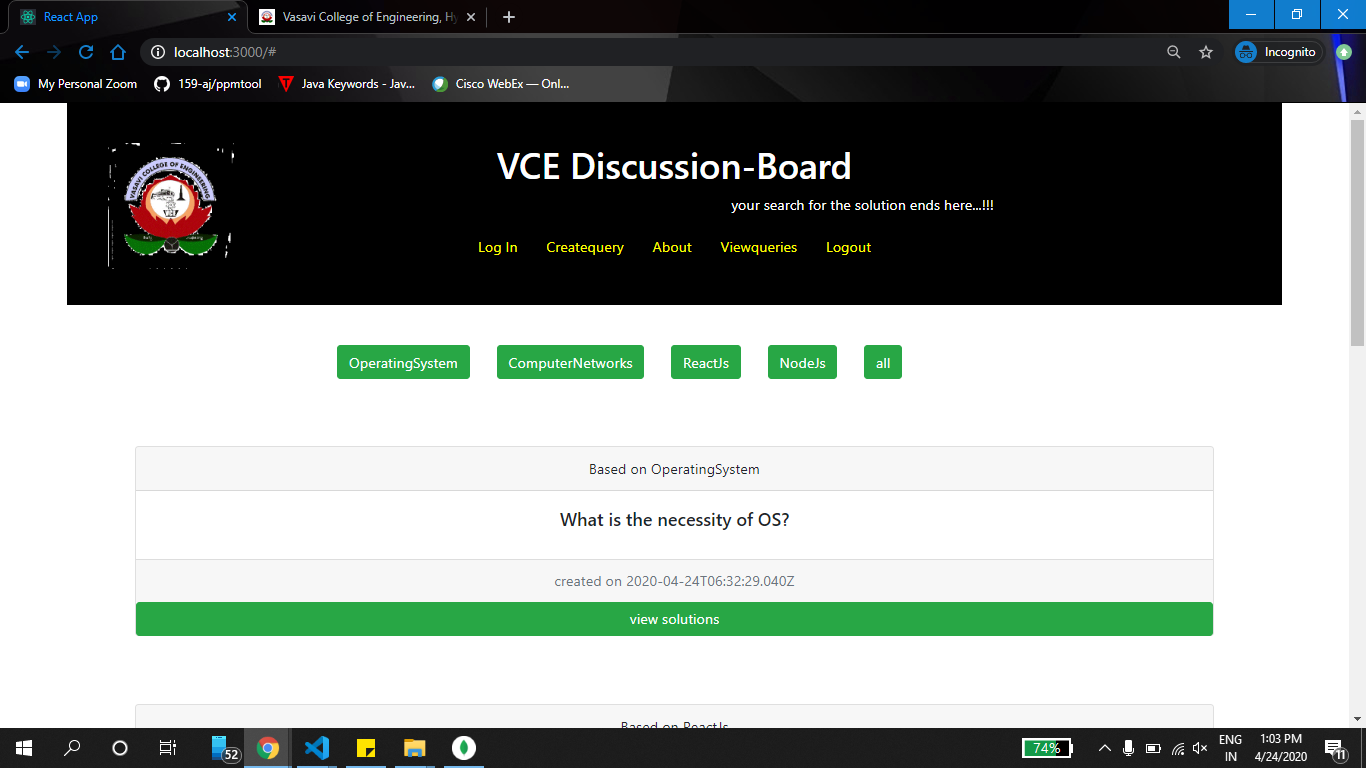
**Login page**

****

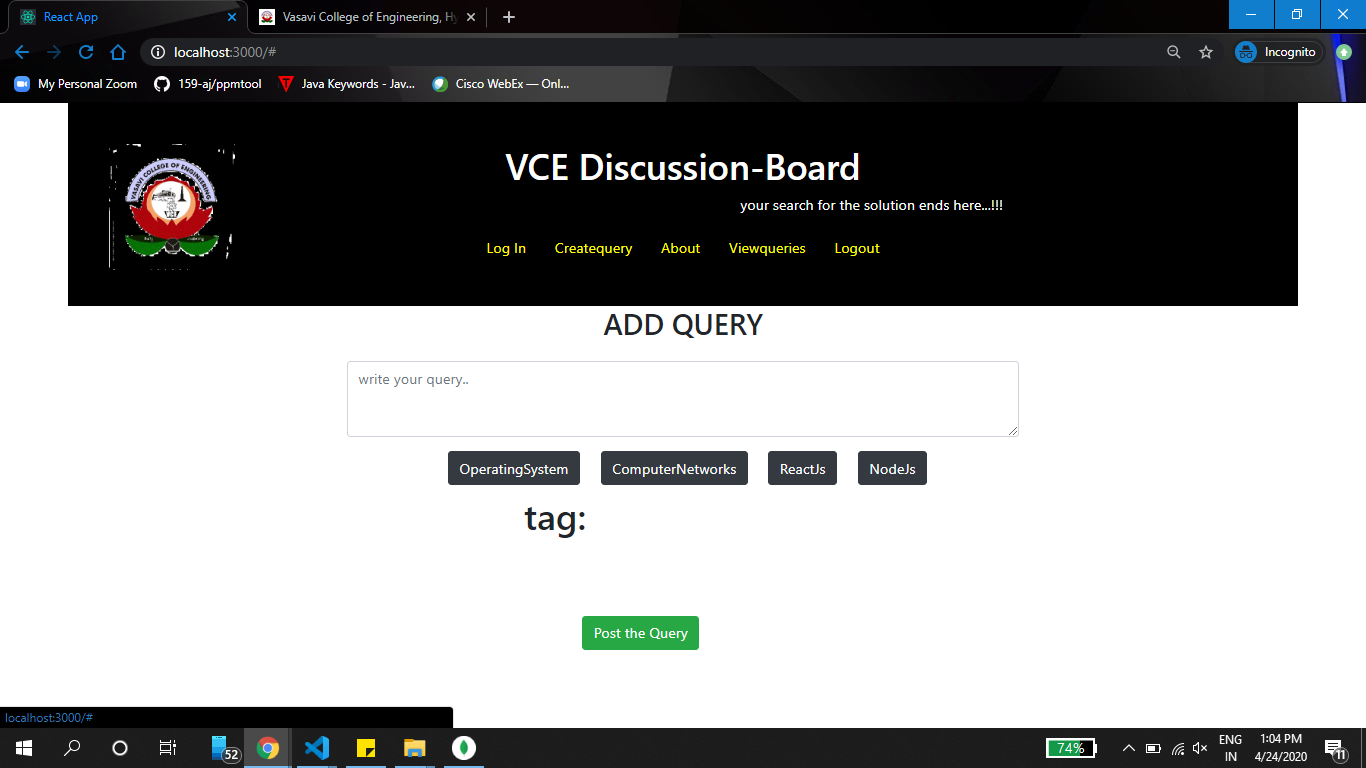
**Registration page**

****

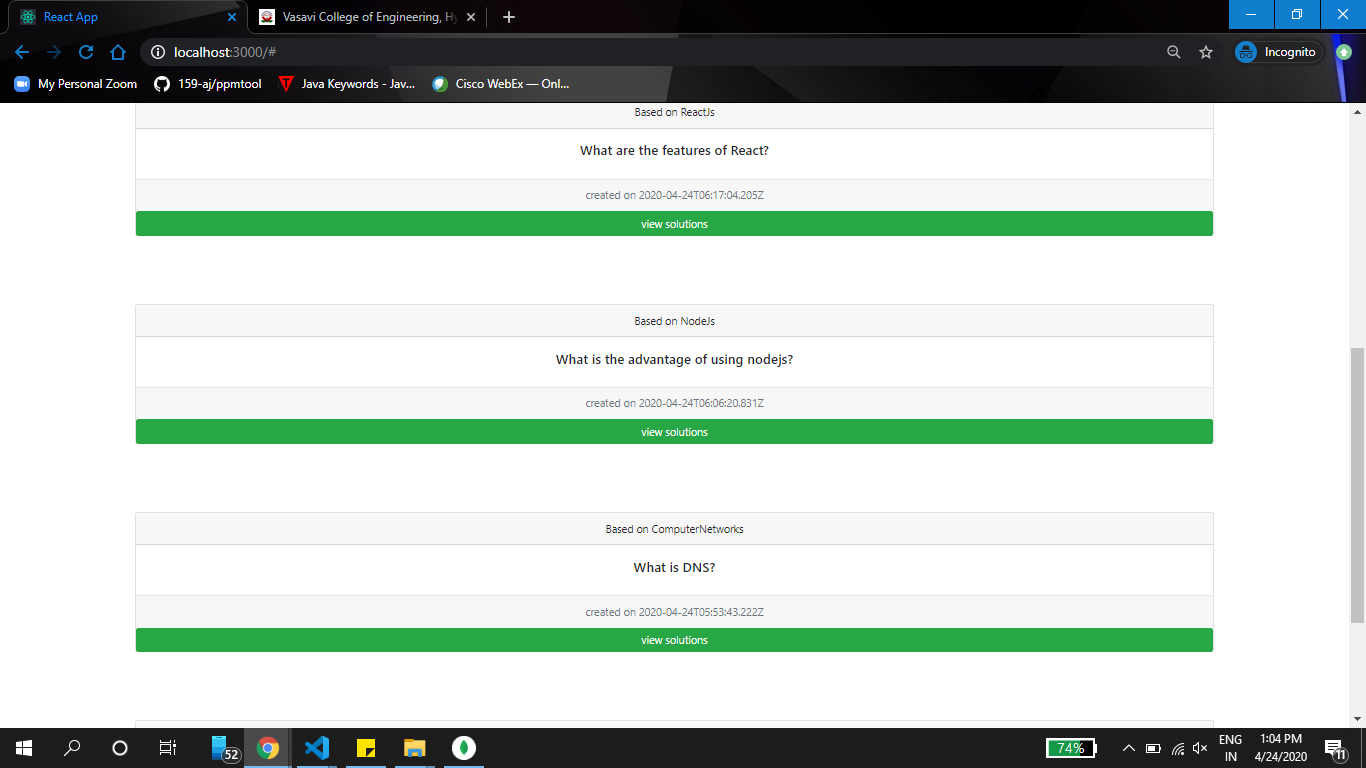
**Home page**

****

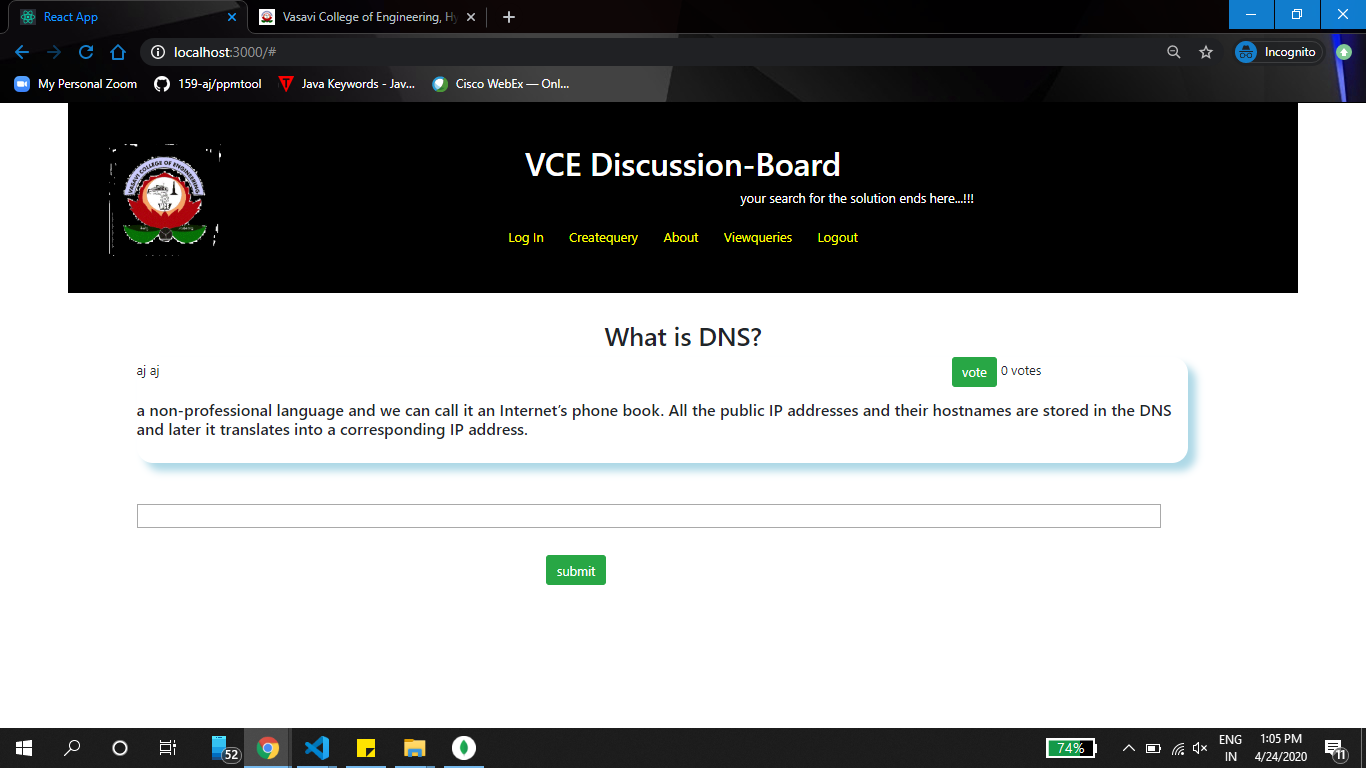
**Create query page**

****

**View query page**

****

**View solution page**

****

**6.TESTING**

Validations are performed on passwords ,emails. Email is validated by checking for ‘@gmail.com’ at the end of user input .Passwords are validated using password hashing algorithm

**7.CONCLUSION AND FUTURE SCOPE**

**CONCLUSION**

I think the online discussion forum is great tool to help both teachers and students maintain their communication beyond the classrooms. It’s also a great way for students to feel more engaged in participating the group discussion; allow them to provide and share their opinions with others at any time and location.

Although some critics feel that using the online discussion forum might have the negative impact on student’s communicate skills and even increase the potential of getting the online bullies. I think there’s always pro and cons on any of the technology inventions and it really depend on how people utilize it. I personally think the online discussion forum is great technology that helps bring the students and teacher more closer and it's a great way to share and transfer knowledge to each other.

**FUTURE SCOPE**

• As per now this project is only restricted to college we can develop it further globally.

• We are trying to add reference links which will redirect to google search engine for more clarification on the topic.

• We are trying to develop our project by adding some modules in such a way that we can get any information about our college also.

**9.REFERENCES**

* <https://www.udemy.com/>
* <https://reactjs.org/docs/getting-started.html>
* <https://nodejs.org/en/docs/>
* <https://www.w3schools.com/>