Date: 07-05-2024

UCS2611 – Internet Programming Lab
Mini Project

A. Karthikeyan	3122 21 5001 041
M Irayanbu	3122 21 5001 036
Anand	3122 21 5001 009

Mini project Description

Develop a full stack web application for conducting *On-line quiz using MVC architecture*. The application should facilitate the normal and admin users to access it. To the normal user, instructions, questions with options and the score needs to be provided as the following snapshots. Admin user can view the registered users and their scores.

Operations:

- Authenticate the user and provide necessary messages
- Keep a timer for the Quiz
- Sending appropriate GET http requests from front end, to the endpoint in the node server.
- Perform necessary operations in the Mongo Collection at the endpoints
- Create a suitable interface in ReactJS to display the results

Tech-Stack:
- ReactJS
- NodeJS
- Express Server
- MongoDB

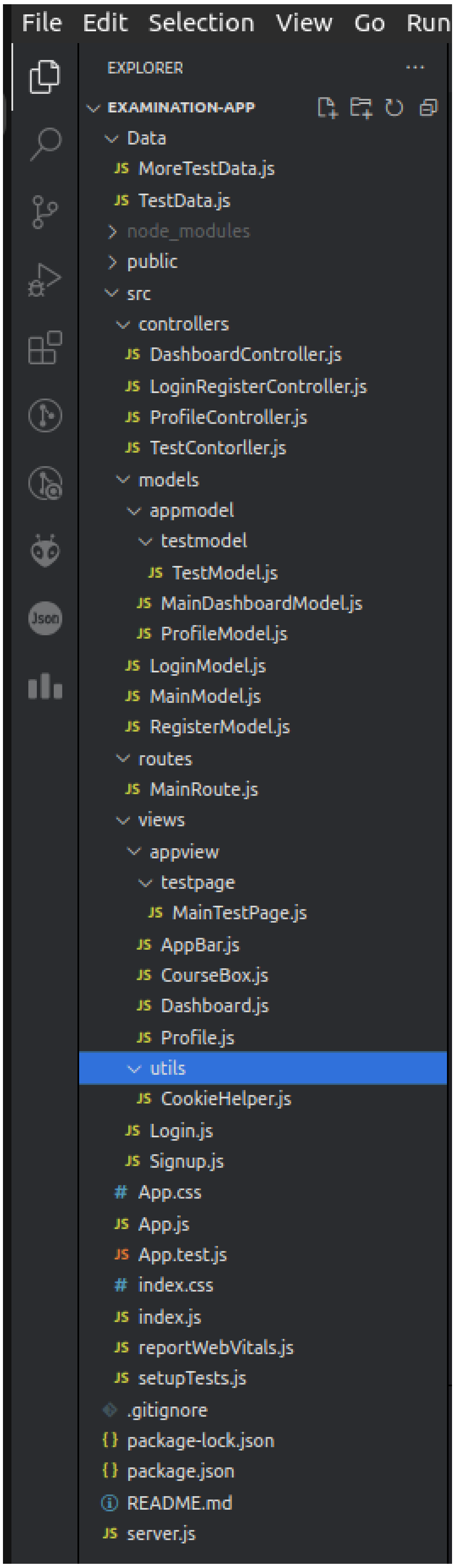
GitHub Link:

https://github.com/KKBUGHUNTER/Internet-Programming-Lab/tree/main/Project/examination-app/

Run the Project: DIR: home

```
npm install
npm start

node server.js
```

Project Structure:



```
EXAMINATION-APP
  Data
    MoreTestData.js
    TestData.js
  node_modules
  public
  src
    controllers
      DashboardController.js
      LoginRegisterController.js
      ProfileController.js
      TestContorller.js
    models
      appmodel
        testmodel
          TestModel.js
        MainDashboardModel.js
        ProfileModel.js
      LoginModel.js
      MainModel.js
      RegisterModel.js
    routes
      MainRoute.js
    views
      appview
        testpage
          MainTestPage.js
        AppBar.js
        CourseBox.js
        Dashboard.js
        Profile.js
      utils
        CookieHelper.js
      Login.js
      Signup.js
    App.css
    App.js
    App.test.js
    index.css
    index.js
    reportWebVitals.js
    setupTests.js
  .gitignore
  package-lock.json
  package.json
  README.md
  server.js
```

Front-end Code:
index.js
```
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter } from 'react-router-dom';
import App from './App';
import reportWebVitals from './reportWebVitals';

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
);

reportWebVitals();
```

App.js
```
// App.js
import React, { useEffect, useState } from 'react';
import { Routes, Route, Navigate, useLocation } from 'react-router-dom';

import Login from "./views/Login";
import Signup from "./views/Signup";
import Dashboard from "./views/appview/Dashboard";
import Profile from "./views/appview/Profile";
import MainTestPage from "./views/appview/testpage/MainTestPage.js";
import Cookies from 'js-cookie';

function App() {
  const [redirected, setRedirected] = useState(false);
  const storedToken = Cookies.get('token');
  const location = useLocation();



  useEffect(() => {
    if (!redirected) {
      if (location.pathname === '/' || location.pathname === '/signup') {
        if (storedToken) {
          setRedirected(true);
          window.location.href = '/dashboard';
        }
      } else if (location.pathname === '/dashboard') {
        if (!storedToken) {
          setRedirected(true);
          window.location.href = '/';
        }
      }
    }
  }, [storedToken, redirected, location]);

  return (
    <Routes>
      {/* Stage 1 */}
      <Route path="/" element={<Login />} />
      <Route path="signup" element={<Signup />} />

      {/* Stage 2 */}
      <Route path="/dashboard" element={<Dashboard />} />
      <Route path="/profile" element={<Profile />} />


      {/* Stage 3 */}
      <Route path="/test" element={<MainTestPage />} />

      <Route path="*" element={<Navigate to="/" />} />
    </Routes>
  );
}

export default App;
```

utils/CookieHelper.js

```
import Cookies from 'js-cookie';

class CookieHelper {
    static getFullCookie() {
        return Cookies.get('token');
    }

    static getUserId() {
        const token = CookieHelper.getFullCookie();
        return token.substring(0, 13);
    }

    static removeTokenCookie() {
        Cookies.remove('token');
    }
}

export default CookieHelper;
```

Views/appviews/AppBar.js

```
import Cookies from 'js-cookie';

function AppBar({ parse }) {
    const Logout = () =>{
        Cookies.remove('token');
        window.location.href = '/';
    }

    const style = {
        appBar: {
            display: "flex",
            justifyContent: "space-between",
            alignItems: "center",
            height: "40px",
            width: "100%",
            backgroundColor: "#1F85DE",
            color: "white",
            padding: "5px",
        },
        profileImage: {
            marginRight: "10px",
        },
        btn:{
            position:"absolute",
            right:"70px",
            padding:"4px",
            backgroundColor:"red",
            border:"0",
            color:"white",
            fontWeight:"bold",
            borderRadius:"5px"
        }
    };

    return (
        <div style={style.appBar}>
            <b>Hello <i>{parse.userId}</i>,</b>
            <button type="submit" onClick={Logout}
style={style.btn}>LogOut</button>
            <a href="http://localhost:3000/profile">
                <img
src="https://cdn-icons-png.flaticon.com/512/3135/3135715.png"
alt="profile img" height="40px" style={style.profileImage} />
            </a>
        </div>
    );
}

export default AppBar;
```

CourseBox.js

```
import React from 'react';

function CourseBox({ data, onTakeTest }) {
   const style = {
      master: {
         margin: "10px",
         padding: "10px",
         border: "0",
         width: "400px",
         border:'3px solid #DE1F85',
         borderRadius: "10px",
      },
      btn:{
         padding:"5px",
         width:"100px",
         border:"0",
         borderRadius:'5px',
         backgroundColor:"#85DE1F",
         float:'right'
      }

   };

   const handleTakeTest = () => {
      onTakeTest(data.testName);
   };

   return (
      <div style={style.master}>
         <h2 style={{color:"darkgreen"}}>{data.testName}</h2>
         <small>{data.testDescription}</small> <br/> <br/>
         <span><b>Score: <i>{data.score}</i></b></span> <br />
         <button onClick={handleTakeTest} style={style.btn}> <big><b
>Take Test</b></big></button>
         <br />
         <br />
      </div>
   );
}

export default CourseBox;
```

Dashboard.js

```
import React, { useEffect, useState } from 'react';
import AppBar from './AppBar';
import CourseBox from "./CourseBox";
import CookieHelper from '../utils/CookieHelper';
```

```
function Dashboard() {
  const [userDetails, setUserDetails] = useState();
  const [testData, setTestData] = useState();

  useEffect(() => {
    const userId = CookieHelper.getUserId();
    if (userId) {
      LoadUserData(userId);
      LoadTestData();
    }
    else{
      window.location.href = '/';
    }
  }, []);

  function LoadTestData(){
    fetch(`http://localhost:7020/test`)
        .then(res => res.json())
        .then(data => {
            setTestData(data);
        })
        .catch(error => console.log(error))
  }

  function LoadUserData(regNo){
    fetch(`http://localhost:7020/userDetails?userId=${regNo}`)
        .then(res => res.json())
        .then(data => {
            setUserDetails(data.username);
        })
        .catch(error => console.log(error))
  }

  const style = {
    master: {
      margin: "0",
      padding: "0",
      border: "0",
      display: "flex",
      flexWrap: "wrap",
    },
  };

  const handleTakeTest = (courseName) => {
    console.log(`Clicked on ${courseName}`);
    window.location.href = `/test?Subject=${courseName}`;
  };
  return (
    <div>
      <AppBar parse={{userId:userDetails}} />
      <h1>Take Your New Test <img src="https://good-samaritan-
school.com/wp-content/uploads/2021/05/new_blink_gif-1.gif" alt="user
profile" height="25px" /> </h1>
      <br/>
      <div style={style.master}>
      {testData && testData.map((test, index) => (
        <CourseBox
          key={index}
          data={{
            testName: test.testName,
            testDescription: test.testDesc,
            score: test.totalNoOfQues
          }}
          onTakeTest={handleTakeTest}
        />
      ))}
      </div>
      {/* {questionSet && <MainTestPage questionSet={questionSet} />}
*/}
    </div>
  );
}

export default Dashboard;
```

Profile.js

```
// /src/views/appviews/Profile.js

import React from 'react';
import {useState, useEffect} from 'react';
import CookieHelper from '../utils/CookieHelper';

function Profile() {
    const [name, setName] = useState();
    const [regNo, setRegNo] = useState();
    const [score, setScore] = useState();
    const [coursesCompleted, setCoursesCompleted] = useState();
    const [courseNames, setCourseNames] = useState([]);

    useEffect(() => {
        const userId = CookieHelper.getUserId();
        setRegNo(userId);
        // console.log("data in useEffect: ", userId);
    }, []);

    useEffect(() => {
```

```
    if (regNo) {
        // console.log("load data", regNo);
        LoadData();
    }
}, [regNo]);


const LoadData = () => {
    console.log("load data", regNo);
    fetch(`http://localhost:7020/userDetails?userId=${regNo}`)
    .then(res => res.json())
    .then(data => {
        setName(data.username);
        setScore(data.score);
        setCoursesCompleted(data.courseCount);
        setCourseNames(data.courseList);
    })
    .catch(error => console.log(error))
}

const DeleteAccount = () =>{
    fetch(`http://localhost:7020/deleteUser?userId=${regNo}`, {
        method: 'DELETE'
    })
    .then(res => res.json())
    .then(data => {
        if (data.message === "DeleteSuccess") {
            // Redirect to the homepage if the deletion was successful
            CookieHelper.removeTokenCookie();
            window.location.href = '/signup';
        }
    })
    .catch(error => console.log(error));
}

const style = {
    container: {
        margin:"auto",
        padding: "30px",
        maxWidth: "600px",
        boxShadow: "0px 0px 20px rgba(0, 0, 0, 0.1)",
        borderRadius: "10px",
        backgroundColor: "#f9f9f9",
    },
    header: {
        textAlign: "center",
        marginBottom: "20px",
        fontSize: "24px",
        fontWeight: "bold",
        color: "#333",
        textTransform: "uppercase",
    },
    detailItem: {
        marginBottom: "10px",
        fontSize: "16px",
        color: "#666",
    },
    courseList: {
        listStyleType: "none",
        padding: 0,
        marginTop: "10px",
    },
    courseItem: {
        marginBottom: "5px",
        paddingLeft: "20px",
        position: "relative",
        fontSize: "14px",
        color: "#444",
    },
    courseBullet: {
        position: "absolute",
        left: "0",
        top: "50%",
        transform: "translateY(-50%)",
        width: "10px",
        height: "10px",
        borderRadius: "50%",
        backgroundColor: "#007bff",
    },
    delBtn:{
        backgroundColor:"red",
        border:"0",
        borderRadius:"8px",
        padding:"8px",
        fontWeight:"bold",
        fontSize:"14px",
        color:"white",
        marginLeft:"40%"
    }
};

return (
    <div style={style.container}>
        {/* <AppBar parse={{userId:regNo}}/> */}
        <div>
            <h2 style={style.header}>Profile</h2>
```

```
                <img
src="https://cdn-icons-png.flaticon.com/512/3135/3135715.png"
alt="profile image" height="140px" />
                <br/>
                <br/>
                <div style={style.detailItem}>
                    <strong>Name:</strong> {name}
                </div>
                <div style={style.detailItem}>
                    <strong>Registration Number:</strong> {regNo}
                </div>
                <div style={style.detailItem}>
                    <strong>Score:</strong> {score}
                </div>
                <div style={style.detailItem}>
                    <strong>Courses Completed:</strong> {coursesCompleted}
                </div>
                {courseNames && (
                    <div style={style.detailItem}>
                        <strong>Course Lists:</strong>
                        <ul style={style.courseList}>
                            {courseNames.map((course, index) => (
                                <li key={index} style={style.courseItem}>
                                    <span style={style.courseBullet}></span>
                                    {course}
                                </li>
                            ))}
                        </ul>
                    </div>
                )}
            </div>
             <button type="submit" onClick={DeleteAccount}
style={style.delBtn}>Delete Account</button>
        </div>
    );
}

export default Profile;
```

src/views/appview/testpage/MainTestPage.js

```
import React, { useState, useEffect } from 'react';
import CookieHelper from '../../utils/CookieHelper';

const MainTestPage = () => {
    const [userDetails, setUserDetails] = useState();
    const [regno, setRegNo] = useState();
    const [questionSet, setQuestionSet] = useState({});
    const [selectedOptions, setSelectedOptions] = useState({});
    const [score, setScore] = useState(null);
    const [prevScore, setPrevScore] = useState();
    const Subject = new
URLSearchParams(window.location.search).get('Subject');
    const [timer, setTimer] = useState(10 * 60); // 10 minutes timer

    useEffect(() => {
        const userId = CookieHelper.getUserId();
        setRegNo(userId);
        if (userId) {
            LoadUserData(userId);
            fetch(`http://localhost:7020/taketest?testName=${Subject}`)
                .then(res => res.json())
                .then(data => {
                    console.log(data);
                    setQuestionSet(data.questionSet || {});
                    const initialSelectedOptions = {};
                    data.questionSet?.Question?.forEach((question, index) => {
                        initialSelectedOptions[index] = '';
                    });
                    setSelectedOptions(initialSelectedOptions);
                })
                .catch(error => console.log(error));

            const countdown = setInterval(() => {
                setTimer(prevTimer => prevTimer - 1);
                if (timer <= 0) {
                    clearInterval(countdown);
                    handleSubmit();
                }
            }, 1000);

            return () => clearInterval(countdown);
        } else {
            window.location.href = '/';
        }
    }, [Subject]);

    const LoadUserData = (regNo) => {
        fetch(`http://localhost:7020/userDetails?userId=${regNo}`)
            .then(res => res.json())
            .then(data => {
                setUserDetails(data.username);
                setPrevScore(data.score);
            })
            .catch(error => console.log(error));
    };

    const handleOptionSelect = (index, option) => {
        setSelectedOptions(prevState => ({
            ...prevState,
            [index]: option,
        }));
    };

    const handleSubmit = () => {
        let correctAnswers = 0;
        const submittedOptions = {};

        questionSet?.Question?.forEach((question, index) => {
            const selectedOption = selectedOptions[index];
            submittedOptions[index] = selectedOption ? selectedOption : '';
            if (selectedOption === question.ans) {
                correctAnswers++;
            }
        });

        setScore(correctAnswers);
        UpdateUserScoreData(regno, correctAnswers, Subject);
    };

    const UpdateUserScoreData = (regno, score, Subject) => {
        fetch(`http://localhost:7020/updateUserData?userId=$
{regno}&score=${score}&testName=${Subject}&prevScore=$
{prevScore}`)
            .then(res => res.json())
            .then(data => {
                setTimeout(() => {
                    window.location.href = '/profile';
                }, 5000);
            })
            .catch(error => console.log(error));
    };

    return (
        <div style={{ margin: '20px auto', maxWidth: '800px', fontFamily:
'Arial, sans-serif', color: '#333', background: '#2E8B57', padding: '20px',
borderRadius: '10px' }}>
            <h1 style={{ fontSize: '24px', textAlign: 'center', marginBottom:
'20px', color: '#fff' }}>Time Remaining: {Math.floor(timer / 60)}:{timer %
60 < 10 ? '0' : ''}{timer % 60}</h1>
            <h2 style={{ fontSize: '20px', textAlign: 'center', marginBottom:
'20px', color: '#fff' }}>{Subject}</h2>
            {questionSet && questionSet.Question && (
                <div>
                    <h3 style={{ fontSize: '18px', marginBottom: '10px', color:
'#fff' }}>{questionSet.testName}</h3>
                    <p style={{ fontSize: '16px', marginBottom: '20px', color:
'#fff' }}>{questionSet.testDescription}</p>
```

```
          {questionSet.Question.map((question, index) => (
            <div key={index} style={{ marginBottom: '20px',
borderBottom: '1px solid #ccc', paddingBottom: '20px' }}>
              <h3 style={{ fontSize: '16px', marginBottom: '10px',
color: '#fff' }}>{question.Q}</h3>
              <ul style={{ listStyleType: 'none', padding: 0 }}>
                {['opt1', 'opt2', 'opt3', 'opt4'].map((optionKey,
optionIndex) => (
                  <li key={optionIndex} style={{ marginBottom:
'10px' }}>
                    <label style={{ display: 'flex', alignItems:
'center', fontSize: '14px', color: '#fff' }}>
                      <input
                        type="radio"
                        name={`question-${index}`}
                        value={question[optionKey]}
                        checked={selectedOptions[index] ===
question[optionKey]}
                        onChange={() =>
handleOptionSelect(index, question[optionKey])}
                        style={{ marginRight: '10px', cursor:
'pointer' }}
                      />
                      {question[optionKey]}
                    </label>
                  </li>
                ))}
              </ul>
            </div>
          ))}
          <button onClick={handleSubmit}
style={{ backgroundColor: '#4CAF50', color: 'white', padding: '10px
20px', borderRadius: '5px', border: 'none', cursor: 'pointer', fontSize:
'16px' }}>Submit</button>
        </div>
      )}
      {score !== null && (
        <div style={{ position: 'fixed', top: '0', left: '0', width: '100%',
height: '100%', backgroundColor: 'rgba(0, 0, 0, 0.5)', zIndex: '9999',
display: 'flex', justifyContent: 'center', alignItems: 'center' }}>
          <div style={{ background: '#fff', padding: '20px',
borderRadius: '10px', textAlign: 'center' }}>
            <h2 style={{ color: '#333', marginBottom:
'20px' }}>Your Score</h2>
            <h1 style={{ color: '#007bff', fontSize: '48px',
marginBottom: '20px' }}>{score}/{questionSet.Question.length}</h1>
          </div>
        </div>
      )}
```

```
        </div>
    );
}

export default MainTestPage;
```

Server side Code:
server.js

```
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');
const mainRoute = require('./src/routes/MainRoute');
const path = require('path');
const cookieParser = require('cookie-parser');

const app = express();
const PORT = 7020;

app.use(bodyParser.json());
app.use(cors());
app.use(cookieParser());

// Serve static files from the React build directory
// app.use(express.static(path.join(__dirname, 'build')));

// API routes
app.use('/', mainRoute);

// Serve index.html for all other routes
// app.get('*', (req, res) => {
//     res.sendFile(path.join(__dirname, 'build', 'index.html'));
// });


app.listen(PORT, () => {
  console.log(`SERVER Listening http://localhost:${PORT}`);
});
```

/src/routes/MainRoute.js

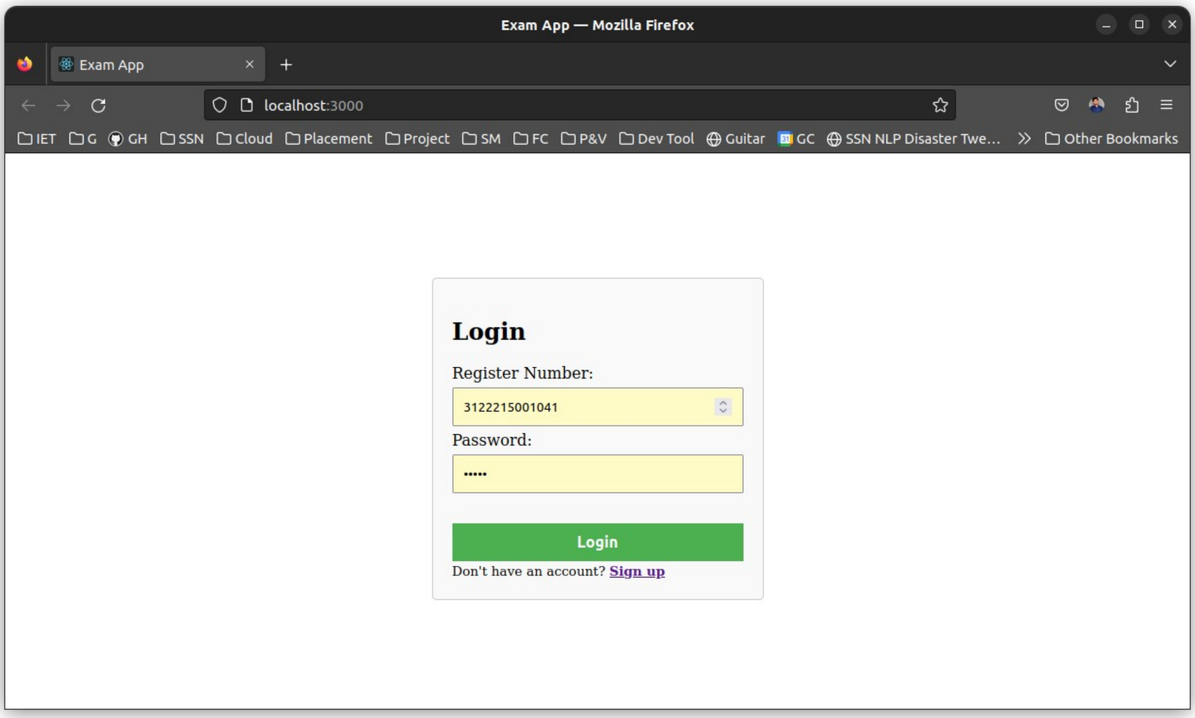
```
const express = require('express');
const router = express.Router();
const LoginRegisterController =
require('../controllers/LoginRegisterController.js');
const dashboardController =
require("../controllers/DashboardController.js");
const profileController = require("../controllers/ProfileController.js");
const testController = require("../controllers/TestContorller.js");
```

```
// stage 1
router.post('/login', LoginRegisterController.login);
router.post('/register', LoginRegisterController.register);

// stage 2
router.get('/userDetails', profileController.getUserDetails);
router.delete('/deleteUser', profileController.deleteAccount);
router.get('/test', dashboardController.getTestList);


// stage 3
router.get("/taketest", testController.testData);
router.get('/updateUserData', profileController.updateUserScore);


module.exports = router;

// /src/modules/RegisterModel.js

const MainModel = require('./MainModel.js');

async function RegisterUser(username, registerNumber, password){
    const mainModule = new MainModel();
    await mainModule.connect();
    const userCollection = await mainModule.getUserCollection();

    try {
        // Insert the user with the given
        const user = {
            _id: registerNumber,
            username: username ,
            password: password,
            solved: [],
            score: 0
        };

        // Insert the user into the user collection
        await userCollection.insertOne(user);

        return user;

    } catch (error) {
        console.error('Error finding user:', error);
        return "ERROR";
    }
    finally {
        await mainModule.close();
    }
}
```

```
module.exports = { RegisterUser };

// /src/modules/MainModule.js

const { MongoClient } = require('mongodb');

class MainModel {

  constructor() {
    this.client = new MongoClient('mongodb://localhost:27017');  //
Database Client
    this.dbName = 'ExaminationApp'; // Database name
    this.userCollection = 'User'; // User Collection
    this.appDataCollection = 'AppData'; // AppData Collection
  }

  async connect() {
    try {
        await this.client.connect();
        // console.log("Connected successfully to server");
    } catch(error) {
        console.error("Connection Error: ", error);
    }
  }

  async getCollection(collectionName) {
    try {
        const database = this.client.db(this.dbName);
        const collection = database.collection(collectionName);
        return collection;
    } catch(error) {
        console.error("Error getting collection: ", error);
    }
  }

  async getUserCollection() {
    return this.getCollection(this.userCollection);
  }

  async getAppDataCollection() {
    return this.getCollection(this.appDataCollection);
  }

  async close() {
    try {
        await this.client.close();
        // console.log("Connection closed");
    } catch(error) {
        console.error("Error closing connection: ", error);
    }
  }
}

module.exports = MainModel;

// /src/modules/LoginModule.js

const MainModel = require('./MainModel.js');

async function LoginUser(registerNumber){
    const mainModule = new MainModel();
    await mainModule.connect();
    const userCollection = await mainModule.getUserCollection();

    try {
        // Find the user with the given registerNumber
        // console.log("Searching for user: ", registerNumber);
        const user = await userCollection.findOne({ _id: registerNumber });
        // console.log(user);
        return user;

    } catch (error) {
        // console.log('Error finding user');
        return "ERROR";
    }
    finally {
        await mainModule.close();
    }
}
module.exports = { LoginUser };

// /src/models/ProfileModel.js

const MainModel = require('../MainModel.js');

async function getUser(userId){
    const mainModule = new MainModel();
    await mainModule.connect();
    const userCollection = await mainModule.getUserCollection();

    try {

        const user = await userCollection.findOne({ _id: userId });
        // console.log("model" , user);
        return user;
    } catch (error) {
        console.log("Error finding user details:", error);
        return null;
    }
}
async function DeleteUser(userId){
    const mainModule = new MainModel();
    await mainModule.connect();
    const userCollection = await mainModule.getUserCollection();

    try {
        // Delete the user document from the collection based on the userId
        const result = await userCollection.deleteOne({ _id: userId });

        if (result.deletedCount === 1) {
            return "DeleteSuccess";
        } else {
            return "DeleteFailed";
        }
    } catch (error) {
        console.log("Error deleting user:", error);
        return null;
    }
}

async function updateUserScore(userId, newScore, testName, prevScore)
{
    const mainModule = new MainModel();
    await mainModule.connect();
    const userCollection = await mainModule.getUserCollection();
    console.log(userId, newScore, testName, prevScore);
    const updatedScore = prevScore + newScore;
    console.log("Updated Score: ", updatedScore);
    try {
        await userCollection.updateOne(
            { _id: userId },
            { $set: { score: updatedScore }, $addToSet: { solved: testName }
} // Update score and add testName to solved array
        );
    } catch (error) {
        console.log("Error updating user score:", error);
        throw error;
    }
}

module.exports = {getUser, DeleteUser, updateUserScore};

// /src/models/appmodel/testmodel/MainDashboardModel.js
const MainModel = require('../MainModel.js');

async function generateTestList(){
    const mainModule = new MainModel();
    await mainModule.connect();

    const client = await mainModule.getAppDataCollection();

    try {
        const cursor = client.find({});

        const data = await cursor.toArray();

        return data;
    } catch (error) {
        console.log("Error finding user details:", error);
        return null;
    }
}

module.exports = { generateTestList };
// /src/models/ProfileModel.js

const MainModel = require('../MainModel.js');

async function getUser(userId){
    const mainModule = new MainModel();
    await mainModule.connect();
    const userCollection = await mainModule.getUserCollection();

    try {

        const user = await userCollection.findOne({ _id: userId });
        // console.log("model" , user);
        return user;
    } catch (error) {
        console.log("Error finding user details:", error);
        return null;
    }
}
async function DeleteUser(userId){
    const mainModule = new MainModel();
    await mainModule.connect();
    const userCollection = await mainModule.getUserCollection();

    try {
        // Delete the user document from the collection based on the userId
        const result = await userCollection.deleteOne({ _id: userId });

        if (result.deletedCount === 1) {
            return "DeleteSuccess";
        } else {
            return "DeleteFailed";
        }
    } catch (error) {
        console.log("Error deleting user:", error);
        return null;
    }
}

async function updateUserScore(userId, newScore, testName, prevScore)
{
    const mainModule = new MainModel();
    await mainModule.connect();
    const userCollection = await mainModule.getUserCollection();
    console.log(userId, newScore, testName, prevScore);
    const updatedScore = prevScore + newScore;
    console.log("Updated Score: ", updatedScore);
    try {
        await userCollection.updateOne(
            { _id: userId },
            { $set: { score: updatedScore }, $addToSet: { solved: testName }
} // Update score and add testName to solved array
        );
    } catch (error) {
        console.log("Error updating user score:", error);
        throw error;
    }
}

module.exports = {getUser, DeleteUser, updateUserScore};

// /src/ models/appmodel/MainDashboardModel.js

const { generateTestList } =
require("../models/appmodel/MainDashboardModel");

exports.getTestList = async (req, res) => {
    try {
        const data = await generateTestList();
        const formattedData = [];

        for (let i = 0; i < data.length; i++) {
            const { testName, testDescription, totalNoOfQuestion } = data[i];
            formattedData.push({
                testName: testName,
                testDesc: testDescription,
                totalNoOfQues: totalNoOfQuestion
            });
        }
        // console.log(formattedData);
        return res.json(formattedData);
    } catch (error) {
        console.log("Error:", error);
        return res.status(500).json({ error: "Internal Server Error" });
    }
}

// src/controllers/LoginRegisterController.js
const { LoginUser } = require('../models/LoginModel');
const { RegisterUser } = require('../models/RegisterModel');

function generateToken(userId) {
  const newToken = userId + Math.random().toString(36).substring(2, 15);
  console.log("New Token Generated - ", newToken);
  return newToken;
}
exports.register = async (req, res) => {
  let { username, registerNumber, password } = req.body;
  registerNumber = parseInt(registerNumber);
  console.log("Register END-POINT Hit:", username, registerNumber,
password);

  try {
    let user = await LoginUser(registerNumber, password);
    if (user) {
      console.log("User already exists:", user.username, registerNumber);
      return res.json({ message: 'User already exists.', username:
user.username, registerNumber: registerNumber });
    } else {
      user = await RegisterUser(username, registerNumber, password);
      if (user) {
        console.log("Signup Success:", username, registerNumber);
        return res.json({ message: "Signup Success", username: username,
registerNumber: registerNumber });
      } else {
        console.log("Signup Fail. Try again:", username);
        return res.json({ message: "Signup Fail. Try again.", username:
username, registerNumber: -1 });
      }
    }
  } catch (error) {
    console.log('...Register ERROR...');
    res.status(500).json({ message: 'Signup Fail. Try again.' });
  }
};

exports.login = async (req, res) => {
  let { registerNumber, password } = req.body;
  registerNumber = parseInt(registerNumber);
  console.log("Login END-POINT Hit:", registerNumber, password);
  try {
    const user = await LoginUser(registerNumber);
    if (user) {
      if (user.password === password) {
        console.log("User:", user._id, "Login Success at", new Date());
        const token = generateToken(user._id);
        res.cookie('token', token, { httpOnly: true });
        return res.json({ message: 'Login successful', username:
user.username, registerNumber: user._id, token: token });
      } else {
        console.log("User:", user._id, "Incorrect Password ", new Date());
        return res.json({ message: 'Incorrect Username or *Password'});
      }
    } else {
      console.log("User:", user._id, "User Not Found ", new Date());
      return res.json({ message: 'Incorrect *Username or Password'});
    }
  } catch (error) {
    console.log("Login API-END POINT Error ", new Date());
    res.status(500).json({ message: 'Server Error.'});
  }
};

// /src/controllers/ProfileController.js
const {getUser, DeleteUser, updateUserScore} =
require("../models/appmodel/ProfileModel");


exports.getUserDetails = async (req, res) => {
    let { userId } = req.query; // Use req.query to get parameters from the
URL
    userId = parseInt(userId);
    console.log("getUserDetails END-POINT Hit:", userId);

    try {
        const user = await getUser(userId); // Call getUserDetails function
with userId
        // console.log(user);
        if (!user) {
            return res.status(404).json({ message: "User not found" });
        }

        const data = {
            username: user.username,
            registerNumber: user._id,
            score: user.score,
            courseCount: user.solved.length, // Corrected to use array.length
            courseList: user.solved
        };
        // console.log(data);
        return res.json(data);
    } catch (error) {
        console.log("Error retrieving user details:", error);
        return res.status(500).json({ message: "Internal server error" });
    }
};


exports.deleteAccount = async (req, res) => {
    let { userId } = req.query; // Use req.query to get parameters from the
URL
    userId = parseInt(userId);
    console.log("getUserDetails END-POINT Hit:", userId);
    // Call the function to delete the user account based on the userId
    const result = await DeleteUser(userId);

    if (result === "DeleteSuccess") {
        return res.json({ message: "DeleteSuccess" });
    } else {
        return res.status(500).json({ message: "Failed to delete user
account" });
    }
};

exports.updateUserScore = async (req, res) => {
    let { userId, score, testName, prevScore } = req.query; // Extract
userId, score, and testName from request query parameters
    userId = parseInt(userId);
    score = parseInt(score);
    prevScore = parseInt(prevScore);
    console.log(userId, score, testName, prevScore);
    try {
        await updateUserScore(userId, score, testName, prevScore); // Call
updateUserScore function with userId, score, and testName
        return res.status(200).json({ message: "User score updated
successfully" });
    } catch (error) {
        console.log("Error updating user score:", error);
        return res.status(500).json({ message: "Internal server error" });
    }
};


// /src/controllers/TestController.js
const { getQuestionSet } =
require("../models/appmodel/testmodel/TestModel");
```

```
exports.testData = async (req, res) => {
    const data = req.query;
    console.log("Test Request: ", data.testName);
    try {
        const questionSet = await getQuestionSet(data.testName);
        // console.log(questionSet);
        if (questionSet) {
            return res.json({questionSet});
        } else {
            console.log("No question set found for test name: ${testName}");
            return res.json({message:`No question set found for test name: $
{testName}`});
        }

    } catch (error) {
        console.error('Error retrieving question set:', error);
    }
}
```

Sample Input and Output:


Exam App — Mozilla Firefox
Exam App
localhost:3000
IET G GH SSN Cloud Placement Project SM FC P&V Dev Tool Guitar GC SSN NLP Disaster Twe... Other Bookmarks

Login

Register Number:

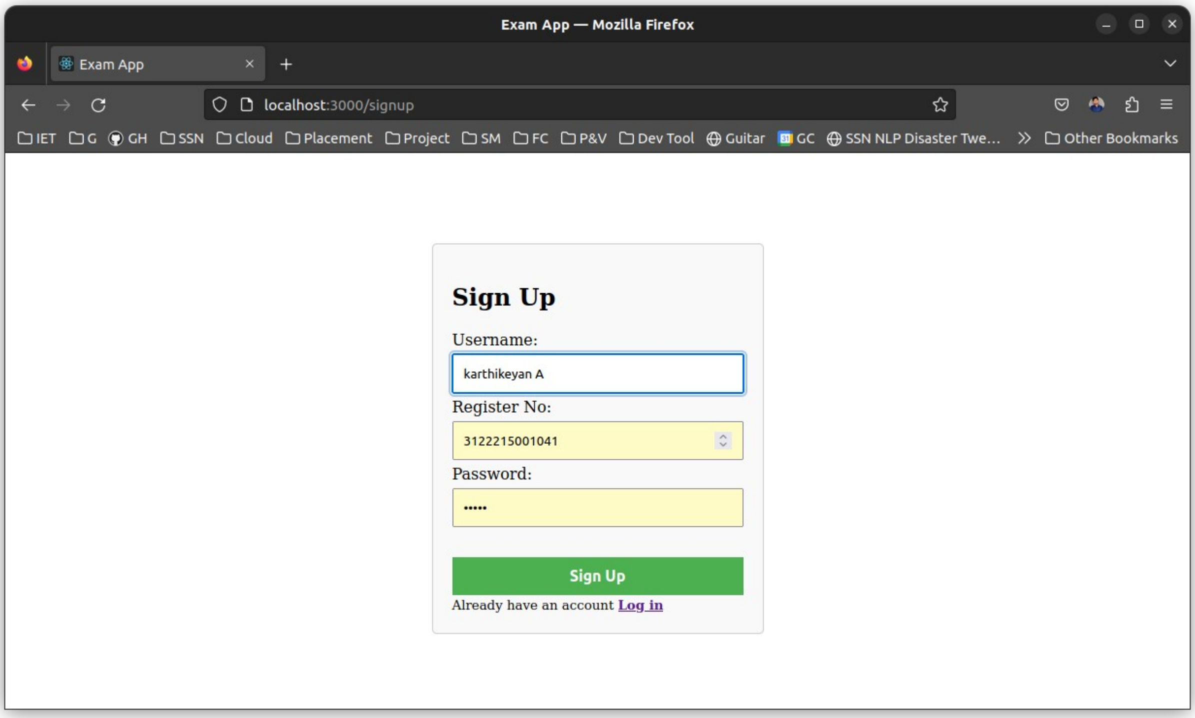
3122215001041

Password:

•••••

Login

Don't have an account? Sign up


Exam App — Mozilla Firefox

localhost:3000/signup

Sign Up

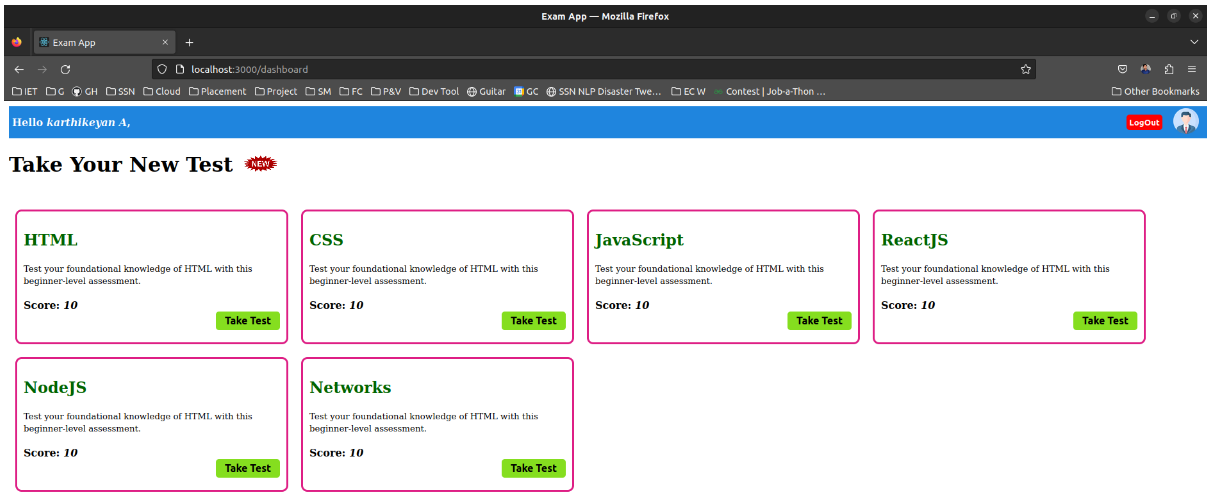
Username: karthikeyan A

Register No: 3122215001041

Password: •••••

Sign Up

Already have an account Log in


Exam App — Mozilla Firefox

localhost:3000/dashboard

Hello karthikeyan A,

LogOut

Take Your New Test

HTML
Test your foundational knowledge of HTML with this beginner-level assessment.
Score: 10
Take Test

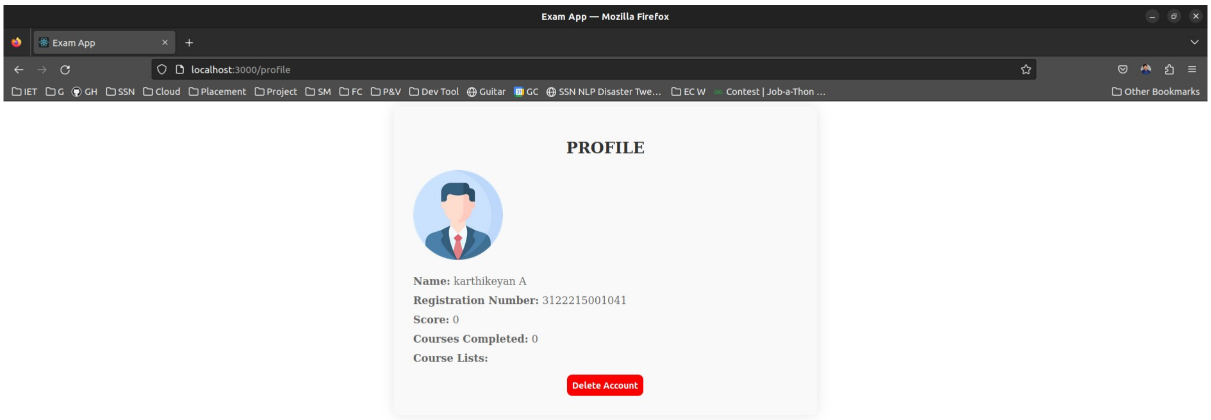
CSS
Test your foundational knowledge of HTML with this beginner-level assessment.
Score: 10
Take Test

JavaScript
Test your foundational knowledge of HTML with this beginner-level assessment.
Score: 10
Take Test

ReactJS
Test your foundational knowledge of HTML with this beginner-level assessment.
Score: 10
Take Test

NodeJS
Test your foundational knowledge of HTML with this beginner-level assessment.
Score: 10
Take Test

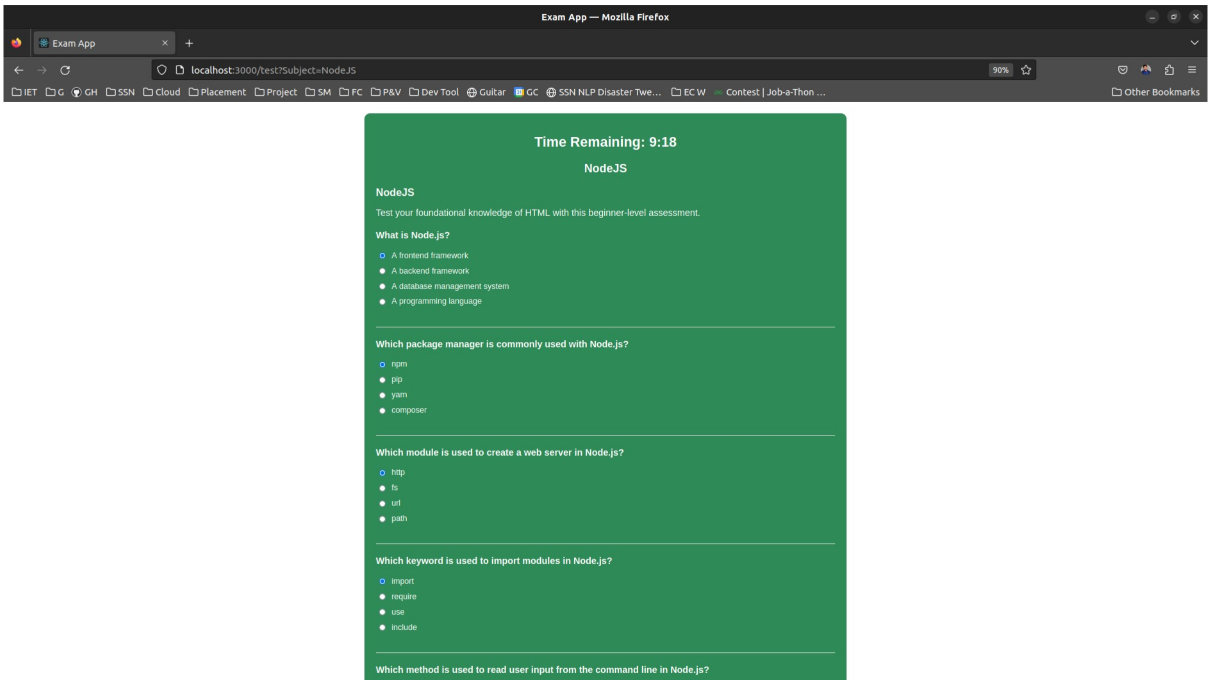
Networks
Test your foundational knowledge of HTML with this beginner-level assessment.
Score: 10
Take Test


Exam App — Mozilla Firefox
Exam App
localhost:3000/profile
IET G GH SSN Cloud Placement Project SM FC P&V Dev Tool Guitar GC SSN NLP Disaster Twe… EC W Contest | Job-a-Thon … Other Bookmarks

PROFILE

Name: karthikeyan A
Registration Number: 3122215001041
Score: 0
Courses Completed: 0
Course Lists:

Delete Account


Exam App — Mozilla Firefox
Exam App
localhost:3000/test?Subject=NodeJS
90%
IET G GH SSN Cloud Placement Project SM FC P&V Dev Tool Guitar GC SSN NLP Disaster Twe… EC W Contest | Job-a-Thon … Other Bookmarks

Time Remaining: 9:18

NodeJS

NodeJS

Test your foundational knowledge of HTML with this beginner-level assessment.

What is Node.js?

- A frontend framework
- A backend framework
- A database management system
- A programming language

Which package manager is commonly used with Node.js?

- npm
- pip
- yarn
- composer

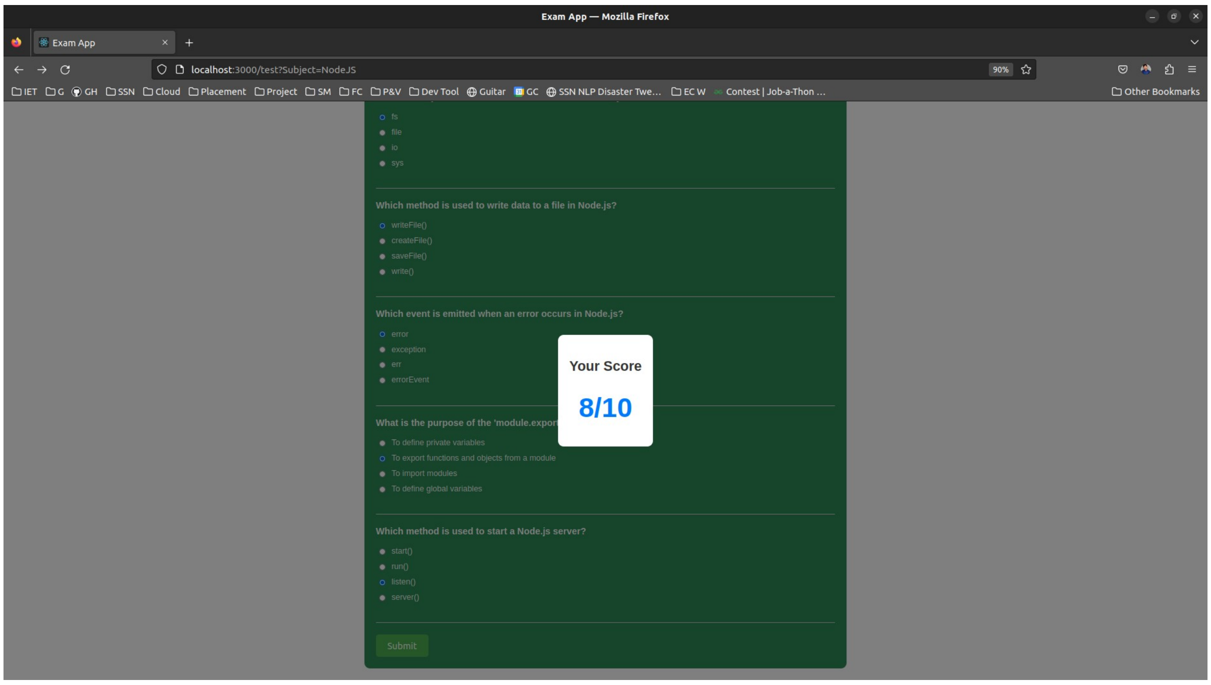
Which module is used to create a web server in Node.js?

- http
- fs
- url
- path

Which keyword is used to import modules in Node.js?

- import
- require
- use
- include

Which method is used to read user input from the command line in Node.js?


Exam App — Mozilla Firefox
localhost:3000/test?Subject=NodeJS
Submit

- fs
- file
- io
- sys

Which method is used to write data to a file in Node.js?

- writeFile()
- createFile()
- saveFile()
- write()

Which event is emitted when an error occurs in Node.js?

- error
- exception
- err
- errorEvent

What is the purpose of the 'module.export

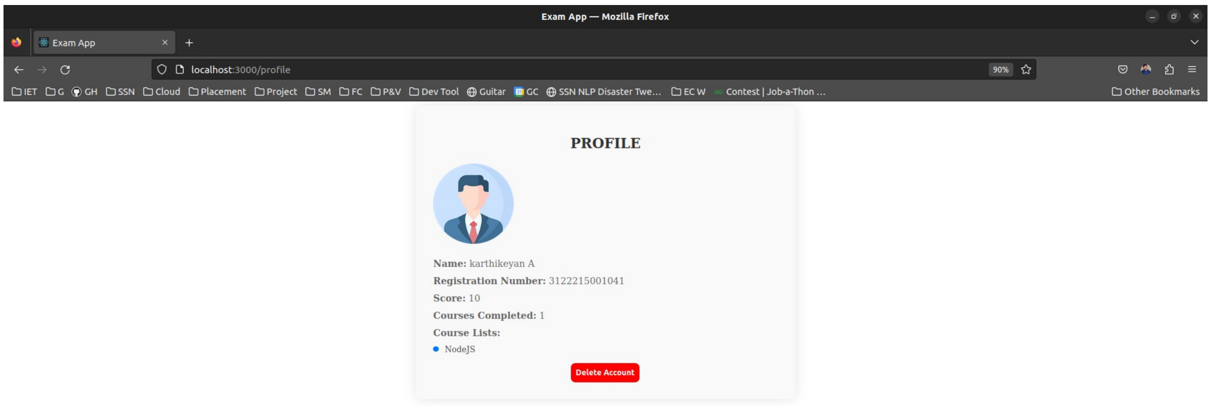
- To define private variables
- To export functions and objects from a module
- To import modules
- To define global variables

Which method is used to start a Node.js server?

- start()
- run()
- listen()
- server()

Your Score

8/10



PROFILE

Name: karthikeyan A

Registration Number: 3122215001041

Score: 10

Courses Completed: 1

Course Lists:

- NodeJS

Best Practices:

- Design before coding
- Incremental coding
- Usage of proper naming convention
- Usage of Comments to the code
- Indentation of code