

Mini Project

Team Members:

Shravan Sreeram 3122215001102

Shreyas S 3122215001103

Y.V.Ojus 3122215001125

Github Link:

[Project](#)

Aim:

To develop Full stack web application for conducting On-line quiz using MVC architecture. The application facilitates the normal and admin users to access it. To the normal user, instructions, questions with options and the score is provided. Admin user can view the registered users and their scores.

Description:

server.js:

This Node.js script configures an Express server, leveraging middleware modules like express, mongoose for MongoDB interaction, and cors for cross-origin resource sharing facilitation. It imports two models, LoginModel and userModel, presumably managing user authentication and quiz data, respectively. Upon initializing the Express application, it incorporates middleware like cors() for cross-origin requests and express.json() for parsing incoming JSON data. Establishing a connection to a MongoDB database located at mongodb://127.0.0.1:27017/quiz, the server defines several routes. The /send route handles user authentication, extracting credentials from the request body and querying the LoginModel for a matching user. If found, it responds with user data; otherwise, it notifies of user absence. The /get route retrieves all users from the database using LoginModel. For updating user scores, the /score route expects a request body containing the user's username and score. It updates the score if the user is found and returns appropriate responses based on the outcome. Additionally, the /getQns route fetches quiz questions

Internet Programming Lab
UCS2611
Mini Project

Lab

102, 103, 125
3122 21 5001 125

from userModel, employing MongoDB's aggregation pipeline to sample ten random questions. The server listens on port 3001, with a logging message indicating its readiness to accept connections. Before starting, it confirms successful database connection. This setup enables the server to manage user authentication, handle quiz-related data, and serve quiz questions via HTTP endpoints efficiently.

App.js:

In App.js, this React component configures a basic user interface for a Quiz App. It imports necessary dependencies like react-router-dom for navigation and axios for making HTTP requests. Within the component, it initializes the useNavigate hook from react-router-dom for programmatic navigation.

The verifyUser function is defined to handle user authentication. It retrieves the username and password from input fields, clears the fields, and sends a POST request to the server at <http://127.0.0.1:3001/send> with the provided credentials and an initial score of 0. Depending on the response received, it navigates the user to different routes. If the username is not admin, it redirects to the /Quiz route, passing the username in the route state. If the username is "admin", it redirects to the /Admin route.

The JSX portion of the component defines the UI elements including a header, login box, input fields for username and password, and a login button. When clicked, the verifyUser function is invoked.

Overall, this setup enables user authentication and navigation based on the response from the server, facilitating a smooth user experience within the Quiz App.

Admin.js:

In this React component named Admin, a functional component is created to manage the administration section of the Quiz App. It utilizes React's useState hook to maintain the state of users, which stores user data fetched from the server.

The getUsers function is defined to fetch user data from the server when the "Get Data" button is clicked. It sends a GET request to <http://localhost:3001/get>, retrieves the response data, updates the users state, and logs the updated state to the console.

Internet Programming Lab
UCS2611
Mini Project

Lab

102, 103, 125
3122 21 5001 125

Within the JSX part, the component renders a header with the title "Admin" and a button labeled "Get Data" to trigger the getUsers function. Below the button, a table is rendered to display user data. For each user in the users array, it maps over the array, rendering a table row (<tr>) with the user's username and score. It skips rendering the admin user by checking if the username is not "admin".

This setup enables the admin section of the Quiz App to fetch user data from the server and display it in a tabular format, excluding the admin user.

Quiz.js:

In this React component named Quiz, it manages the quiz section of the Quiz App. The component utilizes React's useState and useEffect hooks to manage state and perform side effects respectively.

Upon initialization, the component sends a GET request to <http://127.0.0.1:3001/getQns> to fetch quiz questions. Upon successful retrieval, it sets the users state with the fetched questions and initializes the selectedOptions state with default values for each question. If an error occurs during the fetch, it sets the error state with the error message. While waiting for the data to load, it displays a "Loading..." message.

The component also manages the user's selected options for each question using the handleOptionChange function, which updates the selectedOptions state accordingly.

Functionality is provided to navigate to the next question or submit the quiz. Upon submitting, it calculates the user's score based on selected options, updates the score state, and sends a POST request to <http://127.0.0.1:3001/score> to store the user's score in the server.

The UI renders the current question, options, and navigation buttons. Upon completion of the quiz, it displays the user's score. If no questions are found or an error occurs, appropriate messages are displayed.

Overall, this component facilitates the quiz-taking experience within the Quiz App, managing state, fetching questions, handling user interactions, and displaying relevant information to the user.

LoginModel.js:

This script defines a MongoDB schema and model for user login data using Mongoose. The LoginSchema variable creates a schema with fields for username, password, and score. Each field is assigned a data type: username and password as strings, and score as a number.

The LoginModel variable creates a Mongoose model based on the schema, specifying the collection name as "users". This model will be used to interact with the MongoDB database for operations related to user login data.

Finally, the model is exported to be used in other parts of the application.

QuizModel.js:

This script defines a Mongoose schema and model for quiz questions. The userSchema variable creates a schema comprising fields for question, options, and answer. Each field has specific requirements: question and answer are strings and required, while options is an array of strings and required.

The userModel variable creates a Mongoose model based on the schema, specifying the collection name as "questions". This model facilitates interactions with the MongoDB database for operations related to quiz questions.

Finally, the model is exported to be accessible in other parts of the application.

Code:

server.js:

```
const express = require("express");
const mongoose = require("mongoose");
const cors = require("cors");
const LoginModel = require("../Models/LoginModel.js");
const userModel = require("../Models/QuizModel.js");

const app = express();
app.use(cors());
app.use(express.json());

mongoose.connect("mongodb://127.0.0.1:27017/quiz");

app.post("/send", (req, res) => {
  const user = req.body.username;
  const pass = req.body.password;
  console.log(user, pass);
  LoginModel.findOne({ username: user, password: pass })
    .then(result => {
      console.log(result)
      res.json(result)
    })
    .catch(err => console.log(err))
})

app.get("/get", (req, res) => {
  LoginModel.find()
    .then(result => res.json(result))
    .catch(err => console.log(err))
})

app.post("/score", (req, res) => {
  const score = req.body.result;
  const user = req.body.user;

  LoginModel.findOne({ username: user })
    .then(result => {
      if (result) {
        const id = result._id;
```

Internet Programming Lab
UCS2611
Mini Project

Lab

102, 103, 125
3122 21 5001 125

```
        console.log(result);
        console.log(id);

        LoginModel.updateOne({ _id: id }, { $set: { score: score } })
            .then(updateResult => {
                console.log(updateResult);
                res.status(200).send("Received Score");
            })
            .catch(err => {
                console.log(err);
                res.status(500).send("Error updating score");
            });
    } else {
        console.log("User not found");
        res.status(404).send("User not found");
    }
})
.catch(err => {
    console.log(err);
    res.status(500).send("Error finding user");
});
});

console.log("Connected to Database Successfully")
app.get("/getQns", (req, res) => {
    userModel.aggregate([{ $sample: { size: 10 } }])
        .then(questions => {
            res.json(questions);
        })
        .catch(err => {
            res.status(500).json({ error: err.message });
        });
});

app.listen(3001, () => {
    console.log(`Server listening on Port 3001`);
})
```

App.js:

```
// App.js
import { useNavigate } from 'react-router-dom';
import './App.css';
import axios from 'axios';

function App() {
  const navigate = useNavigate(); // Initialize useNavigate hook

  function verifyUser() {
    let user = document.getElementById("username").value;
    let pass = document.getElementById("password").value;

    document.getElementById("username").value = "";
    document.getElementById("password").value = "";

    axios.post("http://127.0.0.1:3001/send", { username: user, password: pass,
score:0 })
      .then(res => {
        const response = res.data?.username || "absent";
        if (response === "abc") {
          navigate('/Quiz',{ state: {name:response} });
        } else if(response === "admin"){
          navigate("/Admin");
        }
        console.log(response);
      })
      .catch(err => console.log(err))
  }

  return (
    <div className="App">
      <div className='Header'>
        <h1>Quiz App</h1>
      </div>
      <div className='box'>
        <div className='admin--login'>
          <h1 id='admin--welcome'>Welcome User 😊 </h1>
          <div className='admin--username'>
            <label htmlFor="username" id='label--username'><b>Username:
</b></label>
            <input type='text' id='username' placeholder='Enter Username'></input>
          </div>
        </div>
      </div>
    </div>
  )
}
```

Internet Programming Lab
UCS2611
Mini Project

Lab

102, 103, 125
3122 21 5001 125

```
        <div className='admin--password'>
          <label htmlFor="password" id='label--password'><b>Password:</b>
</label>
          <input type='password' id='password' placeholder='Enter
Password'></input>
        </div>
        <button id='login--button' onClick={verifyUser}>Login</button>
      </div>
    </div>
  </div>
);
}

export default App;
```

App.css:

```
* {
  padding: 0;
  margin: 0;
  box-sizing: border-box;
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

body {
  background-color: #CEE6F2;
  overflow: auto;
}

.App {
  text-align: center;
}

.Header {
  height: 100px;
  background-color: #962E2A;
  color: white;
  display: flex;
  justify-content: center;
  align-items: center;
}

.box {
```


Internet Programming Lab
UCS2611
Mini Project

Lab

102, 103, 125
3122 21 5001 125

```
display: flex;
justify-content: center;
margin: 5rem;
}

.admin--login {
padding: 2rem;
background-color: #E3867D;
border: 2px solid #962E2A;
border-radius: 15px;
box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
}

#admin--welcome {
margin-bottom: 2rem;
font-size: 1.5rem;
}

.admin--username,
.admin--password {
padding: 1rem;
margin: 0.5rem;
border: 1px solid #ccc;
border-radius: 5px;
display: flex;
justify-content: space-evenly;
width: 20rem;
}

#username,
#password{
text-align: center;
}

#label--username,
#label--password{
text-shadow: 0px 0px 25px black;
}

::-webkit-input-placeholder{
text-align: center;
}

#login--button {
```

Internet Programming Lab
UCS2611
Mini Project

Lab

102, 103, 125
3122 21 5001 125

```
padding: 0.5rem 1rem;
cursor: pointer;
background-color: #962E2A;
color: white;
border: none;
border-radius: 5px;
width: 8rem;
}

#Login--button:hover {
  background-color: #761F1B;
}
```

Admin.js:

```
import { useState } from "react";
import "./Admin.css";
import axios from "axios";

export default function Admin() {
  const [users, setUsers] = useState([]);
  function getUsers() {
    axios.get("http://localhost:3001/get")
      .then(res => {
        setUsers(res.data);
        console.log(users);
      })
      .catch(err => console.log(err))
  }
  return (
    <div>
      <div className='Header'>
        <h1>Admin</h1>
      </div>
      <div className="admin--body">
        <button id="admin--button" onClick={getUsers}>Get Data</button>
      </div>
      <div>
        <table className="display--table">
          <thead>
            <tr>
              <th>User</th>
            </tr>
          </thead>
        </table>
      </div>
    </div>
  )
}
```

Internet Programming Lab
UCS2611
Mini Project

Lab

102, 103, 125
3122 21 5001 125

```
        <th>Score</th>
      </tr>
    </thead>
    <tbody>
      {users.map((user, index) => {
        if (user.username !== "admin") {
          return (
            <tr key={index}>
              <td>{user.username}</td>
              <td>{user.score}</td>
            </tr>
          );
        } else {
          return null; // Skip rendering the admin user
        }
      })}
    </tbody>
  </table>

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

Admin.css:

```
* {
  padding: 0;
  margin: 0;
  box-sizing: border-box;
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

body {
  background-color: #CEE6F2;
  overflow: auto;
}

.Header {
  height: 100px;
  background-color: #962E2A;
  color: white;
}
```

Internet Programming Lab
UCS2611
Mini Project

Lab

102, 103, 125
3122 21 5001 125

```
display: flex;
justify-content: center;
align-items: center;
}

#admin--button{
padding: 0.5rem 1rem;
cursor: pointer;
background-color: #962E2A;
color: white;
border: none;
border-radius: 5px;
width: 8rem;
}

.admin--body{
display: flex;
justify-content: center;
align-items: center;
height: 10rem
}

.display--table {
width: 100%;
border-collapse: collapse;
margin-top: 20px;
}

.display--table th,
.display--table td {
padding: 8px;
border: 1px solid #ddd;
background-color: #f2f2f2;
text-align: center;
}

.display--table th {
background-color: #f2f2f2;
}
```

Quiz.js:

```
import React, { useEffect, useState } from "react";
import axios from "axios";
import { useLocation } from "react-router-dom";
import "./Quiz.css";

function Quiz() {
  const [users, setUsers] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [selectedOptions, setSelectedOptions] = useState({});
  const [score, setScore] = useState(null);
  const [currentQuestionIndex, setCurrentQuestionIndex] = useState(0);

  const { state } = useLocation();
  const { name } = state;

  useEffect(() => {
    axios
      .get("http://127.0.0.1:3001/getQns")
      .then((response) => {
        setLoading(false);
        setUsers(response.data);
        const defaultSelectedOptions = {};
        response.data.forEach((user, index) => {
          defaultSelectedOptions[index] = "";
        });
        setSelectedOptions(defaultSelectedOptions);
      })
      .catch((err) => {
        setLoading(false);
        setError(err.message);
      });
  }, []);

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

  const goToNextQuestion = () => {
```

Internet Programming Lab
UCS2611
Mini Project

Lab

102, 103, 125
3122 21 5001 125

```
    setCurrentQuestionIndex((prevIndex) => prevIndex + 1);
  };

  const calculateScore = () => {
    let score = 0;
    users.forEach((user, index) => {
      if (selectedOptions[index] === user.answer) {
        score += 1;
      }
    });
    setScore(score);

    axios
      .post("http://127.0.0.1:3001/score", { result: score, user: name })
      .then((res) => console.log(res))
      .catch((err) => console.log(err));
  };

  if (loading) return <div>Loading...</div>;
  if (error) return <div>Error: {error}</div>;

  if (users.length === 0) return <div>No questions found.</div>; // Handle no
questions

  const currentQuestion = users[currentQuestionIndex];

  return (
    <div>
      <h1 style={{ textAlign: "center" }}>Quiz Time {name} 😊 </h1>
      <div className="container">
        <div className="question--box">
          <h1 id="question">
            {currentQuestionIndex + 1}. {currentQuestion.question}
          </h1>
          <h3 className="option--box">
            {currentQuestion.options.map((option, index) => (
              <div className="option">
                <label key={index}>
                  <input
                    type="radio"
                    name={`option-${currentQuestionIndex}`}
                    value={option}
                    checked={selectedOptions[currentQuestionIndex] === option}
                    onChange={() =>
```

Internet Programming Lab
UCS2611
Mini Project

Lab

102, 103, 125
3122 21 5001 125

```
        handleOptionChange(currentQuestionIndex, option)
    }
    />
    {String.fromCharCode(65 + index)}. {option}
</label>
</div>
    )}
</h3>
<br />
{currentQuestionIndex < users.length - 1 ? (
    <button className="btn" onClick={goToNextQuestion}>
        Next
    </button>
) : (
    <button className="btn" onClick={calculateScore}>
        Submit
    </button>
)}
</div>

{score !== null && (
    <div>
        <h2>
            Your Score: {score}/{users.length}
        </h2>
    </div>
)}
</div>
</div>
);
}

export default Quiz;
```

Internet Programming Lab
UCS2611
Mini Project

Lab

102, 103, 125
3122 21 5001 125

Quiz.css:

```
*{
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

h3{
  margin-left: 40px;
}

.container h3 {
  margin-left: 50px;
}

.btn {
  display: inline-block;
  padding: 10px 20px;
  font-size: 16px;
  font-weight: bold;
  text-align: center;
  text-decoration: none;
  cursor: pointer;
  border: 2px solid #4CAF50; /* Green border */
  color: #4CAF50; /* Green text */
  background-color: white;
  border-radius: 5px;
  transition: background-color 0.3s, color 0.3s;
}

.btn:hover {
  background-color: #4CAF50; /* Green background */
  color: white; /* White text */
}

body {
  /* fallback for old browsers */
  background-color: #1e5799;
  /* Chrome 10-25, Safari 5.1-6 */
  background: -webkit-linear-gradient(to left, #1e5799, #2989d8, #207cca, #7db9e8);
  /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */
  background: linear-gradient(to left, #1e5799, #2989d8, #207cca, #7db9e8);
}

.container{
```


Internet Programming Lab
UCS2611
Mini Project

Lab

102, 103, 125
3122 21 5001 125

```
display: flex;
flex-direction: column;
justify-content: center;
align-items: center;
}

.question--box{
padding: 5rem;
margin: 3rem;
background-color: #962E2A;
display: flex;
flex-direction: column;
justify-content: space-evenly;
align-items: center;
border-radius: 2rem;
}

#question{
color: white;
margin-bottom: 2rem;
}

.option--box {
margin: 1rem;
padding: 0.5rem;
width: 100%;
}

.option {
background-color: white;
border: 2px solid green;
border-radius: 10px;
margin: 1rem;
padding: 0.5rem;
display: flex;
justify-content: space-around;
cursor: pointer;
transition: background-color 0.3s, color 0.3s;
}

.option:hover {
background-color: #4CAF50;
color: white;
}
```

Internet Programming Lab
UCS2611
Mini Project

Lab

102, 103, 125
3122 21 5001 125

```
}  
  
.option input[type="radio"] {  
  margin-right: 5px;  
}
```

LoginModel.js:

```
const mongoose = require("mongoose");  
const LoginSchema = new mongoose.Schema({  
  username: String,  
  password: String,  
  score: Number  
})  
  
const LoginModel = mongoose.model("users", LoginSchema);  
module.exports = LoginModel;
```

QuizModel.js:

```
const mongoose = require("mongoose");  
  
const userSchema = new mongoose.Schema({  
  question : {  
    type: String,  
    required: true  
  },  
  options :{  
    type: [String] ,  
    required: true  
  },  
  answer :{  
    type:String,  
    required: true  
  }  
});  
  
const userModel=mongoose.model("questions",userSchema);
```

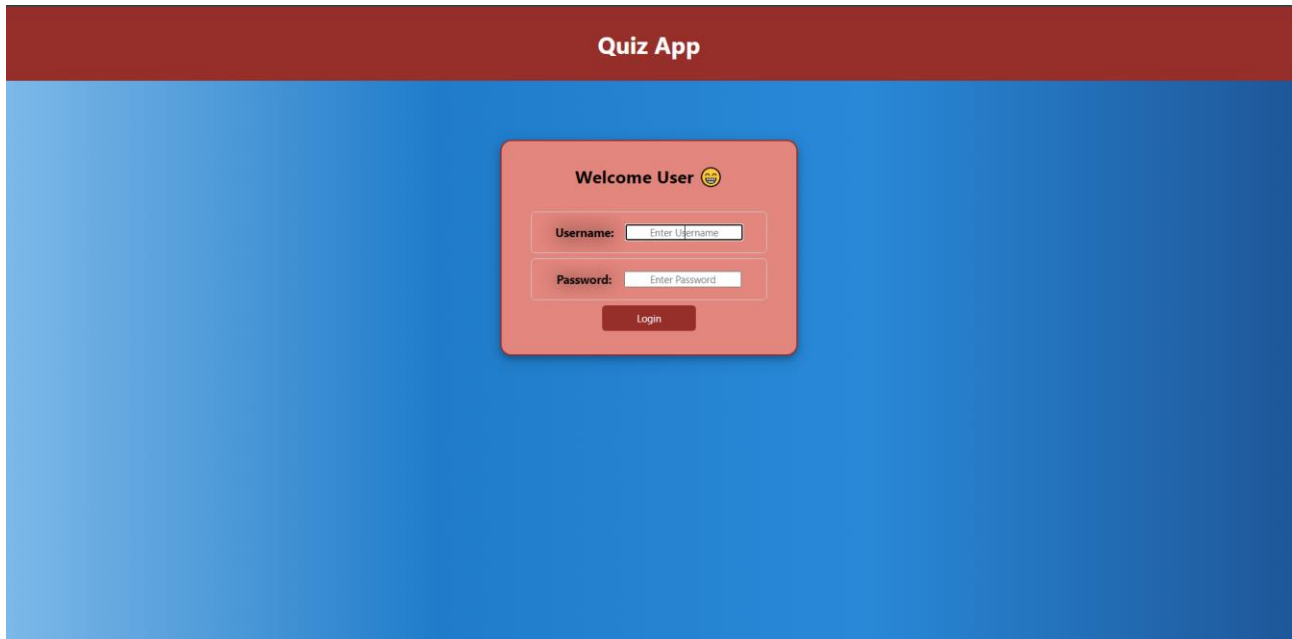
Internet Programming Lab
UCS2611
Mini Project

Lab

102, 103, 125
3122 21 5001 125

```
module.exports= userModel;
```

Output:



The image shows a web application titled "Quiz App" with a dark red header. The main content area has a blue gradient background. In the center, there is a light red rounded rectangle containing a login form. The form has the text "Welcome User" with a smiley face emoji. Below this, there are two input fields: "Username:" and "Password:", each with a placeholder text "Enter Username" and "Enter Password" respectively. At the bottom of the form is a dark red "Login" button.

Frontend Login



The image shows a web application titled "Quiz Time abc" with a dark red header. The main content area has a blue gradient background. In the center, there is a dark red rounded rectangle containing a quiz question. The question is "1. Which country was the first to give women the right to vote?". Below the question are four radio button options: "A. United States", "B. United Kingdom", "C. New Zealand", and "D. Australia". At the bottom of the form is a green "Next" button.

Quiz Page

Internet Programming Lab
UCS2611
Mini Project

Lab

102, 103, 125
3122 21 5001 125

Quiz Time abc 😊

1. Which country was the first to give women the right to vote?

☒ A. United States

☐ B. United Kingdom

☐ C. New Zealand

☐ D. Australia

Next

Hover Effect

Quiz Time abc 😊

10. What is the chemical symbol for the element oxygen?

☒ A. O

☐ B. H

☐ C. C

☐ D. N

Submit

Your Score: 9/10

Score

Internet Programming Lab
UCS2611
Mini Project

Lab

102, 103, 125
3122 21 5001 125

Admin

Get Data

User	Score
------	-------

Admin Page

Get Data

User	Score
abc	9
def	0

User Scores

Internet Programming Lab
UCS2611
Mini Project

Lab

102, 103, 125
3122 21 5001 125



```
_id: ObjectId('663f26e50ba8ba01465a655c')
username : "admin"
password : "admin"
score : 0
```

```
_id: ObjectId('663f270a0ba8ba01465a655d')
username : "abc"
password : "abc"
score : 7
```

```
_id: ObjectId('663f4cd00ba8ba01465a6561')
username : "def"
password : "def"
score : 0
```

MongoDB – Admin

```
_id: ObjectId('663e0bb51c16ec4691d72444')
question : "What is the capital of Canada?"
options : Array (4)
answer : "Ottawa"
```

```
_id: ObjectId('663e0bb51c16ec4691d72445')
question : "Which country is known as the Land of the Rising Sun?"
options : Array (4)
answer : "Japan"
```

```
_id: ObjectId('663e0bb51c16ec4691d72446')
question : "What is the largest ocean on Earth?"
options : Array (4)
answer : "Pacific Ocean"
```

```
_id: ObjectId('663e0bb51c16ec4691d72447')
question : "What is the world's longest river?"
options : Array (4)
answer : "Amazon River"
```

```
_id: ObjectId('663e0bb51c16ec4691d72448')
question : "Which desert is the largest in the world?"
options : Array (4)
answer : "Sahara Desert"
```

```
_id: ObjectId('663e0bb51c16ec4691d72449')
question : "The Great Barrier Reef is located off the coast of which country?"
options : Array (4)
answer : "Australia"
```

MongoDB – Questions

Internet Programming Lab
UCS2611
Mini Project

Lab

102, 103, 125
3122 21 5001 125
