IP LAB Mini Project

M ROHITH 3122215001085

S ROHIT RAM 3122215001086

SANJHAY V 3122215001093

SCHEMA FOR MONGODB:

USER AND ADMIN DETAILS:

LoginModel.js Schema:

- **Schema Name:** LoginSchema
- **Fields:**
  1. **username:**
     - Type: String
     - Description: Represents the username of the user.
  2. **password:**
     - Type: String
     - Description: Stores the password of the user.
  3. **score:**
     - Type: Number
     - Description: Tracks the score associated with the user.

QUESTION STORAGE:

QuizModel.js Schema:

- **Schema Name:** userSchema
- **Fields:**
  1. **question:**
     - Type: String
     - Description: Stores the actual question for the quiz.

- Constraints: Required field.

2. **options:**
   - Type: Array of Strings
   - Description: Represents the multiple choice options for the question.
   - Constraints: Required field.
3. **answer:**
   - Type: String
   - Description: Holds the correct answer to the question.
   - Constraints: Required field.

SEQUENCE FLOW:

- **Server Configuration:**
- Utilizes Express server in Node.js. **Server Configuration:**
  - Utilizes Express server in Node.js.
  - Dependencies include express, mongoose, and cors middleware.
  - Connects to MongoDB database at mongodb://127.0.0.1:27017/quiz.
  - Defines routes for user authentication, data retrieval, score updating, and quiz questions.
  - Listens on port 3001 for connections.
  - Logs successful database connection before starting.
- **User Authentication:**
  - '/send' route handles user authentication.
  - Extracts credentials from request body and queries LoginModel.
  - Responds with user data if found, else notifies of user absence.
- **Data Management:**
  - '/get' route retrieves all users from LoginModel.
  - '/score' route updates user scores based on request body data.
- **Quiz Data Handling:**
  - '/getQns' route fetches quiz questions from userModel.
  - Utilizes MongoDB's aggregation pipeline to sample ten random questions.
- **App.js (React Component):**
  - Configures user interface for Quiz App.
  - Imports react-router-dom for navigation and axios for HTTP requests.
  - Initializes useNavigate hook for navigation.
  - Implements verifyUser function for user authentication.

    - UI includes header, login box, input fields for username and password, and login button.
- **Admin.js (React Component):**
    - Manages administration section of Quiz App.
    - Utilizes useState hook to maintain user state.
    - Implements getUsers function to fetch user data from server.
    - Renders header, "Get Data" button, and table for displaying user data.
    - Excludes admin user from display.
- **Quiz.js (React Component):**
    - Manages quiz section of Quiz App.
    - Uses useState and useEffect hooks for state management and side effects.
    - Sends GET request to fetch quiz questions upon initialization.
    - Handles user interactions for selecting options and navigating questions.
    - Calculates and submits user score to server.
    - Displays current question, options, and navigation buttons.
    - Shows user's score upon quiz completion.
- **LoginModel.js:**
    - Defines MongoDB schema and model for user login data.
    - Schema includes fields for username, password, and score.
    - Model named LoginModel, collection named "users".
- **QuizModel.js:**
    - Defines MongoDB schema and model for quiz questions.
    - Schema includes fields for question, options, and answer.
    - Model named QuizModel, collection named "questions"
    -
    - Dependencies include express, mongoose, and cors middleware.
    - Connects to MongoDB database at mongodb://127.0.0.1:27017/quiz.
    - Defines routes for user authentication, data retrieval, score updating, and quiz questions.
    - Listens on port 3001 for connections.
    - Logs successful database connection before starting.
- **User Authentication:**
    - '/send' route handles user authentication.
    - Extracts credentials from request body and queries LoginModel.
    - Responds with user data if found, else notifies of user absence.
- **Data Management:**
    - '/get' route retrieves all users from LoginModel.
    - '/score' route updates user scores based on request body data.
- **Quiz Data Handling:**

    - '/getQns' route fetches quiz questions from userModel.
    - Utilizes MongoDB's aggregation pipeline to sample ten random questions.
- **App.js (React Component):**
    - Configures user interface for Quiz App.
    - Imports react-router-dom for navigation and axios for HTTP requests.
    - Initializes useNavigate hook for navigation.
    - Implements verifyUser function for user authentication.
    - UI includes header, login box, input fields for username and password, and login button.
- **Admin.js (React Component):**
    - Manages administration section of Quiz App.
    - Utilizes useState hook to maintain user state.
    - Implements getUsers function to fetch user data from server.
    - Renders header, "Get Data" button, and table for displaying user data.
    - Excludes admin user from display.
- **Quiz.js (React Component):**
    - Manages quiz section of Quiz App.
    - Uses useState and useEffect hooks for state management and side effects.
    - Sends GET request to fetch quiz questions upon initialization.
    - Handles user interactions for selecting options and navigating questions.
    - Calculates and submits user score to server.
    - Displays current question, options, and navigation buttons.
    - Shows user's score upon quiz completion.
- **LoginModel.js:**
    - Defines MongoDB schema and model for user login data.
    - Schema includes fields for username, password, and score.
    - Model named LoginModel, collection named "users".
- **QuizModel.js:**
    - Defines MongoDB schema and model for quiz questions.
    - Schema includes fields for question, options, and answer.
    - Model named QuizModel, collection named "questions"

Code:

server.js:

```
//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; 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 !== "absent" && response !== "sanjhay") { navigate('/Quiz',{ state:
                    {name:response} });
                } else if(response === "sanjhay"){ navigate("/Admin");
                }
                console.log(response);
            })
            .catch(err => console.log(err))
    }

    return (
        <div className="App">
            <div className='box'>
                <div className='admin--login'>
                    <h1 id='admin--welcome'>Welcome To the Ultimate Quiz Challenge </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 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:

```
//App.css

* {
    padding: 0;
    margin: 0;
    box-sizing: border-box;
    font-family: 'Roboto', Arial, sans-serif; /* Changed font family for a modern look */
}

body {
    background-image:'https://th.bing.com/th/id/OIP.LWxRXkOQv49IBpY1- b6X6gHaFP?rs=1&pid=ImgDetMain' ;
    background-color: #F0F4F8; /* Lighter background for a softer look */ overflow: auto;
```

```
}

.App {
    text-align: center;
}

.Header { height:
    100px;
    background-color: #30475E; /* Changed to a deep blue shade */ color:
    #E8E8E8; /* Soft white for text */
    display: flex;
    justify-content: center; align-
    items: center;
}

.box {
    display: flex;
    justify-content: center;
    margin: 5rem;
}

.admin--login { padding:
    2rem;
    background-color: #F9D8D6; /* Soft pink for a gentle interface */ border: 3px solid
    #30475E; /* Dark border for contrast */
    border-radius: 15px;
    box-shadow: 0 8px 16px 0 rgba(0, 0, 0, 0.15), 0 12px 30px 0 rgba(0, 0, 0, 0.25); /* Enhanced shadows for 3D
effect */
}

#admin--welcome { margin-
    bottom: 2rem;
    font-size: 1.8rem; /* Larger text for emphasis */
}

.admin--username,
.admin--password {
    padding: 1rem; margin:
    0.5rem;
    border: 2px solid #B0BEC5; /* Subtle border color */
    border-radius: 8px; /* Slightly rounder borders for a modern look */ display: flex;
    justify-content: space-between; /* Ensure space between label and input */ width: 22rem; /*
    Slightly wider for better spacing */
}

#username, #password
{
    text-align: left; /* Align text to the left inside input fields */ flex-grow: 1; /* Allow
    input to fill space */
}

#label--username,
```

```
#label--password {
   color: #30475E; /* Consistent color with header */
   text-shadow: none; /* Remove text shadow for cleaner look */
}

::-webkit-input-placeholder { text-
   align: center;
   color: #6D6875; /* Subtle placeholder color */
}

#login--button {
   padding: 0.75rem 1.5rem;
   cursor: pointer; background-
   color: #30475E; color: #E8E8E8;
   border: 2px solid #F9D8D6; /* Light border for contrast */ border-radius:
   10px; /* More rounded for a friendly feel */ width: 10rem; /* Wider button
   for easier click */
   transition: background-color 0.3s, color 0.3s; /* Smooth transition for hover effect
*/
}

#login--button:hover {
   background-color: #22333B; /* Darker shade on hover for depth */ color: #F9D8D6; /*
   Text color changes on hover */
}
```

Admin.js:

```
/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}>Display
scores</button>
                    </div>
                    <div>
                    <table className="display--table">
        <thead>
            <tr>
                <th>User</th>
                <th>Score</th>
            </tr>
        </thead>
        <tbody>
            {users.map((user, index) => {
                if (user.username !== "sanjhay") { 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:

```
//Admin.css


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

body {
    overflow: auto;
}

.background-container {
    background-image: url('https://th.bing.com/th/id/OIP.LWxRXkOQv49IBpY1-
b6X6gHaFP?rs=1&pid=ImgDetMain');
```

```
    background-size: cover; background-
    repeat: no-repeat;
    height: 100vh; /* Adjust the height as needed */
}

.Header { height:
    100px;
    background-color: #2C3E50;
    color: white;
    display: flex;
    justify-content: center; align-
    items: center;
}

#admin--button { padding:
    0.5rem 1rem; cursor:
    pointer;
    background-color: #E74C3C;
    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 #3498DB;
    background-color: #1ABC9C;
    text-align: center;
    color: white;
}

.display--table th { background-
    color: #3498DB;
}
```

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 = () => {
  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" }}>Welcome to the Ultimate Quiz {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={() =>
                  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;
```

Quiz.css:

```
//Quiz.css



* {
    font-family: 'Roboto', Arial, sans-serif; /* Maintaining a modern font */
}

h3 {
    margin-left: 20px; /* Subtle adjustment */
    color: #5D6D7E; /* Soft slate gray for a less intense emphasis */
}

.container h3 {
    margin-left: 30px; /* A slight increase for visual hierarchy */
}

.btn {
    display: inline-block;
    padding: 12px 24px; font-
    size: 14px;
    font-weight: 600; /* Retain boldness for readability */ text-align:
    center;
    text-decoration: none;
    cursor: pointer;
    border: 2px solid #3498DB; /* Soft blue border */
```

```
    color: #3498DB; /* Soft blue text */

    background-color: white;

    border-radius: 10px; /* Rounded edges for a friendly interface */ transition: all 0.3s ease; /*

    Smooth transition for interaction */

}


.btn:hover {

    background-color: #3498DB; /* Soft blue becomes the background on hover */ color: white; /*

    White text for contrast */

}


body {

    background-color: #EBF5FB; /* Very light blue for a soothing background */ background-

    image:
url('https://th.bing.com/th/id/OIP.hOpNqoi_9zlzTbrUbqODDQHaD9?w=317&h=180&c=7&r=0&o
=5&dpr=1.3&pid=1.7'); /* Add your image URL here */ background-

    size: cover; /* Cover the entire body */

    background-position: center; /* Center the background image */ background-

    repeat: no-repeat; /* Do not repeat the image */

    background-attachment: fixed; /* Fixed background (doesn't scroll with the content) */

    background: -webkit-linear-gradient(to right, rgba(235, 245, 251, 0.8), rgba(214,
234, 248, 0.8)),
url('https://th.bing.com/th/id/OIP.hOpNqoi_9zlzTbrUbqODDQHaD9?w=317&h=180&c=7&r=0&o
=5&dpr=1.3&pid=1.7'); /* Subtle gradient over image for readability */

    background: linear-gradient(to right, rgba(235, 245, 251, 0.8), rgba(214, 234,
248, 0.8)),
url('https://th.bing.com/th/id/OIP.hOpNqoi_9zlzTbrUbqODDQHaD9?w=317&h=180&c=7&r=0&o
=5&dpr=1.3&pid=1.7'); /* Standard gradient syntax */

}


.container { display:

    flex;

    flex-direction: column; justify-

    content: center;
```

```
    align-items: center;
}

.question--box        {
    padding:     2rem;
    margin: 2rem;
    background-color: #AED6F1; /* Light blue for question box */ display: flex;
    flex-direction: column; justify-content:
    space-evenly; align-items: center;
    border-radius: 10px;
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1); /* Lighter shadow for subtlety */
}

#question {
    color: #154360; /* Deep blue for better readability */ margin-
    bottom: 1rem;
    font-size: 1.25rem; /* Enhanced font size */
}

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

.option {
    background-color: #FFFFFF; /* White background for clean look */ border: 2px
    solid #7FB3D5; /* Lighter blue border */
    border-radius: 10px; margin:
    0.5rem;
```

```
    padding: 0.75rem; display:
    flex;
    justify-content: space-between; cursor:
    pointer;
    transition: all 0.3s ease; /* Consistent transitions */
}

.option:hover {
    background-color: #7FB3D5; /* Light blue on hover */ color:
    #FFFFFF; /* Maintain white text for clarity */
}

.option input[type="radio"] {
    margin-right: 10px; /* Proper separation */
}
```

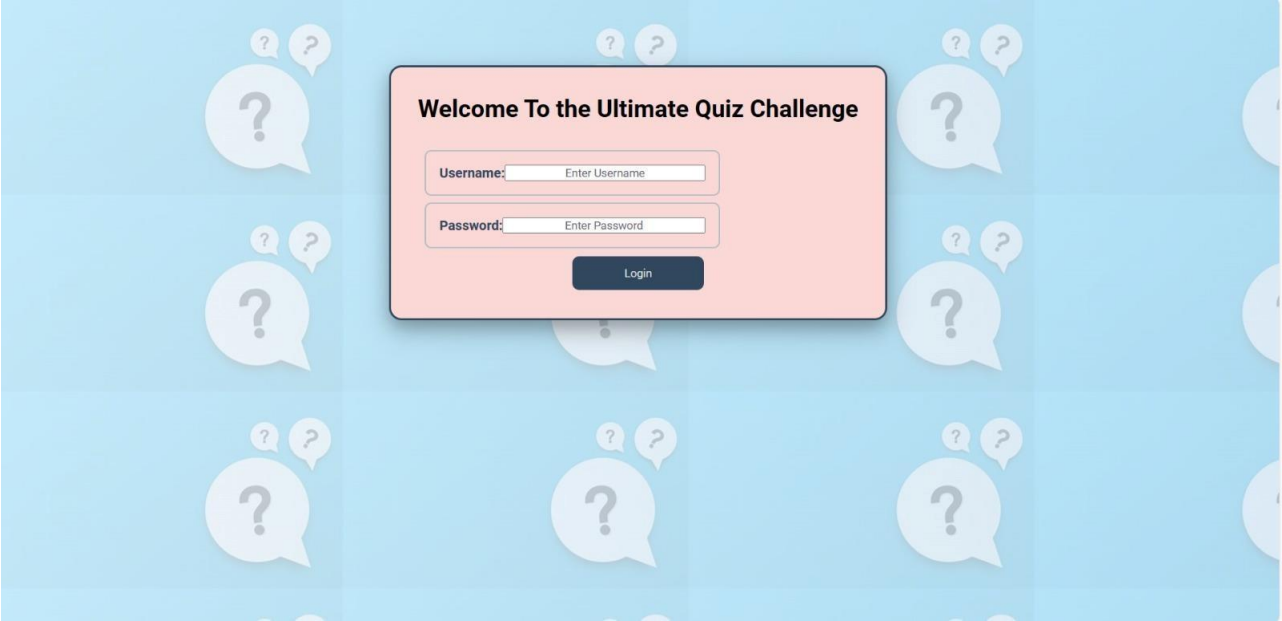
LoginModel.js:

```
//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;
```

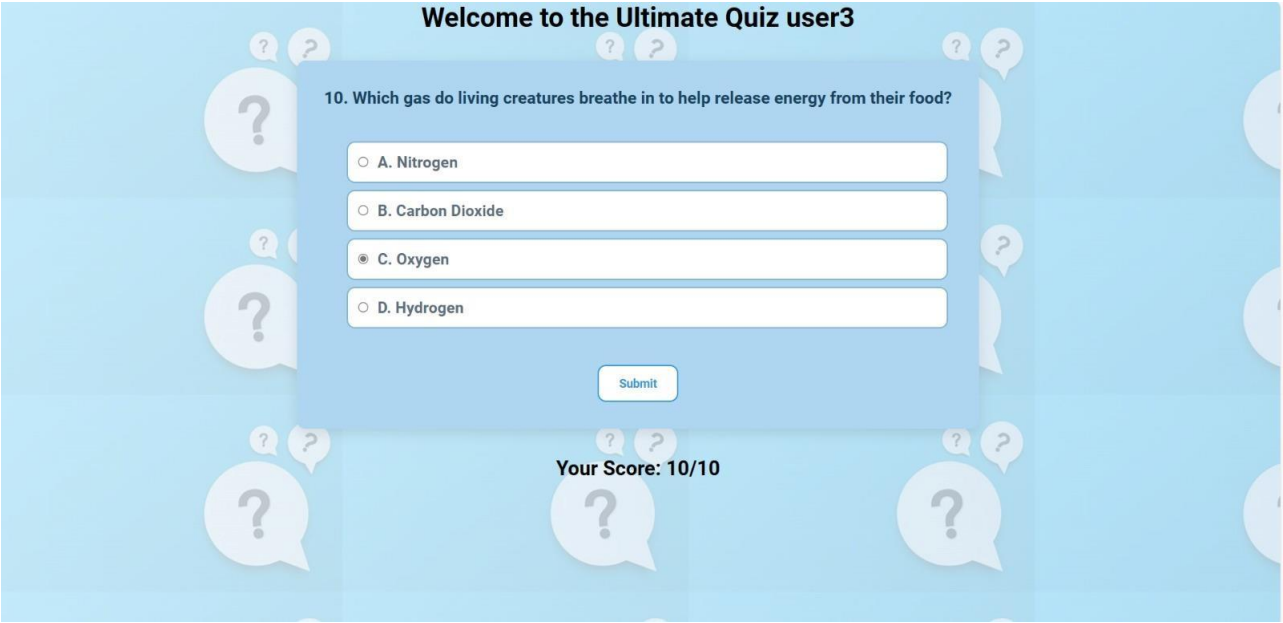
QuizModel.js:

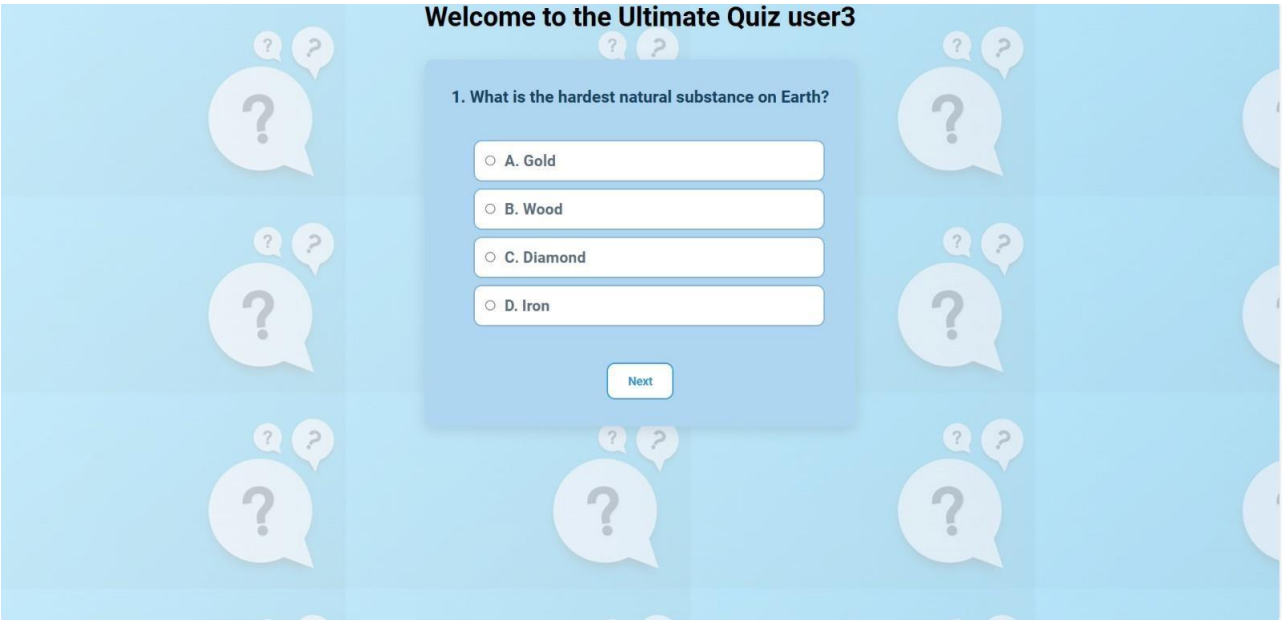
```
//Quizmodel.js

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);
module.exports= userModel;
```

Output:

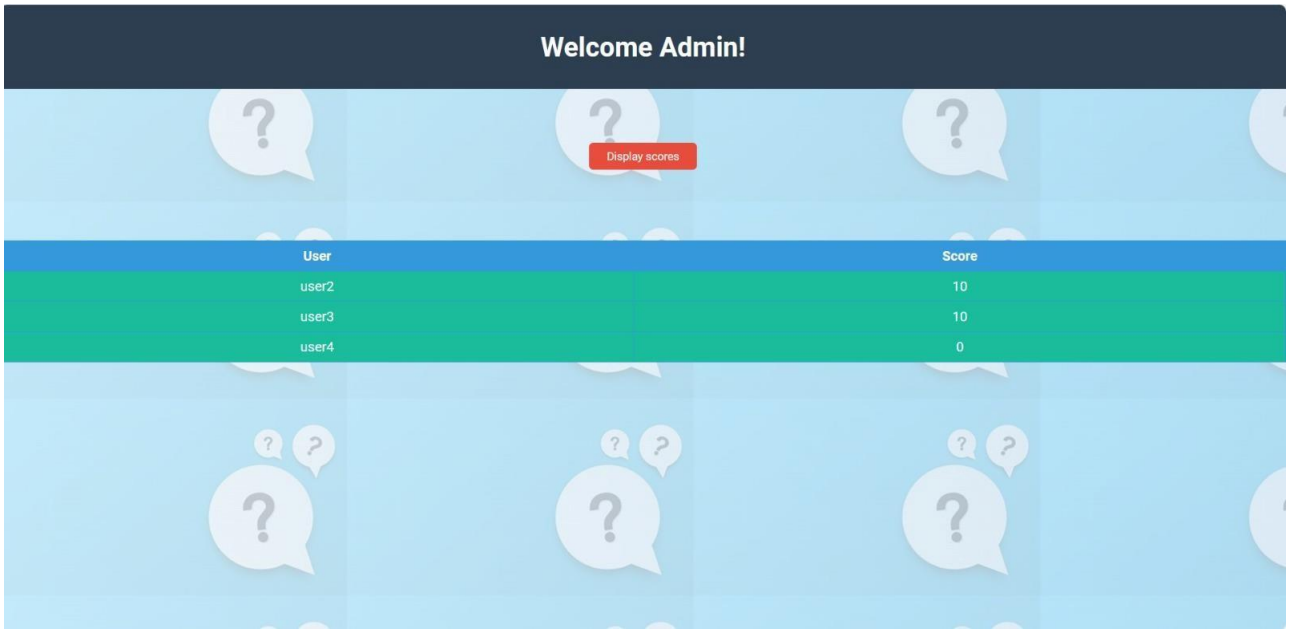


Login page


Welcome to the Ultimate Quiz user3
10. Which gas do living creatures breathe in to help release energy from their food?
A. Nitrogen
B. Carbon Dioxide
C. Oxygen
D. Hydrogen
Submit
Your Score: 10/10


Welcome to the Ultimate Quiz user3
1. What is the hardest natural substance on Earth?
A. Gold
B. Wood
C. Diamond
D. Iron
Next

Quiz Page



Admin Page for displaying scores

```
_id: ObjectId('663f7b3397e91aec375b0d86')
username : "sanjhay"
password : "sanjhay"
score : 0

_id: ObjectId('663f7b4c97e91aec375b0d87')
username : "user2"
password : "pass2"
score : 10

_id: ObjectId('663f7b7197e91aec375b0d88')
username : "user3"
password : "pass3"
score : 10

_id: ObjectId('663f7b8297e91aec375b0d89')
username : "user4"
password : "pass4"
score : 0
```

MongoDB – User and admin details

```
_id: ObjectId('663f7bbb97e91aec375b0d8b')
question : "What is the capital of France?"
options : Array (4)
answer : "Paris"

_id: ObjectId('663f7bc197e91aec375b0d8c')
question : "Which gas do living creatures breathe in to help release energy from t…"
options : Array (4)
answer : "Oxygen"

_id: ObjectId('663f7bce97e91aec375b0d8d')
question : "What is the hardest natural substance on Earth?"
options : Array (4)
answer : "Diamond"

_id: ObjectId('663f7bdc97e91aec375b0d8e')
question : "What planet is known as the 'Red Planet'?"
options : Array (4)
answer : "Mars"

_id: ObjectId('663f7bea97e91aec375b0d8f')
question : "Who painted the Mona Lisa?"
```


MongoDB – Questions storage