Project Title: Quiz Application

(Node.js + MongoDB + Express + HTML/CSS/JS)

Introduction

The Quiz Application is a full-stack web platform that enables users to participate in multiple-choice quizzes, track their scores, and interact with a dynamic frontend UI. Built with Node.js and Express.js on the backend, the application leverages MongoDB (via Mongoose) to store quizzes, user data, and scores persistently. The frontend is styled using responsive HTML5, CSS3, and vanilla JavaScript.

This application serves as a learning and testing tool, showcasing full-cycle web development—from backend API creation to frontend interaction and deployment-ready structure. The project demonstrates CRUD operations for quiz content, RESTful design, and modular Node.js architecture.

Key Features & Demonstrated Skills

- RESTful API development for quizzes and results using Express.js.
- MongoDB integration via Mongoose for dynamic data handling.
- Interactive quiz-taking interface using JavaScript.
- Use of EJS or static HTML templates for frontend rendering.
- Environment-based configuration management.
- Score computation and session tracking.

Objectives

- Develop a responsive web-based quiz platform.
- Enable users to attempt quizzes and view their scores.
- Maintain a clean, modular backend with Express and Mongoose.
- Allow admin CRUD functionalities for quiz management.
- Prepare the project for cloud deployment and real-world usage.

Technologies Used

Category	Technologies
Frontend	HTML5, CSS3, JavaScript (Vanilla JS)
Backend	Node.js, Express.js
Database	MongoDB

VARSHA AGARWAL 2241001035

Project Structure

```
quiz-app-full/
├── backend/
│   ├── models/
│   │   ├── Question.js
│   │   └── User.js
│   ├── node_modules/
│   ├── index.js
│   ├── package-lock.json
│   ├── package.json
│   └── seed.js
├── frontend/
│   ├── index.html
│   ├── script.js
│   └── style.css
```

CODES

- **BACKEND**

Question.js

```
const mongoose = require('mongoose');
const questionSchema = new mongoose.Schema({
 topic: String,
 question: String,
 options: [String],
 correctOption: Number,
});
module.exports = mongoose.model("Question", questionSchema);
```

User.js

```
const mongoose = require('mongoose');
const userSchema = new mongoose.Schema({
 email: String,
 password: String
});
module.exports = mongoose.model("User", userSchema);
```

index.js

```
const express = require('express');
const cors = require('cors');
const mongoose = require('mongoose');
const jwt = require('jsonwebtoken');
const User = require('./models/User');
const Question = require('./models/Question');
const app = express();
app.use(cors());
app.use(express.json());
mongoose.connect('mongodb://localhost:27017/quizdb', {
 useNewUrlParser: true,
 useUnifiedTopology: true,
})
.then(() => console.log("Connected to MongoDB"))
.catch((err) => console.error("MongoDB connection error:", err));
app.post("/api/login", async (req, res) => {
 const { email, password } = req.body;
 const user = await User.findOne({ email, password });
 if (user) {
  const token = jwt.sign({ email }, "secret", { expiresIn: "1h" });
  res.json({ token });
 } else {
  res.status(401).json({ message: "Invalid credentials" });
 }
});
app.get("/api/quiz/:topic", async (req, res) => {
 const topic = req.params.topic;
 const auth = req.headers.authorization?.split(" ")[1];
 if (!auth) return res.status(401).json({ message: "No token provided" });
 try {
  jwt.verify(auth, "secret");
  const questions = await Question.find({ topic }).limit(10);
  res.json({ questions });
 } catch (e) {
  res.status(403).json({ message: "Invalid token" });
 }
});
app.listen(5000, () => console.log("Server started on http://localhost:5000"));
```

seed.js

```
const mongoose = require("mongoose");
const Question = require("./models/Question");
mongoose.connect("mongodb://localhost:27017/quiz-app", {
```

```
  useNewUrlParser: true,
  useUnifiedTopology: true,
});
const questions = [
 // Math Questions (30)
 {
   "topic": "Math",
   "question": "What is the value of $\sqrt{144}$?",
   "options": [
    "10",
    "12",
    "14",
    "16"
   ],
.....
 // Science Questions (30)
 {
   "topic": "Science",
   "question": "What is the chemical symbol for oxygen?",
   "options": [
    "Ox",
    "O",
    "Og",
    "Oy"
   ],
   "correctOption": 1
 },
.....
// History Questions (30)
 {
   "topic": "History",
   "question": "In what year did Christopher Columbus first arrive in the Americas?",
   "options": [
    "1488",
    "1492",
    "1501",
    "1507"
   ],
   "correctOption": 1
 },
.....
 // Geography Questions (30)
 {
   "topic": "Geography",
   "question": "What is the capital city of Japan?",
   "options": [
    "Seoul",
```

```
    "Beijing",
    "Tokyo",
    "Bangkok"
   ],
   "correctOption": 2
 },
.....
// Sports Questions (30)
 {
   "topic": "Sports",
   "question": "How many players are there in a standard soccer team on the field?",
   "options": [
    "9",
    "10",
    "11",
    "12"
   ],
   "correctOption": 2
 },
.....
// Technology Questions (30)
 {
   "topic": "Technology",
   "question": "What does CPU stand for in computer terminology?",
   "options": [
    "Central Processing Unit",
    "Computer Programming Unit",
    "Central Program Utility",
    "Common Protocol Unit"
   ],
   "correctOption": 0
 },
.....
{
   "topic": "Technology",
   "question": "What is a podcast?",
   "options": [
    "A live video stream",
    "An episodic series of digital audio or video files",
    "A type of social media post",
    "An online encyclopedia"
   ],
   "correctOption": 1
 }
];

Question.insertMany(questions)
```

```
.then(() => {
  console.log("Questions inserted successfully");
  mongoose.disconnect();
})
.catch((err) => {
  console.error("Error inserting questions:", err);
  mongoose.disconnect();
});
```

- **FRONTEND**

  index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Quiz App</title>
  <link rel="stylesheet" href="style.css" />
</head>
<body>
  <div id="app"></div>
  <script src="script.js"></script>
</body>
</html>
```

  Script.js

```
document.addEventListener("DOMContentLoaded", () => {
  const app = document.getElementById("app");
  if (!app) {
    console.error("Missing <div id='app'> in index.html");
    return;
  }
  renderLogin();
  function renderLogin() {
    app.innerHTML = `
      <div class="container">
        <h2>Login</h2>
        <form id="loginForm">
          <input type="email" id="email" placeholder="Email" required />
          <input type="password" id="password" placeholder="Password" required />
          <button type="submit">Login</button>
```

```
      </form>
      <p id="loginMessage"></p>
    </div>
  `;
  document.getElementById("loginForm").addEventListener("submit", async (e)
=> {
    e.preventDefault();
    const email = document.getElementById("email").value;
    const password = document.getElementById("password").value;
    try {
      const res = await fetch("http://localhost:5000/api/login", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ email, password }),
      });
      const data = await res.json();
      if (res.ok) {
        localStorage.setItem("token", data.token);
        renderTopicSelection();
      } else {
        document.getElementById("loginMessage").textContent =
          data.message || "Login failed";
      }
    } catch (error) {
      document.getElementById("loginMessage").textContent =
        "Server error. Please try again.";
    }
  });
}

function renderTopicSelection() {
  const topics = [
    "Math",
    "Science",
    "History",
    "Geography",
    "Sports",
    "Technology",
  ];
  app.innerHTML = `
    <div class="container">
      <h2>Select a Topic</h2>
      <div class="topics">
        ${topics
          .map(
            (topic) =>
              `<button class="topic-btn" data-topic="${topic}">${topic}</button>`
```

```
        )
        .join("")}
    </div>
   </div>
  `;

  document.querySelectorAll(".topic-btn").forEach((btn) => {
   btn.addEventListener("click", () => {
    const selectedTopic = btn.getAttribute("data-topic");
    startQuiz(selectedTopic);
   });
  });
}

async function startQuiz(topic) {
  const token = localStorage.getItem("token");
  try {
   const res = await fetch(` http://localhost:5000/api/quiz/${topic}` , {
    headers: { Authorization: "Bearer " + token },
   });
   const data = await res.json();
   renderQuiz(data.questions, topic);
  } catch (err) {
   app.innerHTML =
    "<div class='container'><p>Error fetching questions.</p></div>";
  }
}

function renderQuiz(questions, topic) {
  let current = 0;
  let score = 0;

  function showQuestion() {
   if (current >= questions.length) {
    app.innerHTML = `
     <div class="container">
      <h2>Quiz Finished</h2>
      <p>Score: ${score} / ${questions.length}</p>
     </div>
    `;
    return;
   }

   const q = questions[current];
   app.innerHTML = `
    <div class="container">
     <h2>${topic} - Question ${current + 1}</h2>
```

```
        <p>${q.question}</p>
        ${q.options
          .map(
            (opt, i) =>
              `<button class="opt" data-idx="${i}">${opt}</button>`
          )
          .join("")}
      </div>
    `;

    document.querySelectorAll(".opt").forEach((btn) => {
      btn.addEventListener("click", () => {
        const selected = Number(btn.getAttribute("data-idx"));
        if (selected === q.correctOption) score++;
        current++;
        showQuestion();
      });
    });
  }

  showQuestion();
 }
});
```

Style.css

```
body { font-family: Arial, sans-serif; padding: 20px; background: #f9f9f9; }
.container { max-width: 500px; margin: auto; background: white; padding: 20px;
border-radius: 8px; }
input, button { display: block; width: 100%; margin: 10px 0; padding: 10px; }
.topics button { margin: 5px; }
```

EXECUTION STEPS

Step 1 : Start MongoDB
Step 2 : In Terminal,
>cd backend
>npm install
>node seed.js
>node index.js
Step 3 : Open index.html in any browser

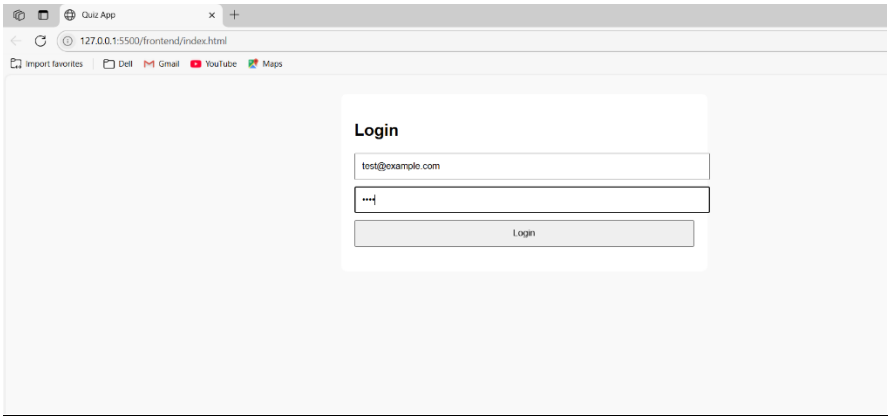
OUTPUTS:

- Terminal

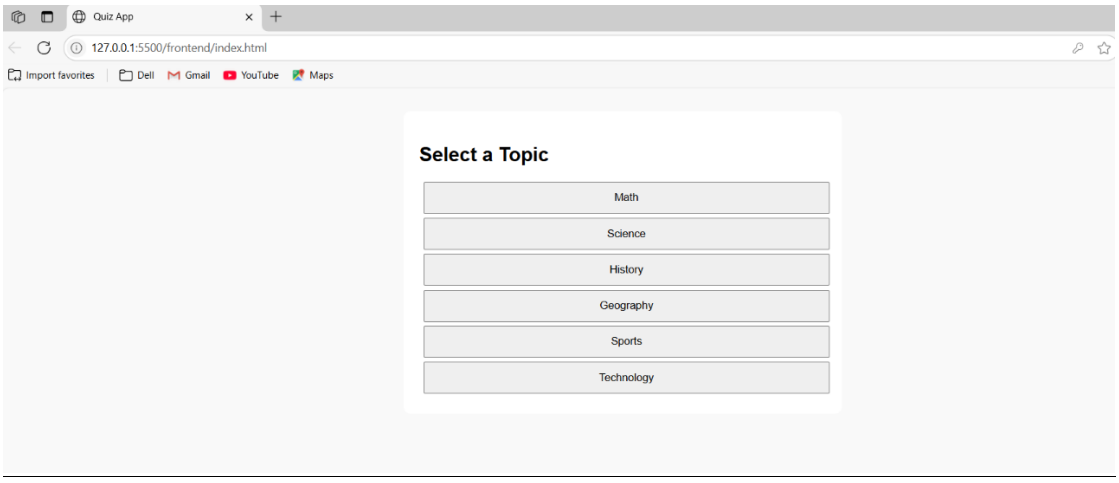
```
D:\3rd year\3rd year\6th sem\SSWD\Lab\quiz-app-full\backend>node seed.js
(node:26316) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next ma
jor version
(Use `node --trace-warnings ...` to show where the warning was created)
(node:26316) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the n
ext major version
(node:26316) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next ma
jor version
(Use `node --trace-warnings ...` to show where the warning was created)
(node:26316) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the n
ext major version
ext major version
Questions inserted successfully

D:\3rd year\3rd year\6th sem\SSWD\Lab\quiz-app-full\backend>node index.js
(node:5412) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next maj
or version
(Use `node --trace-warnings ...` to show where the warning was created)
(node:5412) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the ne
xt major version
Server started on http://localhost:5000
Connected to MongoDB
```

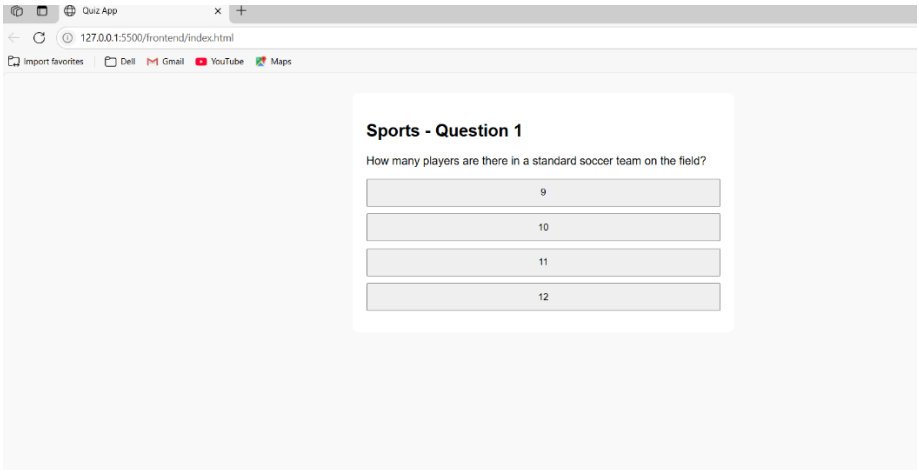
- Login Page



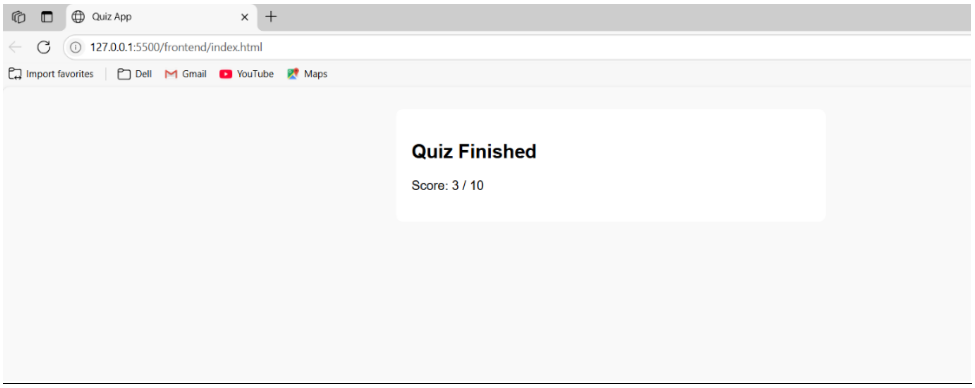
- Topic Selection



➢ Questions



➢ Score



➢ MongoDB View