**A PROJECT REPORT**

**As partial requirement for the degree of**

**Master Of Science (Computer Application)**

**ON**

**Quiz**

**Group No: \_\_\_\_\_\_\_\_**

**SUBMITTED BY : University Exam No.**

**Shrusti B Patel 113**

**UNDER GUIDANCE OF**

**Dr. Nirav Desai**



**Acknowledgement**

I am student of DOLAT-USHA INSTITUTE OF APPLIED SCIENCES AND DHIRUSARLA INSTITUTE OF MANAGEMENT & COMMERCE, VALSAD College of Computer application have worked on the project entitled Restaurant Table Reservation System.

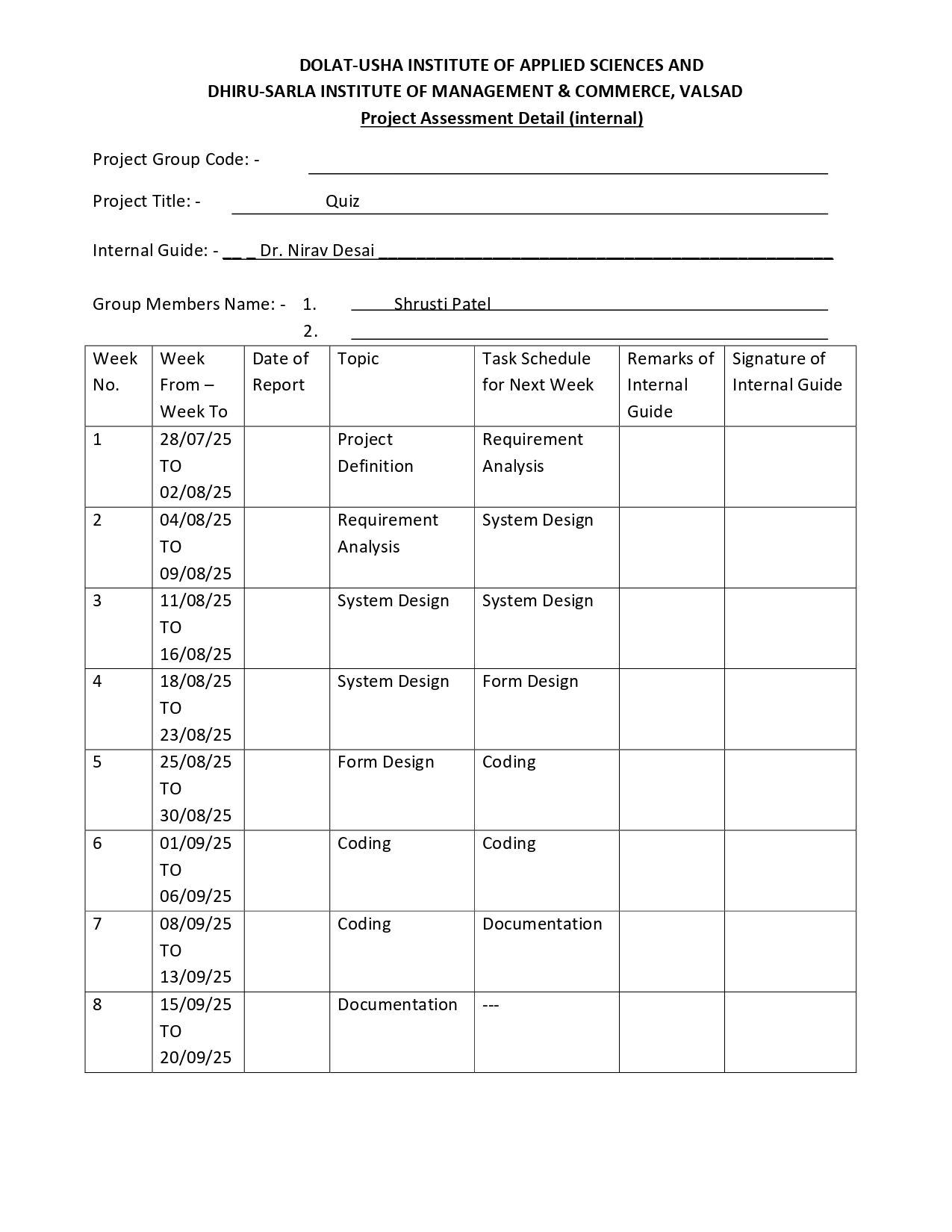
My guide Prof. Dhruvisha Tandel & Dr.Akruti Naik , for imparting me valuable guidance during the preparation of this projects. She helped me by suggesting many references to emerge out with a perfect presentation. Her expertise in the subject matter greatly contributed to the depth and quality of the project.

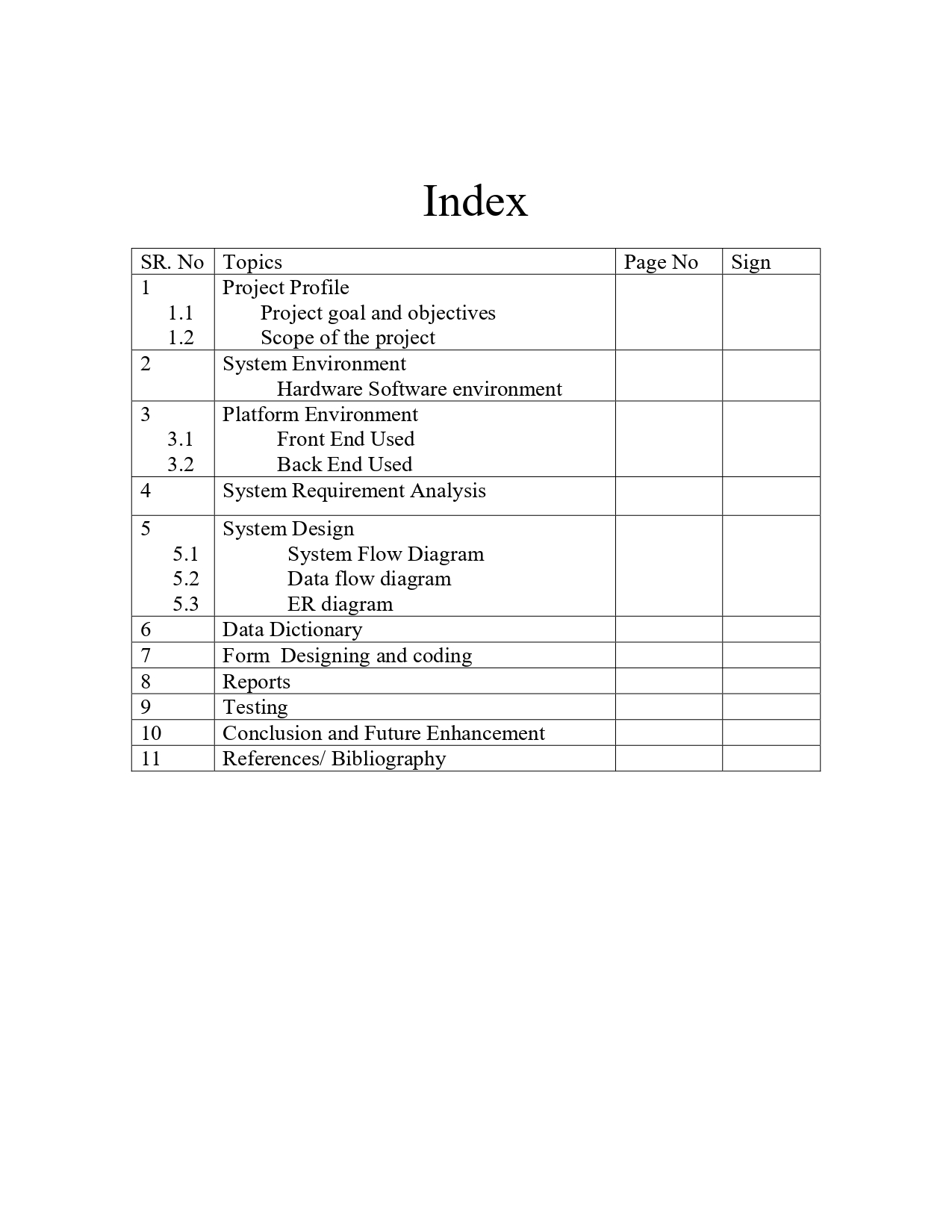
I would like to thank the people who helped me in completion of this project and with content and I express my gratitude towards those people who have given me their precious time and guidance in development of this project.

I am sincerely thankful to all other faculty members who helped me and gave the opportunity and support to take this project respectively.

Thanking Sincerely,

Patel Shrushti Bachubhai

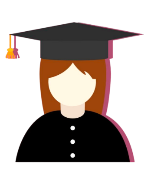
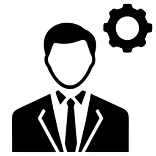




**Introduction**

* **Welcome to the Quiz Application.**
* This application is a simple, efficient, and user-friendly platform for conducting college quizzes, designed to streamline the assessment process for both students and educators. It eliminates the need for extensive manual preparation, grading, and record-keeping, significantly reducing administrative workload. Students can attempt quizzes within a set time limit, creating an engaging and challenging learning experience that encourages quick thinking, focus, and improved knowledge retention.
* The platform supports a variety of question formats, including multiple-choice, true/false, fill-in-the-blanks, and short answer questions, catering to diverse evaluation needs. The time-bound nature of quizzes ensures that students remain attentive and learn to manage time effectively during assessments. Teachers benefit from automated grading and instant generation of detailed performance reports, enabling them to identify areas of improvement and track student progress over time.
* Real-time score tracking and analytics enhance fairness and transparency in evaluation, while instant results provide immediate feedback to students, reinforcing learning outcomes and boosting motivation. The system also supports features like randomized question banks, which minimize cheating and ensure unique quiz experiences for each student.
* Furthermore, the application encourages active participation and healthy competition through leaderboards, badges, and achievement tracking. Administrators can monitor overall performance, generate reports, and manage users efficiently. The platform can be integrated with existing learning management systems (LMS) or used independently, making it a versatile tool for modern education.
* Overall, this application creates a highly effective, interactive, and adaptive learning environment, benefiting students, teachers, and administrators alike by promoting continuous learning, engagement, and academic excellence.

**Objectives**

**Student Teacher Admin**

* **Student Registration Teacher Login Admin Login**
* **Student login Question management Department management**
* **Department selection Profile management Subject management**
* **Subject selection View student's details Teacher management**
* **Start/end quiz View students results Student management**
* **View result Report generation**

**Scope of the Quiz Application**

* The Quiz Application is designed specifically for **students and staff members of the college**, providing a streamlined and efficient platform for conducting quizzes.
* Each quiz contains **15 questions**, structured to evaluate students’ knowledge effectively.
* A **time limit** is imposed for each quiz session, ensuring focused participation and proper time management.
* If the time expires, the quiz is **automatically submitted** with the answers entered so far, maintaining fairness for all participants.
* Students can **view their results instantly** after submission, allowing immediate feedback and self-assessment.
* The system ensures **fast, accurate, and unbiased evaluation**, eliminating human errors in grading.
* Supports multiple **question formats**, including **multiple-choice, true/false, and fill-in-the-blank**, for versatile and comprehensive assessment.
* Teachers can **monitor individual and class performance**, identifying students’ strengths and areas needing improvement.
* The application provides **detailed analytics and performance reports** for both students and classes, aiding in optimizing teaching strategies.
* Features such as **real-time scoring, leaderboards, and performance summaries** encourage **active participation and healthy competition** among students.
* Staff members save significant **time and effort**, as the system automates quiz scheduling, grading, and result generation.
* Administrators can **manage user accounts, monitor system usage, and ensure smooth operation** of the platform.
* The platform promotes a **transparent and organized assessment process**, ensuring fairness and clarity for all users.
* By reducing manual workload and providing **instant feedback**, the application fosters a **more engaging and effective learning environment**.
* The system is **scalable and adaptable**, allowing future integration with other college systems or the addition of new features as needed.

Module Description

* Module Description (Admin)

1 Admin Login – Provides authenticated access with username and password.

2 Department Management – Enables adding departments like MSc CA, MCA, MBA, MCom, MA.

3 Subject Management – Enables administrators to add subjects.

4 Teacher Management – Enables administrators to modify or delete teacher accounts.

5 Student Management – Enables administrators to modify or delete student accounts.

6 Report Generation – Provides daily counts of student logins and results.

* **Module Description (Teacher)**

1 Teacher Login – Provides secure access to teacher-specific functions.

2 Question Management – Allows teachers to create, modify, and delete quiz questions.

3 Profile Management – Enables teachers to customise their profile.

4 View Students Details – Allows teachers to view information of all students.

5 View Students Results – Enables teachers to view results of all students.

Module Description (Student)

1 Student Registration – Add details to register for quiz.

2 Student Login – Access quiz with username and password.

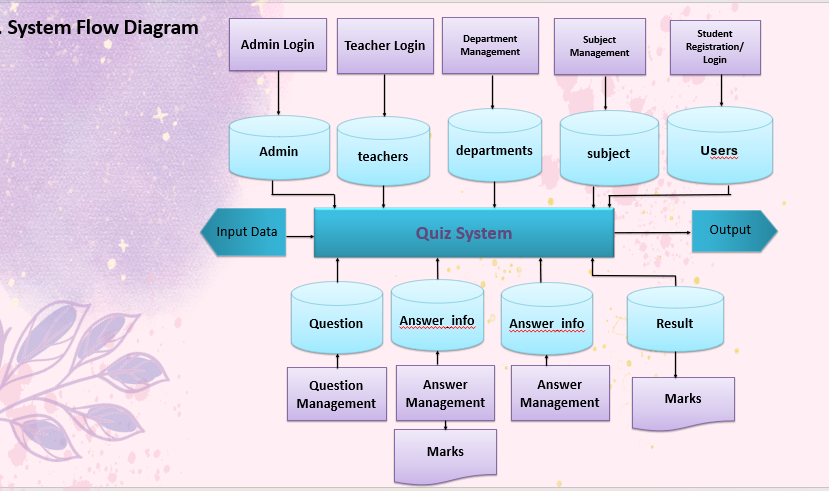
3 Department Selection – Allows students to select department.

4 Subject Selection – Allows students to select subject.

5 Start/End Test – Students can start their test and end within a time limit.

6 View Result – Allows students to view their quiz results.

System Flow Diagram :-

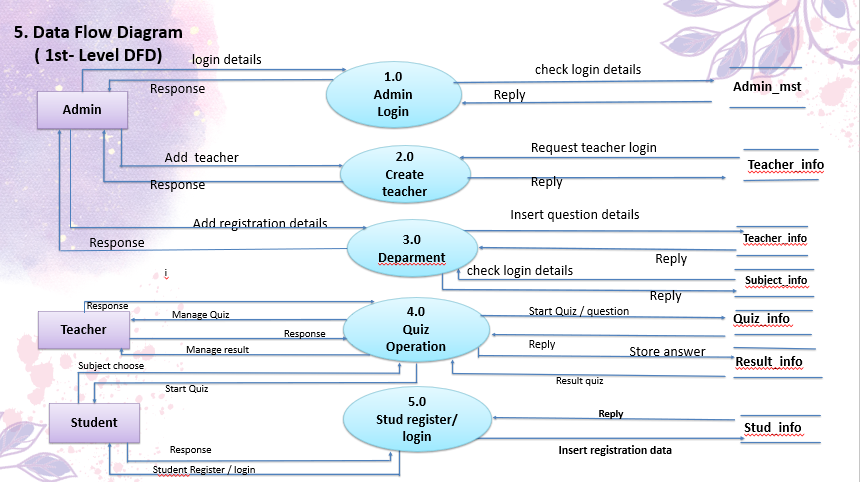


**DFD (Data Flow Diagram).**

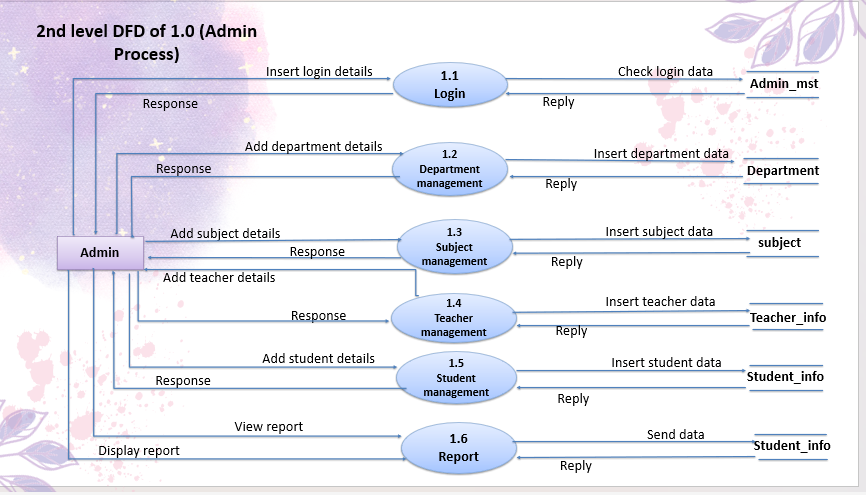
**0 Level DFD**



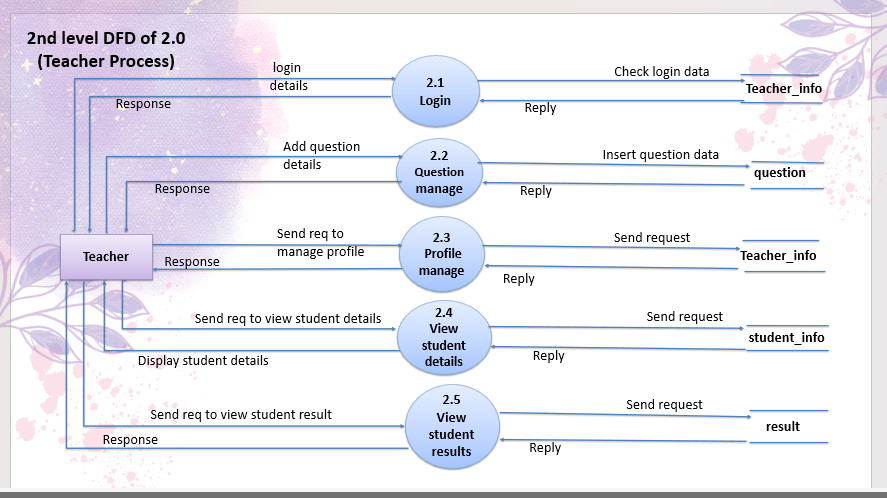
**1st Level DFD**



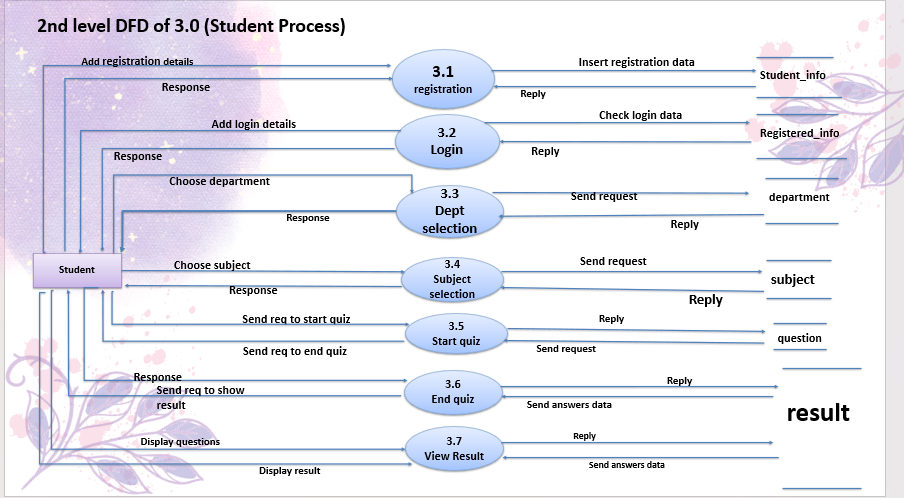
**2nd Level DFD ADMIN**



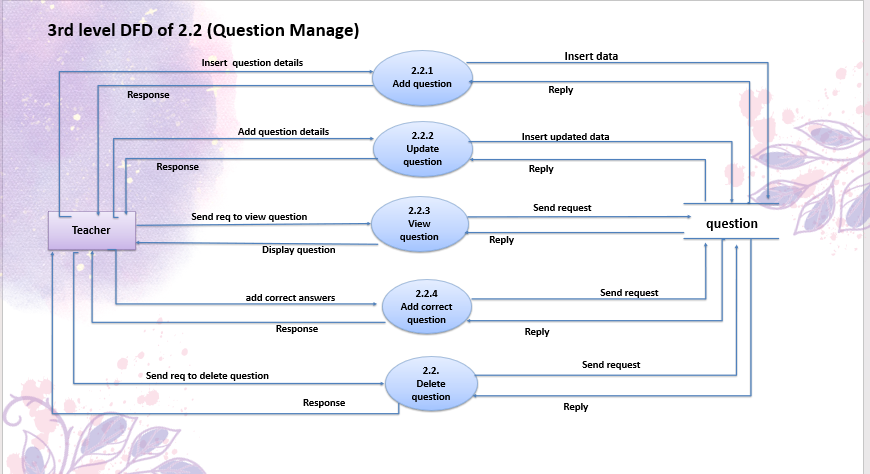
**2nd Level DFD Teacher**



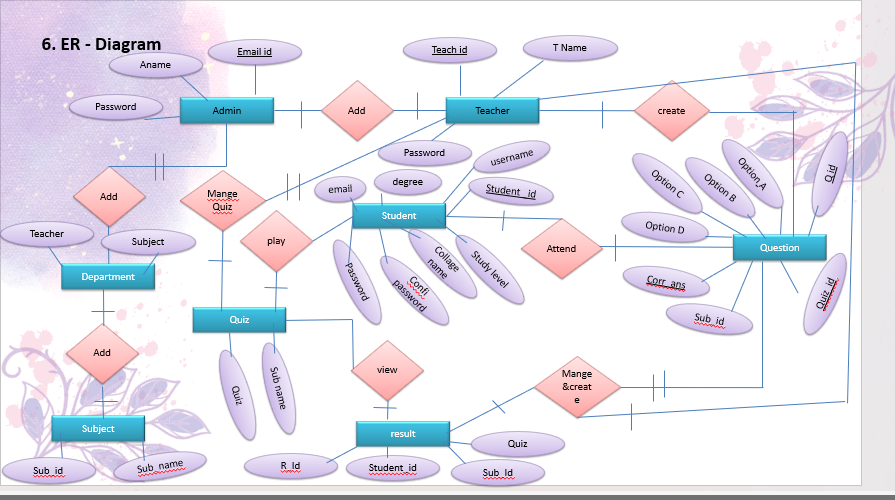
**2nd Level DFD Student**



**2nd Level DFD Student**



ER -Diagram



**Data Dictionary:-**

**Admin\_Mst:-**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraints** | **Description** |
| **admin\_name** | VARCHAR(50) | NOT NULL | The full name of the admin |
| **email\_id** | VARCHAR(100) | NOT NULL, UNIQUE | The admin’s email address (used for login) |
| **password** | VARCHAR(255) | NOT NULL | The password for admin login (hashed) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraints** | **Description** |
| **teacher\_id** | INT | PRIMARY KEY, NOT NULL | Unique identifier for each teacher |
| **sub\_name** | VARCHAR(50) | NOT NULL | Name of the subject the teacher handles |
| **teacher\_name** | VARCHAR(50) | NOT NULL | Full name of the teacher |
| **email\_id** | VARCHAR(100) | NOT NULL, UNIQUE | Teacher’s email address, used for login |

**Teacher:-**

**Deparement :-**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraints** | **Description** |
| **dept\_id** | INT | PRIMARY KEY, NOT NULL | Unique identifier for each department |
| **sub\_id** | INT | NOT NULL, FOREIGN KEY → Subject(sub\_id) | References the subject handled in the department |
| **teacher\_id** | INT | NOT NULL, FOREIGN KEY → Teacher(teacher\_id) | References the teacher assigned to the department |

**Subject:-**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraints** | **Description** |
| **sub\_id** | INT | PRIMARY KEY, NOT NULL | Unique identifier for each subject |
| **sub\_name** | VARCHAR(50) | NOT NULL | Name of the subject |
| **department\_id** | INT | NOT NULL, FOREIGN KEY → Department(dept\_id) | References the department to which the subject belongs |
| **teacher\_id** | INT | NOT NULL, FOREIGN KEY → Teacher(teacher\_id) | References the teacher assigned to this subject |
| **Field Name** | Data Type | Constraints | Description |

**Student**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint** | **Description** |
| **student\_id** | VARCHAR(10) | PK, Unique, Not Null, Auto-gen | Auto ID = 3 letters of username + 3 digits |
| **username** | VARCHAR(50) | Unique, Not Null | Student login name |
| **email** | VARCHAR(100) | Unique, Not Null | Student email ID |
| **study\_level** | ENUM | Not Null | Bachelor / Postgraduate (radio) |
| **college\_name** | VARCHAR(100) | Not Null | College name |
| **password** | VARCHAR(255) | Not Null | Login password (encrypted) |
| **confirm\_password** | VARCHAR(255) | Not Null (Form only) | Confirm password match (not stored) |

**Question:-**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD\_NAMES** | **Data\_Type** | **CONSTRAIN** | **DESCRIPTION** |
| **Q\_Id** | Integer | Primary Key, Auto Increment | Number |
| **Questions** | Text | Not null | Questions |
| **Option\_A** | Text | Not null | A |
| **Option\_B** | Text | Not null | B |
| **Option\_C** | Text | Not null | C |
| **Option\_D** | Text | Not null | D |
| **Corr\_Ans** | Text | Not null | Correct Answer |
| **Quiz\_id** | Integer | Foreign key | Quiz Id |
| **sub\_Id** | Integer | Foreign key | Subcourse Id |

**Result:-**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD\_NAMES** | **Data\_Type** | **CONSTRAIN** | **DESCRIPTION** |
| **R\_id** | Integer | Primary Key, Auto Increment | Result Id |
| **S\_id** | Integer | Not null | Student Id |
| **Final\_Marks** | Integer |  | Final Marks |
| **Exam\_Time** | TIMESTAM | Not null | Exam Time |
| **Quiz\_id** | Integer | Foreign key | Quiz Id |
| **Department\_id** | Integer | Foreign key | Department Id |

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD\_NAMES** | **Data\_Type** | **CONSTRAIN** | **DESCRIPTION** |
| **Quiz\_id** | Integer | Primary Key,Auto Increment | Quiz Id |
| **Quiz\_name** | Integer | Not Null | Quiz Name |

**Quiz:-**

**Coding**

**Admin**

**App.js**

import React from "react";

import { BrowserRouter as Router, Routes, Route } from "react-router-dom";

import AdminLogin from "./components/AdminLogin";

import AdminDashboard from "./components/AdminDashboard";

import DepartmentManagement from "./components/DepartmentManagement";

import SubjectManagement from "./components/SubjectManagement";

import TeacherManagement from "./components/TeacherManagement";

import ManageQuizzes from "./components/ManageQuizzes";

import ManageResults from "./components/ManageResults";

import ManageUsers from "./components/ManageUsers";

function App() {

  return (

    <Router>

      <Routes>

        {*/\* Login \*/*}

        <Route path="/" element={<AdminLogin />} />

        {*/\* Dashboard \*/*}

        <Route path="/admin-dashboard" element={<AdminDashboard />} />

        {*/\* Management Pages \*/*}

        <Route path="/departments" element={<DepartmentManagement />} />

        <Route path="/subjects" element={<SubjectManagement />} />

        <Route path="/teachers" element={<TeacherManagement />} />

        {*/\* Manage Pages \*/*}

        <Route path="/manage-quizzes" element={<ManageQuizzes />} />

        <Route path="/manage-results" element={<ManageResults />} />

        <Route path="/manage-users" element={<ManageUsers />} />

      </Routes>

    </Router>

  );

}

export default App;

**Admin Login.js**

import React, { useState } from "react";

import { useNavigate } from "react-router-dom";

import \* as Yup from "yup";

import { ToastContainer, toast } from "react-toastify";

import "react-toastify/dist/ReactToastify.css";

const AdminLogin = () => {

  const navigate = useNavigate();

  const [email, setEmail] = useState("");

  const [password, setPassword] = useState("");

  const schema = Yup.object().shape({

    email: Yup.string().email("Invalid Email").required("Email is required"),

    password: Yup.string().required("Password is required"),

  });

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

      await schema.validate({ email, password }, { abortEarly: false });

      const res = await fetch("http://localhost:5000/admin/login", {

        method: "POST",

        headers: { "Content-Type": "application/json" },

        body: JSON.stringify({ username: email, password }),

      });

      const data = await res.json();

      if (res.ok) {

        toast.success("✅ Login Successful!", { position: "top-center" });

        localStorage.setItem("adminToken", "loggedin");

        setTimeout(() => navigate("/admin-dashboard"), 1500);

      } else {

        toast.error(data.message || "Login failed", { position: "top-center" });

      }

    } catch (err) {

      if (err.name === "ValidationError") {

        toast.error(err.errors.join("\n"), { position: "top-center" });

      } else {

        toast.error("Server error. Try again later.", { position: "top-center" });

      }

    }

  };

  return (

    <div className="login-container">

      <ToastContainer />

      <div className="login-box">

        <h2 className="login-title">Admin Portal</h2>

        <form onSubmit={handleSubmit} className="login-form">

          <label>Email</label>

          <input

            type="email"

            placeholder="Enter Email"

            value={email}

            onChange={(e) => setEmail(e.target.value)}

          />

          <label>Password</label>

          <input

            type="password"

            placeholder="Enter Password"

            value={password}

            onChange={(e) => setPassword(e.target.value)}

          />

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

            Login

          </button>

        </form>

      </div>

      <style>{`

        body {

          margin: 0;

          font-family: 'Poppins', sans-serif;

        }

        .login-container {

          display: flex;

          justify-content: center;

          align-items: center;

          height: 100vh;

          background: linear-gradient(135deg, #000000 0%, #1b1b1b 40%, #2b2b2b 100%);

        }

        .login-box {

          background: rgba(20, 20, 20, 0.95);

          padding: 50px 40px;

          border-radius: 20px;

          box-shadow: 0 0 25px rgba(255, 255, 255, 0.08);

          width: 360px;

          text-align: center;

          border: 1px solid rgba(255, 255, 255, 0.1);

        }

        .login-title {

          font-size: 28px;

          font-weight: 700;

          color: #00e0ff;

          margin-bottom: 25px;

          text-shadow: 0 0 10px rgba(0, 224, 255, 0.5);

        }

        .login-form {

          display: flex;

          flex-direction: column;

        }

        .login-form label {

          text-align: left;

          color: #aaa;

          margin-bottom: 8px;

          font-size: 14px;

        }

        .login-form input {

          background: #111;

          color: #fff;

          border: 1px solid #333;

          padding: 12px;

          border-radius: 8px;

          margin-bottom: 18px;

          font-size: 14px;

          transition: 0.3s;

        }

        .login-form input:focus {

          outline: none;

          border-color: #00e0ff;

          box-shadow: 0 0 8px rgba(0, 224, 255, 0.5);

        }

        .login-btn {

          background: linear-gradient(135deg, #00e0ff, #0078ff);

          color: #fff;

          padding: 12px;

          border: none;

          border-radius: 8px;

          cursor: pointer;

          font-size: 16px;

          font-weight: 600;

          box-shadow: 0 0 15px rgba(0, 224, 255, 0.3);

          transition: all 0.3s ease;

        }

        .login-btn:hover {

          transform: translateY(-2px);

          box-shadow: 0 0 20px rgba(0, 224, 255, 0.5);

        }

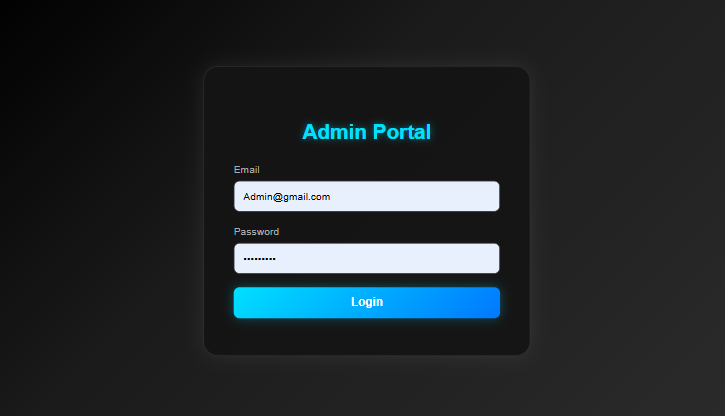
      `}</style>

    </div>

  );

};

export default AdminLogin;

****

**Admin Dashborad.js**

import React, { useState } from "react";

import { useNavigate } from "react-router-dom";

import {

  PieChart,

  Pie,

  Cell,

  ResponsiveContainer,

  BarChart,

  Bar,

  XAxis,

  YAxis,

  Tooltip,

  CartesianGrid,

} from "recharts";

const AdminDashboard = () => {

  const navigate = useNavigate();

  const [menuVisible, setMenuVisible] = useState(false);

  const stats = [

    { label: "Students", value: 120, color: "#00bfff" },

    { label: "Teachers", value: 15, color: "#1cc88a" },

    { label: "Departments", value: 5, color: "#f6c23e" },

    { label: "Subjects", value: 20, color: "#e74a3b" },

  ];

*// ✅ UPDATED: Added 3 new menu items*

  const menuItems = [

    { label: "Dashboard", route: "/admin-dashboard" },

    { label: "Departments", route: "/departments" },

    { label: "Subjects", route: "/subjects" },

    { label: "Teachers", route: "/teachers" },

{ label: "Manage Users", route: "/users" },

    { label: "Manage Quizzes", route: "/manage-quizzes" },

    { label: "Manage Results", route: "/manage-results" },

    { label: "Subject Management", route: "/subject-management" },

    { label: "Logout", route: "/" },

  ];

  const handleMenuClick = (item) => {

    setMenuVisible(false);

    if (item.label === "Logout") {

      alert("Logged out!");

      localStorage.removeItem("adminToken");

    }

    navigate(item.route);

  };

  return (

    <div style={styles.dashboard}>

      <header style={styles.header}>

        <h2 style={styles.title}>Admin Dashboard</h2>

        <div style={styles.userMenu}>

          <span>Welcome, Admin</span>

          <button

            style={styles.menuToggle}

            onClick={() => setMenuVisible(!menuVisible)}

          >

            ☰

          </button>

        </div>

      </header>

      {menuVisible && (

        <div style={styles.sidebarMenu}>

          {menuItems.map((item) => (

            <div

              key={item.label}

              onClick={() => handleMenuClick(item)}

              style={styles.menuItem}

            >

              {item.label}

            </div>

          ))}

        </div>

      )}

      {*/\* KPI Section \*/*}

      <section style={styles.kpiSection}>

        {stats.map((s) => (

          <div key={s.label} style={{ ...styles.kpiCard, background: s.color }}>

            <h3>{s.label}</h3>

            <p>{s.value}</p>

          </div>

        ))}

      </section>

      {*/\* Charts Section \*/*}

      <section style={styles.chartsSection}>

        <div style={styles.chartCard}>

          <h3>Performance Overview</h3>

          <ResponsiveContainer width="100%" height={300}>

            <BarChart

              data={stats}

              barSize={40}

              margin={{ top: 30, right: 30, left: 0, bottom: 10 }}

            >

              <defs>

                <linearGradient id="waveGradient" x1="0" y1="0" x2="0" y2="1">

                  <stop offset="0%" stopColor="#00e0ff" stopOpacity={1} />

                  <stop offset="100%" stopColor="#0078ff" stopOpacity={0.8} />

                </linearGradient>

              </defs>

              <CartesianGrid strokeDasharray="3 3" stroke="#333" />

              <XAxis dataKey="label" stroke="#fff" />

              <YAxis stroke="#fff" />

              <Tooltip

                contentStyle={{

                  background: "#2e2e4f",

                  borderRadius: "10px",

                  border: "none",

                  color: "#fff",

                }}

              />

              <Bar

                dataKey="value"

                fill="url(#waveGradient)"

                radius={[10, 10, 0, 0]}

                animationDuration={2000}

              />

            </BarChart>

          </ResponsiveContainer>

        </div>

        <div style={styles.chartCard}>

          <h3>Data Distribution</h3>

          <ResponsiveContainer width="100%" height={300}>

            <PieChart>

              <Pie

                data={stats}

                dataKey="value"

                nameKey="label"

                outerRadius={120}

                fill="#8884d8"

                label

                animationDuration={2000}

              >

                {stats.map((entry, index) => (

                  <Cell key={`cell-${index}`} fill={entry.color} />

                ))}

              </Pie>

              <Tooltip

                contentStyle={{

                  background: "#2e2e4f",

                  borderRadius: "10px",

                  border: "none",

                  color: "#fff",

                }}

              />

            </PieChart>

          </ResponsiveContainer>

        </div>

      </section>

      <section style={styles.teacherSection}>

        <div style={styles.teacherCard}>

          <h3>👩‍🏫 Manage Teachers</h3>

          <p>

            View, Add, Update, or Delete teachers. Assign subjects and track

            department distribution.

          </p>

          <button

            style={styles.teacherButton}

            onClick={() => navigate("/teachers")}

          >

            Go to Teacher Management

          </button>

        </div>

      </section>

    </div>

  );

};

const styles = {

  dashboard: {

    background: "linear-gradient(135deg, #1e1e2f, #2e2e4f)",

    color: "white",

    minHeight: "100vh",

    padding: "20px",

    fontFamily: "'Poppins', sans-serif",

  },

  header: {

    display: "flex",

    justifyContent: "space-between",

    alignItems: "center",

    background: "#2e2e4f",

    padding: "15px 25px",

    borderRadius: "12px",

    boxShadow: "0 0 15px rgba(0,0,0,0.5)",

  },

  title: {

    color: "#00e0ff",

    textShadow: "0 0 10px rgba(0,224,255,0.7)",

  },

  userMenu: {

    display: "flex",

    alignItems: "center",

    gap: "10px",

  },

  menuToggle: {

    background: "#00bfff",

    color: "white",

    border: "none",

    borderRadius: "6px",

    fontSize: "20px",

    padding: "5px 10px",

    cursor: "pointer",

    boxShadow: "0 0 10px rgba(0,191,255,0.5)",

  },

  sidebarMenu: {

    position: "absolute",

    top: "80px",

    right: "20px",

    background: "#3b3b5c",

    borderRadius: "10px",

    padding: "10px",

    boxShadow: "0 0 10px rgba(0,0,0,0.4)",

  },

  menuItem: {

    padding: "10px",

    cursor: "pointer",

    color: "white",

    borderBottom: "1px solid rgba(255,255,255,0.1)",

  },

  kpiSection: {

    display: "flex",

    justifyContent: "space-around",

    marginTop: "30px",

    flexWrap: "wrap",

    gap: "20px",

  },

  kpiCard: {

    width: "200px",

    borderRadius: "10px",

    padding: "20px",

    textAlign: "center",

    boxShadow: "0 0 15px rgba(0,0,0,0.3)",

  },

  chartsSection: {

    display: "flex",

    justifyContent: "space-around",

    marginTop: "40px",

    flexWrap: "wrap",

    gap: "30px",

  },

  chartCard: {

    background: "#2e2e4f",

    borderRadius: "12px",

    padding: "20px",

    width: "400px",

    boxShadow: "0 0 15px rgba(0,0,0,0.4)",

  },

  teacherSection: {

    marginTop: "50px",

    textAlign: "center",

  },

  teacherCard: {

    background: "#2e2e4f",

    padding: "30px",

    borderRadius: "12px",

    width: "80%",

    margin: "auto",

    boxShadow: "0 0 15px rgba(0,0,0,0.5)",

  },

  teacherButton: {

    background: "linear-gradient(90deg, #00bcd4, #0097a7)",

    border: "none",

    color: "white",

    padding: "12px 20px",

    borderRadius: "8px",

    fontSize: "16px",

    fontWeight: "bold",

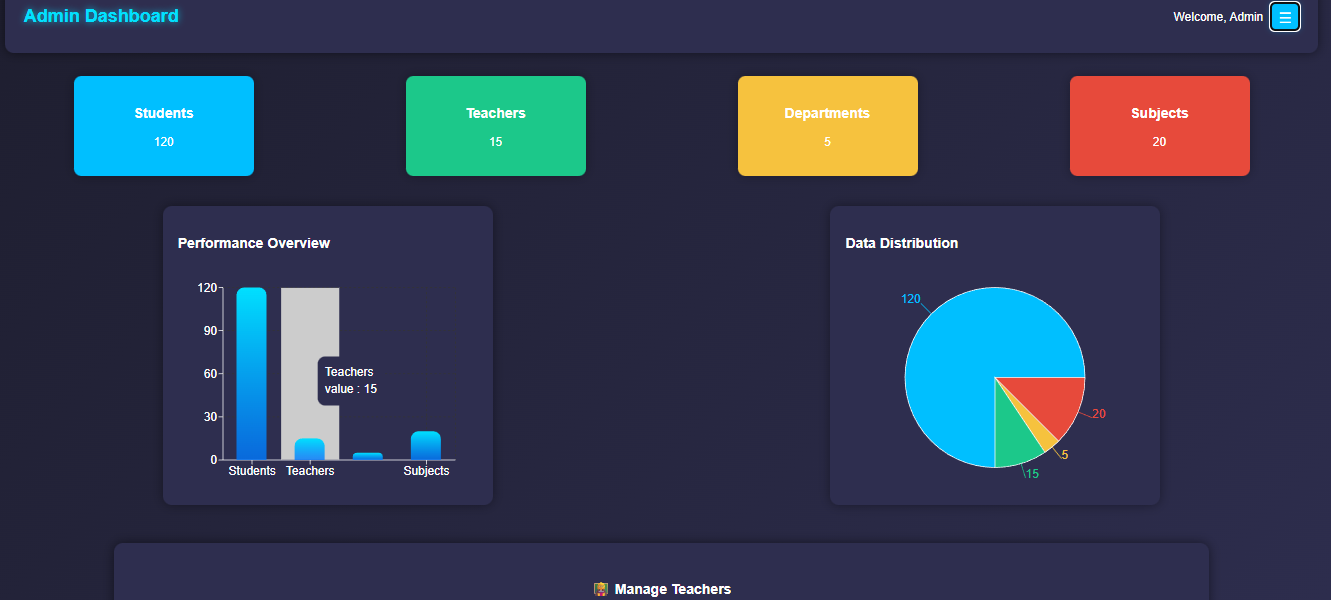
    cursor: "pointer",

    marginTop: "15px",

  },

};

export default AdminDashboard;

****

**Department Mangement.js**

import React, { useEffect, useState } from "react";

import axios from "axios";

import { ToastContainer, toast } from "react-toastify";

import "react-toastify/dist/ReactToastify.css";

const DepartmentManagement = () => {

  const [departments, setDepartments] = useState([]);

  const [newDepartment, setNewDepartment] = useState("");

  const [searchText, setSearchText] = useState("");

  const [editId, setEditId] = useState(null);

  const [editName, setEditName] = useState("");

  useEffect(() => {

    fetchDepartments();

  }, []);

  const fetchDepartments = async () => {

    try {

      const res = await axios.get("http://localhost:5000/departments/display");

      setDepartments(res.data);

    } catch {

      toast.error("❌ Failed to fetch departments");

    }

  };

  const generateDepartmentId = () => {

    if (departments.length === 0) return "D001";

    const lastDept = departments[departments.length - 1];

    const lastNum = parseInt(lastDept.department\_id.slice(1)) || 0;

    return "D" + String(lastNum + 1).padStart(3, "0");

  };

  const handleAddOrUpdate = async () => {

    if (!newDepartment.trim() && !editName.trim()) return;

    if (editId) {

      try {

        await axios.put("http://localhost:5000/departments/update", {

          department\_id: editId,

          department\_name: editName,

        });

        setEditId(null);

        setEditName("");

        fetchDepartments();

        toast.success("✅ Department updated successfully");

      } catch {

        toast.error("❌ Failed to update department");

      }

    } else {

      try {

        const department\_id = generateDepartmentId();

        await axios.post("http://localhost:5000/departments/insert", {

          department\_id,

          department\_name: newDepartment,

        });

        setNewDepartment("");

        fetchDepartments();

        toast.success("✅ Department added successfully");

      } catch {

        toast.error("❌ Failed to add department");

      }

    }

  };

  const handleDelete = async (id) => {

    try {

      await axios.delete(`http://localhost:5000/departments/delete?id=${id}`);

      fetchDepartments();

      toast.info("🗑️ Department deleted");

    } catch {

      toast.error("❌ Failed to delete department");

    }

  };

  const handleEdit = (dept) => {

    setEditId(dept.department\_id);

    setEditName(dept.department\_name);

  };

  const filteredDepartments = departments.filter((dept) =>

    dept.department\_name.toLowerCase().includes(searchText.toLowerCase())

  );

  return (

    <div style={styles.container}>

      <ToastContainer position="top-right" autoClose={2500} />

      <h1 style={styles.title}>Department Management</h1>

      <InputSection

        editId={editId}

        editName={editName}

        setEditName={setEditName}

        newDepartment={newDepartment}

        setNewDepartment={setNewDepartment}

        handleAddOrUpdate={handleAddOrUpdate}

        searchText={searchText}

        setSearchText={setSearchText}

      />

      <DepartmentTable

        departments={filteredDepartments}

        handleEdit={handleEdit}

        handleDelete={handleDelete}

      />

    </div>

  );

};

const InputSection = ({

  editId,

  editName,

  setEditName,

  newDepartment,

  setNewDepartment,

  handleAddOrUpdate,

  searchText,

  setSearchText,

}) => {

  return (

    <>

      <div style={styles.inputContainer}>

        <input

          type="text"

          placeholder="Enter Department Name"

          value={editId ? editName : newDepartment}

          onChange={(e) =>

            editId ? setEditName(e.target.value) : setNewDepartment(e.target.value)

          }

          style={styles.input}

        />

        <button onClick={handleAddOrUpdate} style={styles.addButton}>

          {editId ? "Update" : "Add Department"}

        </button>

      </div>

      <div style={{ ...styles.inputContainer, justifyContent: "center" }}>

        <input

          type="text"

          placeholder="Search by Name..."

          value={searchText}

          onChange={(e) => setSearchText(e.target.value)}

          style={{ ...styles.input, width: "350px" }}

        />

      </div>

    </>

  );

};

const DepartmentTable = ({ departments, handleEdit, handleDelete }) => {

  return (

    <div style={styles.tableWrapper}>

      <table style={styles.table}>

        <thead>

          <tr>

            <th style={styles.th}>Sr No.</th>

            <th style={styles.th}>Department Name</th>

            <th style={styles.th}>Actions</th>

          </tr>

        </thead>

        <tbody>

          {departments.map((dept, index) => (

            <tr key={dept.department\_id}>

              <td style={styles.td}>{index + 1}</td>

              <td style={styles.td}>{dept.department\_name}</td>

              <td style={styles.td}>

                <button style={styles.tdButton} onClick={() => handleEdit(dept)}>

                  Edit

                </button>

                <button style={styles.tdButton} onClick={() => handleDelete(dept.department\_id)}>

                  Delete

                </button>

              </td>

            </tr>

          ))}

        </tbody>

      </table>

    </div>

  );

};

const styles = {

  container: {

    minHeight: "100vh",

    background: "linear-gradient(135deg, #0f0f1a, #1e1e2f, #2c2c3e)",

    padding: "40px",

    fontFamily: "'Poppins', sans-serif",

    color: "#fff",

  },

  title: {

    textAlign: "center",

    fontSize: "32px",

    color: "#00bcd4",

    marginBottom: "30px",

    fontWeight: "bold",

  },

  inputContainer: {

    display: "flex",

    justifyContent: "center",

    gap: "10px",

    marginBottom: "20px",

  },

  input: {

    width: "300px",

    padding: "12px",

    borderRadius: "8px",

    border: "none",

    outline: "none",

    fontSize: "16px",

    background: "#222",

    color: "#fff",

  },

  addButton: {

    padding: "12px 20px",

    background: "linear-gradient(90deg, #00bcd4, #0097a7)",

    border: "none",

    borderRadius: "8px",

    color: "#fff",

    fontWeight: "bold",

    cursor: "pointer",

    transition: "transform 0.2s, box-shadow 0.2s",

  },

  tableWrapper: {

    overflowX: "auto",

    marginTop: "20px",

  },

  table: {

    width: "80%",

    margin: "auto",

    borderCollapse: "separate",

    borderSpacing: "0 10px",

    fontSize: "16px",

  },

  th: {

    background: "linear-gradient(90deg, #00bcd4, #0097a7)",

    color: "#fff",

    padding: "12px",

    textAlign: "center",

    fontWeight: "700",

    borderRadius: "8px",

  },

  td: {

    background: "#1e1e2f",

    color: "#fff",

    padding: "12px",

    textAlign: "center",

    borderRadius: "8px",

    boxShadow: "0 2px 5px rgba(0,0,0,0.3)",

  },

  tdButton: {

    background: "linear-gradient(90deg, #ff5252, #ff1744)",

    color: "#fff",

    padding: "6px 14px",

    margin: "0 5px",

    border: "none",

    borderRadius: "6px",

    cursor: "pointer",

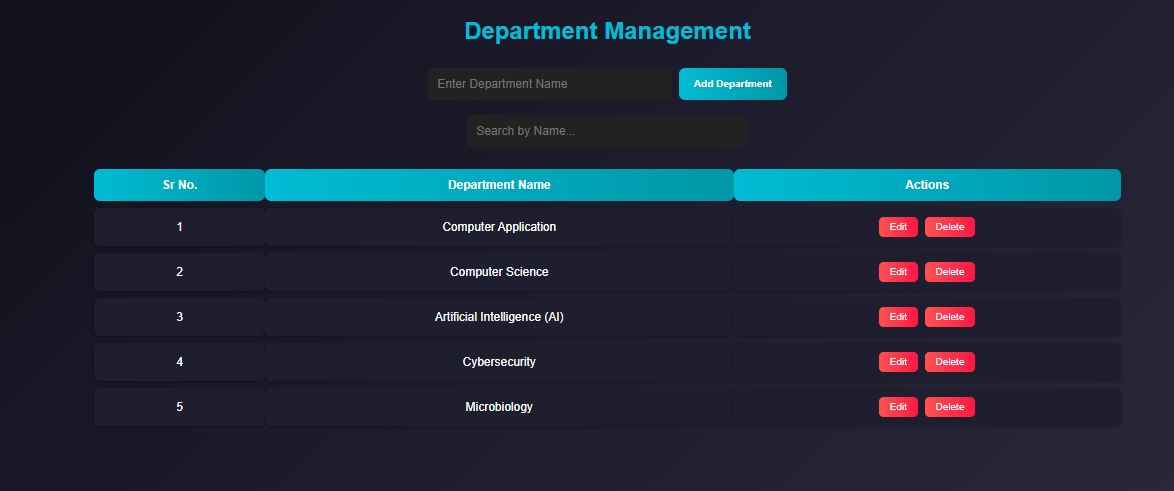
    fontWeight: "500",

    transition: "transform 0.2s, box-shadow 0.2s",

  },

};

export default DepartmentManagement;

****

**Subject Mangement.js**

import React, { useEffect, useState } from "react";

import axios from "axios";

import { ToastContainer, toast } from "react-toastify";

import "react-toastify/dist/ReactToastify.css";

const SubjectManagement = () => {

  const [departments, setDepartments] = useState([]);

  const [subjects, setSubjects] = useState([]);

  const [newSubject, setNewSubject] = useState("");

  const [selectedDept, setSelectedDept] = useState("");

  const [searchText, setSearchText] = useState("");

  const [editId, setEditId] = useState(null);

  const [editName, setEditName] = useState("");

  useEffect(() => {

    fetchDepartments();

    fetchSubjects();

  }, []);

  const fetchDepartments = async () => {

    try {

      const res = await axios.get("http://localhost:5000/departments/display");

      setDepartments(res.data);

    } catch (err) {

      console.error(err);

      toast.error("❌ Failed to fetch departments");

    }

  };

  const fetchSubjects = async () => {

    try {

      const res = await axios.get("http://localhost:5000/subjects/display");

      setSubjects(res.data);

    } catch (err) {

      console.error(err);

      toast.error("❌ Failed to fetch subjects");

    }

  };

  const handleAddOrUpdate = async () => {

    const subjectName = editId ? editName : newSubject;

    if (!subjectName.trim() || !selectedDept) return;

    try {

      if (editId) {

*// UPDATE subject*

        await axios.put("http://localhost:5000/subjects/update", {

          subject\_id: editId,

          subject\_name: subjectName,

          department\_id: selectedDept,

        });

        toast.success("✅ Subject updated successfully");

      } else {

*// ADD new subject*

        const lastSubject = subjects[subjects.length - 1];

        const lastNum = lastSubject

          ? parseInt(lastSubject.subject\_id.slice(1)) || 0

          : 0;

        const newSubjectId = "S" + String(lastNum + 1).padStart(3, "0");

        await axios.post("http://localhost:5000/subjects/insert", {

          subject\_id: newSubjectId,

          subject\_name: subjectName,

          department\_id: selectedDept,

        });

        toast.success("✅ Subject added successfully");

      }

*// Reset form*

      setEditId(null);

      setEditName("");

      setNewSubject("");

      setSelectedDept("");

      fetchSubjects();

    } catch (err) {

      console.error(err);

      toast.error("❌ Failed to save subject");

    }

  };

  const handleDelete = async (id) => {

    try {

      await axios.delete(`http://localhost:5000/subjects/delete?id=${id}`);

      fetchSubjects();

      toast.info("🗑️ Subject deleted");

    } catch (err) {

      console.error(err);

      toast.error("❌ Failed to delete subject");

    }

  };

  const handleEdit = (subj) => {

    setEditId(subj.subject\_id);

    setEditName(subj.subject\_name);

    setSelectedDept(subj.department\_id);

  };

  const filteredSubjects = subjects.filter((subj, index) => {

    const srNo = String(index + 1);

    return (

      subj.subject\_name.toLowerCase().includes(searchText.toLowerCase()) ||

      srNo.includes(searchText)

    );

  });

  return (

    <div style={styles.container}>

      <ToastContainer position="top-right" autoClose={2500} />

      <h1 style={styles.title}>Subject Management</h1>

      <InputSection

        editId={editId}

        editName={editName}

        setEditName={setEditName}

        newSubject={newSubject}

        setNewSubject={setNewSubject}

        selectedDept={selectedDept}

        setSelectedDept={setSelectedDept}

        handleAddOrUpdate={handleAddOrUpdate}

        searchText={searchText}

        setSearchText={setSearchText}

        departments={departments}

      />

      <SubjectTable

        subjects={filteredSubjects}

        departments={departments}

        handleEdit={handleEdit}

        handleDelete={handleDelete}

      />

    </div>

  );

};

const InputSection = ({

  editId,

  editName,

  setEditName,

  newSubject,

  setNewSubject,

  selectedDept,

  setSelectedDept,

  handleAddOrUpdate,

  searchText,

  setSearchText,

  departments,

}) => (

  <>

    <div style={styles.inputContainer}>

      <select

        value={selectedDept}

        onChange={(e) => setSelectedDept(e.target.value)}

        style={styles.select}

      >

        <option value="">Select Department</option>

        {departments.map((dept) => (

          <option key={dept.department\_id} value={dept.department\_id}>

            {dept.department\_name}

          </option>

        ))}

      </select>

      <input

        type="text"

        placeholder="Enter Subject Name"

        value={editId ? editName : newSubject}

        onChange={(e) =>

          editId ? setEditName(e.target.value) : setNewSubject(e.target.value)

        }

        style={styles.input}

      />

      <button onClick={handleAddOrUpdate} style={styles.addButton}>

        {editId ? "Update" : "Add Subject"}

      </button>

    </div>

    <div style={{ ...styles.inputContainer, justifyContent: "center" }}>

      <input

        type="text"

        placeholder="Search by Name..."

        value={searchText}

        onChange={(e) => setSearchText(e.target.value)}

        style={{ ...styles.input, width: "350px" }}

      />

    </div>

  </>

);

const SubjectTable = ({ subjects, departments, handleEdit, handleDelete }) => (

  <div style={styles.tableWrapper}>

    <table style={styles.table}>

      <thead>

        <tr>

          <th style={styles.th}>Sr No.</th>

          <th style={styles.th}>Subject Name</th>

          <th style={styles.th}>Department</th>

          <th style={styles.th}>Actions</th>

        </tr>

      </thead>

      <tbody>

        {subjects.map((subj, index) => (

          <tr key={subj.subject\_id}>

            <td style={styles.td}>{index + 1}</td>

            <td style={styles.td}>{subj.subject\_name}</td>

            <td style={styles.td}>

              {

                departments.find((d) => d.department\_id === subj.department\_id)

                  ?.department\_name

              }

            </td>

            <td style={styles.td}>

              <button style={styles.tdButton} onClick={() => handleEdit(subj)}>

                Edit

              </button>

              <button

                style={styles.tdButton}

                onClick={() => handleDelete(subj.subject\_id)}

              >

                Delete

              </button>

            </td>

          </tr>

        ))}

      </tbody>

    </table>

  </div>

);

const styles = {

  container: {

    minHeight: "100vh",

    background: "linear-gradient(135deg, #0f0f1a, #1e1e2f, #2c2c3e)",

    padding: "40px",

    fontFamily: "'Poppins', sans-serif",

    color: "#fff",

  },

  title: {

    textAlign: "center",

    fontSize: "32px",

    color: "#00bcd4",

    marginBottom: "30px",

    fontWeight: "bold",

  },

  inputContainer: {

    display: "flex",

    justifyContent: "center",

    gap: "10px",

    marginBottom: "20px",

  },

  select: {

    padding: "12px",

    borderRadius: "8px",

    border: "none",

    outline: "none",

    background: "#222",

    color: "#fff",

    fontSize: "16px",

  },

  input: {

    width: "300px",

    padding: "12px",

    borderRadius: "8px",

    border: "none",

    outline: "none",

    fontSize: "16px",

    background: "#222",

    color: "#fff",

  },

  addButton: {

    padding: "12px 20px",

    background: "linear-gradient(90deg, #00bcd4, #0097a7)",

    border: "none",

    borderRadius: "8px",

    color: "#fff",

    fontWeight: "bold",

    cursor: "pointer",

  },

  tableWrapper: {

    overflowX: "auto",

    marginTop: "20px",

  },

  table: {

    width: "80%",

    margin: "auto",

    borderCollapse: "separate",

    borderSpacing: "0 10px",

    fontSize: "16px",

  },

  th: {

    background: "linear-gradient(90deg, #00bcd4, #0097a7)",

    color: "#fff",

    padding: "12px",

    textAlign: "center",

    borderRadius: "8px",

  },

  td: {

    background: "#1e1e2f",

    color: "#fff",

    padding: "12px",

    textAlign: "center",

    borderRadius: "8px",

    boxShadow: "0 2px 5px rgba(0,0,0,0.3)",

  },

  tdButton: {

    background: "linear-gradient(90deg, #ff5252, #ff1744)",

    color: "#fff",

    padding: "6px 14px",

    margin: "0 5px",

    border: "none",

    borderRadius: "6px",

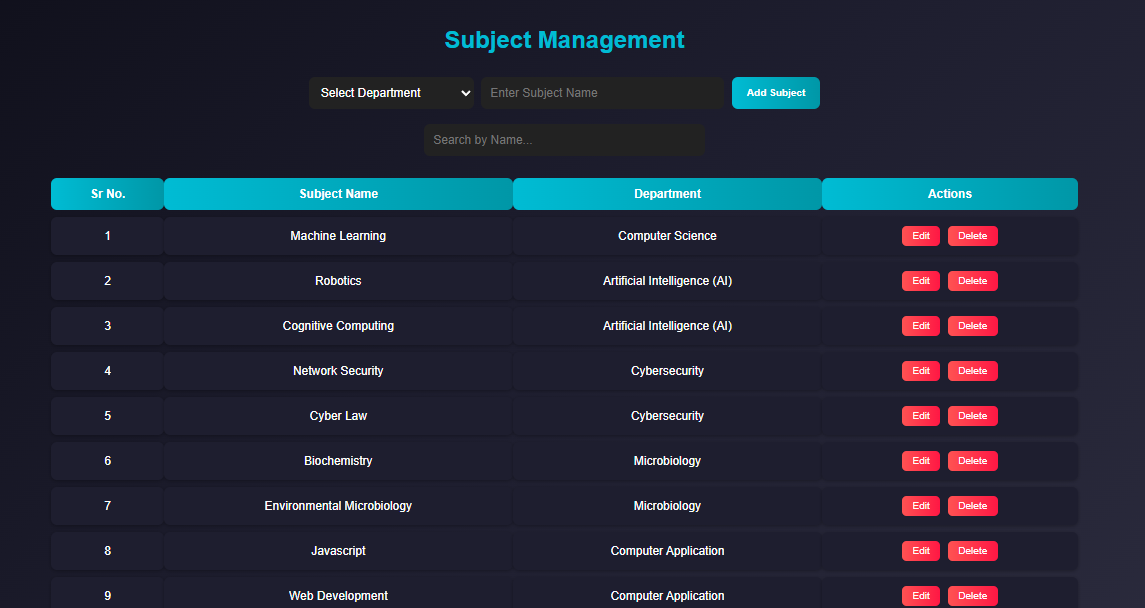
    cursor: "pointer",

    fontWeight: "500",

  },

};

export default SubjectManagement;

****

**Teacher Register.js**

import React, { useEffect, useState } from "react";

import axios from "axios";

import { ToastContainer, toast } from "react-toastify";

import "react-toastify/dist/ReactToastify.css";

import \* as yup from "yup";

const teacherSchema = yup.object().shape({

  teacherName: yup.string().required("Teacher name is required"),

  teacherEmail: yup.string().email().required("Email is required"),

  teacherPassword: yup.string().required("Password is required"),

  department\_id: yup.string().required("Department is required"),

});

const TeacherRegister = () => {

  const [departments, setDepartments] = useState([]);

  const [teachers, setTeachers] = useState([]);

  const [teacherName, setTeacherName] = useState("");

  const [teacherEmail, setTeacherEmail] = useState("");

  const [teacherPassword, setTeacherPassword] = useState("");

  const [selectedDept, setSelectedDept] = useState("");

  const [searchText, setSearchText] = useState("");

  const [editId, setEditId] = useState(null);

  useEffect(() => {

    fetchDepartments();

    fetchTeachers();

  }, []);

  const fetchDepartments = async () => {

    try {

      const res = await axios.get("http://localhost:5000/departments/display");

      setDepartments(res.data);

    } catch {

      toast.error("Failed to fetch departments");

    }

  };

  const fetchTeachers = async () => {

    try {

      const res = await axios.get("http://localhost:5000/teachers/display");

      setTeachers(res.data);

    } catch {

      toast.error("Failed to fetch teachers");

    }

  };

  const handleAddOrUpdate = async () => {

    try {

      await teacherSchema.validate(

        {

          teacherName,

          teacherEmail,

          teacherPassword,

          department\_id: selectedDept,

        },

        { abortEarly: false }

      );

      if (editId) {

        await axios.put("http://localhost:5000/teachers/update", {

          teacher\_id: editId,

          teacherName,

          teacherEmail,

          teacherPassword,

          department\_id: selectedDept,

        });

        toast.success("Teacher updated successfully");

      } else {

        await axios.post("http://localhost:5000/teachers/insert", {

          teacherName,

          teacherEmail,

          teacherPassword,

          department\_id: selectedDept,

        });

        toast.success("Teacher registered successfully");

      }

      setEditId(null);

      setTeacherName("");

      setTeacherEmail("");

      setTeacherPassword("");

      setSelectedDept("");

      fetchTeachers();

    } catch (err) {

      if (err.inner) err.inner.forEach((e) => toast.warning(e.message));

      else toast.error("Failed to save teacher");

    }

  };

  const handleDelete = async (id) => {

    await axios.delete(`http://localhost:5000/teachers/delete?id=${id}`);

    toast.info("Teacher deleted");

    fetchTeachers();

  };

  const handleEdit = (teacher) => {

    setEditId(teacher.teacher\_id);

    setTeacherName(teacher.teacherName);

    setTeacherEmail(teacher.teacherEmail);

    setTeacherPassword(teacher.teacherPassword);

    setSelectedDept(teacher.department\_id);

  };

  const filteredTeachers = teachers.filter((t, index) => {

    const srNo = String(index + 1);

    return (

      t.teacherName.toLowerCase().includes(searchText.toLowerCase()) ||

      t.teacherEmail.toLowerCase().includes(searchText.toLowerCase()) ||

      srNo.includes(searchText)

    );

  });

  return (

    <div style={styles.container}>

      <ToastContainer position="top-right" autoClose={2500} />

      <h1 style={styles.title}>Teacher Register</h1>

      <div style={styles.inputContainer}>

        <input

          placeholder="Teacher Name"

          value={teacherName}

          onChange={(e) => setTeacherName(e.target.value)}

          style={styles.input}

        />

        <input

          type="email"

          placeholder="Email"

          value={teacherEmail}

          onChange={(e) => setTeacherEmail(e.target.value)}

          style={styles.input}

        />

        <input

          type="password"

          placeholder="Password"

          value={teacherPassword}

          onChange={(e) => setTeacherPassword(e.target.value)}

          style={styles.input}

        />

        <select

          value={selectedDept}

          onChange={(e) => setSelectedDept(e.target.value)}

          style={styles.select}

        >

          <option value="">Select Department</option>

          {departments.map((d) => (

            <option key={d.department\_id} value={d.department\_id}>

              {d.department\_name}

            </option>

          ))}

        </select>

        <button onClick={handleAddOrUpdate} style={styles.addButton}>

          {editId ? "Update" : "Register"}

        </button>

      </div>

      <div style={styles.inputContainer}>

        <input

          placeholder="Search..."

          value={searchText}

          onChange={(e) => setSearchText(e.target.value)}

          style={{ ...styles.input, width: "300px" }}

        />

      </div>

      <div style={styles.tableWrapper}>

        <table style={styles.table}>

          <thead>

            <tr>

              <th style={styles.th}>Sr No.</th>

              <th style={styles.th}>Name</th>

              <th style={styles.th}>Email</th>

              <th style={styles.th}>Department</th>

              <th style={styles.th}>Actions</th>

            </tr>

          </thead>

          <tbody>

            {filteredTeachers.map((t, i) => (

              <tr key={t.teacher\_id} style={styles.tr}>

                <td style={styles.td}>{i + 1}</td>

                <td style={styles.td}>{t.teacherName}</td>

                <td style={styles.td}>{t.teacherEmail}</td>

                <td style={styles.td}>

                  {departments.find((d) => d.department\_id === t.department\_id)

                    ?.department\_name || "-"}

                </td>

                <td style={styles.td}>

                  <button style={styles.editBtn} onClick={() => handleEdit(t)}>

                    Edit

                  </button>

                  <button

                    style={styles.deleteBtn}

                    onClick={() => handleDelete(t.teacher\_id)}

                  >

                    Delete

                  </button>

                </td>

              </tr>

            ))}

          </tbody>

        </table>

      </div>

    </div>

  );

};

const styles = {

  container: {

    minHeight: "100vh",

    background: "linear-gradient(135deg, #0f0f1a, #1e1e2f, #2c2c3e)",

    padding: "40px",

    fontFamily: "'Poppins', sans-serif",

    color: "#fff",

  },

  title: {

    textAlign: "center",

    fontSize: "32px",

    color: "#00bcd4",

    marginBottom: "30px",

    fontWeight: "bold",

  },

  inputContainer: {

    display: "flex",

    justifyContent: "center",

    gap: "10px",

    marginBottom: "20px",

    flexWrap: "wrap",

  },

  select: {

    padding: "12px",

    borderRadius: "8px",

    border: "none",

    outline: "none",

    background: "#222",

    color: "#fff",

    fontSize: "16px",

  },

  input: {

    width: "200px",

    padding: "12px",

    borderRadius: "8px",

    border: "none",

    outline: "none",

    fontSize: "16px",

    background: "#222",

    color: "#fff",

  },

  addButton: {

    padding: "12px 20px",

    background: "linear-gradient(90deg, #00bcd4, #0097a7)",

    border: "none",

    borderRadius: "8px",

    color: "#fff",

    fontWeight: "bold",

    cursor: "pointer",

  },

  tableWrapper: {

    overflowX: "auto",

    marginTop: "30px",

  },

*// ✅ Updated spacing between columns and rows*

  table: {

    width: "90%",

    margin: "auto",

    borderCollapse: "separate",

    borderSpacing: "20px 15px", *// <== space between columns (20px) and rows (15px)*

    textAlign: "center",

  },

  th: {

    background: "linear-gradient(90deg, #00bcd4, #0097a7)",

    color: "#fff",

    padding: "15px",

    textAlign: "center",

    fontSize: "18px",

    borderRadius: "10px",

  },

  tr: {

    transition: "transform 0.2s ease, background 0.2s ease",

  },

  td: {

    background: "#1e1e2f",

    color: "#fff",

    padding: "16px 25px", *// ✅ extra padding for inner spacing*

    textAlign: "center",

    fontSize: "16px",

    borderRadius: "10px",

    boxShadow: "0 3px 8px rgba(0,0,0,0.4)",

  },

  editBtn: {

    background: "linear-gradient(90deg, #4caf50, #388e3c)",

    color: "#fff",

    padding: "6px 14px",

    margin: "0 5px",

    border: "none",

    borderRadius: "6px",

    cursor: "pointer",

    fontWeight: "500",

  },

  deleteBtn: {

    background: "linear-gradient(90deg, #ff5252, #ff1744)",

    color: "#fff",

    padding: "6px 14px",

    margin: "0 5px",

    border: "none",

    borderRadius: "6px",

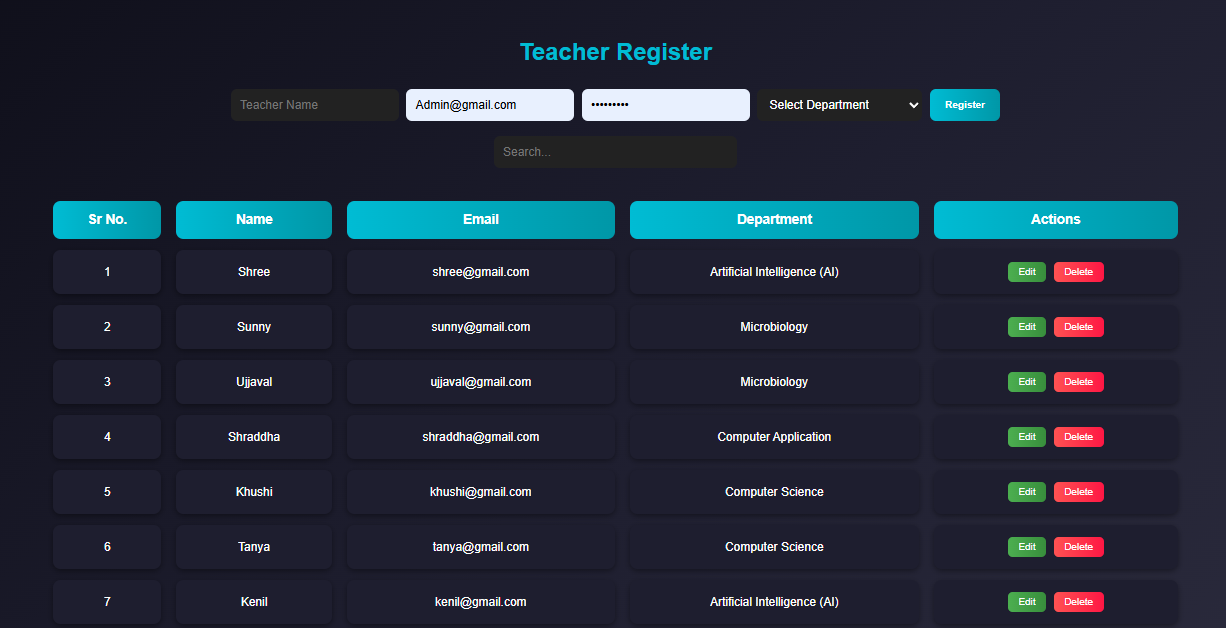
    cursor: "pointer",

    fontWeight: "500",

  },

};

export default TeacherRegister;

****

**Server.js**

const http = require("http");

const { MongoClient, ObjectId } = require("mongodb");

const yup = require("yup");

const uri = "mongodb://127.0.0.1:27017";

const dbName = "Quiz";

const client = new MongoClient(uri);

*// --------------------------- VALIDATION ---------------------------*

const adminLoginSchema = yup.object({

  username: yup.string().email().required(),

  password: yup.string().required(),

});

const departmentSchema = yup.object({

  department\_id: yup.string().required(),

  department\_name: yup.string().required(),

});

const subjectSchema = yup.object({

  subject\_id: yup.string(),

  subject\_name: yup.string().required(),

  department\_id: yup.string().required(),

});

const teacherSchema = yup.object({

  teacher\_id: yup.string(),

  teacherName: yup.string().required(),

  teacherEmail: yup.string().email().required(),

  teacherPassword: yup.string().required(),

  department\_id: yup.string().required(),

  subject\_id: yup.string().nullable(),

});

*// --------------------------- RUN SERVER ---------------------------*

async function runServer() {

  try {

    await client.connect();

    console.log("✅ Connected to MongoDB");

    const db = client.db(dbName);

    const adminCollection = db.collection("Admin");

    const departmentCollection = db.collection("departments");

    const subjectCollection = db.collection("subjects");

    const teacherCollection = db.collection("teachers");

*// USERS COLLECTION*

    const usersCollection = db.collection("Users");

*// --------------------------- DEFAULT ADMIN ---------------------------*

    const defaultAdmin = await adminCollection.findOne({

      email: "Admin@gmail.com",

    });

    if (!defaultAdmin) {

      await adminCollection.insertOne({

        email: "Admin@gmail.com",

        password: "admin@123",

      });

      console.log("✅ Default admin created");

    }

    const server = http.createServer(async (req, res) => {

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

      res.setHeader(

        "Access-Control-Allow-Methods",

        "GET, POST, PUT, DELETE, OPTIONS"

      );

      res.setHeader("Access-Control-Allow-Headers", "Content-Type");

      if (req.method === "OPTIONS") {

        res.writeHead(204);

        return res.end();

      }

      const getBody = async () =>

        new Promise((resolve, reject) => {

          let data = "";

          req.on("data", (chunk) => (data += chunk));

          req.on("end", () => {

            try {

              resolve(JSON.parse(data || "{}"));

            } catch (err) {

              reject(err);

            }

          });

        });

*// --------------------------- ADMIN LOGIN ---------------------------*

      if (req.url === "/admin/login" && req.method === "POST") {

        const body = await getBody();

        try {

          await adminLoginSchema.validate(body);

          const adminUser = await adminCollection.findOne({

            email: body.username,

            password: body.password,

          });

          if (adminUser) {

            res.writeHead(200, { "Content-Type": "application/json" });

            return res.end(JSON.stringify({ message: "Login successful" }));

          } else {

            res.writeHead(401, { "Content-Type": "application/json" });

            return res.end(JSON.stringify({ message: "Invalid credentials" }));

          }

        } catch (err) {

          res.writeHead(400, { "Content-Type": "application/json" });

          return res.end(JSON.stringify({ message: err.message }));

        }

      }

*// --------------------------- DEPARTMENTS ---------------------------*

      if (req.url === "/departments/insert" && req.method === "POST") {

        const body = await getBody();

        try {

          await departmentSchema.validate(body);

          await departmentCollection.insertOne(body);

          res.writeHead(200, { "Content-Type": "application/json" });

          return res.end(

            JSON.stringify({ message: "Department added successfully" })

          );

        } catch (err) {

          res.writeHead(400, { "Content-Type": "application/json" });

          return res.end(JSON.stringify({ message: err.message }));

        }

      }

      if (req.url === "/departments/display" && req.method === "GET") {

        const departments = await departmentCollection.find({}).toArray();

        res.writeHead(200, { "Content-Type": "application/json" });

        return res.end(JSON.stringify(departments));

      }

      if (req.url === "/departments/update" && req.method === "PUT") {

        const body = await getBody();

        await departmentCollection.updateOne(

          { department\_id: body.department\_id },

          { $set: { department\_name: body.department\_name } }

        );

        res.writeHead(200, { "Content-Type": "application/json" });

        return res.end(

          JSON.stringify({ message: "Department updated successfully" })

        );

      }

      if (req.url.startsWith("/departments/delete") && req.method === "DELETE") {

        const urlObj = new URL(req.url, `http://${req.headers.host}`);

        const id = urlObj.searchParams.get("id");

        await departmentCollection.deleteOne({ department\_id: id });

        res.writeHead(200, { "Content-Type": "application/json" });

        return res.end(

          JSON.stringify({ message: "Department deleted successfully" })

        );

      }

*// --------------------------- SUBJECTS ---------------------------*

      if (req.url === "/subjects/insert" && req.method === "POST") {

        const body = await getBody();

        try {

          await subjectSchema.validate(body);

          if (!body.subject\_id) {

            const last = await subjectCollection

              .find({})

              .sort({ subject\_id: -1 })

              .limit(1)

              .toArray();

            const lastNum = last[0]

              ? parseInt(last[0].subject\_id.slice(1))

              : 0;

            body.subject\_id =

              "S" + String(lastNum + 1).padStart(3, "0");

          }

          const dept = await departmentCollection.findOne({

            department\_id: body.department\_id,

          });

          if (!dept)

            return res.end(

              JSON.stringify({ message: "Invalid department\_id" })

            );

          body.department\_name = dept.department\_name;

          await subjectCollection.insertOne(body);

          res.writeHead(200, { "Content-Type": "application/json" });

          return res.end(JSON.stringify({ message: "Subject added successfully" }));

        } catch (err) {

          res.writeHead(400, { "Content-Type": "application/json" });

          return res.end(JSON.stringify({ message: err.message }));

        }

      }

      if (req.url === "/subjects/display" && req.method === "GET") {

        const subjects = await subjectCollection.find({}).toArray();

        res.writeHead(200, { "Content-Type": "application/json" });

        return res.end(JSON.stringify(subjects));

      }

      if (req.url === "/subjects/update" && req.method === "PUT") {

        const body = await getBody();

        try {

          await subjectSchema.validate(body);

          const dept = await departmentCollection.findOne({

            department\_id: body.department\_id,

          });

          if (!dept)

            return res.end(

              JSON.stringify({ message: "Invalid department\_id" })

            );

          await subjectCollection.updateOne(

            { subject\_id: body.subject\_id },

            {

              $set: {

                subject\_name: body.subject\_name,

                department\_id: body.department\_id,

                department\_name: dept.department\_name,

              },

            }

          );

          res.writeHead(200, { "Content-Type": "application/json" });

          return res.end(

            JSON.stringify({ message: "Subject updated successfully" })

          );

        } catch (err) {

          res.writeHead(400, { "Content-Type": "application/json" });

          return res.end(JSON.stringify({ message: err.message }));

        }

      }

      if (req.url.startsWith("/subjects/delete") && req.method === "DELETE") {

        const urlObj = new URL(req.url, `http://${req.headers.host}`);

        const id = urlObj.searchParams.get("id");

        await subjectCollection.deleteOne({ subject\_id: id });

        res.writeHead(200, { "Content-Type": "application/json" });

        return res.end(

          JSON.stringify({ message: "Subject deleted successfully" })

        );

      }

*// --------------------------- TEACHERS ---------------------------*

      if (req.url === "/teachers/display" && req.method === "GET") {

        const teachers = await teacherCollection.find({}).toArray();

        res.writeHead(200, { "Content-Type": "application/json" });

        return res.end(JSON.stringify(teachers));

      }

      if (req.url === "/teachers/insert" && req.method === "POST") {

        const body = await getBody();

        try {

          await teacherSchema.validate(body);

          const last = await teacherCollection

            .find({})

            .sort({ teacher\_id: -1 })

            .limit(1)

            .toArray();

          const lastNum = last[0]

            ? parseInt(last[0].teacher\_id.slice(1))

            : 0;

          body.teacher\_id =

            "T" + String(lastNum + 1).padStart(3, "0");

          const dept = await departmentCollection.findOne({

            department\_id: body.department\_id,

          });

          if (!dept)

            return res.end(

              JSON.stringify({ message: "Invalid department\_id" })

            );

          body.department\_name = dept.department\_name;

          await teacherCollection.insertOne(body);

          res.writeHead(200, { "Content-Type": "application/json" });

          return res.end(

            JSON.stringify({ message: "Teacher added successfully" })

          );

        } catch (err) {

          res.writeHead(400, { "Content-Type": "application/json" });

          return res.end(JSON.stringify({ message: err.message }));

        }

      }

      if (req.url === "/teachers/update" && req.method === "PUT") {

        const body = await getBody();

        try {

          await teacherSchema.validate(body);

          await teacherCollection.updateOne(

            { teacher\_id: body.teacher\_id },

            {

              $set: {

                teacherName: body.teacherName,

                teacherEmail: body.teacherEmail,

                teacherPassword: body.teacherPassword,

                department\_id: body.department\_id,

              },

            }

          );

          res.writeHead(200, { "Content-Type": "application/json" });

          return res.end(

            JSON.stringify({ message: "Teacher updated successfully" })

          );

        } catch (err) {

          res.writeHead(400, { "Content-Type": "application/json" });

          return res.end(JSON.stringify({ message: err.message }));

        }

      }

      if (req.url.startsWith("/teachers/delete") && req.method === "DELETE") {

        const urlObj = new URL(req.url, `http://${req.headers.host}`);

        const id = urlObj.searchParams.get("id");

        await teacherCollection.deleteOne({ teacher\_id: id });

        res.writeHead(200, { "Content-Type": "application/json" });

        return res.end(

          JSON.stringify({ message: "Teacher deleted successfully" })

        );

      }

*// --------------------------- USERS MODULE ADDED HERE ---------------------------*

*// FETCH ALL USERS*

      if (req.url === "/Users" && req.method === "GET") {

        const users = await usersCollection.find().toArray();

        res.end(JSON.stringify(users));

        return;

      }

*// NEXT AUTO ID*

      if (req.url === "/Users/nextId" && req.method === "GET") {

        const last = await usersCollection

          .find({})

          .sort({ studentId: -1 })

          .limit(1)

          .toArray();

        const nextId = last.length > 0 ? last[0].studentId + 1 : 1;

        res.end(JSON.stringify({ nextId }));

        return;

      }

*// ADD USER*

      if (req.url === "/Users/register" && req.method === "POST") {

        const body = await getBody();

        await usersCollection.insertOne(body);

        res.end(JSON.stringify({ message: "User registered successfully!" }));

        return;

      }

*// UPDATE USER*

      if (req.url.startsWith("/Users/") && req.method === "PUT") {

        const id = req.url.split("/").pop();

        const body = await getBody();

        await usersCollection.updateOne(

          { \_id: new ObjectId(id) },

          { $set: body }

        );

        res.end(JSON.stringify({ message: "User updated successfully!" }));

        return;

      }

*// DELETE USER*

      if (req.url.startsWith("/Users/") && req.method === "DELETE") {

        const id = req.url.split("/").pop();

        await usersCollection.deleteOne({ \_id: new ObjectId(id) });

        res.end(JSON.stringify({ message: "User deleted successfully!" }));

        return;

      }

*// --------------------------- ROUTE NOT FOUND ---------------------------*

      res.writeHead(404, { "Content-Type": "application/json" });

      res.end(JSON.stringify({ message: "Route not found" }));

    });

    server.listen(5000, () =>

      console.log("🚀 Server running at http://localhost:5000")

    );

  } catch (err) {

    console.error("❌ Error:", err);

  }

}

runServer();

**package.json**

{

  "name": "server",

  "version": "1.0.0",

  "main": "index.js",

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1"

  },

  "keywords": [],

  "author": "",

  "license": "ISC",

  "description": "",

  "dependencies": {

    "cors": "^2.8.5",

    "http": "^0.0.1-security",

    "mongodb": "^6.19.0"

  }

}

**Teacher**

**App.js**

import React, { useState } from "react";

import TeacherLogin from "./Components/TeacherLogin";

import Dashboard from "./Components/Dashboard";

import CreateQuiz from "./Components/CreateQuiz";

import ViewStudents from "./Components/ViewStudents";

import ViewProfileStud from "./Components/viewprofileStud";  *// Import new component*

export default function App() {

  const [teacher, setTeacher] = useState(null);

  const [subject, setSubject] = useState(null);

  const [selectedStudent, setSelectedStudent] = useState(null); *// For profile*

  const [page, setPage] = useState("login");

  if (page === "login")

    return (

      <TeacherLogin

        onLogin={(t) => {

          setTeacher(t);

          setPage("dashboard");

        }}

      />

    );

  if (page === "dashboard")

    return (

      <Dashboard

        teacher={teacher}

        onCreateQuiz={(s) => {

          setSubject(s);

          setPage("createQuiz");

        }}

        onViewStudents={() => setPage("students")}

      />

    );

  if (page === "createQuiz")

    return (

      <CreateQuiz

        teacher={teacher}

        subject={subject}

        onBack={() => setPage("dashboard")}

      />

    );

  if (page === "students")

    return (

      <ViewStudents

        teacher={teacher}

        onBack={() => setPage("dashboard")}

        onViewProfile={(student) => {

          setSelectedStudent(student);

          setPage("viewProfile");

        }}

      />

    );

  if (page === "viewProfile")

    return (

      <ViewProfileStud

        student={selectedStudent}

        onBack={() => setPage("students")}

      />

    );

  return null;

}

**Teacher Login.js**

import React, { useState, useEffect } from "react";

import { ToastContainer, toast } from "react-toastify";

import "react-toastify/dist/ReactToastify.css";

export default function TeacherLogin({ onLogin }) {

  const [email, setEmail] = useState("");

  const [password, setPassword] = useState("");

  const [bgColor, setBgColor] = useState("#0f2027");

*// Background animation*

  useEffect(() => {

    const colors = ["#0f2027", "#203a43", "#2c5364", "#485563", "#29323c"];

    let i = 0;

    const interval = setInterval(() => {

      setBgColor(colors[i]);

      i = (i + 1) % colors.length;

    }, 2000);

    return () => clearInterval(interval);

  }, []);

  const handleLogin = async (e) => {

    e.preventDefault();

    try {

      const response = await fetch("http://localhost:5002/login", {

        method: "POST",

        headers: { "Content-Type": "application/json" },

        body: JSON.stringify({ email, password }),

      });

      const data = await response.json();

      if (data.success) {

        toast.success("Login Successful!", {

          position: "top-center",

          autoClose: 1500,

        });

        setTimeout(() => {

          onLogin(data.teacher);

        }, 1500);

      } else {

        toast.error("Invalid Email or Password", {

          position: "top-center",

          autoClose: 1500,

        });

      }

    } catch (error) {

      console.error(error);

      toast.error("Server Error! Try again.", {

        position: "top-center",

        autoClose: 1500,

      });

    }

  };

  return (

    <div

      style={{

        ...styles.container,

        background: bgColor,

        transition: "background 2s ease",

      }}

    >

      <div style={styles.formBox}>

        <h2 style={styles.heading}>Teacher Login</h2>

        <form onSubmit={handleLogin} style={styles.form}>

          <label style={styles.label}>Email:</label>

          <input

            style={styles.input}

            type="email"

            placeholder="Enter email"

            value={email}

            onChange={(e) => setEmail(e.target.value)}

            required

          />

          <label style={styles.label}>Password:</label>

          <input

            style={styles.input}

            type="password"

            placeholder="Enter password"

            value={password}

            onChange={(e) => setPassword(e.target.value)}

            required

          />

          <button style={styles.button} type="submit">

            Login

          </button>

        </form>

      </div>

      {*/\* Toast Fixed \*/*}

      <ToastContainer newestOnTop={false} />

    </div>

  );

}

const styles = {

  container: {

    height: "100vh",

    display: "flex",

    justifyContent: "center",

    alignItems: "center",

  },

  formBox: {

    width: "350px",

    padding: "40px",

    background: "rgba(255,255,255,0.1)",

    borderRadius: "10px",

    backdropFilter: "blur(6px)",

    color: "#fff",

  },

  heading: {

    textAlign: "center",

    marginBottom: "20px",

  },

  form: {

    display: "flex",

    flexDirection: "column",

  },

  label: {

    marginBottom: "8px",

    fontWeight: "bold",

  },

  input: {

    marginBottom: "15px",

    padding: "10px",

    borderRadius: "5px",

    border: "none",

    width: "100%",

  },

  button: {

    padding: "10px",

    backgroundColor: "#00bcd4",

    color: "#fff",

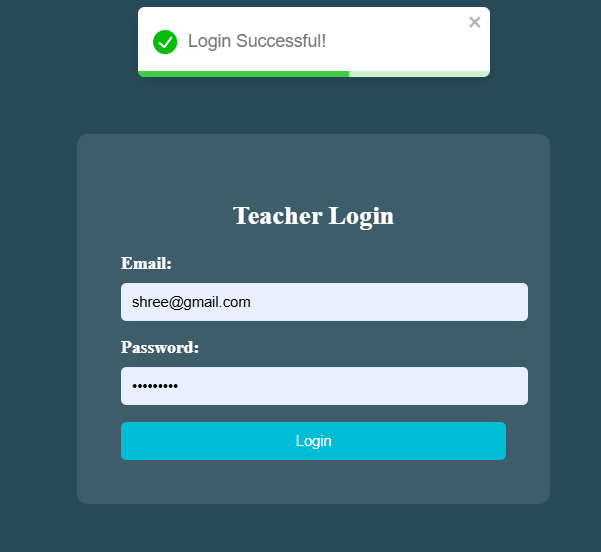
    border: "none",

    borderRadius: "5px",

    cursor: "pointer",

  },

};

****

**Teacher Dashboard-**

import React, { useEffect, useState } from "react";

import { ToastContainer, toast } from "react-toastify";

import "react-toastify/dist/ReactToastify.css";

export default function TeacherDashboard({

  teacher: initialTeacher,

  onCreateQuiz,

  onViewStudents,

  onViewStudentProfile, *// This will be called when profile button is clicked*

  onLogout,

  approach = "normalized",

}) {

  const [teacher, setTeacher] = useState(null);

  const [subjects, setSubjects] = useState([]);

  const [selectedSubject, setSelectedSubject] = useState("");

*// Persist teacher in localStorage*

  useEffect(() => {

    if (initialTeacher) {

      localStorage.setItem("teacher", JSON.stringify(initialTeacher));

      setTeacher(initialTeacher);

    } else {

      const saved = localStorage.getItem("teacher");

      if (saved) setTeacher(JSON.parse(saved));

    }

  }, [initialTeacher]);

*// Dashboard styling*

  useEffect(() => {

    document.body.style.margin = "0";

    document.body.style.height = "100vh";

    document.body.style.background = "#eef3fb";

    document.body.style.fontFamily = "'Poppins', sans-serif";

    const style = document.createElement("style");

    style.innerHTML = `

      @keyframes fadeIn {

        from { opacity: 0; transform: translateY(10px); }

        to { opacity: 1; transform: translateY(0); }

      }

      button:hover {

        opacity: 0.9;

      }

      select:hover {

        border-color: #3b82f6;

      }

    `;

    document.head.appendChild(style);

  }, []);

*// Fetch subjects*

  useEffect(() => {

    if (!teacher?.department\_id) return;

    const fetchSubjects = async () => {

      try {

        let data;

        if (approach === "embedded") {

          const res = await fetch(

            `http://localhost:5002/departments/${teacher.department\_id}`

          );

          data = await res.json();

          if (data.success && data.department?.subjects?.length > 0) {

            setSubjects(data.department.subjects);

            setSelectedSubject(data.department.subjects[0].subject\_id);

          } else {

            setSubjects([]);

            setSelectedSubject("");

          }

        } else {

          const res = await fetch(`http://localhost:5002/subjects/byDepartment`, {

            method: "POST",

            headers: { "Content-Type": "application/json" },

            body: JSON.stringify({ department\_id: teacher.department\_id }),

          });

          data = await res.json();

          if (data.success && data.subjects?.length > 0) {

            setSubjects(data.subjects);

            setSelectedSubject(data.subjects[0].subject\_id);

          } else {

            setSubjects([]);

            setSelectedSubject("");

          }

        }

      } catch (err) {

        console.error("Error fetching subjects:", err);

        toast.error("Failed to fetch subjects. Please try again.");

      }

    };

    fetchSubjects();

  }, [teacher, approach]);

  const handleCreateQuiz = () => {

    if (!selectedSubject) {

      toast.error("Please select a subject first.");

      return;

    }

    const subject = subjects.find((s) => s.subject\_id === selectedSubject);

    if (subject) {

      onCreateQuiz(subject);

      toast.success(`Quiz for "${subject.subject\_name}" created successfully!`);

    }

  };

*// Styles*

  const styles = {

    page: { minHeight: "100vh", color: "#1e293b", display: "flex", flexDirection: "column", animation: "fadeIn 0.8s ease-in-out" },

    navbar: { background: "#1e40af", padding: "18px 0", display: "flex", justifyContent: "center", gap: "25px", boxShadow: "0 2px 15px rgba(0,0,0,0.2)", alignItems: "center" },

    navButton: { background: "#2563eb", color: "white", border: "1px solid rgba(255,255,255,0.7)", borderRadius: "6px", padding: "8px 20px", fontSize: "15px", cursor: "pointer", fontWeight: "500", transition: "0.3s" },

    container: { background: "white", margin: "40px auto", padding: "40px", borderRadius: "12px", width: "90%", maxWidth: "700px", boxShadow: "0 2px 10px rgba(0,0,0,0.1)", textAlign: "center" },

    heading: { color: "#0f172a", fontSize: "26px", marginBottom: "10px", fontWeight: "600" },

    info: { fontSize: "16px", marginBottom: "8px", color: "#334155" },

    dropdown: { padding: "10px", fontSize: "16px", borderRadius: "8px", border: "1px solid #cbd5e1", backgroundColor: "white", color: "#1e293b", outline: "none", transition: "0.3s" },

    button: { marginLeft: "10px", padding: "10px 18px", fontSize: "16px", borderRadius: "8px", border: "none", cursor: "pointer", fontWeight: "600", transition: "0.3s" },

    createBtn: { background: "#3b82f6", color: "white" },

    viewBtn: { background: "#10b981", color: "white", marginTop: "25px", display: "inline-block" },

    noSubjects: { color: "#dc2626", fontWeight: "600" },

  };

  if (!teacher) return null;

  return (

    <>

      <div style={styles.page}>

        {*/\* NAVBAR \*/*}

        <div style={styles.navbar}>

          <button style={styles.navButton} onClick={() => window.location.reload()}>Dashboard</button>

          <button style={styles.navButton} onClick={onViewStudents}>👨‍🎓 View Student Scores</button>

          <button style={styles.navButton} onClick={onViewStudentProfile}>👤 View Student Profile</button>

          <button style={styles.navButton} onClick={() => { localStorage.removeItem("teacher"); onLogout(); }}>Logout</button>

        </div>

        {*/\* MAIN CONTAINER \*/*}

        <div style={styles.container}>

          <h2 style={styles.heading}>Welcome, {teacher.teacherName}</h2>

          <p style={styles.info}><strong>Teacher ID:</strong> {teacher.teacher\_id}</p>

          <p style={styles.info}><strong>Department:</strong> {teacher.department\_name}</p>

          {*/\* SUBJECT SELECT \*/*}

          <div>

            <h4 style={{ color: "#1e293b", marginBottom: "10px" }}>Select a Subject to Create Quiz:</h4>

            {subjects.length === 0 ? (

              <p style={styles.noSubjects}>No subjects available for your department.</p>

            ) : (

              <div style={{ display: "flex", alignItems: "center", justifyContent: "center" }}>

                <select style={styles.dropdown} value={selectedSubject} onChange={(e) => setSelectedSubject(e.target.value)}>

                  {subjects.map((s) => <option key={s.subject\_id} value={s.subject\_id}>{s.subject\_name}</option>)}

                </select>

                <button style={{ ...styles.button, ...styles.createBtn }} onClick={handleCreateQuiz}>➕ Create Quiz</button>

              </div>

            )}

          </div>

          {*/\* BOTTOM GREEN BUTTONS \*/*}

          <button style={{ ...styles.button, ...styles.viewBtn }} onClick={onViewStudents}>👨‍🎓 View Student Scores</button>

          <button style={{ ...styles.button, ...styles.viewBtn }} onClick={onViewStudentProfile}>👤 View Student Profile</button>

        </div>

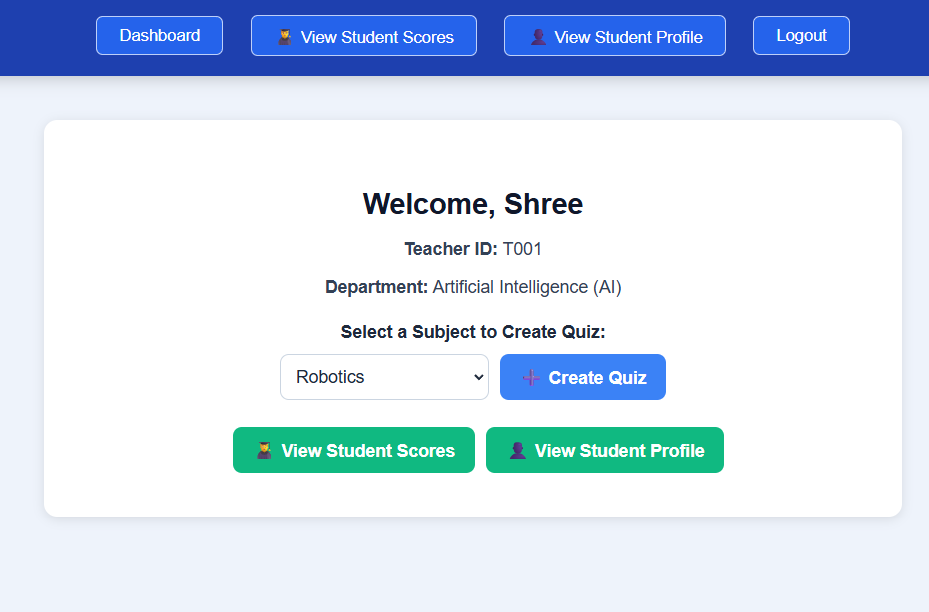
        <ToastContainer position="top-right" autoClose={3000} />

      </div>

    </>

  );

}

****

**Teacher Create Qiuz-**

import React, { useState } from "react";

export default function CreateQuiz({ teacher, subject, onBack }) {

  const [quizName, setQuizName] = useState("");

  const [questions, setQuestions] = useState([]);

*// Add new question*

  const addQuestion = () => {

    setQuestions([

      ...questions,

      {

        question\_text: "",

        options: ["", "", "", ""],

        correctAnswer: "",

        marks: 2,

      },

    ]);

  };

*// Remove question*

  const removeQuestion = (index) => {

    const newQ = questions.filter((\_, idx) => idx !== index);

    setQuestions(newQ);

  };

*// Update question text*

  const handleChangeQuestion = (index, value) => {

    const newQ = [...questions];

    newQ[index].question\_text = value;

    setQuestions(newQ);

  };

*// Update individual option*

  const handleChangeOption = (qIndex, optIndex, value) => {

    const newQ = [...questions];

    newQ[qIndex].options[optIndex] = value;

    setQuestions(newQ);

  };

*// Update correct answer*

  const handleChangeCorrectAnswer = (index, value) => {

    const newQ = [...questions];

    newQ[index].correctAnswer = value;

    setQuestions(newQ);

  };

*// Update marks dynamically*

  const handleChangeMarks = (index, value) => {

    const newQ = [...questions];

    newQ[index].marks = Number(value) || 0;

    setQuestions(newQ);

  };

  const handleSubmit = async () => {

*// Filter out empty questions*

    const cleanQuestions = questions.filter((q) => q.question\_text.trim() !== "");

    if (!quizName || cleanQuestions.length === 0) {

      alert("Please enter quiz name and at least one question.");

      return;

    }

    const totalMarks = cleanQuestions.reduce((sum, q) => sum + q.marks, 0);

    const quizData = {

      quizName,

      teacherId: teacher.\_id,

      subjectId: subject.subject\_id,

      questions: cleanQuestions,

      totalMarks,

      createdAt: new Date(),

    };

    try {

      const res = await fetch("http://localhost:5002/quizzes/create", {

        method: "POST",

        headers: { "Content-Type": "application/json" },

        body: JSON.stringify(quizData),

      });

      const data = await res.json();

      if (data.success) {

        alert("✅ Quiz created successfully!");

        onBack();

      } else {

        alert("❌ Error creating quiz");

      }

    } catch (err) {

      console.error(err);

      alert("⚠️ Server error");

    }

  };

  return (

    <div style={styles.container}>

      <h2 style={styles.heading}>Create Quiz for {subject.subject\_name}</h2>

      <input

        type="text"

        placeholder="Enter Quiz Name"

        value={quizName}

        onChange={(e) => setQuizName(e.target.value)}

        style={styles.quizInput}

      />

      <hr style={styles.divider} />

      <button style={styles.addBtn} onClick={addQuestion}>

        ➕ Add Question

      </button>

      {questions.map((q, i) => (

        <div key={i} style={styles.questionBlock}>

          <div style={{ display: "flex", justifyContent: "space-between" }}>

            <label style={styles.label}>Q{i + 1}</label>

            <button style={styles.removeBtn} onClick={() => removeQuestion(i)}>

              ❌ Remove

            </button>

          </div>

          <input

            type="text"

            placeholder="Enter question"

            value={q.question\_text}

            onChange={(e) => handleChangeQuestion(i, e.target.value)}

            style={styles.questionInput}

          />

          <div style={styles.optionsContainer}>

            {q.options.map((opt, idx) => (

              <input

                key={idx}

                type="text"

                placeholder={`Option ${String.fromCharCode(65 + idx)}`}

                value={opt}

                onChange={(e) => handleChangeOption(i, idx, e.target.value)}

                style={styles.optionInput}

              />

            ))}

          </div>

          <div style={styles.answerRow}>

            <label>Correct Answer:</label>

            <select

              value={q.correctAnswer}

              onChange={(e) => handleChangeCorrectAnswer(i, e.target.value)}

              style={styles.select}

            >

              <option value="">Select</option>

              {q.options.map((\_, idx) => (

                <option key={idx} value={String.fromCharCode(65 + idx)}>

                  {String.fromCharCode(65 + idx)}

                </option>

              ))}

            </select>

            <label style={{ marginLeft: "10px" }}>

              Marks:

              <input

                type="number"

                value={q.marks}

                min={1}

                onChange={(e) => handleChangeMarks(i, e.target.value)}

                style={{ width: "60px", marginLeft: "5px" }}

              />

            </label>

          </div>

        </div>

      ))}

      <hr style={styles.divider} />

      <div style={styles.btnContainer}>

        <button style={styles.submitBtn} onClick={handleSubmit}>

          🚀 Submit Quiz

        </button>

        <button style={styles.backBtn} onClick={onBack}>

          ⬅ Back

        </button>

      </div>

    </div>

  );

}

const styles = {

  container: {

    padding: "25px",

    background: "linear-gradient(135deg, #120625ff, #7597de)",

    color: "#fff",

    fontFamily: "Poppins, sans-serif",

  },

  heading: { textAlign: "center", fontSize: "26px", fontWeight: "bold", color: "#f1c40f" },

  quizInput: { width: "100%", padding: "12px", fontSize: "16px", borderRadius: "8px", border: "none", outline: "none", marginBottom: "15px" },

  addBtn: { backgroundColor: "#00cec9", color: "#fff", padding: "8px 12px", border: "none", borderRadius: "8px", marginBottom: "15px", cursor: "pointer" },

  questionBlock: { backgroundColor: "rgba(22, 10, 10, 0.15)", padding: "15px", borderRadius: "10px", marginBottom: "20px" },

  label: { fontWeight: "bold", fontSize: "18px" },

  removeBtn: { backgroundColor: "#d63031", color: "#fff", border: "none", borderRadius: "6px", cursor: "pointer" },

  questionInput: { width: "100%", padding: "10px", borderRadius: "8px", border: "none", outline: "none", marginTop: "5px", marginBottom: "10px", backgroundColor: "#fff", color: "#333" },

  optionsContainer: { display: "grid", gap: "6px", marginTop: "10px" },

  optionInput: { padding: "8px", borderRadius: "8px", border: "none", outline: "none", backgroundColor: "#f8f9fa", color: "#333" },

  answerRow: { display: "flex", justifyContent: "flex-start", alignItems: "center", marginTop: "10px", gap: "15px" },

  select: { padding: "6px", borderRadius: "8px", border: "none", backgroundColor: "#fff", color: "#333" },

  divider: { margin: "20px 0", border: "1px solid rgba(255,255,255,0.3)" },

  btnContainer: { display: "flex", justifyContent: "center", gap: "15px" },

  submitBtn: { backgroundColor: "#00b894", color: "#fff", padding: "12px 25px", border: "none", borderRadius: "8px", fontSize: "16px", cursor: "pointer", fontWeight: "bold" },

  backBtn: { backgroundColor: "#0984e3", color: "#fff", padding: "12px 25px", border: "none", borderRadius: "8px", fontSize: "16px", cursor: "pointer", fontWeight: "bold" },

};

*// import React, { useState } from "react";*

*// export default function CreateQuiz({ teacher, subject, onBack }) {*

*//   const [quizName, setQuizName] = useState("");*

*//   const [questions, setQuestions] = useState(*

*//     Array.from({ length: 15 }, (\_, idx) => ({*

*//       question: "",*

*//       options: ["", "", "", ""],*

*//       correctAnswer: "",*

*//       marks: 2,*

*//     }))*

*//   );*

*//   // Update question text*

*//   const handleChangeQuestion = (index, value) => {*

*//     const newQ = [...questions];*

*//     newQ[index].question = value;*

*//     setQuestions(newQ);*

*//   };*

*//   // Update individual option*

*//   const handleChangeOption = (qIndex, optIndex, value) => {*

*//     const newQ = [...questions];*

*//     newQ[qIndex].options[optIndex] = value;*

*//     setQuestions(newQ);*

*//   };*

*//   // Update correct answer*

*//   const handleChangeCorrectAnswer = (index, value) => {*

*//     const newQ = [...questions];*

*//     newQ[index].correctAnswer = value;*

*//     setQuestions(newQ);*

*//   };*

*//   const handleSubmit = async () => {*

*//     const cleanQuestions = questions.map((q) => ({*

*//       question\_text: q.question,*

*//       options: q.options,*

*//       correctAnswer: q.correctAnswer,*

*//       marks: Number(q.marks) || 0,*

*//     }));*

*//     const totalMarks = cleanQuestions.reduce((sum, q) => sum + q.marks, 0);*

*//     const quizData = {*

*//       quizName,*

*//       teacherId: teacher.\_id,*

*//       subjectId: subject.subject\_id,*

*//       questions: cleanQuestions,*

*//       totalMarks,*

*//       createdAt: new Date(),*

*//     };*

*//     try {*

*//       const res = await fetch("http://localhost:5002/quizzes/create", {*

*//         method: "POST",*

*//         headers: { "Content-Type": "application/json" },*

*//         body: JSON.stringify(quizData),*

*//       });*

*//       const data = await res.json();*

*//       if (data.success) {*

*//         alert("✅ Quiz created successfully!");*

*//         onBack();*

*//       } else {*

*//         alert("❌ Error creating quiz");*

*//       }*

*//     } catch (err) {*

*//       console.error(err);*

*//       alert("⚠️ Server error");*

*//     }*

*//   };*

*//   return (*

*//     <div style={styles.container}>*

*//       <h2 style={styles.heading}>Create Quiz for {subject.subject\_name}</h2>*

*//       <input*

*//         type="text"*

*//         placeholder="Enter Quiz Name"*

*//         value={quizName}*

*//         onChange={(e) => setQuizName(e.target.value)}*

*//         style={styles.quizInput}*

*//       />*

*//       <hr style={styles.divider} />*

*//       {questions.map((q, i) => (*

*//         <div key={i} style={styles.questionBlock}>*

*//           <label style={styles.label}>Q{i + 1}</label>*

*//           <input*

*//             type="text"*

*//             placeholder="Enter question"*

*//             value={q.question}*

*//             onChange={(e) => handleChangeQuestion(i, e.target.value)}*

*//             style={styles.questionInput}*

*//           />*

*//           <div style={styles.optionsContainer}>*

*//             {q.options.map((opt, idx) => (*

*//               <input*

*//                 key={idx}*

*//                 type="text"*

*//                 placeholder={`Option ${String.fromCharCode(65 + idx)}`}*

*//                 value={opt}*

*//                 onChange={(e) => handleChangeOption(i, idx, e.target.value)}*

*//                 style={styles.optionInput}*

*//               />*

*//             ))}*

*//           </div>*

*//           <div style={styles.answerRow}>*

*//             <label>Correct Answer:</label>*

*//             <select*

*//               value={q.correctAnswer}*

*//               onChange={(e) => handleChangeCorrectAnswer(i, e.target.value)}*

*//               style={styles.select}*

*//             >*

*//               <option value="">Select</option>*

*//               {q.options.map((\_, idx) => (*

*//                 <option key={idx} value={String.fromCharCode(65 + idx)}>*

*//                   {String.fromCharCode(65 + idx)}*

*//                 </option>*

*//               ))}*

*//             </select>*

*//             <span style={styles.marks}>Marks: {q.marks}</span>*

*//           </div>*

*//         </div>*

*//       ))}*

*//       <hr style={styles.divider} />*

*//       <div style={styles.btnContainer}>*

*//         <button style={styles.submitBtn} onClick={handleSubmit}>*

*//           🚀 Submit Quiz*

*//         </button>*

*//         <button style={styles.backBtn} onClick={onBack}>*

*//           ⬅ Back*

*//         </button>*

*//       </div>*

*//     </div>*

*//   );*

*// }*

*// const styles = {*

*//   container: {*

*//     padding: "25px",*

*//     background: "linear-gradient(135deg, #120625ff, #7597de)",*

*//     color: "#fff",*

*//     fontFamily: "Poppins, sans-serif",*

*//   },*

*//   heading: {*

*//     textAlign: "center",*

*//     fontSize: "26px",*

*//     fontWeight: "bold",*

*//     color: "#f1c40f",*

*//   },*

*//   quizInput: {*

*//     width: "100%",*

*//     padding: "12px",*

*//     fontSize: "16px",*

*//     borderRadius: "8px",*

*//     border: "none",*

*//     outline: "none",*

*//     marginBottom: "15px",*

*//   },*

*//   questionBlock: {*

*//     backgroundColor: "rgba(22, 10, 10, 0.15)",*

*//     padding: "15px",*

*//     borderRadius: "10px",*

*//     marginBottom: "20px",*

*//   },*

*//   label: {*

*//     fontWeight: "bold",*

*//     fontSize: "18px",*

*//   },*

*//   questionInput: {*

*//     width: "100%",*

*//     padding: "10px",*

*//     borderRadius: "8px",*

*//     border: "none",*

*//     outline: "none",*

*//     marginTop: "5px",*

*//     marginBottom: "10px",*

*//     backgroundColor: "#fff",*

*//     color: "#333",*

*//   },*

*//   optionsContainer: {*

*//     display: "grid",*

*//     gap: "6px",*

*//     marginTop: "10px",*

*//   },*

*//   optionInput: {*

*//     padding: "8px",*

*//     borderRadius: "8px",*

*//     border: "none",*

*//     outline: "none",*

*//     backgroundColor: "#f8f9fa",*

*//     color: "#333",*

*//   },*

*//   answerRow: {*

*//     display: "flex",*

*//     justifyContent: "space-between",*

*//     alignItems: "center",*

*//     marginTop: "10px",*

*//   },*

*//   select: {*

*//     padding: "6px",*

*//     borderRadius: "8px",*

*//     border: "none",*

*//     backgroundColor: "#fff",*

*//     color: "#333",*

*//   },*

*//   marks: {*

*//     color: "#f1c40f",*

*//     fontWeight: "bold",*

*//   },*

*//   divider: { margin: "20px 0", border: "1px solid rgba(255,255,255,0.3)" },*

*//   btnContainer: {*

*//     display: "flex",*

*//     justifyContent: "center",*

*//     gap: "15px",*

*//   },*

*//   submitBtn: {*

*//     backgroundColor: "#00b894",*

*//     color: "#fff",*

*//     padding: "12px 25px",*

*//     border: "none",*

*//     borderRadius: "8px",*

*//     fontSize: "16px",*

*//     cursor: "pointer",*

*//     fontWeight: "bold",*

*//   },*

*//   backBtn: {*

*//     backgroundColor: "#0984e3",*

*//     color: "#fff",*

*//     padding: "12px 25px",*

*//     border: "none",*

*//     borderRadius: "8px",*

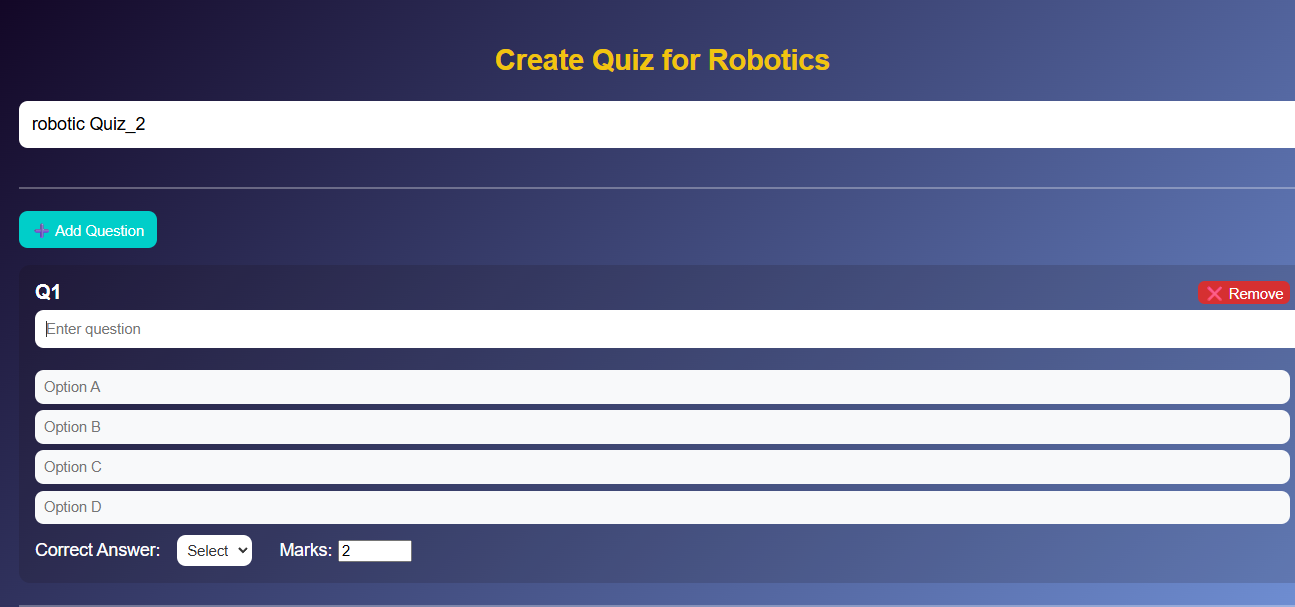
*//     fontSize: "16px",*

*//     cursor: "pointer",*

*//     fontWeight: "bold",*

*//   },*

*// };*

****

**Server.js**

const express = require("express");

const cors = require("cors");

const { MongoClient, ObjectId } = require("mongodb");

const app = express();

const PORT = 5002;

app.use(cors());

app.use(express.json());

*// MongoDB connection*

const uri = "mongodb://127.0.0.1:27017";

const client = new MongoClient(uri);

let db,

  teacherCollection,

  subjectCollection,

  departmentCollection,

  quizCollection,

  questionCollection,

  userCollection;

async function connectDB() {

  try {

    await client.connect();

    db = client.db("Quiz");

    teacherCollection = db.collection("teachers");

    subjectCollection = db.collection("subjects");

    departmentCollection = db.collection("departments");

    quizCollection = db.collection("quizzes");

    questionCollection = db.collection("questions");

    userCollection = db.collection("users");

    console.log("✅ Connected to MongoDB");

  } catch (err) {

    console.error("❌ MongoDB Connection Error:", err);

    process.exit(1);

  }

}

connectDB();

*// ---------- TEACHER LOGIN ----------*

app.post("/login", async (req, res) => {

  try {

    const { email, password } = req.body;

    if (!email || !password)

      return res

        .status(400)

        .json({ success: false, message: "Email and password required" });

    const teacher = await teacherCollection.findOne({

      teacherEmail: email,

      teacherPassword: password,

    });

    if (!teacher)

      return res

        .status(200)

        .json({ success: false, message: "Invalid credentials" });

    res.json({ success: true, teacher });

  } catch (err) {

    res.status(500).json({ success: false, message: err.message });

  }

});

*// ---------- FETCH TEACHER BY EMAIL & SUBJECTS ----------*

app.post("/teachers/fetchByEmail", async (req, res) => {

  try {

    const { email } = req.body;

    if (!email)

      return res.status(400).json({ success: false, message: "Email is required" });

    const teacher = await teacherCollection.findOne({ teacherEmail: email });

    if (!teacher)

      return res.status(200).json({ success: false, message: "Teacher not found" });

    const subjects = await subjectCollection

      .find({ department\_id: teacher.department\_id })

      .toArray();

    res.json({

      success: true,

      teacher: {

        teacher\_id: teacher.teacher\_id,

        teacherName: teacher.teacherName,

        teacherEmail: teacher.teacherEmail,

        department\_id: teacher.department\_id,

        department\_name: teacher.department\_name,

      },

      subjects,

    });

  } catch (err) {

    res.status(500).json({ success: false, message: err.message });

  }

});

*// ---------- FETCH SUBJECTS BY DEPARTMENT ----------*

app.post("/subjects/byDepartment", async (req, res) => {

  try {

    const { department\_id } = req.body;

    if (!department\_id)

      return res

        .status(400)

        .json({ success: false, message: "department\_id is required" });

    const subjects = await subjectCollection

      .find({ department\_id })

      .project({ \_id: 0, subject\_id: 1, subject\_name: 1 })

      .toArray();

    res.json({ success: true, subjects });

  } catch (err) {

    res.status(500).json({ success: false, message: err.message });

  }

});

*// ---------- FETCH ALL SUBJECTS ----------*

app.get("/subjects/display", async (req, res) => {

  try {

    const subjects = await subjectCollection.find({}).toArray();

    res.json(subjects);

  } catch (err) {

    res.status(500).json({ success: false, message: err.message });

  }

});

*// ---------- FETCH DEPARTMENT ----------*

app.get("/departments/:department\_id", async (req, res) => {

  try {

    const { department\_id } = req.params;

    const department = await departmentCollection.findOne({ department\_id });

    if (!department)

      return res.json({ success: false, message: "Department not found" });

    res.json({ success: true, department });

  } catch (err) {

    res.status(500).json({ success: false, message: err.message });

  }

});

*// ---------- CREATE QUIZ + STORE QUESTIONS SEPARATELY WITH QUESTION IDs ----------*

app.post("/quizzes/create", async (req, res) => {

  try {

    const { quizName, teacherId, subjectId, questions, totalMarks } = req.body;

    if (!quizName || !questions || !teacherId || !subjectId) {

      return res

        .status(400)

        .json({ success: false, message: "All fields required" });

    }

*// Create quiz record*

    const quizDoc = {

      quizName,

      teacherId,

      subjectId,

      totalMarks,

      createdAt: new Date(),

    };

    const quizResult = await quizCollection.insertOne(quizDoc);

    const quizId = quizResult.insertedId;

*// Insert each question with a question\_id like Q001, Q002*

    const questionDocs = questions.map((q, idx) => {

      const options = q.options.map((opt) => opt.trim() || "N/A");

      return {

        quizRef: quizId,

        question\_id: `Q${(idx + 1).toString().padStart(3, "0")}`,

        question\_text: q.question\_text,

        options,

        correctAnswer: q.correctAnswer || "N/A",

        marks: Number(q.marks) || 0,

      };

    });

    await questionCollection.insertMany(questionDocs);

    res.json({ success: true, quizId });

  } catch (err) {

    console.error("❌ Quiz creation error:", err);

    res.status(500).json({ success: false, message: "Server error" });

  }

});

*// ---------- FETCH STUDENT BY EMAIL ----------*

app.get("/getStudent/:email", async (req, res) => {

  try {

    const { email } = req.params;

    const student = await userCollection.findOne({ email });

    if (!student) return res.status(404).json({ message: "Student not found" });

    res.json(student);

  } catch (err) {

    res.status(500).json({ message: "Server error" });

  }

});

app.listen(PORT, () => console.log(`🚀 Server running at http://localhost:${PORT}`));

**Package.json**

{

  "name": "server",

  "version": "1.0.0",

  "main": "index.js",

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1"

  },

  "keywords": [],

  "author": "",

  "license": "ISC",

  "description": "",

  "dependencies": {

    "cors": "^2.8.5",

    "express": "^5.1.0",

    "http": "^0.0.1-security",

    "mongodb": "^6.20.0",

    "yup": "^1.7.1"

  }

}

**Student**

**App.js**

import React from "react";

import { BrowserRouter as Router, Routes, Route } from "react-router-dom";

import Home from "./components/Home";

import QuizPage from "./components/QuizPage";

import StudentLogin from "./components/StudentLogin";

import StudentRegister from "./components/StudentRegister";

import Playquiz from "./components/Playquiz";

import QuizList from "./components/QuizList";

import ResultPage from "./components/ResultPage";

function App() {

  return (

    <Router>

      <Routes>

        <Route path="/" element={<Home />} />

        <Route path="/quizpage" element={<QuizPage />} />

        <Route path="/login" element={<StudentLogin />} />

        <Route path="/register" element={<StudentRegister />} />

        <Route path="/quiz-list" element={<QuizList />} />

        <Route path="/start-quiz/:quizId" element={<Playquiz />} />

        <Route path="/result" element={<ResultPage />} /> {*/\* Fixed route \*/*}

      </Routes>

    </Router>

  );

}

export default App;

**Student Home-**

*// src/components/HomePage.js*

import React from "react";

import { Link } from "react-router-dom";

const Home = () => {

  return (

    <div className="home-container">

      {*/\* Navbar \*/*}

      <nav className="navbar">

        <div className="logo">QuizMaster</div>

        <ul className="nav-links">

          <li><a href="#about">About</a></li>

          <li><a href="#contact">Contact</a></li>

          <li><Link to="/register" className="btn-nav">Register</Link></li>

          <li><Link to="/login" className="btn-nav">Login</Link></li>

        </ul>

      </nav>

      {*/\* Hero Section \*/*}

      <section className="hero-section">

        <h1>Challenge Your Knowledge!</h1>

        <p>Test yourself, learn new skills, and enjoy interactive quizzes.</p>

        <div className="hero-buttons">

          <Link to="/register" className="btn">Register</Link>

          <Link to="/login" className="btn">Login</Link>

        </div>

      </section>

      {*/\* About Section \*/*}

      <section className="about-section" id="about">

        <div className="about-text">

          <h2>About QuizMaster</h2>

          <p>

            QuizMaster is an engaging platform that encourages learners to challenge themselves and expand their knowledge.

            With a variety of quizzes covering multiple disciplines, users can improve problem-solving skills,

            critical thinking, and practical knowledge in an interactive and fun way.

          </p>

        </div>

        <div className="about-image">

          <img src="https://source.unsplash.com/400x300/?learning,books" alt="About Illustration" />

        </div>

      </section>

      {*/\* Contact Section \*/*}

      <section className="contact-section" id="contact">

        <div className="contact-text">

          <h2>Contact Us</h2>

          <p>

            Have questions or feedback? Reach out to us and we'll get back to you promptly.

            Your input helps us make QuizMaster better for everyone!

          </p>

          <form className="contact-form">

            <input type="text" placeholder="Your Name" required />

            <input type="email" placeholder="Your Email" required />

            <textarea placeholder="Your Message" required></textarea>

            <button type="submit" className="btn">Send Message</button>

          </form>

        </div>

        <div className="contact-image">

          <img src="https://source.unsplash.com/400x300/?contact,message" alt="Contact Illustration" />

        </div>

      </section>

      {*/\* Footer \*/*}

      <footer className="footer">

        <p>&copy; 2025 QuizMaster. All rights reserved.</p>

        <div className="footer-links">

          <a href="#about">About</a>

          <a href="#contact">Contact</a>

        </div>

      </footer>

      {*/\* Background Animation \*/*}

      <div className="bg-animation">

        <span></span><span></span><span></span><span></span><span></span>

      </div>

      {*/\* CSS \*/*}

      <style>{`

        .home-container {

          font-family: 'Arial', sans-serif;

          color: #f5f5f5;

          position: relative;

          overflow-x: hidden;

          background-color: #0d0d0d;

        }

        /\* Navbar \*/

        .navbar {

          display: flex;

          justify-content: space-between;

          align-items: center;

          padding: 10px 25px;

          background: rgba(10,10,10,0.95);

          border-bottom: 2px solid #ffd700;

          box-shadow: 0 4px 15px rgba(0,0,0,0.5);

          font-weight: 500;

          position: fixed;

          width: 95%;

          top: 0;

          z-index: 100;

        }

        .navbar .logo { font-size: 20px; font-weight: 400; color: #ffd700; letter-spacing: 1px; }

        .navbar .nav-links { list-style: none; display: flex; gap: 5px; }

        .navbar .nav-links li a {

          color: #fff;

          text-decoration: none;

          font-size: 14px;

          transition: 0.3s;

        }

        .navbar .nav-links li a:hover { color: #ffd700; text-shadow: 0 0 5px #ffd700; }

        .btn-nav {

          background: #ffd700;

          color: #0d0d0d;

          padding: 5px 10px;

          border-radius: 6px;

          font-weight: bold;

          transition: 0.3s;

        }

        .btn-nav:hover { background: #e6c200; transform: scale(1.05); }

        /\* Hero Section \*/

        .hero-section {

          display: flex;

          flex-direction: column;

          justify-content: center;

          align-items: center;

          height: 100vh;

          text-align: center;

          background: linear-gradient(rgba(0,0,0,0.5), rgba(0,0,0,0.5)),

                      url('https://source.unsplash.com/1600x900/?quiz,learning') center/cover no-repeat;

          padding-top: 55px;

        }

        .hero-section h1 { font-size: 48px; margin-bottom: 20px; animation: fadeIn 2s ease-in-out; text-shadow: 0 0 15px #ffd700; }

        .hero-section p { font-size: 18px; margin-bottom: 30px; color: #e0e0e0; }

        .hero-buttons { display: flex; gap: 20px; flex-wrap: wrap; }

        .btn {

          padding: 12px 28px;

          background-color: #ffd700;

          color: #0d0d0d;

          border: none;

          border-radius: 10px;

          text-decoration: none;

          font-weight: bold;

          transition: 0.3s;

          box-shadow: 0 5px 20px rgba(255,215,0,0.4);

        }

        .btn:hover { background-color: #e6c200; transform: scale(1.08); box-shadow: 0 8px 25px rgba(255,215,0,0.6); }

        /\* Sections \*/

        .about-section, .contact-section { display: flex; align-items: center; justify-content: center; gap: 50px; padding: 80px 50px; flex-wrap: wrap; }

        .about-section { background: #111111; }

        .contact-section { background: #111111; }

        .about-text, .contact-text { max-width: 500px; }

        h2 { font-size: 36px; margin-bottom: 20px; color: #ffd700; text-shadow: 0 0 5px #ffd700; }

        p { font-size: 16px; line-height: 1.5; color: #e0e0e0; }

        .about-image img, .contact-image img { width: 400px; border-radius: 15px; box-shadow: 0 5px 25px rgba(0,0,0,0.6); transition: transform 0.4s ease, box-shadow 0.4s ease; }

        .about-image img:hover, .contact-image img:hover { transform: scale(1.06); box-shadow: 0 8px 35px rgba(255,215,0,0.4); }

        /\* Contact Form \*/

        .contact-form { display: flex; flex-direction: column; gap: 15px; margin-top: 20px; }

        .contact-form input, .contact-form textarea { padding: 12px; border-radius: 10px; border: none; font-size: 15px; background: #222; color: #fff; box-shadow: inset 0 2px 5px rgba(0,0,0,0.5); }

        .contact-form input::placeholder, .contact-form textarea::placeholder { color: #ccc; }

        .contact-form button { cursor: pointer; }

        /\* Footer \*/

        .footer { background: #000; padding: 25px 50px; text-align: center; }

        .footer-links a { color: #ffd700; margin: 0 12px; text-decoration: none; transition: 0.3s; }

        .footer-links a:hover { text-decoration: underline; }

        /\* Background Animation \*/

        .bg-animation span { position: absolute; display: block; width: 20px; height: 20px; background: rgba(255, 215, 0, 0.25); animation: animate 18s linear infinite; bottom: -150px; }

        .bg-animation span:nth-child(1) { left: 10%; animation-delay: 0s; }

        .bg-animation span:nth-child(2) { left: 30%; animation-delay: 3s; }

        .bg-animation span:nth-child(3) { left: 50%; animation-delay: 6s; }

        .bg-animation span:nth-child(4) { left: 70%; animation-delay: 9s; }

        .bg-animation span:nth-child(5) { left: 90%; animation-delay: 12s; }

        @keyframes animate { 0% { transform: translateY(0) rotate(0deg); opacity: 1; } 100% { transform: translateY(-1000px) rotate(720deg); opacity: 0; } }

        @keyframes fadeIn { 0% { opacity: 0; transform: translateY(-20px); } 100% { opacity: 1; transform: translateY(0); } }

        @media screen and (max-width: 768px) {

          .about-section, .contact-section { flex-direction: column; text-align: center; }

          .about-image img, .contact-image img { width: 300px; }

          .hero-section h1 { font-size: 36px; }

          .hero-buttons { flex-direction: column; gap: 15px; }

          .navbar .nav-links { gap: 12px; }

        }

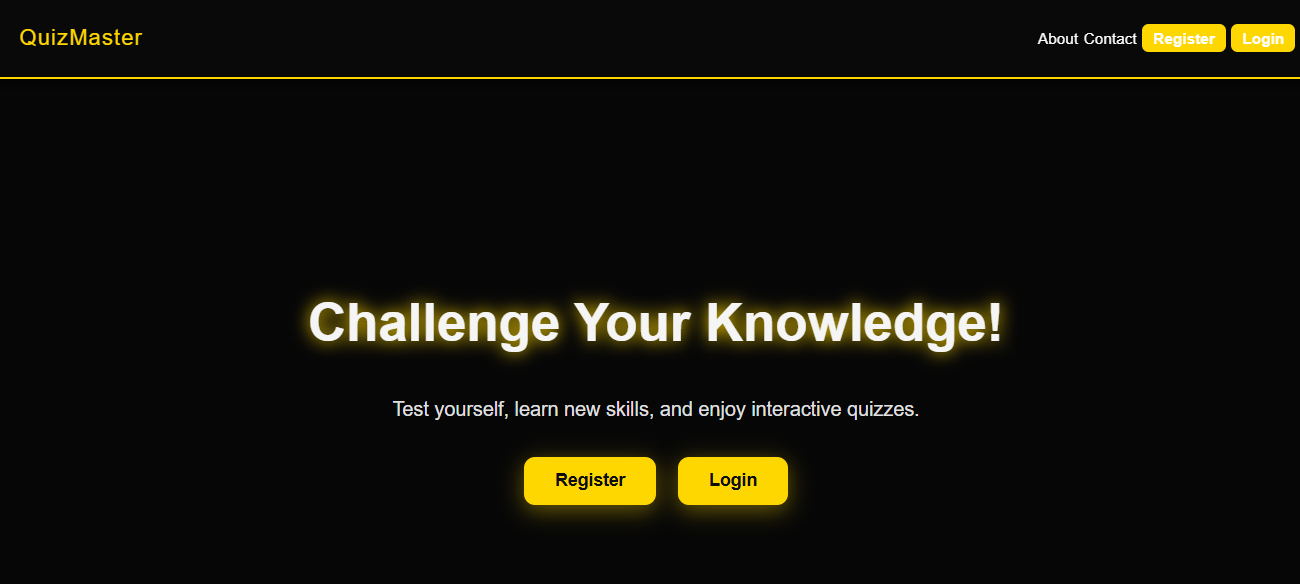
      `}</style>

    </div>

  );

};

export default Home;



**Student Registration**

*// src/components/StudentLogin.js*

import React, { useState } from "react";

import \* as yup from "yup";

import { useNavigate } from "react-router-dom";

const StudentLogin = () => {

  const [form, setForm] = useState({ email: "", password: "" });

  const [message, setMessage] = useState("");

  const navigate = useNavigate();

*// ✅ Validation Schema*

  const schema = yup.object().shape({

    email: yup.string().email("Invalid email").required("Email is required"),

    password: yup.string().required("Password is required"),

  });

*// ✅ Handle Change*

  const handleChange = (e) => {

    setForm({ ...form, [e.target.name]: e.target.value });

  };

*// ✅ Handle Submit*

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

      await schema.validate(form, { abortEarly: false });

      const response = await fetch("http://localhost:5003/Users/login", {

        method: "POST",

        headers: { "Content-Type": "application/json" },

        body: JSON.stringify(form),

      });

      const data = await response.json();

      if (response.ok) {

        alert("🎉 Login Successful!");

*// Optionally, save token or user info*

        localStorage.setItem("studentData", JSON.stringify(data));

        navigate("/quizpage"); *// Redirect to quiz page*

      } else {

        setMessage(data.message || "Invalid credentials");

      }

    } catch (err) {

      if (err.inner) {

        setMessage(err.inner.map((e) => e.message).join("\n"));

      } else {

        setMessage(err.message);

      }

    }

  };

  return (

    <div className="login-container">

      <h2>Student Login</h2>

      <form className="login-form" onSubmit={handleSubmit}>

        <input

          type="email"

          name="email"

          placeholder="Enter Email"

          value={form.email}

          onChange={handleChange}

        />

        <input

          type="password"

          name="password"

          placeholder="Enter Password"

          value={form.password}

          onChange={handleChange}

        />

        <button type="submit">Login</button>

      </form>

      {message && <p className="error">{message}</p>}

      <style>{`

        .login-container {

          background: linear-gradient(to right, #141E30, #243B55);

          color: white;

          min-height: 100vh;

          display: flex;

          flex-direction: column;

          align-items: center;

          justify-content: center;

          padding: 20px;

        }

        h2 {

          font-size: 2rem;

          margin-bottom: 1rem;

        }

        .login-form {

          background: rgba(255, 255, 255, 0.1);

          padding: 30px;

          border-radius: 10px;

          display: flex;

          flex-direction: column;

          width: 100%;

          max-width: 400px;

          box-shadow: 0 4px 15px rgba(0,0,0,0.3);

        }

        input {

          margin: 10px 0;

          padding: 10px;

          border: none;

          border-radius: 5px;

          outline: none;

          font-size: 1rem;

        }

        input:focus {

          border: 2px solid #00BFFF;

        }

        button {

          margin-top: 15px;

          padding: 10px;

          border: none;

          border-radius: 5px;

          background: #00BFFF;

          color: white;

          font-size: 1rem;

          cursor: pointer;

          transition: 0.3s;

        }

        button:hover {

          background: #008CBA;

        }

        .error {

          color: #ff5555;

          margin-top: 10px;

          font-size: 0.9rem;

          text-align: center;

          white-space: pre-line;

        }

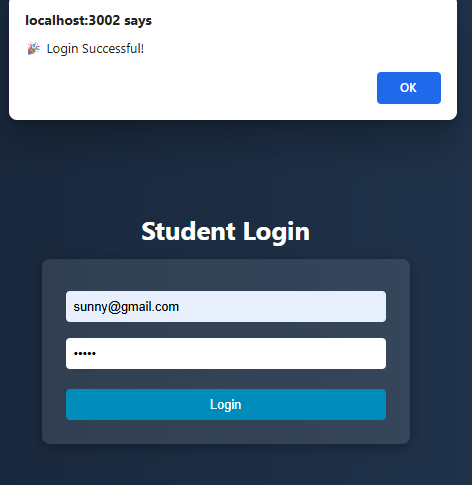
      `}</style>

    </div>

  );

};

export default StudentLogin;



**Student Dashboard-**

import React, { useEffect, useState } from "react";

import { useNavigate } from "react-router-dom";

export default function StudentQuiz() {

  const [student, setStudent] = useState(null);

  const navigate = useNavigate();

  useEffect(() => {

*// Get student data from localStorage*

    const data = JSON.parse(localStorage.getItem("studentData"));

    if (!data) {

      navigate("/StudentLogin"); *// Redirect if not logged in*

      return;

    }

    setStudent(data);

  }, [navigate]);

  const goToQuizList = () => {

    navigate("/quiz-list"); *// Navigate straight to quiz list page*

  };

  return (

    <div style={styles.container}>

      <h2 style={styles.heading}>📚 Student Dashboard</h2>

      {student && (

        <div style={styles.studentBox}>

          <p><strong>Name:</strong> {student.username}</p>

          <p><strong>Degree:</strong> {student.degree}</p>

          <p><strong>Email:</strong> {student.email}</p>

        </div>

      )}

      <button style={styles.startBtn} onClick={goToQuizList}>

        🎮 Play Quizzes

      </button>

    </div>

  );

}

*// Inline CSS*

const styles = {

  container: {

    width: "700px",

    margin: "50px auto",

    padding: "25px",

    background: "#111",

    color: "white",

    borderRadius: "12px",

    boxShadow: "0 0 10px rgba(255,255,255,0.2)",

  },

  heading: {

    textAlign: "center",

    marginBottom: "20px",

  },

  studentBox: {

    background: "#222",

    padding: "15px",

    borderRadius: "8px",

    marginBottom: "20px",

    lineHeight: "1.6",

  },

  startBtn: {

    width: "100%",

    padding: "12px",

    background: "black",

    border: "2px solid white",

    color: "white",

    borderRadius: "8px",

    cursor: "pointer",

    fontSize: "18px",

  },

};

*// import React, { useEffect, useState } from "react";*

*// import axios from "axios";*

*// import { useNavigate } from "react-router-dom";*

*// export default function StudentQuiz() {*

*//   const [student, setStudent] = useState(null);*

*//   const [subjects, setSubjects] = useState([]);*

*//   const [selectedSubject, setSelectedSubject] = useState("");*

*//   const navigate = useNavigate();*

*//   useEffect(() => {*

*//     // Get student data from localStorage*

*//     const data = JSON.parse(localStorage.getItem("studentData"));*

*//     if (!data) {*

*//       navigate("/StudentLogin"); // Redirect if not logged in*

*//       return;*

*//     }*

*//     setStudent(data);*

*//     // Fetch all subjects from backend*

*//     axios*

*//       .get("http://localhost:5003/subjects/display") // Matches updated backend route*

*//       .then((res) => setSubjects(res.data))*

*//       .catch((err) => console.log("Error fetching subjects:", err));*

*//   }, [navigate]);*

*//   const startQuiz = () => {*

*//     if (!selectedSubject) {*

*//       alert("Please select a subject!");*

*//       return;*

*//     }*

*//     navigate(`/quiz/start/${selectedSubject}`);*

*//   };*

*//   return (*

*//     <div style={styles.container}>*

*//       <h2 style={styles.heading}>📚 Student Dashboard</h2>*

*//       {student && (*

*//         <div style={styles.studentBox}>*

*//           <p><strong>Name:</strong> {student.username}</p>*

*//           <p><strong>Degree:</strong> {student.degree}</p>*

*//           <p><strong>Email:</strong> {student.email}</p>*

*//         </div>*

*//       )}*

*//       <h3 style={styles.subHeading}>Select Subject</h3>*

*//       {/\* Dropdown Box \*/}*

*//       <select*

*//         style={styles.dropdown}*

*//         value={selectedSubject}*

*//         onChange={(e) => setSelectedSubject(e.target.value)}*

*//       >*

*//         <option value="">-- Choose Subject --</option>*

*//         {subjects.map((sub) => (*

*//           <option key={sub.\_id} value={sub.subject\_name}>*

*//             {sub.subject\_name}*

*//           </option>*

*//         ))}*

*//       </select>*

*//       <button style={styles.startBtn} onClick={startQuiz}>*

*//         Start Quiz*

*//       </button>*

*//     </div>*

*//   );*

*// }*

*// // Inline CSS*

*// const styles = {*

*//   container: {*

*//     width: "700px",*

*//     margin: "50px auto",*

*//     padding: "25px",*

*//     background: "#111",*

*//     color: "white",*

*//     borderRadius: "12px",*

*//     boxShadow: "0 0 10px rgba(255,255,255,0.2)",*

*//   },*

*//   heading: {*

*//     textAlign: "center",*

*//     marginBottom: "20px",*

*//   },*

*//   studentBox: {*

*//     background: "#222",*

*//     padding: "15px",*

*//     borderRadius: "8px",*

*//     marginBottom: "20px",*

*//     lineHeight: "1.6",*

*//   },*

*//   subHeading: {*

*//     marginTop: "20px",*

*//     marginBottom: "10px",*

*//     borderBottom: "1px solid white",*

*//     paddingBottom: "5px",*

*//   },*

*//   dropdown: {*

*//     width: "100%",*

*//     padding: "12px",*

*//     fontSize: "16px",*

*//     borderRadius: "8px",*

*//     background: "#333",*

*//     color: "white",*

*//     border: "1px solid white",*

*//     marginBottom: "20px",*

*//   },*

*//   startBtn: {*

*//     width: "100%",*

*//     padding: "12px",*

*//     background: "black",*

*//     border: "2px solid white",*

*//     color: "white",*

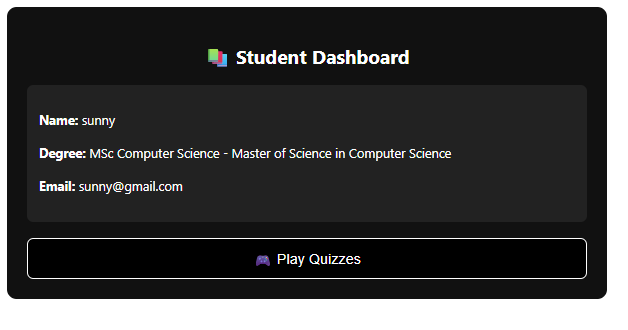
*//     borderRadius: "8px",*

*//     cursor: "pointer",*

*//     fontSize: "18px",*

*//   },*

*// };*



**QuizList.js**

import React, { useEffect, useState } from "react";

import axios from "axios";

import { useNavigate } from "react-router-dom";

export default function QuizList() {

  const [subjects, setSubjects] = useState([]);

  const [selectedSubject, setSelectedSubject] = useState(null);

  const [quizzes, setQuizzes] = useState([]);

  const [loadingQuizzes, setLoadingQuizzes] = useState(false);

  const navigate = useNavigate();

*// Load subjects*

  useEffect(() => {

    const fetchSubjects = async () => {

      try {

        const res = await axios.get("http://localhost:5003/subjects/display");

        setSubjects(res.data);

      } catch (err) {

        console.error("Error loading subjects:", err);

      }

    };

    fetchSubjects();

  }, []);

*// Load quizzes for selected subject*

  const loadQuizzes = async (subject) => {

    setSelectedSubject(subject);

    setQuizzes([]);

    setLoadingQuizzes(true);

    const subjectId = subject.subject\_id; *// Use correct field*

    try {

      const res = await axios.get(

        `http://localhost:5003/quizzes/by-subject/${subjectId}`

      );

      setQuizzes(res.data);

    } catch (err) {

      console.error("Error loading quizzes:", err);

    } finally {

      setLoadingQuizzes(false);

    }

  };

*// Navigate to PlayQuiz page*

  const startQuiz = (quiz) => {

    navigate(`/start-quiz/${quiz.\_id}`);

  };

  return (

    <div style={styles.container}>

      <h1 style={styles.title}>📚 Quiz Selection</h1>

      {!selectedSubject && (

        <>

          <h2 style={styles.subTitle}>Select a Subject</h2>

          <div style={styles.grid}>

            {subjects.map((sub) => (

              <div

                key={sub.\_id}

                style={styles.subjectCard}

                onClick={() => loadQuizzes(sub)}

              >

                <h3 style={styles.subjectName}>{sub.subject\_name}</h3>

              </div>

            ))}

          </div>

        </>

      )}

      {selectedSubject && (

        <>

          <button

            style={styles.backBtn}

            onClick={() => {

              setSelectedSubject(null);

              setQuizzes([]);

            }}

          >

            ⬅ Back to Subjects

          </button>

          <h2 style={styles.subTitle}>

            📝 Quizzes for {selectedSubject.subject\_name}

          </h2>

          {loadingQuizzes ? (

            <p>Loading quizzes...</p>

          ) : quizzes.length === 0 ? (

            <p style={styles.noQuiz}>No quizzes available.</p>

          ) : (

            <div style={styles.quizGrid}>

              {quizzes.map((quiz) => (

                <div key={quiz.\_id} style={styles.quizCard}>

                  <h3>{quiz.quizName}</h3>

                  <p>Marks: {quiz.totalMarks}</p>

                  <button

                    style={styles.startBtn}

                    onClick={() => startQuiz(quiz)}

                  >

                    Start Quiz

                  </button>

                </div>

              ))}

            </div>

          )}

        </>

      )}   </div>);}

const styles = {

  container: { width: "850px", margin: "40px auto", padding: "20px", background: "#111", color: "white", borderRadius: "12px" },

  title: { textAlign: "center", marginBottom: "25px" },

  subTitle: { marginBottom: "20px", borderBottom: "1px solid white", paddingBottom: "5px" },

  grid: { display: "grid", gridTemplateColumns: "repeat(4, 1fr)", gap: "12px" },

  subjectCard: { background: "#222", padding: "15px", borderRadius: "8px", border: "1px solid white", cursor: "pointer", textAlign: "center", height: "90px", display: "flex", alignItems: "center", justifyContent: "center" },

  subjectName: { fontSize: "16px", fontWeight: "bold" },

  backBtn: { background: "black", border: "1px solid white", padding: "8px 15px", color: "white", borderRadius: "6px", cursor: "pointer", marginBottom: "20px" },

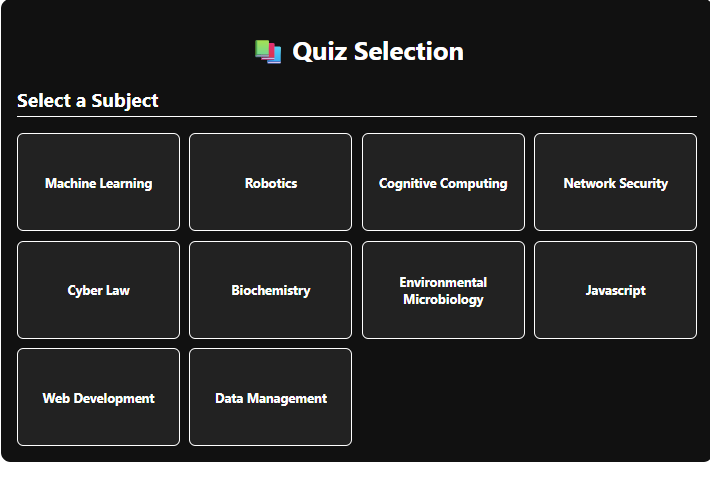
  quizGrid: { display: "grid", gridTemplateColumns: "repeat(3, 1fr)", gap: "15px" },

  quizCard: { background: "#222", padding: "15px", borderRadius: "10px", border: "1px solid white", textAlign: "center" },

  startBtn: { marginTop: "10px", padding: "10px", width: "100%", background: "black", color: "white", border: "1px solid white", borderRadius: "8px", cursor: "pointer" },

  noQuiz: { textAlign: "center", opacity: 0.6 },

};



**PlayQuiz.js**

import React, { useEffect, useState, useCallback } from "react";

import { useParams, useNavigate } from "react-router-dom";

import axios from "axios";

export default function Playquiz() {

  const { quizId } = useParams();

  const navigate = useNavigate();

  const [quiz, setQuiz] = useState(null);

  const [loading, setLoading] = useState(true);

  const [error, setError] = useState("");

  const [userAnswers, setUserAnswers] = useState([]);

  const [score, setScore] = useState(null);

  const [currentStep, setCurrentStep] = useState(0);

  const [timeLeft, setTimeLeft] = useState(10 \* 60);

*// ---------------- STUDENT DETAILS ----------------*

  const storedStudentId = localStorage.getItem("studentId");

  const [student, setStudent] = useState(null);

  useEffect(() => {

    if (!storedStudentId) return;

    axios

      .get(`http://localhost:5003/student/${storedStudentId}`)

      .then((res) => setStudent(res.data))

      .catch(() => console.log("Cannot fetch student"));

  }, [storedStudentId]);

*// ------------------ FETCH QUIZ -------------------*

  useEffect(() => {

    const fetchQuiz = async () => {

      try {

        const res = await axios.get(`http://localhost:5003/quizzes/full/${quizId}`);

        setQuiz(res.data);

        setUserAnswers(Array(res.data.questions.length).fill(""));

        setLoading(false);

      } catch (err) {

        console.error("Error fetching quiz:", err);

        setError("Failed to load quiz.");

        setLoading(false);

      }

    };

    fetchQuiz();

  }, [quizId]);

*// ---------------- SAVE RESULT --------------------*

  const saveResult = async (finalScore) => {

    try {

      await axios.post("http://localhost:5003/save-result", {

        studentName: student?.username,

        studentId: storedStudentId,

        quizName: quiz.quizName,

        subject: quiz.subjectId,

        score: finalScore,

        totalMarks: quiz.totalMarks,

        quizId,

      });

    } catch (err) {

      console.error("Error saving result:", err);

    }

  };

*// ---------------- SUBMIT QUIZ ---------------------*

  const autoSubmit = useCallback(async () => {

    try {

      const res = await axios.post("http://localhost:5003/submit-quiz", {

        userAnswers,

        questions: quiz.questions,

      });

      const finalScore = res.data.score;

      setScore(finalScore);

      await saveResult(finalScore);

*// Navigate to Result page*

      navigate("/result", {

        state: {

          studentName: student?.username,

          quizName: quiz.quizName,

          subject: quiz.subjectId,

          score: finalScore,

          totalMarks: quiz.totalMarks,

        },

      });

    } catch (err) {

      console.error("Auto-submit error:", err);

    }

  }, [userAnswers, quiz, student, navigate]);

*// ---------------- TIMER ---------------------------*

  useEffect(() => {

    if (score !== null) return;

    const timer = setInterval(() => {

      setTimeLeft((t) => {

        if (t <= 1) {

          autoSubmit();

          clearInterval(timer);

          return 0;

        }

        return t - 1;

      });

    }, 1000);

    return () => clearInterval(timer);

  }, [score, autoSubmit]);

*// ---------------- CHANGE ANSWERS -------------------*

  const handleAnswerChange = (index, answer) => {

    const updated = [...userAnswers];

    updated[index] = answer;

    setUserAnswers(updated);

  };

*// Manual Submit*

  const submitQuiz = async () => {

    autoSubmit();

  };

*// ---------------- TIME FORMAT ---------------------*

  const formatTime = (sec) => {

    const m = String(Math.floor(sec / 60)).padStart(2, "0");

    const s = String(sec % 60).padStart(2, "0");

    return `${m}:${s}`;

  };

*// ---------------- UI RENDER ------------------------*

  if (loading) return <p className="whiteText pad20">Loading quiz...</p>;

  if (error) return <p className="whiteText pad20">{error}</p>;

  return (

    <div className="quizContainer">

      <style>{`

        body { background-color: #001f3f; }

        .quizContainer {

          min-height: 100vh;

          display: flex;

          justify-content: center;

          align-items: center;

          flex-direction: column;

          padding: 20px;

          color: white;

        }

        .quizBox {

          width: 90%;

          max-width: 700px;

          background: #003366;

          padding: 25px;

          border-radius: 15px;

          box-shadow: 0 0 15px rgba(0,255,255,0.3);

        }

        .stepper {

          display: flex;

          gap: 8px;

          flex-wrap: wrap;

          margin-bottom: 15px;

        }

        .step {

          width: 40px;

          height: 40px;

          border-radius: 50%;

          background: #004080;

          display: flex;

          justify-content: center;

          align-items: center;

          cursor: pointer;

          font-weight: bold;

          color: white;

          transition: 0.2s;

        }

        .step.active { background: aqua; color: black; }

        .step.answered { background: #0099cc; }

        .question { margin-bottom: 15px; font-size: 18px; line-height: 1.4; }

        .optionBox {

          margin-bottom: 10px;

          padding: 10px;

          background: #00284d;

          border-radius: 8px;

          display: flex;

          align-items: center;

        }

        .optionBox label { margin-left: 10px; cursor: pointer; }

        .navButtons {

          margin-top: 20px;

          display: flex;

          justify-content: space-between;

        }

        .btn {

          background: aqua;

          padding: 10px 20px;

          border: none;

          border-radius: 10px;

          cursor: pointer;

          font-weight: bold;

          color: black;

          transition: 0.2s;

        }

        .btn:hover { transform: scale(1.05); }

        .timer {

          border: 3px solid aqua;

          padding: 10px 20px;

          border-radius: 30px;

          font-size: 20px;

          font-weight: bold;

          margin-bottom: 20px;

          box-shadow: 0 0 12px aqua;

        }

      `}</style>

      {student && (

        <div style={{ fontSize: "20px", marginBottom: "10px" }}>

          👤 Student: <b>{student.username}</b> ({student.email})

        </div>

      )}

      <div className="timer">⏳ Time Left: {formatTime(timeLeft)}</div>

      <div className="quizBox">

        <h2 style={{ textAlign: "center" }}>📝 {quiz.quizName}</h2>

        <div className="stepper">

          {quiz.questions.map((q, index) => (

            <div

              key={index}

              className={`step ${index === currentStep ? "active" : ""} ${

                userAnswers[index] ? "answered" : ""

              }`}

              onClick={() => setCurrentStep(index)}

            >

              {index + 1}

            </div>

          ))}

        </div>

        <p className="question">

          <strong>Q{currentStep + 1}:</strong> {quiz.questions[currentStep].question\_text}

        </p>

        {quiz.questions[currentStep].options.map((opt, i) => (

          <div className="optionBox" key={i}>

            <input

              type="radio"

              id={`q${currentStep}a${i}`}

              name={`q${currentStep}`}

              checked={userAnswers[currentStep] === opt}

              onChange={() => handleAnswerChange(currentStep, opt)}

            />

            <label htmlFor={`q${currentStep}a${i}`}>{opt}</label>

          </div>

        ))}

        <div className="navButtons">

          <button

            className="btn"

            disabled={currentStep === 0}

            onClick={() => setCurrentStep(currentStep - 1)}

          >

            Previous

          </button>

          {currentStep < quiz.questions.length - 1 ? (

            <button className="btn" onClick={() => setCurrentStep(currentStep + 1)}>

              Next

            </button>

          ) : (

            <button className="btn" onClick={submitQuiz}>

              Submit Quiz

            </button>

          )}

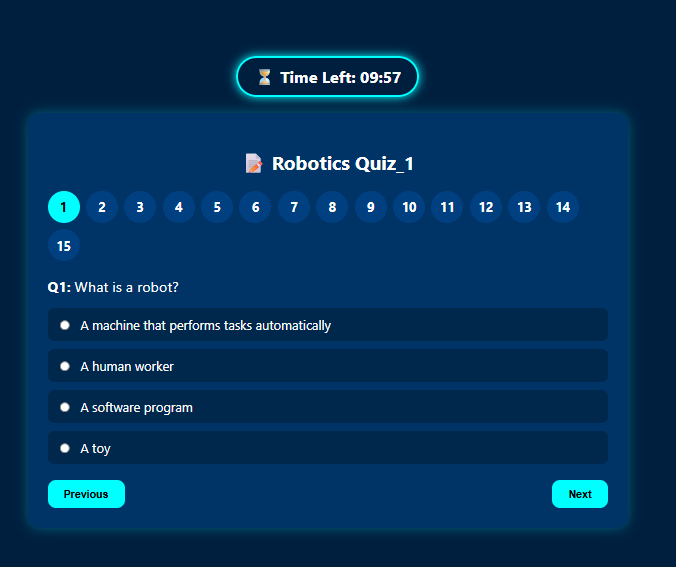
        </div>

      </div>

    </div>

  );

}



**Result.js**

import React, { useEffect, useState } from "react";

import { useLocation, useNavigate } from "react-router-dom";

export default function ResultPage() {

  const { state } = useLocation();

  const navigate = useNavigate();

  const [student, setStudent] = useState(null);

*// Get student from localStorage*

  useEffect(() => {

    const data = JSON.parse(localStorage.getItem("studentData"));

    if (!data) {

      navigate("/login");

      return;

    }

    setStudent(data);

  }, [navigate]);

  if (!state || !student) {

    return <p style={{ color: "white", textAlign: "center", marginTop: "50px" }}>Loading...</p>;

  }

  return (

    <div style={{ padding: "40px", backgroundColor: "#001f3f", minHeight: "100vh", color: "white", textAlign: "center" }}>

      <h1>📝 Quiz Result</h1>

      {*/\* Student Info \*/*}

      <div style={{ fontSize: "22px", margin: "20px 0" }}>

        <p>👤 <strong>Name:</strong> {student.username}</p>

        <p>🎓 <strong>Degree:</strong> {student.degree}</p>

        <p>📧 <strong>Email:</strong> {student.email}</p>

      </div>

      {*/\* Quiz Info \*/*}

      <div style={{ fontSize: "22px", margin: "20px 0" }}>

        <p>📚 <strong>Quiz Name:</strong> {state.quizName}</p>

        <p>✅ <strong>Score:</strong> {state.score} / {state.totalMarks}</p>

      </div>

      <button

        style={{

          padding: "12px 25px",

          background: "aqua",

          border: "none",

          borderRadius: "10px",

          cursor: "pointer",

          color: "black",

          fontWeight: "bold",

          fontSize: "18px",

          marginTop: "30px",

        }}

        onClick={() => navigate("/quiz-list")}

      >

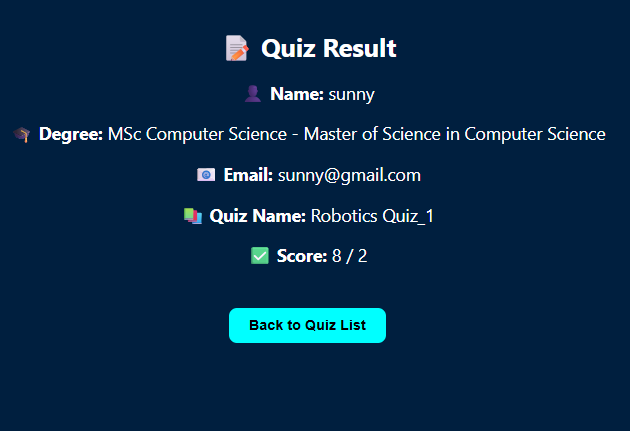
        Back to Quiz List

      </button>

    </div>

  );

}



**Server.js**

const http = require("http");

const { MongoClient, ObjectId } = require("mongodb");

*// ---- MONGO CONNECTION ----*

const url = "mongodb://127.0.0.1:27017";

const dbName = "Quiz";

let db;

MongoClient.connect(url)

  .then((client) => {

    db = client.db(dbName);

    console.log("✅ MongoDB connected to:", dbName);

    startServer();

  })

  .catch((err) => console.log("❌ MongoDB Error:", err));

*// ---- START SERVER ----*

function startServer() {

  const server = http.createServer((req, res) => {

*// CORS*

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

    res.setHeader("Access-Control-Allow-Methods", "GET, POST, OPTIONS");

    res.setHeader("Access-Control-Allow-Headers", "Content-Type");

    if (req.method === "OPTIONS") {

      res.writeHead(200);

      res.end();

      return;

    }

    let body = "";

    req.on("data", (chunk) => (body += chunk.toString()));

    const sendJSON = (status, data) => {

      res.writeHead(status, { "Content-Type": "application/json" });

      res.end(JSON.stringify(data));

    };

    req.on("end", async () => {

      try {

        let parsed = {};

        if (body.length > 0) parsed = JSON.parse(body);

*// --------------------------------------------------------*

*// 1️⃣ AUTO-INCREMENT STUDENT ID*

*// --------------------------------------------------------*

        if (req.url === "/Users/nextId" && req.method === "GET") {

          const collection = db.collection("Users");

          const last = await collection.find().sort({ studentId: -1 }).limit(1).toArray();

          const nextId = last.length === 0 ? 1 : last[0].studentId + 1;

          return sendJSON(200, { nextId });

        }

*// --------------------------------------------------------*

*// 2️⃣ USER REGISTRATION*

*// --------------------------------------------------------*

        if (req.url === "/Users/register" && req.method === "POST") {

          const { studentId, username, email, password, degree, collegeName } = parsed;

          const exists = await db.collection("Students").findOne({ email });

          if (exists) return sendJSON(400, { message: "Email already registered" });

          await db.collection("Users").insertOne({ studentId, username, email, password, degree, collegeName });

          return sendJSON(200, { message: "Registration successful" });

        }

*// --------------------------------------------------------*

*// 3️⃣ USER LOGIN*

*// --------------------------------------------------------*

        if (req.url === "/Users/login" && req.method === "POST") {

          const { email, password } = parsed;

          const user = await db.collection("Users").findOne({ email, password });

          if (!user) return sendJSON(401, { message: "Invalid credentials" });

          return sendJSON(200, {

            message: "Login successful",

            username: user.username,

            degree: user.degree,

            email: user.email,

            studentId: user.studentId

          });

        }

*// --------------------------------------------------------*

*// 4️⃣ DISPLAY SUBJECTS*

*// --------------------------------------------------------*

        if (req.url === "/subjects/display" && req.method === "GET") {

          const subjects = await db.collection("subjects").find({}).toArray();

          return sendJSON(200, subjects);

        }

*// --------------------------------------------------------*

*// 5️⃣ QUIZZES BY SUBJECT*

*// --------------------------------------------------------*

        if (req.url.startsWith("/quizzes/by-subject/") && req.method === "GET") {

          const subjectId = req.url.split("/")[3];

          const quizzes = await db.collection("quizzes").find({ subjectId }).toArray();

          return sendJSON(200, quizzes);

        }

*// --------------------------------------------------------*

*// 6️⃣ GET QUESTIONS BY QUIZ ID*

*// --------------------------------------------------------*

        if (req.url.startsWith("/quiz/questions/") && req.method === "GET") {

          let quizId = req.url.split("/")[3];

          try { quizId = new ObjectId(quizId); } catch { return sendJSON(400, { error: "Invalid quiz ID" }); }

          const questions = await db.collection("questions").find({ quizRef: quizId }).toArray();

          return sendJSON(200, questions);

        }

*// --------------------------------------------------------*

*// 7️⃣ FULL QUIZ DETAILS (Quiz + Questions)*

*// --------------------------------------------------------*

        if (req.url.startsWith("/quizzes/full/") && req.method === "GET") {

          let quizId = req.url.split("/")[3];

          let objId;

          try { objId = new ObjectId(quizId); } catch { return sendJSON(400, { error: "Invalid quiz ID" }); }

          const quiz = await db.collection("quizzes").findOne({ \_id: objId });

          if (!quiz) return sendJSON(404, { error: "Quiz not found" });

          const questions = await db.collection("questions").find({ quizRef: objId }).toArray();

          return sendJSON(200, { quizName: quiz.quizName, totalMarks: quiz.totalMarks, questions });

        }

*// --------------------------------------------------------*

*// 8️⃣ SUBMIT QUIZ*

*// --------------------------------------------------------*

        if (req.url === "/submit-quiz" && req.method === "POST") {

          const { userAnswers, questions } = parsed;

          let score = 0;

          questions.forEach((q, i) => { if (userAnswers[i] === q.correctAnswer) score += q.marks; });

          return sendJSON(200, { score });

        }

*// --------------------------------------------------------*

*// 9️⃣ SAVE RESULT*

*// --------------------------------------------------------*

        if (req.url === "/save-result" && req.method === "POST") {

          const resultData = {

            studentName: parsed.studentName,

            quizName: parsed.quizName,

            subject: parsed.subject,

            score: parsed.score,

            totalMarks: parsed.totalMarks,

            quizId: parsed.quizId,

            submittedAt: new Date()

          };

          await db.collection("result").insertOne(resultData);

          return sendJSON(200, { message: "Result saved successfully" });

        }

*// --------------------------------------------------------*

*// 🔟 GET SINGLE SUBJECT BY ID*

*// --------------------------------------------------------*

        if (req.url.startsWith("/subject/") && req.method === "GET") {

          const subjectId = req.url.split("/")[2];

          try {

*// Convert to ObjectId if needed*

            let objId;

            try { objId = new ObjectId(subjectId); } catch { return sendJSON(400, { error: "Invalid subject ID" }); }

            const subject = await db.collection("subjects").findOne({ \_id: objId });

            if (!subject) return sendJSON(404, { error: "Subject not found" });

            return sendJSON(200, subject);

          } catch (err) {

            return sendJSON(500, { error: "Failed to fetch subject" });

          }

        }

*// --------------------------------------------------------*

*// ❌ UNKNOWN ROUTE*

*// --------------------------------------------------------*

        sendJSON(404, { message: "Route not found" });

      } catch (err) {

        console.log("❌ SERVER ERROR:", err);

        sendJSON(500, { error: "Internal Server Error" });

      }

    });

  });

  server.listen(5003, () => {

    console.log("🚀 Server running on PORT 5003");

  });

}

**Package.js**

{

  "name": "server",

  "version": "1.0.0",

  "main": "index.js",

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1"

  },

  "keywords": [],

  "author": "",

  "license": "ISC",

  "description": "",

  "dependencies": {

    "cors": "^2.8.5",

    "http": "^0.0.1-security",

    "mongodb": "^6.19.0"

  }

}