🧠 **MERN Stack Project Assignment**

IT Subjects Quiz Application

🎯 **Project Title**

IT Quiz Portal – MERN Stack Application

📌 **Project Objective**

Build a full-stack Quiz Application using:

- **Frontend:** React
- **Backend:** Express.js
- **Runtime:** Node.js
- **Database:** MongoDB

The system should allow:

- Admins to manage IT subject questions
- Students to take quizzes
- Automatic score evaluation
- Leaderboard and result tracking

👥 **User Roles**

1️⃣ **Admin**

- Login/Register
- Create subjects
- Add/Edit/Delete questions
- View student results
- View leaderboard

2️⃣ **Student**

- Register/Login
- Select subject
- Attempt quiz
- View result & previous attempts

🏗️ Core Features

🔐 Authentication

- JWT based login system
- Role-based access (Admin / Student)
- Password hashing using bcrypt

📚 Subject Management (Admin)

Admin can create subjects like:

- Data Structures
- Operating Systems
- DBMS
- Computer Networks
- JavaScript
- MERN Stack

❓ Question Management (Admin)

Each question must contain:

```
{
 subjectId,
 questionText,
 options: ["A", "B", "C", "D"],
 correctAnswer,
 difficultyLevel, // Easy | Medium | Hard
 marks
}
```

Features:

- Add question
- Update question
- Delete question
- Filter by difficulty

📝 Quiz Attempt (Student)

Rules:

- 10 random questions per quiz
- Timer (15 minutes)
- One attempt per subject per day
- Auto submit when timer ends

📊 Result System

After submission:

- Total Questions
- Correct Answers
- Wrong Answers
- Score
- Percentage
- Pass/Fail (Pass if >= 50%)

🏆 Leaderboard

- Top 10 scorers per subject
- Based on highest score

🗄️ Database Design (MongoDB Models)

👤 User Model

```
{
 name,
 email,
 password,
 role: "admin" | "student",
 createdAt
}
```

📘 Subject Model

```
{
 title,
 description,
 createdBy,
 createdAt
}
```

❓ Question Model

```
{
 subjectId,
 questionText,
 options: [],
 correctAnswer,
 difficultyLevel,
 marks
}
```

📄 QuizAttempt Model

```
{
 studentId,
 subjectId,
 answers: [
  {
   questionId,
   selectedOption,
   isCorrect
  }
 ],
 totalScore,
 percentage,
```

submittedAt

}

🔌 Backend API Routes

🔐 Auth Routes

POST /api/auth/register

POST /api/auth/login

📚 Subject Routes

POST /api/subjects (Admin)

GET /api/subjects (All)

GET /api/subjects/:id

PUT /api/subjects/:id (Admin)

DELETE /api/subjects/:id (Admin)

❓ Question Routes

POST /api/questions (Admin)

GET /api/questions/:subjectId

PUT /api/questions/:id (Admin)

DELETE /api/questions/:id (Admin)

📝 Quiz Routes

GET /api/quiz/:subjectId // Get random 10 questions

POST /api/quiz/submit // Submit answers

GET /api/quiz/result/:attemptId

GET /api/quiz/leaderboard/:subjectId

🖥️ Frontend Pages (React)

Public

- Home Page
- Login Page

- Register Page

Student Dashboard

- Subject List Page
- Quiz Page
- Result Page
- Attempt History Page
- Leaderboard Page

Admin Dashboard

- Manage Subjects Page
- Manage Questions Page
- View Results Page

🎨 **UI Requirements**

- Clean dashboard layout
- Timer component
- Progress bar during quiz
- Score chart (optional)
- Responsive design

⚙️ **Technical Requirements**

Backend

- MVC Folder Structure
- Middleware:
    - Auth middleware
    - Role middleware
- Environment variables
- Proper validation

Frontend

- React Router
- Axios for API calls
- Protected Routes

- Global state (Context API)

📦 Submission Requirements

Students must submit:

1. GitHub Repository
2. README file with:
    - Setup instructions
    - API documentation
    - Screenshots
3. Database schema explanation
4. Deployment link (Optional)