Project Title: Unified Assessment Platform

Objective

The objective of this platform is to provide a robust and secure web application where:

- **Students** can attend various types of assessments (mock tests, coding, MCQs, subjective tests).
- **Colleges/Companies (Instructors)** can create, schedule, and evaluate these assessments.
- **Admins** can oversee the functioning of the platform, manage user roles, monitor content, and configure institutional branding.

Authentication & User Onboarding:

Login / Signup Page

- **Design: Clean, minimal, and modern UI using Tailwind CSS or Material UI.**
- **Functionality:**
  - **Role selection during signup: Student, Instructor, or Admin**
  - **Fields: Name, Email, Password, Role, Institute Code (if needed), etc.**
  - **Validation using Formik + Yup**
  - **Error feedback (e.g., incorrect password, email not found)**

Role-Based Routing

- **Implement using React Router + JWT decoding to dynamically redirect users after login:**
  - **/student-dashboard**
  - **/instructor-dashboard**
  - **/admin-dashboard**

---

2. Student Dashboard

Dashboard Overview

- **Welcome message with name and profile pic**
- **Quick Stats: Total Assessments Taken, Scores, Pending Evaluations**

Assessment Catalog Page

- **Browse available assessments (cards or list view)**
- **Filters: Subject, Instructor, Date, Type (MCQ/Written)**
- **"Start Assessment" button**
- **Timer countdown if assessment has a time limit**

Assessment Interface

- **MCQs: Single or multiple select**
- **Written Questions: Text editor (like React Quill)**
- **Navigation Pane for moving between questions**
- **Auto-save feature**
- **Submit button with confirmation modal**

My Submissions

- **View list of assessments taken**
- **Status Tags: "Pending", "Evaluated"**
- **Evaluation Details: Score, Instructor feedback (if evaluated)**

Profile Section

- **Editable details (except role/email)**
- **Change Password**
- **View Performance Graph (use Chart.js or Recharts)**

---

3. Instructor Dashboard

Dashboard Overview

- **Total Assessments Created**
- **Submissions Pending Review**
- **Quick access to "Create New Assessment"**

Create Assessment Page

- **Assessment Builder UI:**
  - **Title, Description, Duration, Tags**
  - **Add Sections: MCQ, Short Answer, Long Answer**
  - **Use drag-and-drop or dynamic form fields (e.g., React Hook Form)**
- **Publish or Save as Draft**

- **Assign to Student Groups or public**

Manage Assessments

- **List view of all created assessments**
- **Edit/Delete options**
- **View Submissions count per assessment**

Evaluate Submissions

- **Filter submissions by assessment or student**
- **For MCQs: Auto-evaluated**
- **For Written:**
    - **Display student response**
    - **Text editor with comment box**
    - **Score entry and submit evaluation**
- **Status change: "Pending" → "Evaluated"**

Student Overview

- **View list of students who have taken assessments**
- **Performance summary**
- **Search and filter**

Profile

- **Editable info**
- **View created assessments and stats**

---

4. Admin Dashboard

Overview

- **Number of active Students, Instructors, Assessments**
- **Admin announcements or flags**

User Management

- **View, Add, Delete Users**
- **Role update option (only to Instructor or Student)**
- **Status (Active, Suspended)**
- **Search/Filter by role**

Assessment Oversight

- **View all assessments created**
- **Flag reports or duplicate content**
- **Manage or archive old assessments**

Logs & Reports

- **Downloadable logs of assessment activity**
- **Filters by date, role, action**
- **Suspicious behavior alerts (e.g., too many submissions in short time)**

Platform Settings

- **Institute Name/Logo upload**
- **Reset or archive old data**
- **Manage user roles**

Other Shared Frontend Features

Notifications System

- **Toasts or Badge Alerts**
- **New assessment assigned**
- **Submission evaluated**
- **Admin announcements**

Calendar View

- **For students: Upcoming assessments**
- **For instructors: Submission due evaluations**

Navigation & Sidebars

- **Role-based sidebar menu using conditionals**
- **Mobile responsive collapsible menu**
- **Persistent dark/light mode (stored in localStorage)**

Chat/Support (Optional)

- **Integrate support ticket/chat system**
- **Instructor ↔ Student Q&A per assessment**
- **Use Socket.io if going real-time**

Search & Filters

- **Global search bar (top navbar)**

- **Filterable data tables using react-table or similar**

Responsive UI

- **Fully mobile-optimized layouts using Tailwind Flex/Grid**
- **Breakpoints for cards, tables, and forms**

Module Overview

Module 1: Authentication & Role-Based Access

- Unified login/signup page with role selection
- Secure form validation and error handling
- Role-based routing to direct users to their specific dashboards
- Persistent login sessions with token-based authentication

Module 2: Student Dashboard

- Overview with test stats and progress
- Browse assessments by subject, instructor, or type
- Auto-saving, timer, and question navigation while attempting tests
- View all submissions with status (pending, evaluated)
- Visual charts for score trends and performance metrics
- Profile management (name, image, password)

Module 3: Instructor Dashboard

- Overview of total assessments, pending submissions
- Test creation with multi-section builder (MCQ, short, long)
- Publish, draft, or delete assessments
- Assign tests to selected student groups
- Manual evaluation for subjective responses
- View submissions and assign scores/feedback
- Access to student performance summaries

Module 4: Admin Dashboard

- Manage all users (search, filter, add/remove)
- Modify roles and change account status (active/suspended)
- Global oversight of assessments and submissions

- Download logs and audit trails of activities
- Detect anomalies like bulk submissions or timing irregularities
- Configure platform branding (logo, institute name)
- Reset/archive test data periodically

Authentication and Authorization (High-Level Overview)

- Role-based system using JSON Web Tokens (JWT)
- Each token contains user identity and role info
- Protected routes based on role verification
- Admin access includes additional validation layer

Technology Stack

Frontend (Client-Side)

- **React.js**: SPA architecture with component-based UI
- **React Router**: Role-based navigation
- **Tailwind CSS**: Modern, responsive styling
- **Formik + Yup**: Form handling with validation
- **Chart.js / Recharts**: Graphical representation of results
- **React Quill**: Rich text input for descriptive answers
- **React Table**: Filterable, sortable data views
- **React Toastify**: Alerts and notification banners
- **Responsive Layouts**: Grid & flexbox support for all devices

Backend (Server-Side)

- **Node.js**: Runtime environment
- **Express.js**: Backend framework to create REST APIs
- **JWT**: Secure authentication & role-based authorization
- **bcryptjs**: Password encryption
- **Multer**: File upload middleware for images and files
- **Winston/Morgan**: Logging and activity tracking

Database

- **MongoDB Atlas**: Cloud-hosted NoSQL database
- **Mongoose**: ODM for structured schema modeling

Deployment

- **Frontend**: Vercel or Netlify
- **Backend**: Render or Railway
- **Database**: MongoDB Atlas
- **CI/CD Pipeline**: GitHub Actions or Render Auto Deploy
- **Monitoring Tools (Optional)**: PostHog, LogRocket, or Sentry

---

Non-Functional Requirements (NFR)

1. Scalability

- Support 10,000+ concurrent users.
- Use stateless APIs, database indexing, and load balancers.
- Implement pagination, caching, and CDN for optimal performance.

2. Security

- Enforce HTTPS for all communication.
- Use hashed passwords (e.g., bcrypt) and JWT for secure sessions.
- Role-based access control and input sanitization to prevent injection attacks.

3. Responsiveness

- Fully mobile-optimized UI using Tailwind's Flex/Grid.
- Seamless experience across devices and screen sizes.

4. Accessibility

- Implement ARIA tags, semantic HTML, and keyboard navigation.
- Ensure compatibility with screen readers.

5. Performance

- Optimize APIs for fast data retrieval.
- Use lazy loading, debounce search, and async data fetching.

6. Reliability

- Auto-save test responses periodically.
- Retry failed network requests to prevent data loss.

7. Audit Logging

- Maintain logs for logins, test attempts, and admin actions.
- Use structured log formats and filters by user/date/type.

8. Dark/Light Mode

- User preference stored in localStorage.
- Consistent UI theme across sessions.

9. Error Handling

- Show toast notifications for success/errors.
- Fallback UIs for broken or delayed data loads.

Implementation Phases

Phase 1: Project Setup

- Initialize GitHub repository with base project structure
- Set up separate directories for frontend and backend
- Create MongoDB cluster and initialize connection with backend

Phase 2: Authentication Module

- Implement user registration and login flow
- Add form validations using Formik & Yup
- Set up JWT authentication with role-based tokens
- Configure protected frontend routes with React Router

Phase 3: Student Module

- Design student dashboard layout with performance widgets
- Implement assessment catalog with filter/search
- Build test-taking interface with:
  - Timer logic
  - Auto-save mechanism
  - Question navigation pane
- Create submissions history and result visualization (charts)

Phase 4: Instructor Module

- Dashboard showing test stats and quick actions
- Build dynamic test creation interface (drag & drop form)
- Include MCQ, short, and long answer question types

- Add assign-to-students flow with group creation
- Evaluate submissions (manual for subjective; auto for MCQ)
- Add comment/feedback box and scoring

Phase 5: Admin Module

- Build admin control panel with:
    - Role management
    - Platform user list with filter, add, suspend options
- View and manage all assessments across instructors
- View full logs of platform activities (filterable by date/user)
- Add branding section to upload institute name/logo
- Enable reset/archive feature for old test data

Phase 6: Shared Features

- Global notification system for:
    - New test assigned
    - Submission evaluated
    - Admin alerts
- Role-based sidebar navigation and responsive mobile layout
- Calendar integration for upcoming tests and due evaluations
- Dark/light mode toggle and preference storage
- Optional real-time chat with Socket.io (future enhancement)
