Nextastra Technologies pvt ltd Assignment:

“Student Project Management Portal”

Objective

Build a **MERN stack web application** that allows **multiple types of users (Admin, Teacher, Student)** to manage and interact with **projects, tasks, and evaluations**.
The app must use **both MongoDB and MySQL**, **JWT authentication**, **Redux for state management**, and include **filtering, search, and pagination** on both client and server sides.

This assignment evaluates your ability to:

- Design and implement a **minimum viable product (MVP)** with clear priorities.
- Manage **data models across both SQL and NoSQL databases**.
- Implement **authentication and authorization** securely.
- Handle **frontend state** efficiently with Redux.
- Deliver **usable and responsive UI** under time constraints.

Project Theme

Student Project Management Portal

Use Case Summary:

- **Admins** manage users (teachers & students).
- **Teachers** can create projects, assign tasks, and grade submissions.
- **Students** can view their assigned projects, submit tasks, and view grades.
- Each user logs in with **JWT authentication**.

Technical Requirements

1. Stack

Layer	**Tech**
Frontend	React.js (with Redux Toolkit)
Styling	Tailwind CSS or Bootstrap

Layer	Tech
Backend	Node.js + Express.js
Databases	MongoDB (NoSQL) + MySQL (SQL)
Authentication	JWT (JSON Web Token)
Deployment (optional)	Render / Vercel / Railway

---

2. Database Architecture

Use Case	Database	Reason
User management (Admin, Teacher, Student)	**MySQL**	Structured, relational data
Project and task details	**MongoDB**	Flexible, nested data
Audit logs / user activity	**MongoDB**	Schema-less event tracking

Example:

MySQL tables:

- users (id, name, email, password, role)
- user_roles (role_id, role_name)

MongoDB collections:

- projects: { title, description, teacher_id, students: [], tasks: [] }
- tasks: { project_id, title, description, status, submissions: [] }
- logs: { user_id, action, timestamp }

---

3. Features & Functional Requirements

Authentication

- Login & Registration (JWT-based)
- Password encryption using bcrypt
- Role-based access (Admin, Teacher, Student)

- Persist auth state in Redux

User Roles

Admin:

- Create / Delete users
- Assign roles (teacher/student)
- View all projects

Teacher:

- Create, update, delete projects
- Assign students to projects
- Add / edit / delete tasks
- Grade student submissions

Student:

- View assigned projects
- Submit task responses
- View feedback and grades

---

4. Integration of MongoDB + MySQL

You must use **both databases** meaningfully:

- Fetch user info (MySQL) and combine with project data (MongoDB).
- For example, when a student views a project:
    - Get user from MySQL (user_id, role)
    - Fetch their projects from MongoDB
    - Combine in one API response.

---

5. Evaluation Criteria

Criteria	**Description**
MVP Completion	Core features working end-to-end

Criteria	Description
Architecture	Clean folder structure, modular code
Database Design	Proper schema setup in both SQL/NoSQL
Authentication & Security	JWT, bcrypt, route guards
Frontend Quality	Responsive, intuitive, and functional
Redux Usage	Centralized state, no prop-drilling
Filtering/Search/Pagination	Efficient and responsive
Code Quality	Reusable components, clean API calls
Time Management	Completed within given timeframe
Bonus	Deployment or dockerization

---

8. Deliverables

- GitHub repository (frontend + backend)
- ReadMe file with:
    - Setup instructions
    - ER diagrams (optional)
    - API documentation (Postman/Swagger) (optional)
    - Credentials for demo users (Admin, Teacher, Student)

---

6. Bonus Enhancements (Optional)

- Dockerize both frontend & backend
- Deploy using Render/Vercel
- Add dark mode
- Include activity logs with timestamps (MongoDB)
- Implement file upload for student submissions