**Name : Mohammed Sami Nadaf**

**SRN : 01FE23MCA022**

**Title of the Project:** **ProjectEase**

**Introduction / Background**  
ProjectEase is a full-stack MERN (MongoDB, Express.js, React, Node.js) web platform designed to streamline project requests, approvals, management, and delivery for students and organizations. As academic and professional demands increase, ProjectEase offers an efficient, transparent, and automated solution that connects clients and administrators in a single, unified portal.

Objectives

* Provide a seamless multi-role portal for users (clients) and admins (service providers).
* Enable transparent project requesting (existing or custom) for both registered and guest users.
* Streamline approval workflows with status updates, admin notes, and project timelines.
* Integrate secure authentication with email verification and password recovery.
* Automate email notifications for all key events (requests, approvals, rejections, payments).
* Implement real-time payment processing via Razorpay (advance/full payments).
* Deliver a modern, responsive UI/UX with search/filter and smooth animations.

**Problem Statement**  
Manual project coordination leads to delays, miscommunication, and lack of tracking. Institutions and students need a centralized system for submitting requests, approving work, monitoring progress, and handling payments—reducing overhead and improving satisfaction.

**Scope**

**User Side:**

* Signup/login, profile management (picture, contact, GitHub link, change/forgot password)
* Browse, search, and request projects
* Track all requests (pending → completed), view admin updates and history
* Make advance or full payments via Razorpay

**Admin Side:**

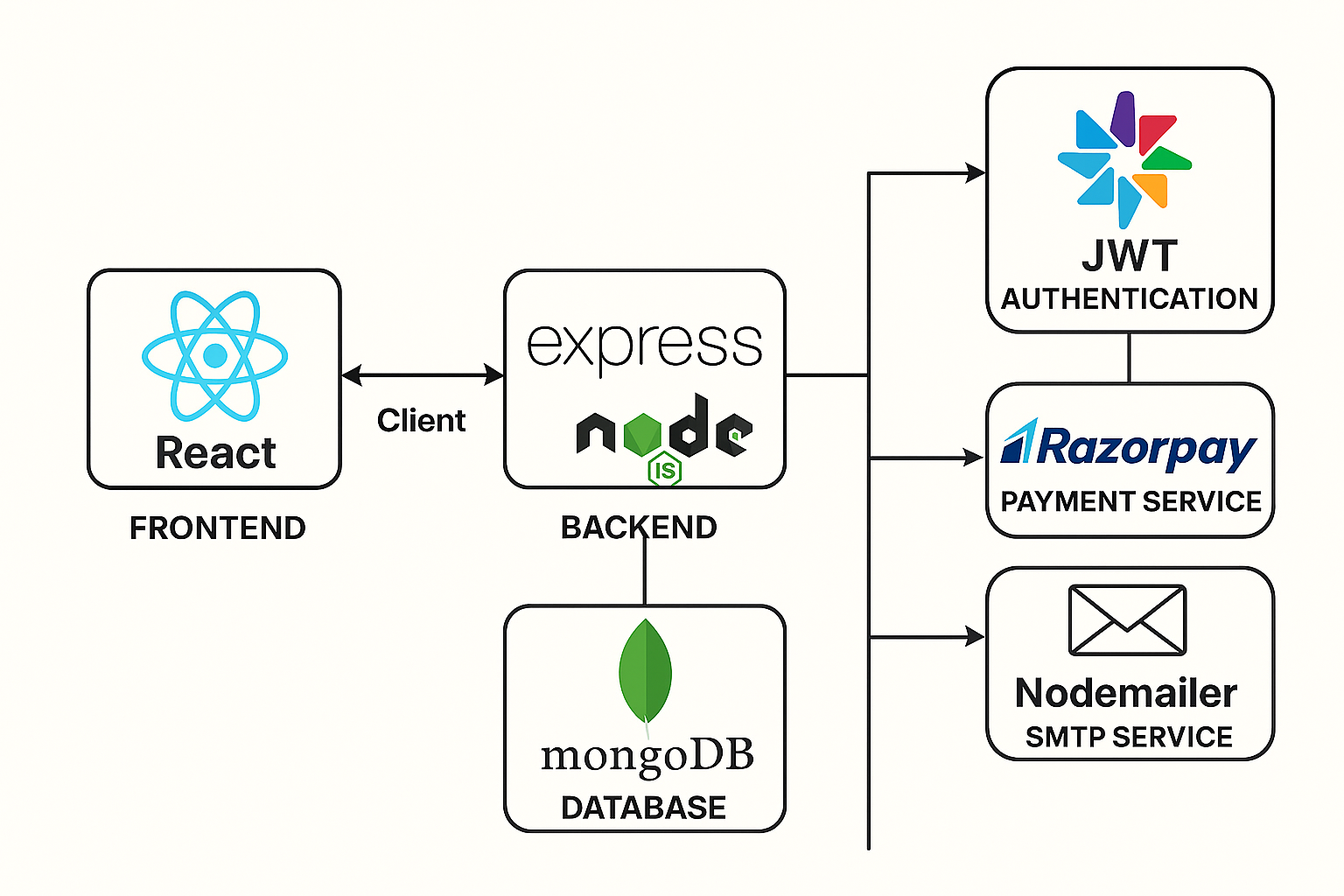
* Dashboard to manage projects and users
* CRUD operations on projects and users
* Approve/reject requests, add notes, update status, set milestones
* Review searchable lists of all requests, with inline editing

**System:**

* Automated email notifications (Nodemailer/SMTP)
* JWT-based role-based security
* MongoDB data models with status history
* Responsive React UI with Tailwind CSS and Framer Motion
* Vite for fast development & builds

**Proposed Solution**  
A MERN-stack application where users can effortlessly request and track projects, and admins can approve, update, and communicate—all through a single dashboard. Payments, notifications, and progress history are fully integrated.

Technical Details  
Frontend: React, React Router, React Hook Form, Tailwind CSS, Framer Motion, React-Toastify  
Backend: Node.js, Express, Mongoose, JWT authentication, role middleware  
Database: MongoDB (local or Atlas)  
Payments: Razorpay integration (sandbox & live)  
Email: Nodemailer via SMTP or SendGrid  
Deployment: Separate front-end/back-end with environment configs for production  
 **System Design**



**Methodology / Timeline**

| **Phase** | **Activities** | **Duration** |
| --- | --- | --- |
| Planning | Requirements, database & API design, UI mockups | 15 days |
| Backend Setup | Models, routes, auth, email & payment integration | 3 days |
| Frontend Dev | Pages, forms, modals, dashboards, search/filter | 7 days |
| Integration | Connect front & back, test all workflows | 2 days |
| Testing & QA | Bug fixes, security, performance, cross-browser tests | 2 days |
| Deployment | CI/CD setup, environment configs, documentation | 1 day |

**Expected Outcomes**

* An intuitive platform for project requests and tracking
* Clear, real-time communication between users and admins
* Automated notifications and payment workflows
* A modern, responsive UI enhancing user engagement and productivity

**References**

* MERN Stack documentation (MongoDB, Express, React, Node.js)
* Razorpay API docs
* Nodemailer guides
* Tailwind CSS & Framer Motion
* JWT authentication best practices

**ProjectEase Workflow**

**1. User Journey**

**a. Registration & Verification**

* User (student/client) signs up with username, email, and password.
* System sends a verification email with a secure link.
* User verifies their email by clicking the link (required for login).

**b. Login & Authentication**

* User logs in with email and password.
* JWT token issued on successful login (email must be verified).

**c. Browsing & Project Request**

* User lands on dashboard.
* Two tabs available:
  + **“My Projects”**: Shows projects the user has requested (pending, approved, rejected, completed).
  + **“All Projects”**: Shows catalog of all available projects; user can request any for themselves.
* User can:
  + Request a new (catalog/existing) project
  + Submit a custom project request (detailed form)

**d. Project Tracking & Update Notifications**

* User can view request status (pending, approved, in-progress, completed, rejected).
* Receives email and dashboard notifications when:
  + Admin responds/updates status
  + Payment is due or completed
  + Progress update or admin note is added
* Can view project timeline/history and admin notes.

**e. Payments**

* On approval, user pays upfront (70% advance or 100% full) via Razorpay.
* On completion, user pays outstanding amount if only advance was paid.

**f. Security & Profile**

* User can update profile, picture, and contact info.
* User can change password (confirmation email sent).
* User can delete their account (with confirmation modal).

**2. Admin Journey**

**a. Secure Login**

* Admin logs in from the login page; access is role-based.

**b. Admin Dashboard**

* Can view, filter, and search all project requests (across all users).
* Can approve/reject/mark project requests as in-progress, completed, or rejected.
* Can add admin notes, set project pricing, and update progress timeline.
* All actions trigger email updates to the relevant user.

**c. Project Management**

* Admin can create, update, or delete existing projects in the catalog.
* Can view and manage all user profiles for support.

**3. Email, Security, and Payment Flows**

**a. Email**

* Every status change, important event (signup, password update, approval, completion) triggers a transactional email.

**b. Authentication & Roles**

* JWT token required for all protected API endpoints.
* Role middleware distinguishes between user and admin endpoints.

**c. Payment**

* Payments made via Razorpay; backend verifies payment and updates request status.
* Payment history visible to users via dashboard.