



COLLEGE CODE:8203

COLLEGE NAME:AVC COLLEGE OF ENGINEERING

DEPARTMENT:B.E-CSE

STUDENTNMID: 6D5783801C4BE3BA82F47B3F14A4C2CB

ROLL NO:23CS71

DATE:22.09.2025

Completed the project named as Phase 3

**TECHNOLOGY PROJECT NAME: ADMIN DASHBOARD
WITH CHARTS**

SUBMITTED BY,

NAME: P.NITHYA SRI

MOBILE NO:9597290644

1. Project Setup

- **Technology Stack Selection:**
 - **Frontend:** React.js (for UI and chart rendering)
 - **Backend:** Node.js / Express (for API handling)
 - **Database:** MongoDB or PostgreSQL (for storing admin and chart-related data)
 - **Visualization Library:** Chart.js, Recharts, or D3.js for charts
 - **Version Control:** GitHub for collaboration and code management
- **Environment Setup:**
 - Initialize GitHub repository with proper branching strategy (main, dev, feature branches).
 - Install required dependencies (React, chart libraries, backend frameworks, database drivers).
 - Configure .env file for environment-specific variables.
- **Folder Structure Setup:**
 - **Frontend:** /src/components, /src/pages, /src/services
 - **Backend:** /routes, /controllers, /models, /middleware

2. Core Features Implementation

- **Authentication & Authorization:**
 - Secure login system for admins.
 - Role-based access control to manage dashboards and reports.
- **Dashboard Features:**

- Overview page with key performance indicators (KPIs).
- Interactive charts (bar, line, pie) to visualize user activity, transactions, or system performance.
- Filter and search options (date ranges, categories, departments).
- **User/Admin Management:**
 - Add/edit/remove admin users.
 - Track admin activities through logs.
- **Notifications & Alerts:**
 - Real-time alerts for system errors, usage thresholds, or critical activities.

3. Data Storage (Local State / Database)

- **Local State Management:**
 - Use Redux or React Context API to manage UI state (theme, filters, chart view preferences).
- **Database Storage:**
 - Store user/admin information, logs, and chart datasets in a structured database.
 - Schema Example:
 - users - { id, name, email, role }
 - charts - { id, type, dataset, timestamp }
 - logs - { action, admin_id, time }
- **Data APIs:**
 - RESTful APIs for fetching and updating dashboard data.
 - Secure endpoints with JWT authentication.

4. Testing Core Features

- **Unit Testing:**
 - Test React components (charts, forms, filters).
 - Test backend API routes (data fetching, authentication).
- **Integration Testing:**
 - Validate interaction between frontend and backend.
 - Ensure chart data is correctly fetched and rendered.
- **User Acceptance Testing (UAT):**
 - Admins test the dashboard for usability, responsiveness, and accuracy of charts.

5. Version Control (GitHub)

<https://nithyapraba20.github.io/IBMNMVC/>