



COLLEGE CODE:8203

COLLEGE NAME:AVC COLLEGE OF ENGINEERING

DEPARTMENT: B.E-CSE

STUDENT NM ID: 28833E772254004849F4415917202520

ROLL NO:23CS69

DATE:22.09.2025

Completed the project named as

Phase3

TECHNOLOGY PROJECT NAME: ADMIN DASHBOARD WITH CHARTS

SUBMITTED BY,

NAME: S.NISHA

MOBILE NO:9626210848

1. Project Setup

• Technology Stack Selection:

- Frontend: React.js (for UI and chart rendering)
- Backend: Node.js / Express (for API handling)
- o **Database**: MongoDB or PostgreSQL (for storing admin and chart-related data)
- o Visualization Library: Chart.js, Recharts, or D3.js for charts
- o **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).
- o 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.
- o Filter and search options (date ranges, categories, departments).

• User/Admin Management:

- o Add/edit/remove admin users.
- Track admin activities through logs.

Notifications & Alerts:

o 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.
- o Schema Example:
 - users → { id, name, email, role }
 - charts → { id, type, dataset, timestamp }
 - logs → { action, admin_id, time }

Data APIs:

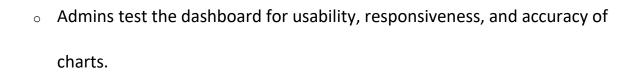
- o RESTful APIs for fetching and updating dashboard data.
- o Secure endpoints with JWT authentication.

4. Testing Core Features

- Unit Testing:
 - o Test React components (charts, forms, filters).
 - Test backend API routes (data fetching, authentication).

• Integration Testing:

- o Validate interaction between frontend and backend.
- o Ensure chart data is correctly fetched and rendered.
- User Acceptance Testing (UAT):



5. Version Control (GitHub)

https://nisha-17-06.github.io/IBMNMAVC/