COLLEGE CODE: 8203

COLLEGE: AVC COLLEGE OF ENGINEERING

DEPARTMENT: INFORMATION TECHNOLOGY

STUDENT NM-ID: 4D2198FA90720AD8F834BBD4C69121C9

ROLL NO: 23IT100

DATE:15-09-2025

Completed the project named as Phase 2

TECHNOLOGY PROJECT NAME: Admin dashboard with charts

SUBMITTED BY,

NAME: Sivarajaganapathi S

MOBILE NO: 7845102808

Solution Design & Architecture

Tech Stack Selection

Component	Technology	Rationale
Frontend (UI)	**HTML, CSS, JavaScript** (Frameworks like React/Vue optional)	To create a rich, dynamic, and responsive dashboard interface capable of rendering complex charts.
Backend (API)	**Node.js** with **Express.js**	A high-performance, non-blocking environment ideal for building fast and scalable RESTful APIs to serve data to the dashboard.
Database	**MongoDB**	A flexible NoSQL database that is excellent for storing application data, user metrics, logs, and easily retrieving/aggregating data for real-time charting.
Visualization	**Chart.js**	A lightweight, open-source JavaScript charting library to display dynamic and interactive line, bar, and pie charts on the frontend.
API Client	**Axios** (in frontend)	HTTP client used to reliably fetch metric data from the backend Node.js APIs.
Security	**JWT (JSON Web Tokens)**	To secure API endpoints and implement **Role-Based Access Control (RBAC)** to restrict data visibility.
Design/Testing	**Figma, Postman, Git/GitHub**	Tools for wireframing, designing, version control, and rigorous API testing.

UI Structure / API Schema Design

UI Structure – Key Elements

- **Header**: "Admin Dashboard" with user profile and logout.
- **Sidebar Navigation**: Links to different metric views (e.g., Overview, Sales, Users, System Health).
- **Metric Cards**: High-level, real-time numerical summaries (e.g., Total Users: 5,000, Today's Sales: $1,200).
- **Chart Visualization Area**: The main area where dynamic charts are rendered using Chart.js.
    - **Bar Chart**: e.g., Sales by Product Category.
    - **Line Chart**: e.g., User Signups Over 30 Days.
    - **Pie/Doughnut Chart**: e.g., Traffic Source Distribution.
- **Role-Based Visibility**: Elements/charts are dynamically hidden or displayed based on the user's role (Admin, Manager, Analyst).

API Schema Design (Core Endpoints)

Endpoint	Method	Request Parameter	Successful Response Body (Example)	Error Response Body (Example)
/api/auth/login	POST	username, password (body)	{"token": "jwt_string", "role": "Admin"}	{"error": "Invalid credentials"}
/api/metrics/summary	GET	**Header**: Authorization: Bearer <JWT>	{"totalUsers": 5000, "dailySales": 1200, "trafficSources": {...}}	{"error": "Unauthorized"} (401/403)
/api/metrics/salesTrend	GET	**Header**: Authorization: Bearer <JWT>	{"labels": ["W1", "W2", "W3"], "data": [1000, 1500, 900]}	{"error": "Access Denied"} (403)
/api/metrics/userGrowth	GET	**Header**: Authorization: Bearer <JWT>	{"labels": ["Jan", "Feb", "Mar"], "data": [500, 650, 800]}	N/A

Data Handling Approach

The system uses a direct API-to-Chart approach, leveraging MongoDB's aggregation pipeline for efficient data preparation.

1. **Authentication & Authorization**:
   - User logs in via /api/auth/login. The backend issues a **JWT** containing the user's **role** (e.g., 'Admin', 'Manager').
   - Every subsequent API call requires this JWT in the Authorization header. Express middleware checks the JWT validity and the associated role before granting access to data endpoints.
2. **Backend Data Computation (MongoDB/Express)**:
   - When an API (e.g., /api/metrics/salesTrend) is called, the Express handler triggers a **MongoDB aggregation pipeline**.

    - This pipeline processes large datasets (e.g., all sales records), performs computations (e.g., summing sales per week), and formats the output into a structure suitable for Chart.js.

3. **API Response Formatting**:
    - The backend ensures the response is a clean, chart-ready **JSON object** with labels (x-axis) and data (y-axis values).
    - Example: {"labels": ["Mon", "Tue"], "data": [10, 20]}.
4. **Frontend Data Rendering (Chart.js)**:
    - The frontend uses **Axios** to fetch the data from the secure API endpoints.
    - Upon successful retrieval, **Chart.js** initializes or updates the corresponding chart (line, bar, pie) using the labels and data arrays.
    - Charts are configured to **auto-update** every 30 seconds by repeating the API call.
5. **Error Handling**:
    - **401/403 Errors**: If a user without the necessary role attempts to access a protected endpoint (e.g., a 'Manager' accessing 'System Health'), the API returns a 403 Forbidden. The frontend redirects the user or displays an **"Access Denied"** message.
    - **500 Errors**: If the MongoDB connection or computation fails, the API returns a 500 Internal Server Error, and the frontend displays a general **"Dashboard data unavailable"** message.

Component / Module Diagram

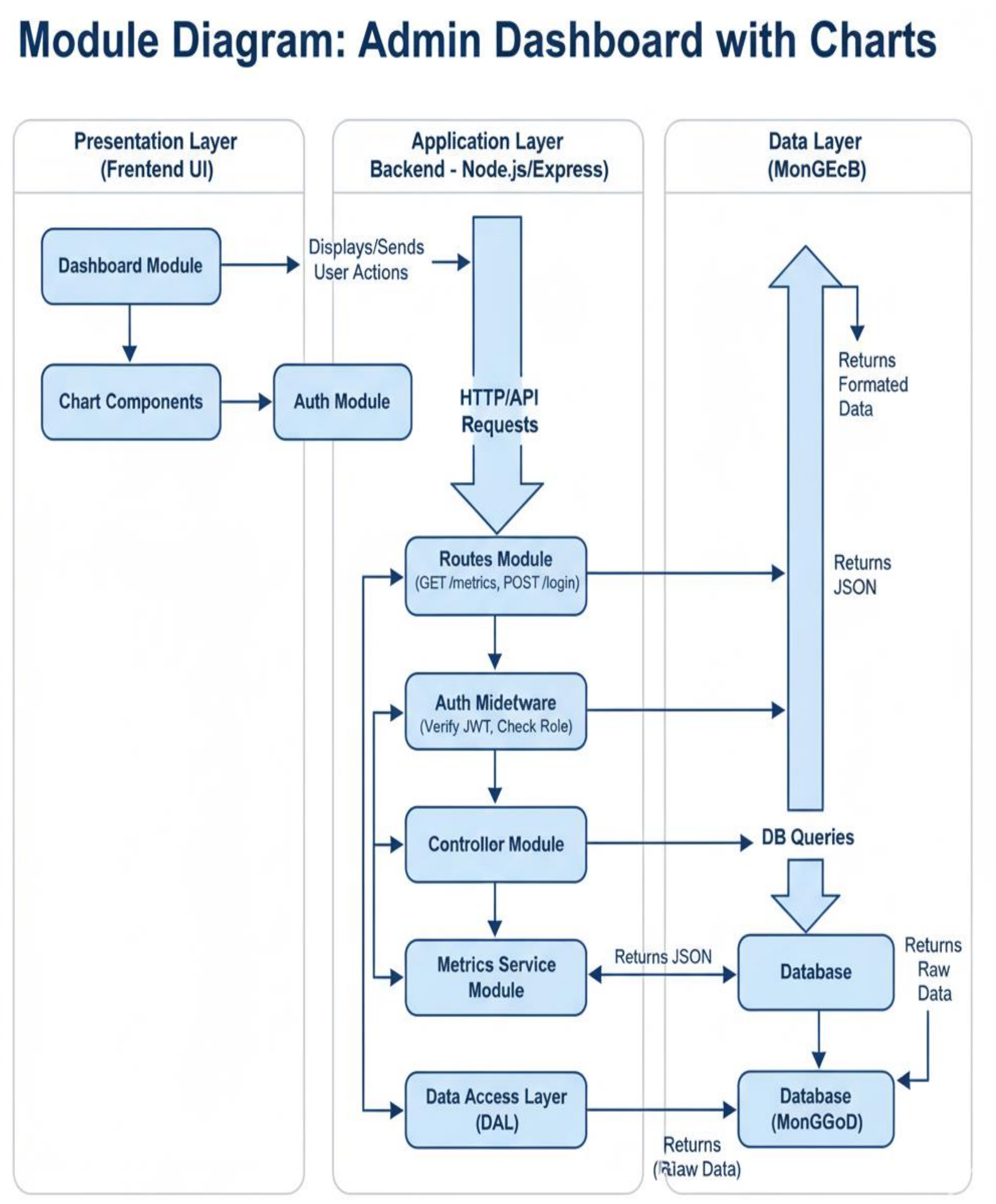
The system is logically divided into three main layers:

1. **Presentation Layer (Frontend UI)**:
    - **Dashboard Module**: Manages the overall layout and navigation.
    - **Chart Component**: Reusable components that encapsulate Chart.js logic (e.g., LineChartComponent, BarChartComponent).
    - **Auth Module**: Handles user login/logout and stores the JWT.
2. **Application Layer (Backend - Node.js/Express)**:
    - **Routes Module**: Defines API endpoints (/auth/login, /metrics/*).
    - **Auth Middleware**: Checks JWT and validates user roles for RBAC.

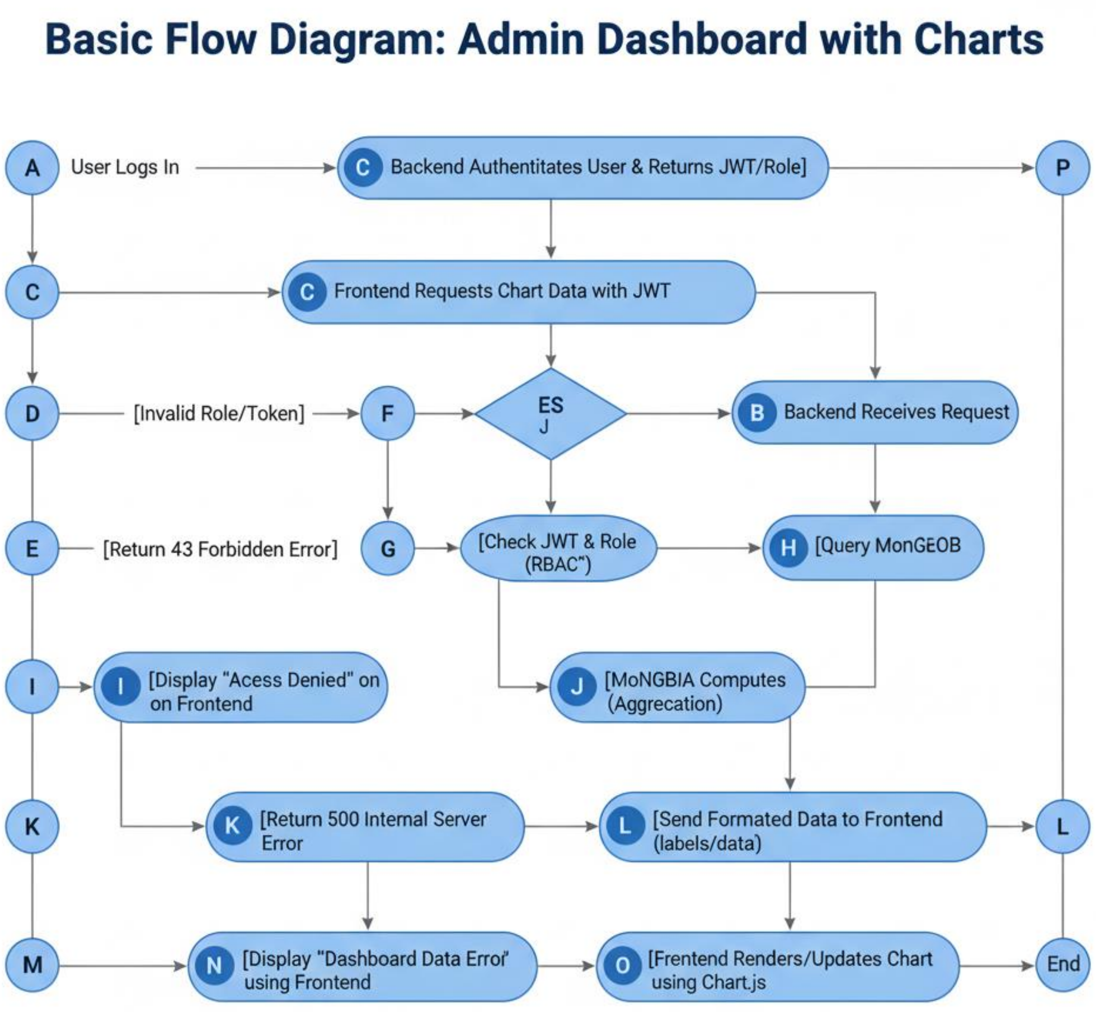
- **Controller Module**: Coordinates between the Routes and the Service Layer.
- **Metrics Service Module**: Contains the business logic for data retrieval.

3. **Data Layer (MongoDB)**:

- **Database**: Stores all raw and transactional data.
- **Data Access Layer (DAL)**: Contains functions for connecting to MongoDB and executing optimized queries/aggregation pipelines.



Basic Flow Diagram


Basic Flow Diagram: Admin Dashboard with Charts
A
User Logs In
C Backend Authentitates User & Returns JWT/Role]
P
C
C Frontend Requests Chart Data with JWT
D
[Invalid Role/Token]
F
ES J
B Backend Receives Request
E
[Return 43 Forbidden Error]
G
[Check JWT & Role (RBAC")
H [Query MonGEOB
I
I [Display "Acess Denied" on on Frontend
J [MoNGBIA Computes (Aggrecation)
K
K [Return 500 Internal Server Error
L [Send Formated Data to Frontend (labels/data)
L
M
N [Display "Dashboard Data Error' using Frontend
O [Frentend Renders/Updates Chart using Chart.js
End