COLLEGE CODE: 8203

COLLEGE: AVC COLLEGE OF ENGINEERING

DEPARTMENT: INFORMATION TECHNOLOGY

STUDENT NM-ID: 8EFD03039C61BA1B601456D562D0067A

ROLL NO: 23IT106

DATE:15/09/2025

Completed the project named as phase_2

TECHNICAL PROJECT:JOB APPLICATION TRACKER

SUBMITTED BY,

NAME: R SRINATH

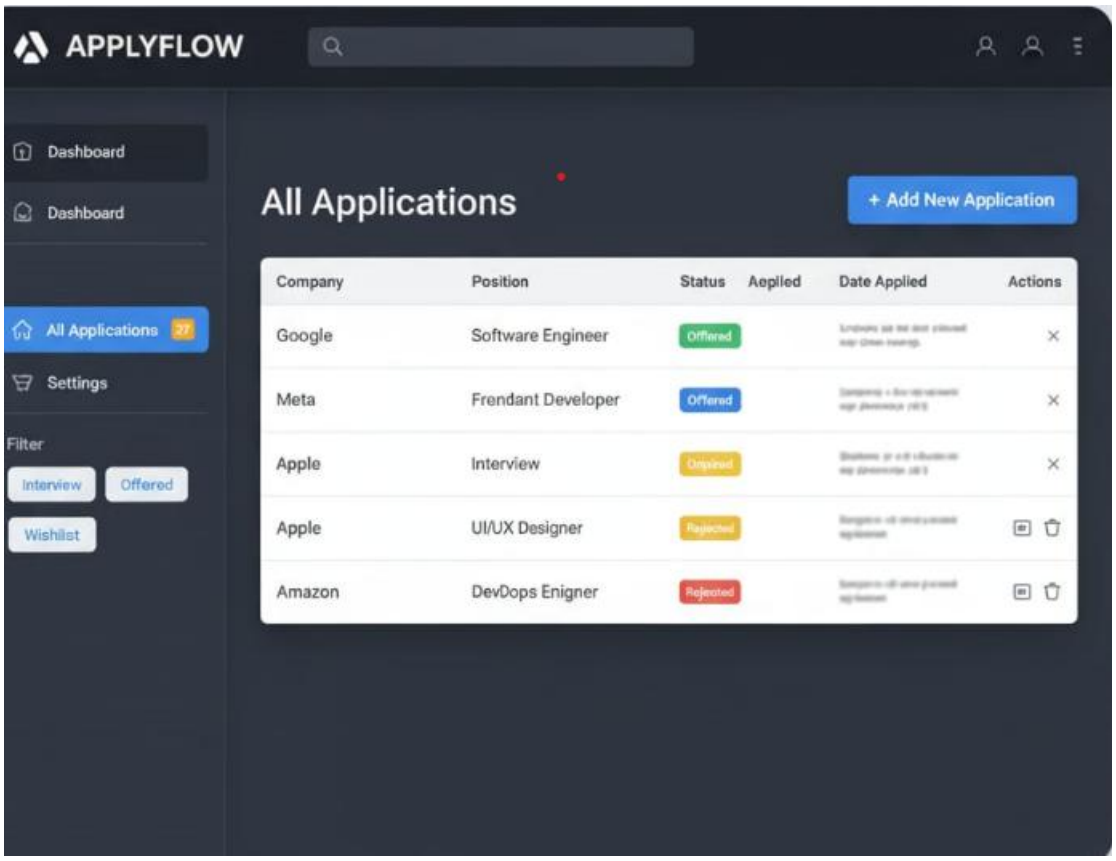
MOBILE NO: 9342680867

Phase 2 — Solution Design & Architecture

Tech Stack Selection

Component	Technology	Role in Project	Reason for Selection
Frontend (UI)	**HTML, CSS, JavaScript** (Optional: React.js)	Simple, responsive interface for input, viewing, and filtering applications.	Easy to use and ensures responsiveness across devices.
Backend (Server Logic)	**Node.js**	Executes server-side code efficiently using non-blocking I/O.	Scalable, fast, and widely adopted for REST API development.
Web Framework	**Express.js**	Simplifies routing, middleware implementation, and API endpoint handling.	Minimalist and flexible framework built on Node.js.
Database	**MongoDB**	Stores all application data (company, status, notes, user IDs).	Flexible, schema-less (NoSQL), and aligns well with JavaScript-based development (JSON-like documents).
HTTP Client	**Axios**	Handles API requests to and from the backend for CRUD operations.	Simple, promise-based HTTP client for external and internal API calls.
Auth	**JWT (JSON Web Tokens)**	Secures API endpoints and separates user data.	Standard, stateless method for secure authentication.
Tools & Utilities	**Git/GitHub, Postman**	Version control/collaboration and API testing.	Essential for development and testing.

UI Structure / API Schema Design



UI Structure – Job Application Tracker

- **Header:** "Job Application Tracker" with a **Logout** button.
- **Application List / Dashboard (Main View):**
  - **Filter Section:** Dropdown or buttons to filter applications by **Status** (Applied, Interview, Offered, Rejected).
  - **"Add New Application"** button.
  - **Table/Card List:** Displays key fields for each application: **Company Name** (bold), **Status** (color-coded), **Date Applied**.
  - **Action Buttons: Edit** (pencil icon) and **Delete** (trash icon) for each entry.

- **Add/Edit Application Form (Modal/Page):**
    - Input fields for: **Company Name**, **Role/Title**, **Date Applied** (Date Picker).
    - Dropdown for **Status** (Applied, Interview, Offered, Rejected).
    - Large text area for **Notes** (Interview feedback, contact names).
    - **Save** and **Cancel** buttons.

API Schema – Job Application Tracker

The application data is stored as a JSON object in MongoDB.

Field Name	Data Type	Required	Description
company	String	Yes	Name of the company.
role	String	Yes	Job title/position applied for.
status	String	Yes	Current application stage (e.g., 'Applied', 'Interview', 'Offered').
dateApplied	Date	Yes	Date the application was submitted.
notes	String	No	Details like contact names, interview notes, etc.
userId	MongoDB ObjectID	Yes	Automatically linked to the authenticated user for data separation.

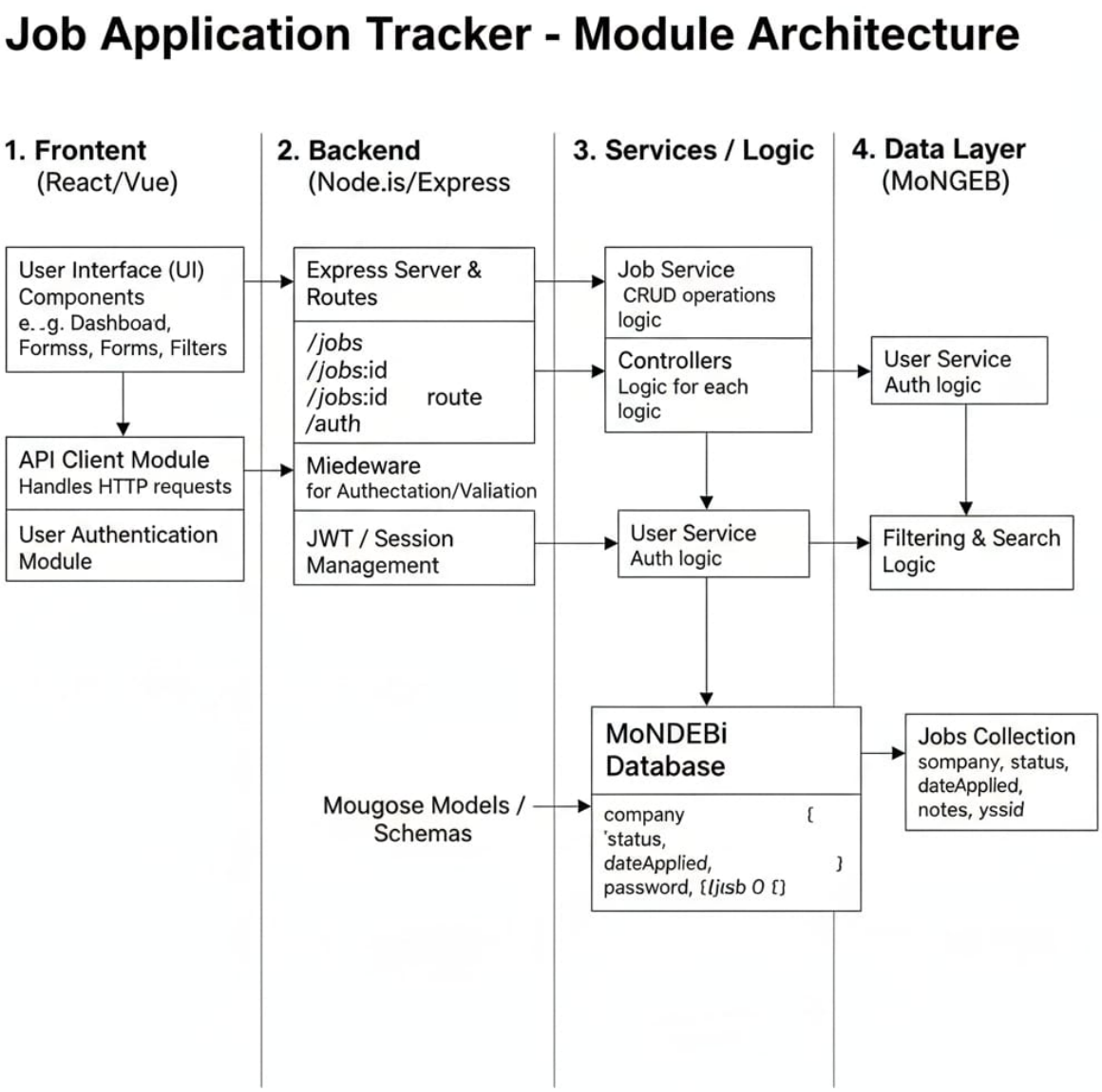
Data Handling Approach

1. **User Authentication:**
    - User sends **username** and **password** to `/api/auth/login`.
    - Backend validates credentials and returns a **JWT (JSON Web Token)**.
    - **Frontend** stores the JWT (e.g., in Local Storage) and includes it in the **Authorization Header** for all subsequent protected API calls.
2. **Application Submission (POST):**
    - **Frontend** sends application data and the JWT to the `/api/applications` endpoint.
    - **Backend (Express Middleware)**: Verifies the JWT and extracts the `userId`.
    - **Controller:** Creates a new application object, adding the extracted `userId`, and saves it to **MongoDB**.
3. **Application Retrieval (GET):**
    - **Frontend** sends a request with an optional `status` query parameter (e.g., `/api/applications?status=Interview`).
    - **Backend:** Uses the `userId` from the JWT to query **MongoDB** (e.g., `db.applications.find({ userId: '...' })`).
    - If a `status` filter is present, the MongoDB query includes this filter.
    - **Backend** returns the filtered, user-specific data to the frontend.
4. **Error Handling:**
    - **Authentication Errors:** Invalid token or credentials result in a **401 Unauthorized** response.
    - **Validation Errors:** Missing required fields (e.g., company name) result in a **400 Bad Request** with a descriptive message.
    - **Database Errors:** Connection or query failures result in a **500 Internal Server Error**.
    - **Frontend** handles these errors by displaying a user-friendly message (e.g., "Login Failed" or "Please fill in all fields").

Component / Module Diagram

The system is split into five main logical modules:

1. **Frontend (UI):**
    - **Components:** Login Form, Dashboard List, Filter Bar, Add/Edit Application Form.
    - **Function:** Renders data, captures user input, and manages user session (JWT storage).
2. **Backend (Express):**
    - **Components:** Router Module (defines API endpoints), Controller Module (handles business logic).
    - **Function:** Routes requests, authenticates users, and manages data flow between the UI and the database.
3. **Authentication Module:**
    - **Components:** JWT generation/validation logic, Password Hashing.
    - **Function:** Secures endpoints and verifies user identity for all application data access.
4. **Database Layer (MongoDB):**
    - **Components:** Application Schema, User Schema.
    - **Function:** Persistent storage for all user and application data.
5. **Data Models/Mongoose:**
    - **Components:** Data schemas that enforce structure for MongoDB documents.
    - **Function:** Translates application logic into database operations.



Basic Flow Diagram

This flow illustrates the process for **Filtering Applications by Status** after a user is logged in.

1. **Start** (User is logged in).
2. **User Selects Status Filter** (e.g., "Interview") on the UI.
3. **Frontend Sends GET Request** with JWT and query parameter to /api/applications?status=Interview.
4. **Backend (Express) Receives Request**.

5. **Check JWT** (Is user authenticated?).
    - **No:** Return **401 Unauthorized** Error. **(End)**
    - **Yes:** Extract userId.
6. **Query MongoDB:** Search for applications where userId matches and status is "Interview".
7. **Data Found?**
    - **No:** Return **Empty Array** (or 200 OK).
    - **Yes: Send Data to Frontend**.
8. **Display Filtered List** on the Dashboard UI. **(End)**