COLLEGE CODE: 8203

COLLEGE NAME: A.V.C. College of Engineering

DEPARTMENT: Information Technology

STUDENT NM-ID: 2CF9A0E1E49C80C86124EED021004FAD

ROLL NO: 23IT114

DATE: 15/09/2025

Completed the project named as Phase 2

TECHNOLOGY PROJECTN NAME: *USER AUTHENTICATION SYSTEM*

SUBMITTED BY,

VEDIKA.D

NAME: Vedika.D

MOBILE NO: 9486786011

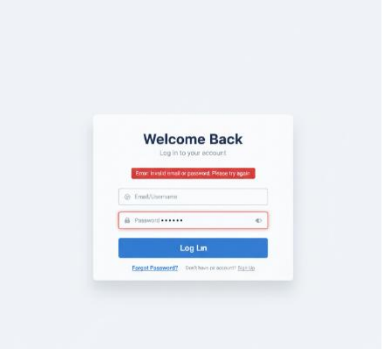
1. Tech Stack Selection

Layer / Feature	Technology / Tool	Purpose
Backend	Node.js	Runtime for server-side logic
Web Framework	Express.js	Simplifies REST API creation & route handling
Database	MongoDB	NoSQL DB for storing user accounts, roles, and session info
Password Security	bcrypt	Hash and salt passwords securely
Authentication	JWT (JSON Web Token)	Stateless authentication and token verification
Middleware	Express Middleware	Protect routes, validate tokens, enforce RBAC
Session Management	Optional: express-session	Persistent sessions for hybrid approach (cookies)
Validation	Joi / express-validator	Validate incoming request data (email, password, etc.)
Logging	Winston / Morgan	Track login attempts, errors, and security events
Frontend	React.js / HTML-CSS-JS	User interfaces for signup/login/dashboard
API Testing	Postman / Swagger	Test API endpoints and document API
Security	Helmet, CORS	Protect against common web attacks (XSS, CSRF, etc.)

2. UI Structure / API Schema Design:

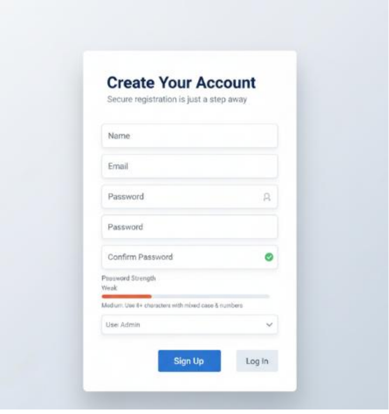
UI Structure (Frontend Pages)

1. **Login Page**



Error: Invalid email or password. Please try again.

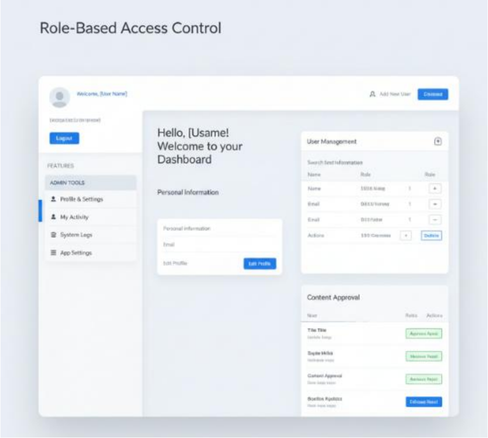
- ○ Fields: Email/Username, Password
- ○ Buttons: Login, Signup, Forgot Password
- ○ Features: Show/hide password, error messages for invalid credentials



2. **Signup Page**

- ○ Fields: Name, Email, Password, Confirm Password, Role (if admin adds user)
- ○ Buttons: Signup, Login
- ○ Features: Password strength indicator, validation messages.

3. **Dashboard (Post-login)**



- ○ Sections: Welcome message, Role-based content.
  - ▪ Regular User: Profile info, personal settings
  - ▪ Admin: User management panel, role assignment
  - ▪ Moderator (optional): Limited actions

- Buttons: Logout, Edit Profile

Method	Endpoint	Description	Access
POST	/api/signup	Register new user (bcrypt hashed)	Public
POST	/api/login	Authenticate user & issue JWT	Public
GET	/api/logout	Invalidate session/JWT	Authenticated
GET	/api/profile	Fetch logged-in user profile	Authenticated
GET	/api/admin	Admin-only route (RBAC check)	Admin only
PUT	/api/admin/role	Update user role	Admin only
GET	/api/logs	Fetch authentication logs	Admin only

3.

Data Handling Approach:

User Schema (MongoDB)

```
{
  _id: ObjectId,
  name: String,
  email: String, // unique
  password: String, // hashed using bcrypt
  role: String, // 'user' | 'admin' | 'moderator'
  createdAt: Date,
  updatedAt: Date
}
```

Session / Token Handling

- JWT tokens stored on client (HTTP-only cookies or localStorage)
- Optional server-side session table for hybrid approach
- Token expiration: 1 hour (configurable)
- Middleware verifies JWT and role for protected routes

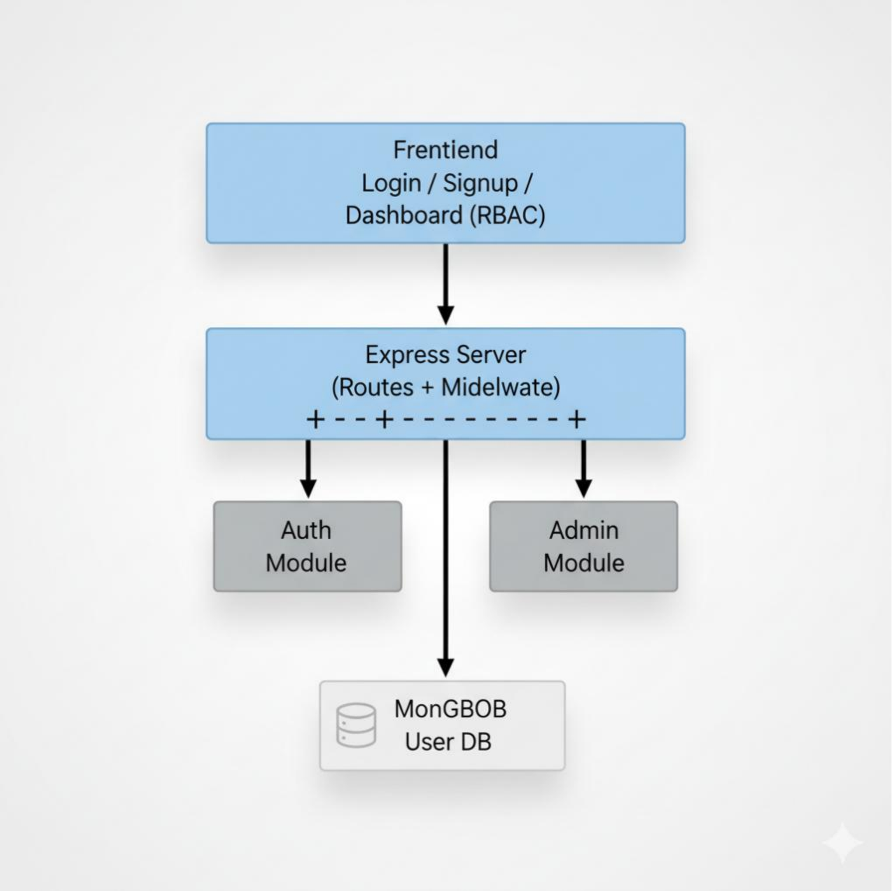
Password & Validation

- Hash passwords using bcrypt before saving
- Validate email format, password strength (min 8 chars, uppercase, lowercase, number)

4. Component / Module Diagram

Modules:

1. **User Management Module**
   - Signup



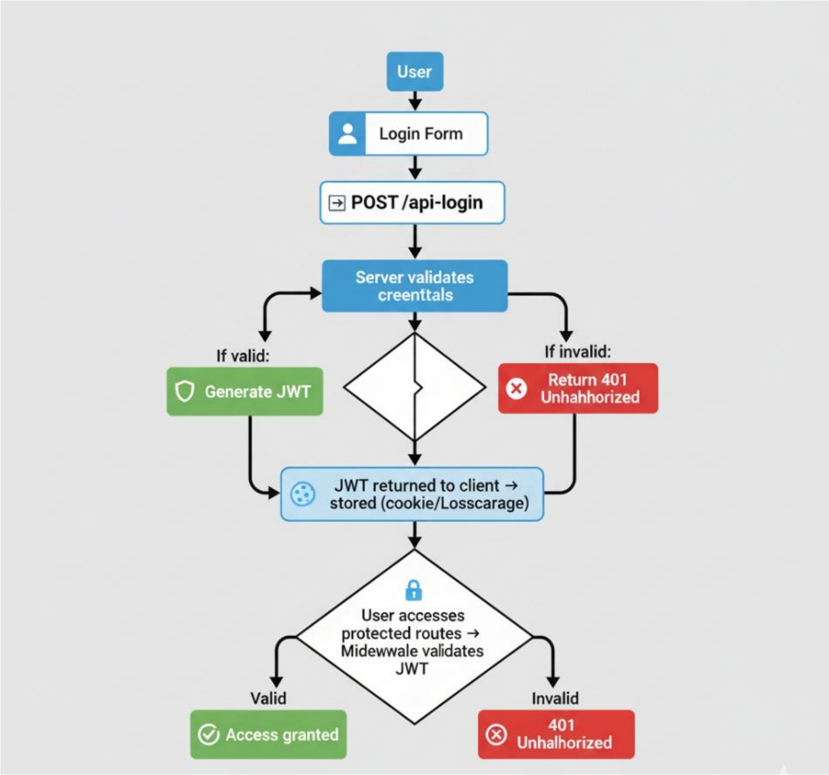
   - Login
   - Logout
   - Profile retrieval
   - Role management (admin)
2. **Authentication Module**
   - JWT token generation/verification

- Middleware for protected routes

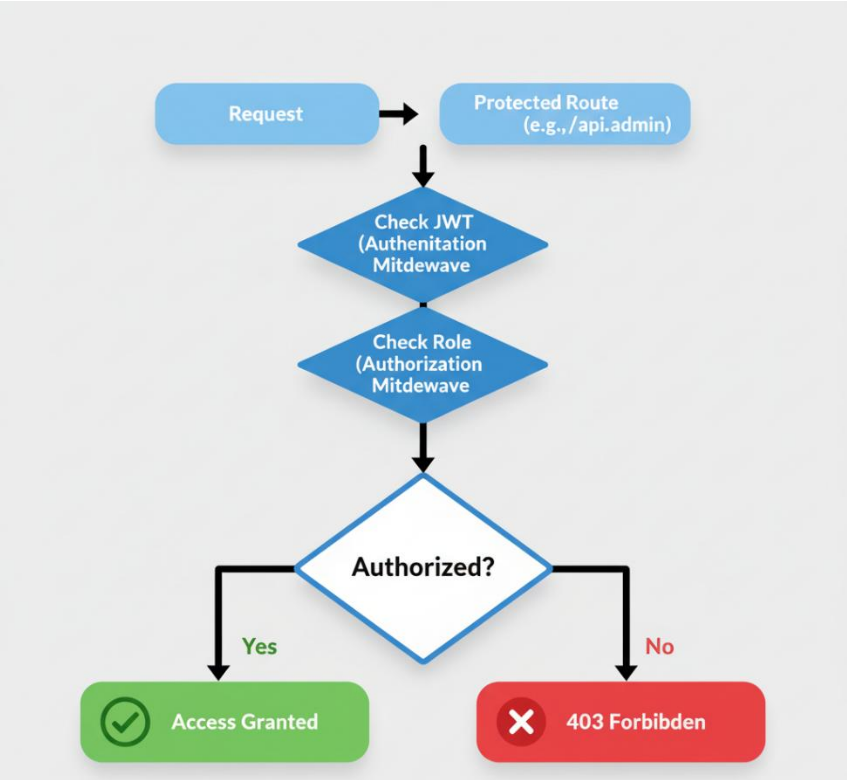
3. **Authorization Module**
   - RBAC enforcement
   - Route guards based on roles
4. **Validation & Error Handling Module**
   - Input validation
   - Standard error responses
5. **Logging & Monitoring Module**
   - Authentication logs
   - Failed login attempts tracking
6. **Frontend Module**
   - Login/Signup/Dashboard
   - Role-based UI rendering.

5. Basic Flow Diagram

User Authentication Flow:


User
Login Form
POST /api-login
Server validates creenttals
If valid:
Generate JWT
If invalid:
Return 401 Unhahhorized
JWT returned to client → stored (cookie/Losscarage)
User accesses protected routes → Midewwale validates JWT
Valid
Access granted
Invalid
401 Unhalhorized

Role-based Access Flow


Request
Protected Route
(e.g.,/api.admin)
Check JWT
(Authenitation
Mitdewave
Check Role
(Authorization
Mitdewave
Authorized?
Yes
No
Access Granted
403 Forbibden