Secure User Profile & Access Control System

(JWT Authentication + AES-256 Encryption + React Dashboard)

Github Link

Project Overview

This project implements a **Secure User Profile & Access Control System** as part of the **GET 2026 Full Stack Assignment**.

The application focuses on **secure authentication**, **encryption of sensitive data**, and **protected profile access**, demonstrating real-world backend security practices integrated with a modern React frontend.

Key Features

- Secure user **registration and login**
- **JWT-based authentication** for protected APIs
- **AES-256 encryption** for Aadhaar number stored at rest
- **Password hashing** using bcrypt
- Protected **Profile Dashboard**
- Clean, responsive **React UI**
- Logout functionality to revoke access

Technology Stack

Backend

- **Node.js**
- **Express.js**
- **MongoDB Atlas**
- **JWT (jsonwebtoken)** – authentication
- **bcrypt** – password hashing
- **crypto (AES-256)** – Aadhaar encryption
- **dotenv**, **cors**

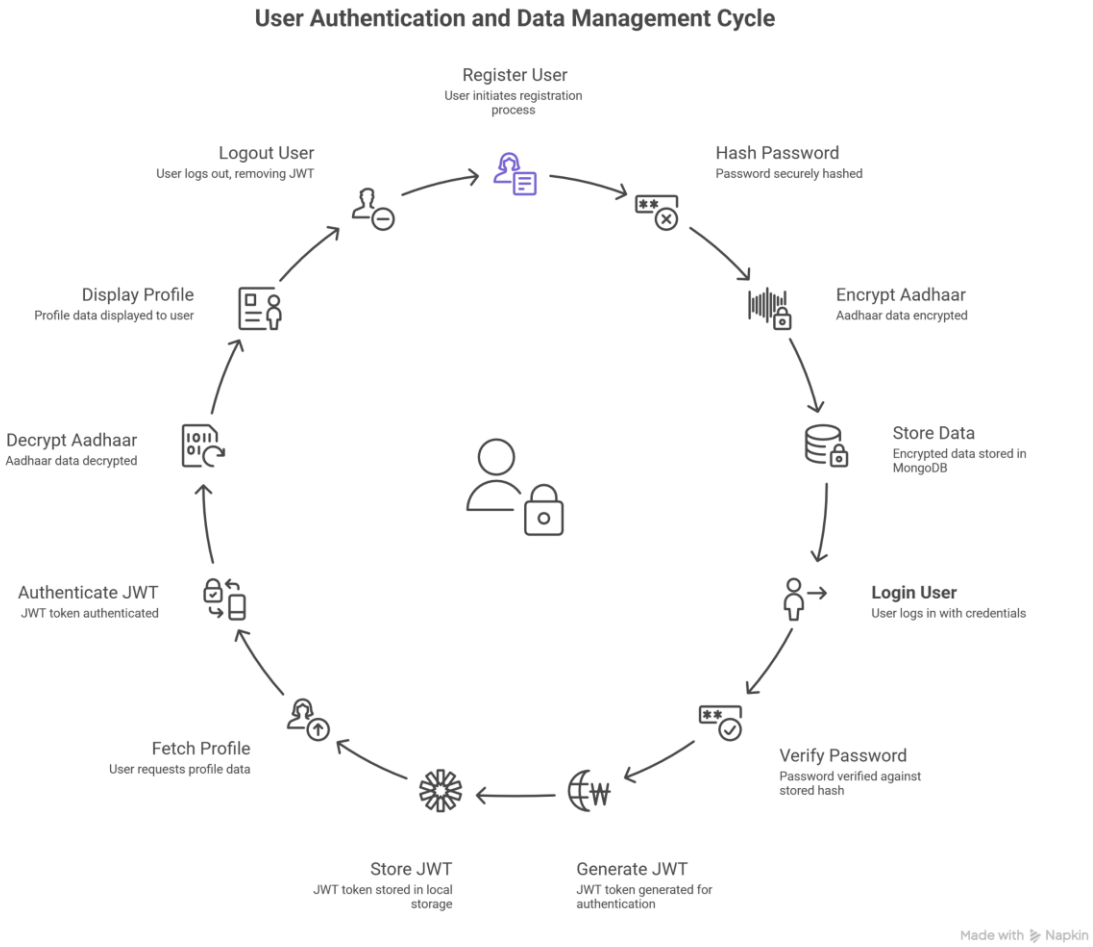
Frontend

- **React (Vite)**
- **Axios**

- **LocalStorage** (JWT persistence)
- Custom **blue-gradient UI**

System Architecture & Data Flow

The diagram below illustrates the complete **authentication and encrypted data management lifecycle** of the system.



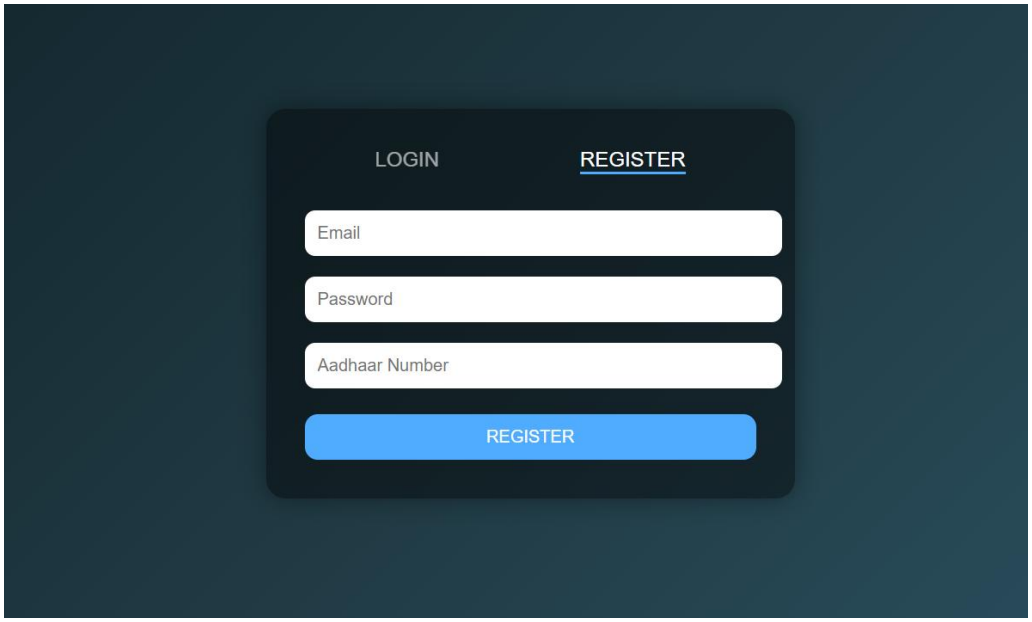
Flow Summary

1. User registers with email, password, Aadhaar
2. Password is hashed, Aadhaar is encrypted
3. Encrypted data stored in MongoDB
4. User logs in with credentials
5. JWT token generated and stored client-side
6. Protected profile API accessed using JWT
7. Aadhaar decrypted only before sending response

8. User logs out, JWT removed

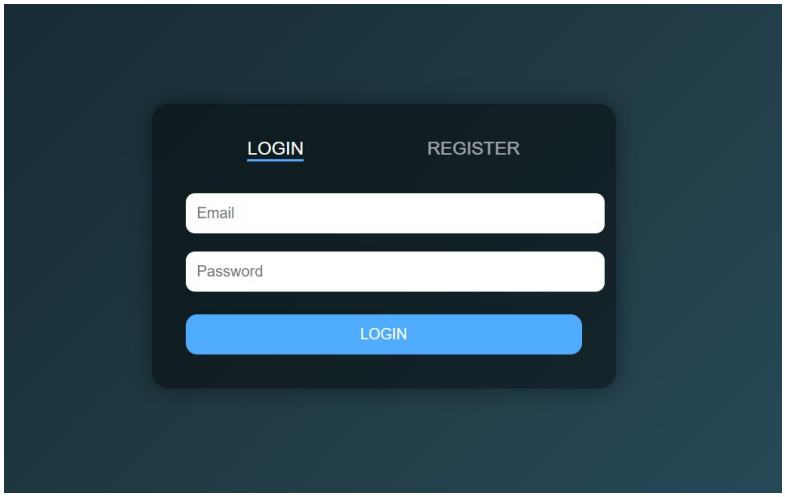
Application Flow (Step-by-Step)

1.User Registration



- User submits **email, password, Aadhaar**
- Frontend validates:
    - Email format
    - Aadhaar is exactly 12 digits
- Backend:
    - Hashes password using bcrypt
    - Encrypts Aadhaar using AES-256
    - Stores encrypted data in MongoDB

2. **User Login**



- Backend verifies password
- JWT token generated on success
- Token returned to frontend and stored in **LocalStorage**

Security Considerations

- Passwords are **never stored in plaintext**
- Aadhaar is **encrypted at rest** using AES-256
- Decryption occurs **only on backend**
- JWT ensures stateless, secure authentication
- Protected routes prevent unauthorized access

Note: In production systems, Aadhaar can be masked at UI level.
For demonstration purposes, the full Aadhaar is displayed after authentication.

Edge Cases Considered

Backend

- Duplicate user registration
- Invalid login credentials
- Missing or invalid JWT
- Expired JWT
- Encryption/decryption failures
- Database connectivity issues

Frontend

- Invalid email format

- Invalid Aadhaar length
- Login failure handling
- Token persistence on refresh
- Logout cleanup
- Unauthorized profile access

🖥️ **Backend Setup**

cd backend

npm install

Create a .env file inside backend:

PORT=5000

MONGO_URI=mongodb+srv://<username>:<password>@<cluster>.mongodb.net/<database>

JWT_SECRET=your_jwt_secret

CRYPTO_SECRET=64_character_hex_key

Start backend server:

npm run devStart

Expected output:

MongoDB connected

Server running on port 5000

Frontend Setup

cd frontend

npm install

npm run dev

Frontend runs at:

http://localhost:5173

Testing & Verification

- Register → Login → Profile → Logout tested end-to-end
- JWT verified via:
    - Browser DevTools → Application → Local Storage

- MongoDB verified to store **encrypted Aadhaar**

AI Tool Usage Log (Mandatory)

AI-Assisted Tasks

- Generated JWT authentication middleware boilerplate
- Designed AES-256 encryption/decryption utility
- Assisted in backend route structuring
- Helped debug frontend-backend integration
- Refined frontend validation logic
- Structured README documentation

Effectiveness Score

4 / 5

Justification:
AI tools significantly reduced development time by assisting with boilerplate code, debugging, and documentation, while final implementation and integration decisions were handled manually.

Conclusion

This project demonstrates a **secure, production-oriented authentication system** with encryption at rest, protected APIs, and a clean frontend dashboard.
All assignment requirements and evaluation criteria have been fulfilled with a strong emphasis on security and clarity.