9530

St. MOTHER THERESA ENGINEERING COLLEGE

COMPUTER SCIENCE ENGINEERING

NM-ID: 5CAA608A93FF57485DC4D0B381BDF4F6

REG NO: 953023104105

DATE:22-09-2025

Completed the project named as

Phase 3

FRONT END TECHNOLOGY

Login Authentication System

SUBMITTED BY:

Sam Giftson K

9600282616

1. Project Setup

The first step in MVP implementation is establishing the development environment. For the Login Authentication System, we set up both frontend and backend environments:

- **Frontend Setup:**
    - React.js initialized using create-react-app or Vite.
    - Project structured into components (LoginForm, RegisterForm, Dashboard).
    - UI libraries like TailwindCSS/Bootstrap integrated for styling.
    - Axios (or Fetch API) configured for communicating with backend APIs.
- **Backend Setup:**
    - Node.js environment initialized using npm init.
    - Express.js framework installed for building REST APIs.
    - MongoDB or MySQL database connected using Mongoose/Sequelize ORM.
    - Project folders organized into routes, controllers, models, and middlewares.
    - Environment variables (.env) used to secure database credentials and JWT secret keys.
- **Basic Tools Setup:**
    - ESLint and Prettier integrated for maintaining code quality.
    - Postman/Insomnia used for API testing during development.

This setup ensures that both the client and server layers communicate seamlessly and securely.

---

2. Core Features Implementation

The MVP focuses on implementing the **essential features** of authentication:

1. **User Registration**
    - Users provide email, password, and name.
    - Backend validates inputs and hashes the password using bcrypt.
    - User record stored in the database with role (default: user).

2. **User Login**
    - Users enter email and password.
    - Backend verifies credentials against hashed passwords.
    - On success, JWT is generated and returned to the client.
    - Token is stored in browser localStorage or secure cookies.
3. **Session Management**
    - JWT tokens are validated for each protected API call.
    - Middleware checks token validity before granting access.
4. **Password Reset**
    - Users can request a password reset link via email.
    - A one-time token is generated and sent.
    - On clicking the link, users can set a new password.
5. **Role-Based Access**
    - Regular users can only view/update their own profile.
    - Admins can view all users, lock accounts, or reset credentials.
6. **Account Security Features**
    - Account lockout after 5 failed login attempts.
    - Error messages are generic (e.g., "Invalid credentials") to prevent information leakage.

---

3. Data Storage (Local State / Database)

Frontend (Local State)

- Login status, JWT tokens, and user profile details stored in React state using Context API or Redux.
- LocalStorage or SessionStorage used for persistence across sessions.

Backend (Database)

- **User Collection/Table** (MongoDB Example):
- {
- "id": "string",
- "name": "string",
- "email": "string",
- "passwordHash": "string",
- "role": "user/admin",
- "isLocked": "boolean",
- "createdAt": "date"
- }
- Security logs stored for auditing failed login attempts.
- Refresh tokens (if implemented) stored in DB for token rotation.

---

4. Testing Core Features

Testing ensures the reliability and security of the authentication system.

- **Unit Testing:**
    - Password hashing and validation functions tested with sample data.
    - JWT generation and expiry tested.
- **Integration Testing:**
    - API endpoints tested using Postman for valid/invalid cases.
    - Registration → Login → Profile retrieval flow tested.
- **Security Testing:**
    - SQL Injection and NoSQL Injection prevented via validation.
    - Brute-force prevention verified by attempting multiple logins.
    - HTTPS enforced for all communication.

- **User Acceptance Testing (UAT):**
    - End users asked to test login, registration, and reset flows.
    - Feedback collected for UI clarity and usability.

---

5. Version Control (GitHub)

Version control is managed using **Git** and hosted on **GitHub**:

- **Repository Structure:**
    - Separate branches for frontend and backend development.
    - main branch reserved for stable releases.
- **Branching Strategy:**
    - feature/authentication for login/signup implementation.
    - feature/admin-dashboard for admin functionalities.
    - bugfix/* for issue resolution.
- **Collaboration:**
    - Pull Requests (PRs) reviewed before merging.
    - GitHub Issues used to track bugs and feature requests.
    - GitHub Actions (optional) configured for CI/CD (automatic builds and tests).

Using GitHub ensures transparency, collaboration, and safe rollback in case of errors.

Conclusion:

The MVP Implementation phase successfully transforms the design into a **working prototype**. At this stage:

- The **authentication flow** (register, login, session, reset) is functional.
- Data is securely stored and retrieved from the database.
- Core features are **tested for functionality and security**.
- The project is properly version-controlled, ensuring stability and scalability.