COLLEGE CODE:9628

COLLEGE NAME:UNIVERSITY COLLEGE OF ENGINEERING, NAGERCOIL.

DEPARTMENT:COMPUTER SCIENCE AND ENGINEERING.

STUDENT NM-ID:

443E0C77651A61FB2E34B443B6BDB61B

ROLLNO:962823104103

DATE:22/9/2025

Completed the project named as

Phase:1 – FE

NAME:LOGIN AUTHENTICATION SYSTEM

SUBMITTED BY,

NAME:ABINOV I

MOBILE NO:6383764595

Login Authentication System

Phase 1 – Problem Understanding & Requirements

---

1. Problem Statement

In today's digital landscape, security plays a critical role in web applications. Almost every online platform, from e-commerce portals to educational systems, requires a secure and reliable way for users to create accounts and authenticate themselves. Unauthorized access can lead to severe consequences such as data theft, privacy violations, and reputational damage to organizations. Therefore, a login authentication system serves as the first line of defense in protecting sensitive data and ensuring that only authorized users gain access to application resources.

The proposed project focuses on designing and implementing a **Login Authentication System using AngularJS**. The system will include two primary modules:

1. **Registration Module** – Enables new users to create accounts using their email and password.
2. **Login Module** – Allows registered users to securely log in with their credentials and access the system's dashboard.
3.

The system will also integrate important features such as:

- Input validation (ensuring email format correctness and password strength).
- Error handling for incorrect login attempts.
- Session persistence (using AngularJS `$rootScope` or browser `localStorage`).
- Mock API services for login and registration to simulate backend behavior.
-

The system ensures that users are guided through the authentication process with clear validation messages, preventing both technical errors and poor user experience. By implementing this system, organizations can demonstrate secure application access, developers can practice modular AngularJS authentication workflows, and end-users benefit from safe access to web services.

2. Users and Stakeholders

To properly design the system, it is essential to identify all users and stakeholders, as each plays a unique role in the authentication lifecycle.

2.1 End Users

- Individuals who will interact with the system by registering and logging in.
- Their primary interest is secure, easy-to-use, and error-free access.
- Example: Students accessing a university portal, customers logging into an e-commerce site, or employees accessing a company dashboard.

2.2 Developers

- Responsible for building the authentication logic, ensuring security best practices, and maintaining system stability.
- They design features like form validation, error handling, and session simulation.
- Developers also simulate backend APIs (since Phase 1 emphasizes mock services).

2.3 System Administrators

- Ensure that the system runs smoothly in deployment.
- They handle session security, prevent breaches, and manage system reliability.
- Though not directly coding, administrators set policies regarding password rules, session limits, and user account management.

2.4 Organizations

- Represent the businesses or institutions deploying the authentication system.
- They benefit from protecting sensitive customer and employee information.
- Ensuring data privacy and compliance (GDPR, HIPAA, etc.) is crucial for reputation and legal adherence.

3. User Stories

User stories help capture the requirements from different perspectives. They also define how the system should behave under real-world usage scenarios.

1. **As a new user,** I want to create an account with my email and password so that I can access the system.

2. **As a user,** I want to log in using my registered email and password so that I can reach the dashboard.
3. **As a user,** I want to see error messages if I enter invalid credentials so that I know what went wrong.
4. **As a system,** I want to validate email format and password strength during registration to ensure account security.
5. **As a system,** I want to simulate a session using $rootScope or localStorage so that the user remains logged in until logout.
6. **As a user,** I want to log out of the system whenever I choose, so that my account is not left open on shared devices.
7. **As an administrator,** I want to ensure that sessions are cleared when users log out or close the browser.

Each user story highlights the needs of both the system and the individuals interacting with it.

4. MVP Features

For the Minimum Viable Product (MVP), the goal is to implement the **essential features** that demonstrate the system's working authentication flow.

Core MVP Features

1. **Registration Form**
   - Fields: Email, Password.
   - Validates that the email is properly formatted.
   - Password must meet minimum length criteria (e.g., 6+ characters).
   -
2. **Login Form**
   - Fields: Email, Password.
   - Validates inputs (non-empty, proper format).
   - Accepts only registered/mock credentials.
   -
3. **AngularJS Form Validation**
   - Required field validation.
   - Email regex validation.
   - Password length and strength rules.
   -
4. **Mock Authentication Service**
   - Uses hardcoded values or in-memory data structures for credentials.
   - Simulates backend login and registration functionality.
   -
5. **Session Simulation**
   - Uses $rootScope or browser localStorage to track logged-in users.

    - Sessions persist until logout or browser closure.
    -
6. **Redirection Logic**
    - Successful login redirects users to the dashboard.
    - Failed login remains on the login page with error feedback.
    -
7. **Error Messaging System**
    - Displays appropriate error messages for invalid login attempts, empty inputs, or incorrect formats.
    -
8. **Logout Functionality**
    - Clears session and redirects users to the login page.

5. Wireframes and API Endpoint List

5.1 Wireframes

1. **Registration Page**
    - Fields: Email, Password
    - Button: “Register”
    - Validation messages below fields
2. **Login Page**
    - Fields: Email, Password
    - Button: “Login”
    - Error messages for invalid credentials
3. **Dashboard Page**
    - Displays: “Welcome, [User Email]”
    - Button: “Logout”

(Wireframes can be created with tools like Figma, Balsamiq, or even drawn by hand in final submission.)

5.2 Mock API Endpoints (AngularJS Service Simulation)

1. **POST /register**
    - Input: { email, password }
    - Action: Stores new user credentials (mocked in service).
    - Output: { success: true, message: "Registration successful" }
    -
2. **POST /login**
    - Input: { email, password }
    - Action: Compares against registered/mock credentials.

    - Output:
        - Success → { success: true, token: "mockToken123" }
        - Failure → { success: false, message: "Invalid credentials" }
        -
3. **GET /dashboard**
    - Input: token
    - Action: Returns dashboard data if authenticated.
    - Output: { message: "Welcome to the dashboard" }

6. Acceptance Criteria

The project will be considered successful if the following conditions are met:

1. **Registration**
    - A new user can register with valid email and password.
    - Invalid emails or weak passwords are rejected.
2. **Login**
    - Registered users can log in with correct credentials.
    - Invalid login attempts display a meaningful error message.
3. **Validation**
    - Registration and login forms prevent submission of empty inputs.
    - Incorrect email formats trigger an error.
4. **Session Handling**
    - Session persists until logout or browser closure.
    - Successful login redirects users to the dashboard.
5. **Logout**
    - Logout clears session data.
    - Users are redirected back to the login page.
6. **Usability**
    - Error messages are clear and positioned near the input fields.
    - System behavior is consistent with real-world authentication systems.

7. Conclusion

The Login Authentication System provides a foundational understanding of **secure user authentication workflows** in AngularJS. By implementing registration, login, validation, mock APIs, and session persistence, the project meets both academic and practical requirements.

This **Phase 1 document** establishes problem understanding and requirements, ensuring

that all stakeholders—end users, developers, administrators, and organizations—are aligned. The next phase (Phase 2) will focus on **Solution Design & Architecture**, including component diagrams, data flow, and integration strategies.