COLLEGE CODE:9628

COLLEGE NAME:UNIVERSITY COLLEGE OF ENGINEERING, NAGERCOIL.

DEPARTMENT:COMPUTER SCIENCE AND ENGINEERING.

STUDENT NM-ID:

443E0C77651A61FB2E34B443B6BDB61B

ROLLNO:962823104103

DATE:29/9/2025

Completed the project named as

Phase:3 – FE

NAME:LOGIN AUTHENTICATION SYSTEM

SUBMITTED BY,

NAME:ABINOV I

MOBILE NO:6383764595

Login Authentication System – Phase 3

---

1. Introduction

Authentication is one of the most crucial aspects of any modern web application. It ensures that only authorized users gain access to specific resources or functionalities. Without proper authentication, sensitive data and system integrity could be compromised.

This project focuses on developing a **Login Authentication System** using **AngularJS**, a front-end JavaScript framework. The system provides a **secure and simple login process** using **hardcoded credentials** for this MVP (Minimum Viable Product) phase. The project ensures that a user can log in, be redirected to a secure dashboard, and maintain their session using $rootScope or localStorage.

1.1 Objective

The main objective of this MVP is to:

- Develop a basic authentication mechanism without a backend.
- Perform form validation on the login inputs using AngularJS.
- Simulate authentication with hardcoded credentials.
- Redirect authenticated users to a dashboard.
- Manage user sessions using localStorage or $rootScope.
- Prepare the system for future integration with real backend APIs.

---

2. Phase 3 Goals

This phase involves building the **minimum viable product**, focusing on the essential features required to demonstrate the authentication workflow.

2.1 Scope

- Creation of the login and dashboard pages.
- Implementation of AngularJS routing to navigate between views.
- Session handling using localStorage.
- Mock authentication logic with hardcoded credentials.
- GitHub-based version control.

2.2 Deliverables

At the end of this phase, the deliverables include:

1. Fully functional login system with validation.
2. Dashboard page accessible only to authenticated users.
3. Code repository on GitHub.
4. Documentation outlining the development and testing process.

---

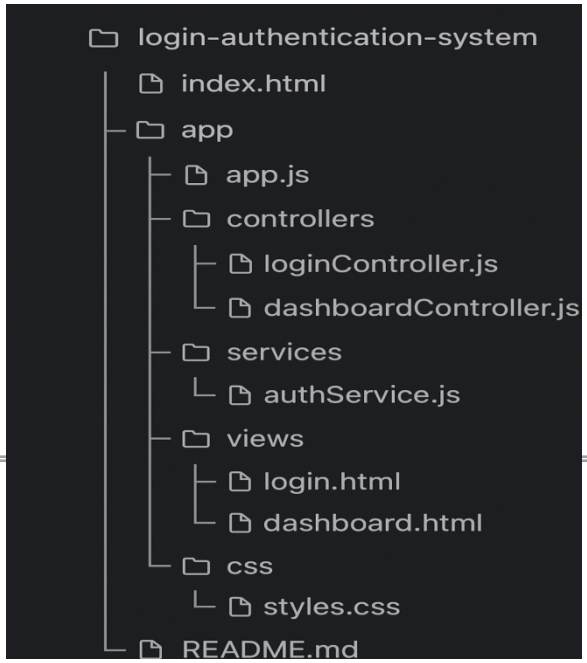
3. Project Setup

3.1 Tools & Technologies Used

Tool/Technology	Purpose
AngularJS	Front-end framework for dynamic UI and routing
HTML5 & CSS3	Structure and styling of pages
JavaScript	Application logic and service integration
Visual Studio Code	Code editor
Git & GitHub	Version control and collaboration
Google Chrome	Browser for testing

3.2 Project Structure

A well-organized folder structure was designed to maintain modularity and clarity.



3.3 Installing AngularJS

AngularJS was included via **Content Delivery Network (CDN)** for quick setup.

Code snippet for index.html:

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular-route.min.js"></script>
```

4. Core Features Implementation

4.1 Login Page Design

The **login page** contains two input fields – **Email** and **Password**. AngularJS validations ensure:

- The email format is valid.
- The password field is not empty.
- Invalid inputs are flagged before form submission.

HTML Code (login.html):

```
<div class="login-container" ng-controller="LoginController">
  <form name="loginForm" ng-submit="login()" novalidate>
    <h2>Login</h2>

    <div>
      <label>Email:</label>
      <input type="email" name="email" ng-model="user.email" required />
      <span ng-show="loginForm.email.$dirty && loginForm.email.$invalid">
        Valid email is required
      </span>
    </div>

    <div>
      <label>Password:</label>
      <input type="password" name="password" ng-model="user.password" required />
      <span ng-show="loginForm.password.$dirty && loginForm.password.$invalid">
        Password is required
      </span>
    </div>

    <button type="submit" ng-disabled="loginForm.$invalid">Login</button>
    <p class="error">{{errorMessage}}</p>
  </form>
</div>
```

4.2 Authentication Logic

The authentication mechanism is handled by a dedicated AngularJS service, authService. This service compares user input with hardcoded credentials.

Code (authService.js):

```
app.service('authService', function() {
  var validUser = {
    email: "user@example.com",
    password: "password123"
  };

  this.validateUser = function(email, password) {
    return (email === validUser.email && password === validUser.password);
  };
});
```

Note:
In future phases, this hardcoded data will be replaced with a backend API and secure database.

4.3 Login Controller

The controller manages user interaction, invokes the authentication service, and handles redirection upon success.

Code (loginController.js):

```
app.controller('LoginController', function($scope, $location, $rootScope, authService) {
  $scope.user = {};
  $scope.errorMessage = "";

  $scope.login = function() {
    if (authService.validateUser($scope.user.email, $scope.user.password)) {
      $rootScope.isLoggedIn = true;
      localStorage.setItem("loggedIn", true);
      $location.path("/dashboard");
    } else {
      $scope.errorMessage = "Invalid Email or Password!";
    }
  };
});
```

4.4 Dashboard Page

The **dashboard page** is accessible only after successful login.
Unauthorized users attempting to access this route are redirected back to the login page.

HTML Code (dashboard.html):

```
<div class="dashboard-container" ng-controller="DashboardController">
  <h2>Welcome to Dashboard!</h2>
  <p>You have successfully logged in.</p>
  <button ng-click="logout()">Logout</button>
</div>
```

Dashboard Controller (dashboardController.js):

```
app.controller('DashboardController', function($scope, $location, $rootScope) {
  if (!localStorage.getItem("loggedIn")) {
    $location.path("/login");
  }

  $scope.logout = function() {
    $rootScope.isLoggedIn = false;
    localStorage.removeItem("loggedIn");
    $location.path("/login");
  };
});
```

4.5 Routing Configuration

AngularJS $routeProvider was used to define application routes.

Code (app.js):

```
var app = angular.module('loginApp', ['ngRoute']);

app.config(function($routeProvider) {
  $routeProvider
    .when('/login', {
      templateUrl: 'app/views/login.html',
      controller: 'LoginController'
    })
    .when('/dashboard', {
      templateUrl: 'app/views/dashboard.html',
      controller: 'DashboardController'
    })
    .otherwise({
      redirectTo: '/login'
    });
});
```

---

5. Data Storage Using Local State

Since there is no backend in this MVP, **localStorage** was chosen to simulate session management.

5.1 How It Works

- When a user logs in successfully:
  - localStorage.setItem("loggedIn", true) is executed.
- Before accessing the dashboard, the application checks:
  - localStorage.getItem("loggedIn").
- On logout:
  - localStorage.removeItem("loggedIn") clears the session.

This approach ensures the session persists even after refreshing the browser window.

---

6. Testing Core Features

6.1 Testing Approach

The system was tested using **manual testing** techniques to verify:

- Input validation
- Authentication logic
- Navigation and redirection
- Session persistence

6.2 Test Cases

Test Case ID	Description	Input	Expected Result
TC-01	Empty Email	""	Error: "Valid email required"
TC-02	Empty Password	"user@example.com", ""	Error: "Password required"
TC-03	Invalid Credentials	"abc@test.com", "wrongpass"	Error: "Invalid Email or Password!"
TC-04	Valid Credentials	"user@example.com", "password123"	Redirect to /dashboard
TC-05	Logout Functionality	Click Logout	Redirect to /login, session cleared
TC-06	Session Persistence	Refresh browser after login	Remain logged in until logout

6.3 Test Results

- All input validation checks passed successfully.
- Dashboard was inaccessible without proper login credentials.
- Logout correctly cleared session data.
- LocalStorage ensured session persistence across page refreshes.

---

7. Version Control Using GitHub

Version control plays a vital role in team collaboration and maintaining code integrity. Git and GitHub were used for tracking changes.

7.1 Git Workflow

1. **Initialize Repository:**
2. git init

3. **Add Project Files:**
4. git add .

5. **Commit Changes:**
6. git commit -m "Initial commit - Login Authentication MVP"

7. **Connect to GitHub:**
8. git remote add origin https://github.com/username/login-auth-system.git

9. **Push Changes:**
10. git push -u origin main

7.2 Benefits of GitHub

- Centralized code repository.
- Collaboration with multiple developers.
- Easy rollback to previous versions.
- Transparent version history.

---

8. Future Enhancements

Once the MVP is complete, several advanced features can be added:

1. **Backend Integration:**
   - Replace hardcoded credentials with REST API calls.
   - Connect to a database for storing user information.
2. **Security Improvements:**
   - Implement password hashing and encryption.
   - Add CSRF protection and secure session handling.
3. **Role-Based Access Control:**
   - Create separate roles like Admin, User, etc.
   - Control dashboard access based on role.
4. **Enhanced UI/UX:**
   - Improve design with responsive frameworks like Bootstrap.
   - Provide a smoother user experience.
5. **Unit Testing:**
   - Add automated test scripts for AngularJS components.

---

9. Conclusion

The **Login Authentication System MVP** successfully implements a functional authentication process using AngularJS. It demonstrates:

- **Secure Login** with basic validation.
- **Session Management** using localStorage.
- **Routing and Navigation** between pages.
- **Logout and Access Restriction** for unauthorized users.

This MVP lays a solid foundation for future development, where backend integration, enhanced security measures, and a better UI will transform it into a production-ready system.

By completing Phase 3, the team has achieved a critical milestone in building a robust authentication framework, setting the stage for scalability and advanced features.