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:4 – FE

NAME:LOGIN AUTHENTICATION SYSTEM

SUBMITTED BY,

NAME:ABINOV I

MOBILE NO:6383764595

Login Authentication System (AngularJS)– Phase 4

1. Introduction

Overview:

The Login Authentication System provides secure authentication for users through email and password. Phase 4 extends the system to include:

- Full dashboard with user info.
- Profile management.
- Logout functionality.
- Security enhancements and session management.
- Multi-page navigation.
  Deployment-ready setup.

Objectives:

- Multi-page AngularJS application. Role-based
- routing and access control. Enhanced UX/UI for
- mobile and desktop. API-driven authentication
- replacing mock services. Performance, caching, and
- security improvements.

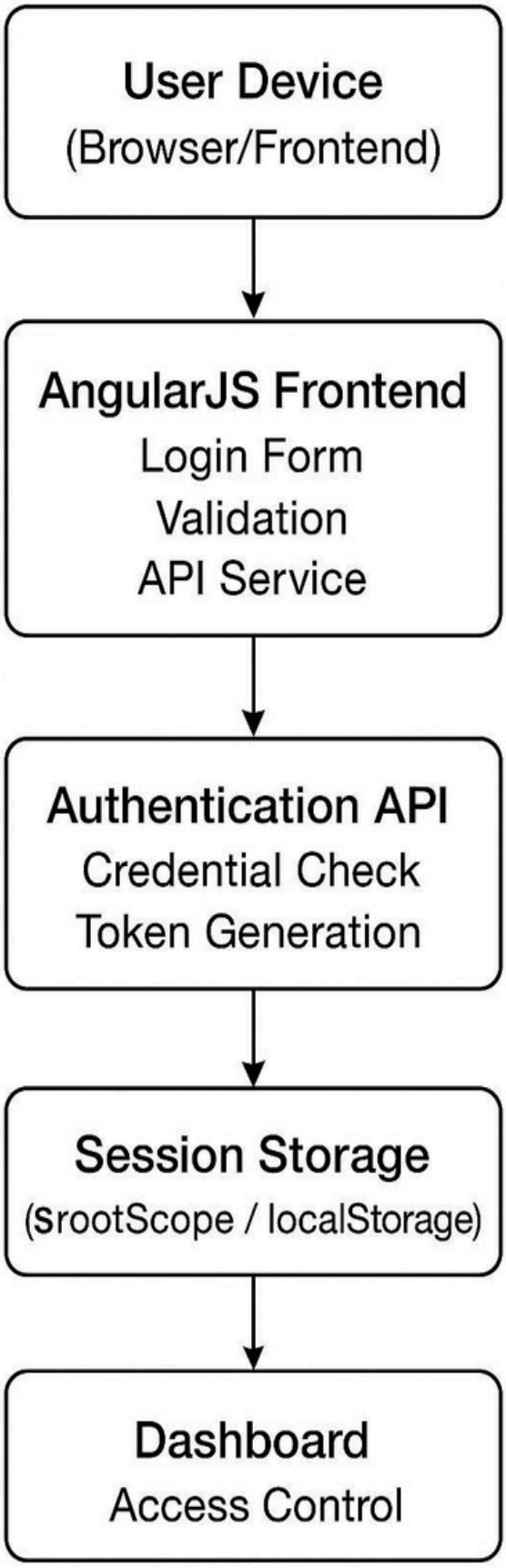
Benefits:

- Realistic login workflow.
- Secure session management.
- Improved navigation and usability.
- Ready for production deployment.

---

2. System Architecture (Extended)

Architecture Diagram:


User Device
(Browser/Frontend)
AngularJS Frontend
Login Form
Validation
API Service
Authentication API
Credential Check
Token Generation
Session Storage
(srootScope / localStorage)
Dashboard
Access Control

- Frontend: Handles multi-page navigation, validation, and API interaction.
- API: Verifies credentials, returns session tokens.
- Session Storage: Maintains logged-in state securely.
- Dashboard & Profile Pages: Displays user info and allows updates.
- Logout: Clears session and redirects to login.
-

---

3. UI/UX Improvements (Extended)

3.1 Multi-Page Navigation

- **Login Page:** Email/password entry with validation.
- **Dashboard:** User welcome message, recent activity, stats.
- **Profile Page:** Editable fields like name, email, password.
- **Logout Button:** Always visible in header/navbar.
-

3.2 Responsive Layout

- Mobile-first design using Flexbox and Grid.
- Collapsible menu for smaller devices.
- Adaptive font size and buttons.
-

3.3 Enhanced Feedback

- Inline validation messages.
- Success/failure toasts for actions like login or profile update.
- Smooth transitions between pages.
-

3.4 Accessibility

- ARIA labels on all buttons and input fields.
- Keyboard navigation across forms.
- High contrast theme option.
-

---

4. API & Authentication Enhancements

4.1 AuthService (AngularJS Service)

```
angular.module('loginApp').service('AuthService', function($http, $q) {
    this.login = function(user) {
        return $http.post('http://localhost:3000/login', user)
            .then(response => response.data)
            .catch(error => $q.reject(error.data));
    };
    this.getProfile = function(token) {

        return $http.get('http://localhost:3000/profile', {
            headers: { 'Authorization': 'Bearer ' + token }
        }).then(res => res.data);
    };
    this.updateProfile = function(user, token) {

        return $http.put('http://localhost:3000/profile', user, {
            headers: { 'Authorization': 'Bearer ' + token }
        }).then(res => res.data);
    };
});
```

4.2 API Workflow

1. User submits login form.
2. Frontend sends credentials via AuthService.
3. API validates credentials:
    - Success → Returns user info + token.
    - Failure → Returns error message.
    -
4. Frontend stores session info in $rootScope or localStorage.
5. AuthGuard checks session before accessing protected pages.

---

6. Multi-Page AngularJS Routing

```
angular.module('loginApp', ['ngRoute'])
.config(function($routeProvider, $locationProvider) {
    $routeProvider
        .when('/login', {
            templateUrl: 'login.html',
            controller: 'LoginCtrl'
        })
        .when('/dashboard', {
```

```
            templateUrl: 'dashboard.html',
            controller: 'DashboardCtrl',
            resolve: { auth: authCheck }
        })
        .when('/profile', {
            templateUrl: 'profile.html',
            controller: 'ProfileCtrl',
            resolve: { auth: authCheck }
    }) .otherwise({ redirectTo: '/login' });

    $locationProvider.html5Mode(true);

});
functionauthCheck($q, $location, $rootScope) {

    if($rootScope.user || localStorage.getItem('user')) {
        return$q.resolve();
    } else {
        $location.path('/login');
        return$q.reject();
    }
}
```

---

7. Dashboard Page

```
<!-- dashboard.html -->
<div>
    <h2>Welcome, {{user.name}}</h2>
    <p>Email: {{user.email}}</p>
    <button ng-click="logout()">Logout</button>
</div>
// Dashboard Controller
angular.module('loginApp')
.controller('DashboardCtrl', function($scope, $rootScope, $location) {
    $scope.user = $rootScope.user || JSON.parse(localStorage.getItem('user'));

    $scope.logout = function() {
        $rootScope.user = null;
        localStorage.removeItem('user');
        $location.path('/login');
    };
});
```

---

8. Profile Page

```
<!-- profile.html -->
<div>
  <h2>Profile</h2>
  <form ng-submit="updateProfile()">
    <label>Name</label>
    <input type="text" ng-model="user.name" required>
    <label>Email</label>
    <input type="email" ng-model="user.email" required>
    <label>Password</label>
    <input type="password" ng-model="user.password">
    <button type="submit">Update</button>

  </form>
  <span>{{message}}</span>
</div>
// Profile Controller
angular.module('loginApp')
.controller('ProfileCtrl', function($scope, AuthService, $rootScope) {
  $scope.user = $rootScope.user;
  $scope.updateProfile = function() {
    consttoken = localStorage.getItem('token');
    AuthService.updateProfile($scope.user, token)
      .then(res => {
        $scope.message = "Profile updated successfully!";
        $rootScope.user = res.user;
        localStorage.setItem('user', JSON.stringify(res.user));
      }).catch(err => {
        $scope.message = err.message;
      });
  };
});
```

---

9. Security Enhancements (Extended)

- HTTPS for all API calls.
- JWT token expiration handled in frontend.
- Password strength meter and client-side hashing.
- XSS protection via input sanitization.
- Logout clears session and token.

---

10. Deployment Steps

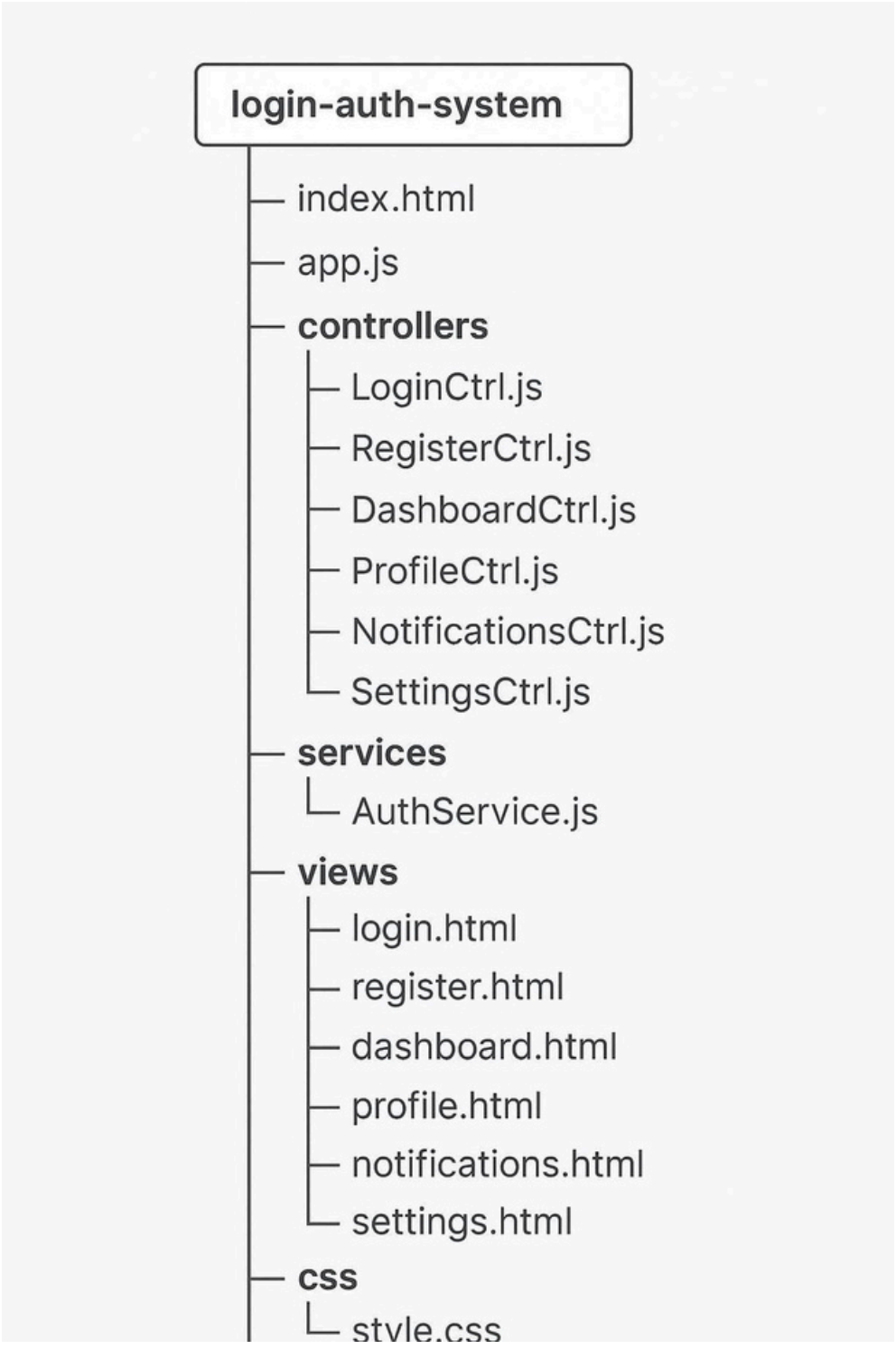
1. Minify AngularJS files for production.
2. Host frontend on **Firebase Hosting**, **GitHub Pages**, or **Nginx**.
3. Deploy backend API on **Heroku** or **Node.js server**.
4. Configure environment variables for API endpoints and tokens.
5. Test login, session persistence, routing, and logout.
6. Ensure HTTPS and security headers enabled.

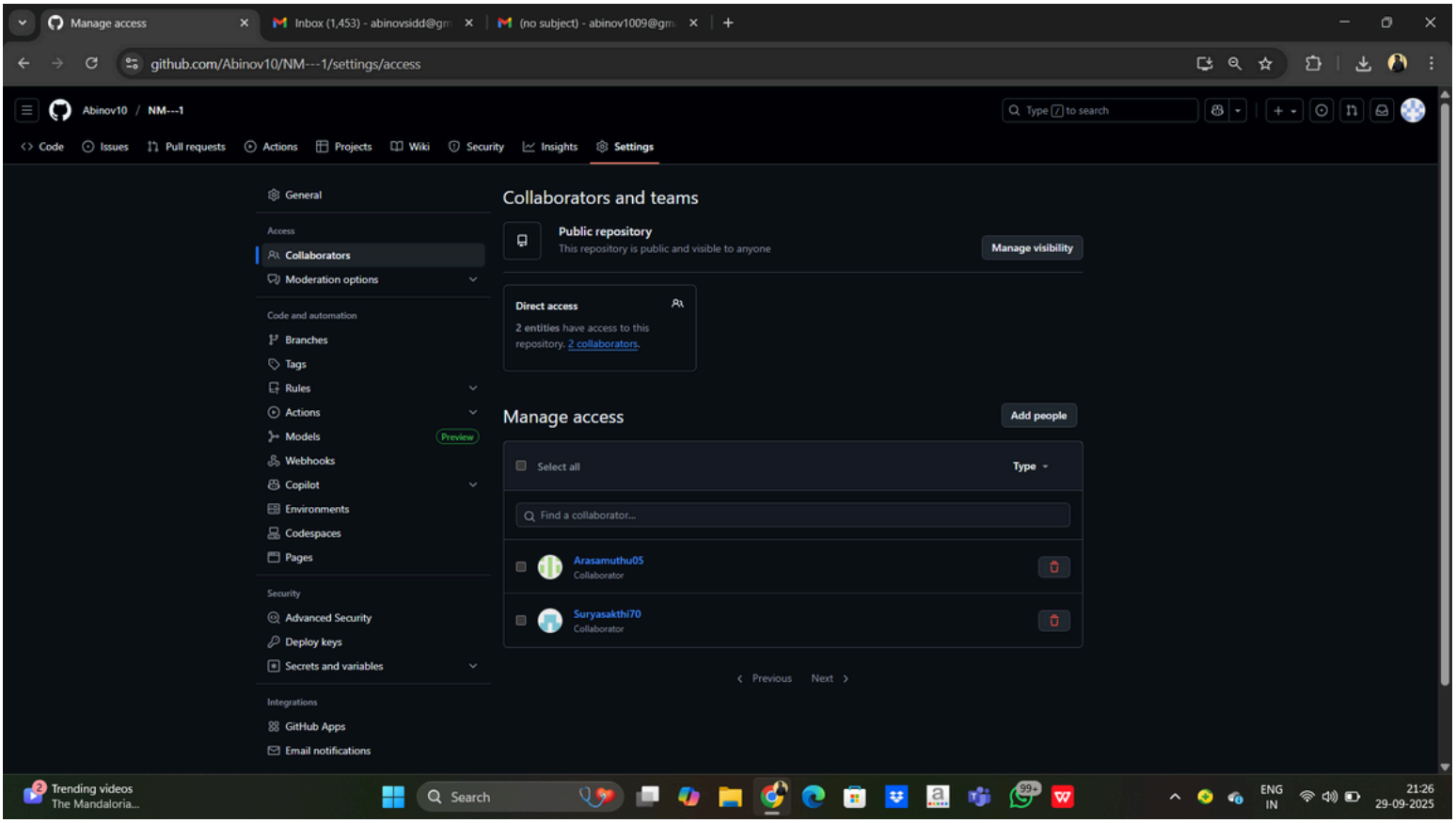
---

11. Future Scope

- Multi-factor authentication (MFA).
- Social login integration (Google, Facebook).
- Role-based dashboard (Admin, User).
- Analytics dashboard for users and admin.
- Notification system for login activity or profile changes.

9. Project Structure (Extended)


login-auth-system
index.html
app.js
controllers
LoginCtrl.js
RegisterCtrl.js
DashboardCtrl.js
ProfileCtrl.js
NotificationsCtrl.js
SettingsCtrl.js
services
AuthService.js
views
login.html
register.html
dashboard.html
profile.html
notifications.html
settings.html
css
style.css



GitHub Link

https://github.com/Abinov10/NM---1