EXPERIMENT NO. 8 - AngularJS

Name of Student	Atharva Prabhu
Class Roll No	42
D.O.P.	20/03/2025
D.O.S.	27/03/2025
Sign and Grade	

AIM : To study AngularJS

PROBLEM STATEMENT:

a. Demonstrate with an AngularJS code one way data binding and two way data binding in AngularJS
b. Implement a basic authentication system for a web application using AngularJS. Create a simple login page that takes a username and password, and upon submission, checks for a hardcoded set of credentials. If the credentials are valid, display a success message; otherwise, show an error message.
   Demonstrate AngularJS controller, module and form directives.
c. Users want to search for books by title, author, or genre. To accomplish this, develop an AngularJS custom filter named bookFilter and include it into the application.
d. Create a reusable and modular custom AngularJS service to handle user authentication. Include this service into an application.

GITHUB LINK - https://github.com/spandandeb/WebXex8

THEORY :

1. **What are directives? Name some of the most commonly used directives in AngularJS application**
   Directives are special attributes in AngularJS that extend HTML functionality by adding custom behavior to elements. They help in data binding, DOM manipulation, and component creation.
   Commonly Used Directives:

- **ng-app** – Defines the root element of an AngularJS application.
- **ng-model** – Binds input fields to the model (two-way data binding).
- **ng-bind** – Displays model data in HTML (one-way binding).
- **ng-repeat** – Iterates over an array to display dynamic lists.
- **ng-click** – Binds a function to a click event.
- **ng-if / ng-show / ng-hide** – Controls element visibility based on conditions.
- **ng-class** – Dynamically applies CSS classes.
- **ng-submit** – Handles form submissions.

2. **What is data binding in AngularJS?**

   Data binding in AngularJS is the process of synchronizing data between the model (JavaScript) and the view (HTML). It ensures automatic updates whenever data changes.

   Types of Data Binding:

   - **One-Way Data Binding (**ng-bind**)** – Updates the view when the model changes but not vice versa.
   - **Two-Way Data Binding** (ng-model**)** – Keeps both the model and view in sync automatically.

     Example:

     ```
     <input type="text" ng-model="name">
     <p>Hello, {{ name }}!</p>
     ```

3. **How is form validation done in angularJS**

   AngularJS provides built-in form validation using **form and input directives**. It tracks user inputs and displays validation errors dynamically.

   Key Directives for Validation:

   - **ng-required** – Marks a field as required.
   - **ng-minlength / ng-maxlength** – Sets minimum and maximum input length.
   - **ng-pattern** – Validates input against a regex pattern.
   - **ng-change** – Triggers a function on input change.
   - **$dirty / $pristine** – Tracks if the field has been modified.
   - **$valid / $invalid** – Checks if the field meets validation rules.

     Example:

     ```
     <form name="myForm">
         <input type="text" name="username" ng-
     model="user.name" ng-required="true">
         <span ng-show="myForm.username.$error.required"> Username is required!
     </span>
     ```

</form>

4. **What is the use of AngularJS Controllers in the application?**
   AngularJS **Controllers** manage application logic by handling data and interactions between the view (HTML) and the model (data). They are defined using ng-controller and provide functions and variables to the view.
   Key Functions of Controllers:
   - **Manage Data** – Store and manipulate scope variables ($scope).
   - **Handle Events** – Process user interactions like clicks and form submissions.
   - **Apply Business Logic** – Perform calculations, validations, and API calls.
   - **Control View Behavior** – Dynamically update UI based on data changes.

5. **What is the use of AngularJS Filters in the application?**
   Filters in AngularJS format and transform data before displaying it in the UI. They help in refining output without modifying the original data.
   Filters are applied using the | (pipe) symbol in expressions, e.g., {{ name | uppercase }}. They enhance data presentation dynamically.
   Common Uses:
   - **Formatting text** – uppercase, lowercase
   - **Number formatting** – currency, number
   - **Filtering arrays** – filter(searching within lists)
   - **Sorting data** – orderBy
   - **Date formatting** – date

CODE:

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>AngularJS Examples</title>
  <link rel="stylesheet" href="styles.css">
  <script src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.8.3/angular.min.js"></script>
  <script src="app.js"></script>
</head>
<body ng-app="myApp">
  <div class="container">
    <h1>AngularJS Examples</h1>

    <!-- A. ONE-WAY AND TWO-WAY DATA BINDING -->
    <section ng-controller="DataBindingController">
      <h2>A. One-Way and Two-Way Data Binding</h2>
```

```
        <div class="form-group">
            <h3>Two-Way Data Binding</h3>
            <label for="twoWayInput">Enter text (two-way binding):</label>
            <input type="text" id="twoWayInput" ng-model="twoWayValue">
            <p>You typed: <strong>{{ twoWayValue }}</strong></p>
        </div>

        <div class="form-group">
            <h3>One-Way Data Binding</h3>
            <p>Current time (one-way binding): <strong>{{ currentTime }}</strong></p>
            <button ng-click="updateTime()">Update Time</button>
        </div>

        <div class="form-group">
            <h3>One-Way Data Binding with ng-bind</h3>
            <p>Using ng-bind: <strong ng-bind="oneWayValue"></strong></p>
            <button ng-click="changeValue()">Change Value</button>
        </div>
    </section>

    <!-- B. AUTHENTICATION SYSTEM -->
    <section ng-controller="AuthController">
        <h2>B. Basic Authentication System</h2>

        <form ng-submit="login()" name="loginForm" novalidate>
            <div class="form-group">
                <label for="username">Username:</label>
                <input type="text" id="username" ng-model="credentials.username" required>
                <p ng-show="loginForm.username.$touched && loginForm.username.$invalid"
class="error">Username is required</p>
            </div>

            <div class="form-group">
                <label for="password">Password:</label>
                <input type="password" id="password" ng-model="credentials.password" required>
                <p ng-show="loginForm.password.$touched && loginForm.password.$invalid"
class="error">Password is required</p>
            </div>

            <button type="submit" ng-disabled="loginForm.$invalid">Login</button>
        </form>

        <p ng-show="loginSuccess" class="success">Login successful! Welcome, {{
credentials.username }}.</p>
        <p ng-show="loginError" class="error">Invalid username or password.</p>

        <div ng-show="isLoggedIn">
            <h3>Welcome to the Dashboard</h3>
            <p>You are now logged in as {{ credentials.username }}</p>
            <button ng-click="logout()">Logout</button>
        </div>
```

```
    </section>

    <!-- C. CUSTOM FILTER FOR BOOK SEARCH -->
    <section ng-controller="BookController">
      <h2>C. Custom Book Filter</h2>

      <div class="form-group">
        <label for="searchQuery">Search books:</label>
        <input type="text" id="searchQuery" ng-model="searchQuery" placeholder="Enter title,
author or genre">
      </div>

      <div class="form-group">
        <label>Search by:</label>
        <select ng-model="searchType">
          <option value="all">All Fields</option>
          <option value="title">Title</option>
          <option value="author">Author</option>
          <option value="genre">Genre</option>
        </select>
      </div>

      <table>
        <thead>
          <tr>
            <th>Title</th>
            <th>Author</th>
            <th>Genre</th>
          </tr>
        </thead>
        <tbody>
          <tr ng-repeat="book in books | bookFilter:searchQuery:searchType">
            <td>{{ book.title }}</td>
            <td>{{ book.author }}</td>
            <td>{{ book.genre }}</td>
          </tr>
        </tbody>
      </table>
    </section>

    <!-- D. AUTH SERVICE IMPLEMENTATION -->
    <section ng-controller="ServiceAuthController">
      <h2>D. Auth Service Implementation</h2>

      <div ng-show="!isAuthenticated()">
        <form ng-submit="login()" name="serviceLoginForm" novalidate>
          <div class="form-group">
            <label for="serviceUsername">Username:</label>
            <input type="text" id="serviceUsername" ng-model="userCredentials.username"
required>
          </div>
```

```
                <div class="form-group">
                    <label for="servicePassword">Password:</label>
                    <input type="password" id="servicePassword" ng-
model="userCredentials.password" required>
                </div>

                <button type="submit">Login with Service</button>
            </form>

            <p ng-show="authError" class="error">{{ authError }}</p>
        </div>

        <div ng-show="isAuthenticated()">
            <h3>Service Authentication Success!</h3>
            <p>You are logged in as: <strong>{{ currentUser().username }}</strong></p>
            <p>User role: <strong>{{ currentUser().role }}</strong></p>
            <button ng-click="logout()">Logout</button>
        </div>
    </section>
  </div>
</body>
</html>

App.js
// Main Application Module
var app = angular.module('myApp', []);

// A. DATA BINDING CONTROLLER
app.controller('DataBindingController', function($scope) {
  // Two-way data binding
  $scope.twoWayValue = "Hello AngularJS";

  // One-way data binding
  $scope.currentTime = new Date().toLocaleTimeString();
  $scope.updateTime = function() {
    $scope.currentTime = new Date().toLocaleTimeString();
  };

  // One-way data binding with ng-bind
  $scope.oneWayValue = "Initial Value";
  $scope.changeValue = function() {
    $scope.oneWayValue = "Updated Value: " + new Date().toLocaleTimeString();
  };
});

// B. AUTHENTICATION CONTROLLER
app.controller('AuthController', function($scope) {
  // Initialize variables
  $scope.credentials = {
    username: '',
```

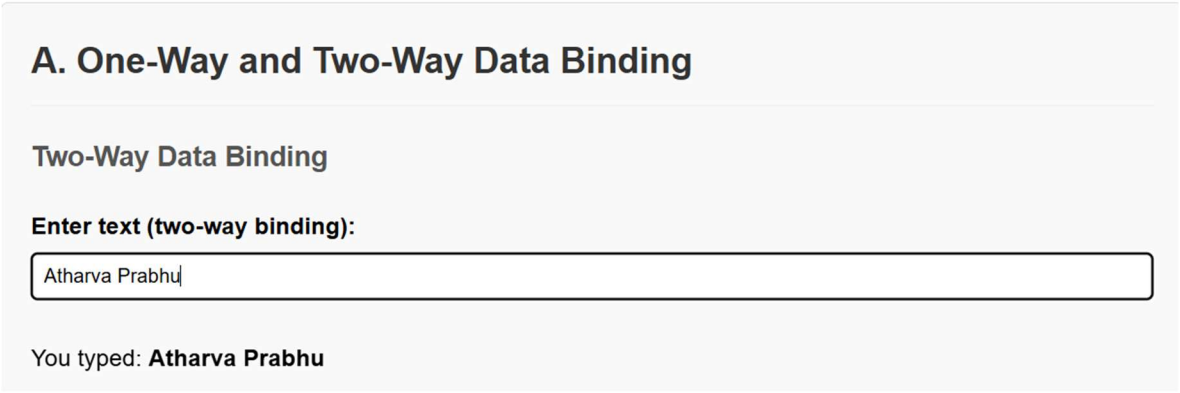
```
        password: ''
    };
    $scope.loginSuccess = false;
    $scope.loginError = false;
    $scope.isLoggedIn = false;

    // Valid credentials (hardcoded for demo)
    var validCredentials = {
        username: 'spandandeb',
        password: '123456'
    };

    // Login function
    $scope.login = function() {
        if ($scope.credentials.username === validCredentials.username &&
            $scope.credentials.password === validCredentials.password) {
            $scope.loginSuccess = true;
            $scope.loginError = false;
            $scope.isLoggedIn = true;
        } else {
            $scope.loginSuccess = false;
            $scope.loginError = true;
            $scope.isLoggedIn = false;
        }
    };

    // Logout function
    $scope.logout = function() {
        $scope.isLoggedIn = false;
        $scope.loginSuccess = false;
    };
});

// C. BOOK CONTROLLER AND CUSTOM FILTER
app.controller('BookController', function($scope) {
    // Sample book data
    $scope.books = [
        { title: 'To Kill a Mockingbird', author: 'Harper Lee', genre: 'Fiction' },
        { title: '1984', author: 'George Orwell', genre: 'Dystopian' },
        { title: 'The Great Gatsby', author: 'F. Scott Fitzgerald', genre: 'Classic' },
        { title: 'Pride and Prejudice', author: 'Jane Austen', genre: 'Romance' },
        { title: 'The Hobbit', author: 'J.R.R. Tolkien', genre: 'Fantasy' },
        { title: 'Brave New World', author: 'Aldous Huxley', genre: 'Dystopian' }
    ];

    // Default search settings
    $scope.searchQuery = '';
    $scope.searchType = 'all';
});

// Custom filter implementation
```

```
app.filter('bookFilter', function() {
    return function(books, query, searchType) {
        if (!query) {
            return books;
        }

        query = query.toLowerCase();
        var result = [];

        for (var i = 0; i < books.length; i++) {
            var book = books[i];

            if (searchType === 'all') {
                if (book.title.toLowerCase().indexOf(query) !== -1 ||
                    book.author.toLowerCase().indexOf(query) !== -1 ||
                    book.genre.toLowerCase().indexOf(query) !== -1) {
                    result.push(book);
                }
            } else if (searchType === 'title' && book.title.toLowerCase().indexOf(query) !== -1) {
                result.push(book);
            } else if (searchType === 'author' && book.author.toLowerCase().indexOf(query) !== -1) {
                result.push(book);
            } else if (searchType === 'genre' && book.genre.toLowerCase().indexOf(query) !== -1) {
                result.push(book);
            }
        }

        return result;
    };
});

// D. AUTH SERVICE IMPLEMENTATION
app.service('AuthService', function() {
    var currentUser = null;

    // Sample user database
    var users = [
        { username: 'atharva', password: 'pass1', role: 'User' },
        { username: 'admin1', password: 'admin123', role: 'Admin' }
    ];

    return {
        // Authenticate user
        login: function(username, password) {
            // Find matching user
            for (var i = 0; i < users.length; i++) {
                if (users[i].username === username && users[i].password === password) {
                    currentUser = {
                        username: users[i].username,
                        role: users[i].role
                    };
```

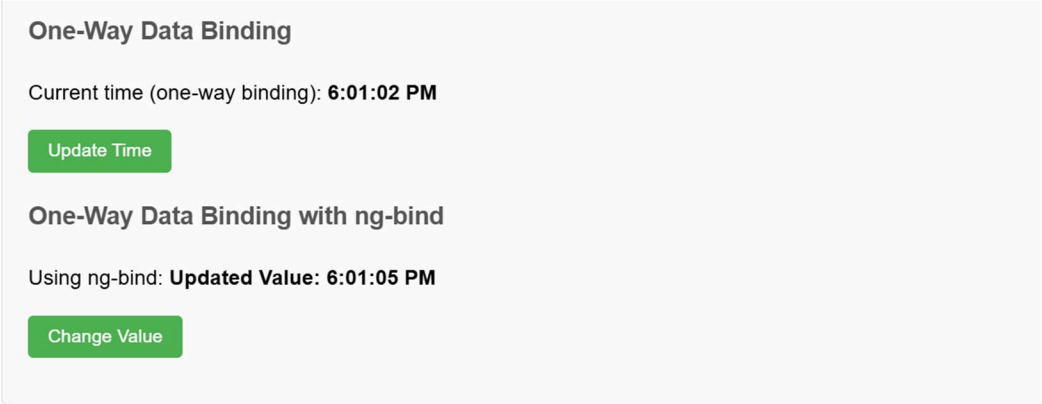
```
                return { success: true };
            }
        }
        return { success: false, message: 'Invalid username or password' };
    },

    // Logout user
    logout: function() {
        currentUser = null;
    },

    // Check if user is authenticated
    isAuthenticated: function() {
        return currentUser !== null;
    },

    // Get current user info
    getCurrentUser: function() {
        return currentUser;
    }
  };
});

// Controller that uses the Auth Service
app.controller('ServiceAuthController', function($scope, AuthService) {
  $scope.userCredentials = {
    username: '',
    password: ''
  };
  $scope.authError = null;

  // Login function using service
  $scope.login = function() {
    var result = AuthService.login($scope.userCredentials.username,
$scope.userCredentials.password);
    if (!result.success) {
      $scope.authError = result.message;
    } else {
      $scope.authError = null;
    }
  };

  // Check authentication status
  $scope.isAuthenticated = function() {
    return AuthService.isAuthenticated();
  };

  // Get current user
  $scope.currentUser = function() {
    return AuthService.getCurrentUser();
  };
```

```
    // Logout function
    $scope.logout = function() {
        AuthService.logout();
    };
});
```

OUTPUT:



This screenshot demonstrates the concept of two-way data binding in a web application. Two-way data binding allows the synchronization of data between the user interface (UI) and the application's data model in real-time



This screenshot illustrates the concepts of one-way data binding and one-way data binding with ng-bind in a web application

B. Basic Authentication System

Username:

Atharva

Password:

••••••

Login

Login successful! Welcome, Atharva.

Welcome to the Dashboard

You are now logged in as Atharva

Logout

C. Custom Book Filter

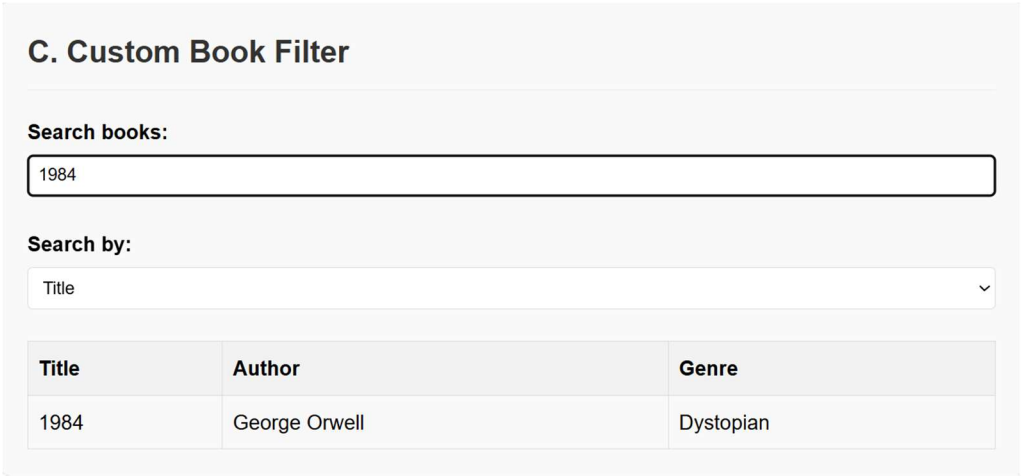
Search books:

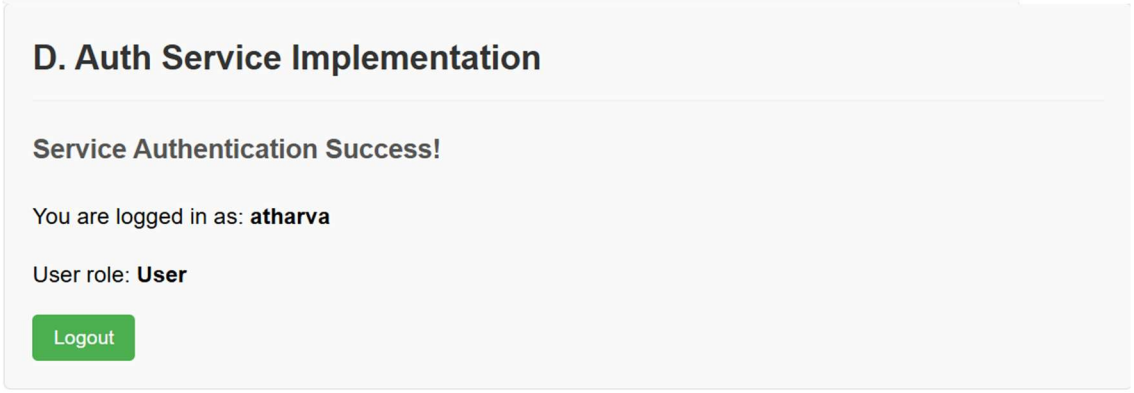
Enter title, author or genre

Search by:

Title

Title	Author	Genre
To Kill a Mockingbird	Harper Lee	Fiction
1984	George Orwell	Dystopian
The Great Gatsby	F. Scott Fitzgerald	Classic
Pride and Prejudice	Jane Austen	Romance
The Hobbit	J.R.R. Tolkien	Fantasy
Brave New World	Aldous Huxley	Dystopian





Here a user can login as a user and admin and they can logout as well it showcases a typical authentication service implementation.

CONCLUSION:

In this experiment, we successfully explored AngularJS by implementing one-way and two-way data binding, a basic authentication system, and a custom book search filter. We used AngularJS directives, controllers, services, and filters to build an interactive web application. The login system validated user credentials, while the book search feature demonstrated custom filtering. Additionally, we implemented form validation using built-in AngularJS directives. This experiment provided hands-on experience in developing dynamic, modular, and responsive applications using AngularJS.