EXPERIMENT NO. 8 - AngularJS

Name of Student	Soham Satpute
Class Roll No	51
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" ng-app="bookApp">
<head>
  <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Book Library - By Soham Satpute</title>
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <header class="text-center">
      <h1>Book Library Application</h1>
      <p>Developed by Soham Satpute</p>
    </header>

    <!-- Main content area -->
    <div ng-controller="MainController as main">
      <!-- Authentication Section -->
      <div ng-if="!main.isAuthenticated" class="auth-container">
        <h2>Login</h2>
        <div class="row">
          <!-- One-way Data Binding Example -->
          <div class="col-md-6">
            <div class="panel panel-default">
              <div class="panel-heading">
                <h3 class="panel-title">One-way Data Binding Example</h3>
              </div>
              <div class="panel-body">
                <p>Enter your name: <input type="text" ng-model="main.userName" class="form-control"></p>
                <p>Hello, {{main.userName}}!</p>
                <p><small>This demonstrates one-way data binding where changes in the model update the view.</small></p>
```

```
                </div>
            </div>
        </div>

        <!-- Two-way Data Binding Example -->
        <div class="col-md-6">
            <div class="panel panel-default">
                <div class="panel-heading">
                    <h3 class="panel-title">Two-way Data Binding Example</h3>
                </div>
                <div class="panel-body">
                    <p>Select your role:</p>
                    <select ng-model="main.userRole" class="form-control">
                        <option value="admin">Admin</option>
                        <option value="user">User</option>
                        <option value="guest">Guest</option>
                    </select>
                    <p>Your selected role: {{main.userRole}}</p>
                    <button class="btn btn-info" ng-click="main.setRole('admin')">Set as Admin</button>
                    <button class="btn btn-info" ng-click="main.setRole('user')">Set as User</button>
                    <p><small>This demonstrates two-way data binding where changes in the view update the model and vice versa.</small></p>
                </div>
            </div>
        </div>

        <!-- Login Form -->
        <div class="panel panel-primary">
            <div class="panel-heading">
                <h3 class="panel-title">Authentication</h3>
            </div>
            <div class="panel-body">
                <form name="loginForm" ng-submit="main.login()" novalidate>
                    <div class="form-group" ng-class="{'has-error': loginForm.username.$invalid && loginForm.username.$touched}">
                        <label for="username">Username:</label>
                        <input type="text" id="username" name="username" class="form-control" ng-model="main.credentials.username" required>
                        <span class="help-block" ng-show="loginForm.username.$invalid && loginForm.username.$touched">Username is required</span>
                    </div>
                    <div class="form-group" ng-class="{'has-error': loginForm.password.$invalid && loginForm.password.$touched}">
                        <label for="password">Password:</label>
                        <input type="password" id="password" name="password" class="form-control" ng-model="main.credentials.password" required>
                        <span class="help-block" ng-show="loginForm.password.$invalid && loginForm.password.$touched">Password is required</span>
                    </div>
                    <button type="submit" class="btn btn-primary" ng-disabled="loginForm.$invalid">Login</button>
                </form>
                <div class="alert alert-danger" ng-if="main.loginError" style="margin-top: 15px;">
                    {{main.loginError}}
                </div>
            </div>
```

```
        </div>
      </div>

      <!-- Book Library Section (shown after authentication) -->
      <div ng-if="main.isAuthenticated">
        <div class="row">
          <div class="col-md-12">
            <div class="alert alert-success">
              Welcome, {{main.currentUser}}! You have successfully logged in.
            </div>
            <button class="btn btn-warning" ng-click="main.logout()">Logout</button>
          </div>
        </div>

        <div class="book-section">
          <h2>Book Library</h2>

          <!-- Search and Filter -->
          <div class="row">
            <div class="col-md-12">
              <div class="form-group">
                <label for="searchBooks">Search Books:</label>
                <input type="text" id="searchBooks" class="form-control" ng-model="main.searchText" placeholder="Search by title, author, or genre">
              </div>
              <div class="form-group">
                <label>Filter by:</label>
                <div class="btn-group">
                  <button class="btn btn-default" ng-class="{'btn-primary': main.filterType === 'all'}" ng-click="main.setFilterType('all')">All</button>
                  <button class="btn btn-default" ng-class="{'btn-primary': main.filterType === 'title'}" ng-click="main.setFilterType('title')">Title</button>
                  <button class="btn btn-default" ng-class="{'btn-primary': main.filterType === 'author'}" ng-click="main.setFilterType('author')">Author</button>
                  <button class="btn btn-default" ng-class="{'btn-primary': main.filterType === 'genre'}" ng-click="main.setFilterType('genre')">Genre</button>
                </div>
              </div>
            </div>
          </div>

          <!-- Book List -->
          <div class="row">
            <div class="col-md-4" ng-repeat="book in main.books | bookFilter:main.searchText:main.filterType">
              <div class="panel panel-default">
                <div class="panel-heading">
                  <h3 class="panel-title">{{book.title}}</h3>
                </div>
                <div class="panel-body">
                  <p><strong>Author:</strong> {{book.author}}</p>
                  <p><strong>Genre:</strong> {{book.genre}}</p>
                  <p><strong>Year:</strong> {{book.year}}</p>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>

      <!-- AngularJS Library -->
```

```
                                        <script
    src="https://ajax.googleapis.com/ajax/libs/a
    ngularjs/1.8.2/angular.min.js"></script>

      <!-- Application Scripts -->
      <script src="app.js"></script>
                                        <script
    src="controllers/mainController.js"></script
    >
                                        <script
    src="services/authService.js"></script>
                                        <script
    src="filters/bookFilter.js"></script>
    </body>
    </html>

    // B. MAIN CONTROLLER
angular.module('bookApp')
   .controller('MainController', ['$scope',
'AuthService', function($scope, AuthService) {
      var vm = this;

      // Initialize controller properties
      vm.userName = 'Guest';
      vm.userRole = 'guest';
      vm.isAuthenticated = false;
      vm.currentUser = '';
      vm.loginError = '';
      vm.credentials = {
         username: '',
         password: ''
      };

      // Book data
      vm.books = [
         { title: 'To Kill a Mockingbird', author:
'Harper Lee', genre: 'Fiction', year: 1960 },
         { title: '1984', author: 'George Orwell',
genre: 'Dystopian', year: 1949 },
         { title: 'The Great Gatsby', author: 'F.
Scott Fitzgerald', genre: 'Fiction', year: 1925 },
         { title: 'Pride and Prejudice', author:
'Jane Austen', genre: 'Romance', year: 1813 },
         { title: 'The Hobbit', author: 'J.R.R.
Tolkien', genre: 'Fantasy', year: 1937 },
         { title: 'The Catcher in the Rye', author:
'J.D. Salinger', genre: 'Fiction', year: 1951 },
         { title: 'Harry Potter and the Sorcerer\'s
Stone', author: 'J.K. Rowling', genre: 'Fantasy',
year: 1997 },
         { title: 'The Lord of the Rings', author:
'J.R.R. Tolkien', genre: 'Fantasy', year: 1954 },
         { title: 'The Da Vinci Code', author: 'Dan
Brown', genre: 'Mystery', year: 2003 }
      ];

      // Search and filter settings
      vm.searchText = '';
      vm.filterType = 'all';

      // Two-way data binding example method
      vm.setRole = function(role) {
         vm.userRole = role;
      };

      // Set filter type
      vm.setFilterType = function(type) {
         vm.filterType = type;
      };

      // Login method
      vm.login = function() {
         vm.loginError = '';

         // Use the AuthService to authenticate
         AuthService.login(vm.credentials.userna
me, vm.credentials.password)
            .then(function(response) {
               if (response.success) {
                  vm.isAuthenticated = true;
                  vm.currentUser =
response.username;
                  vm.loginError = '';
               } else {
                  vm.loginError =
response.message;
               }
            });
      };

      // Logout method
      vm.logout = function() {
         AuthService.logout();
         vm.isAuthenticated = false;
```

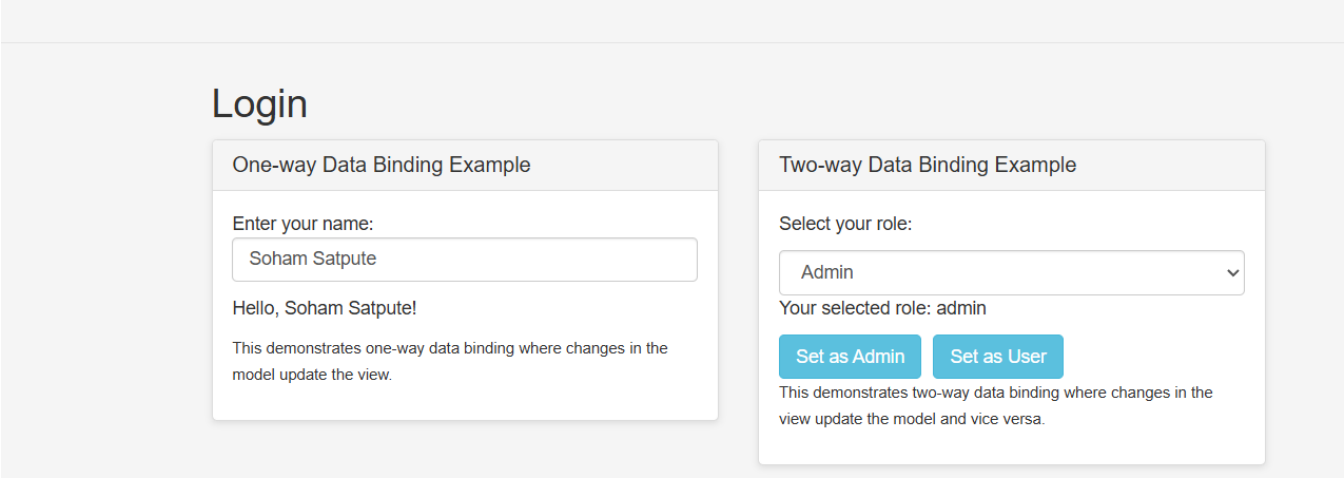
```
    vm.currentUser = '';
    vm.credentials = {
      username: '',
      password: ''
    };
  };

  // Check if user is already authenticated
  if (AuthService.isAuthenticated()) {
    vm.isAuthenticated = true;
    vm.currentUser = AuthService.getCurrentUser();
  }
}]);

// D. AUTH SERVICE IMPLEMENTATION
angular.module('bookApp')
  .service('AuthService', ['$q', function($q) {
    // Private variables
    var currentUser = null;
    var isAuth = false;

    // Hardcoded valid credentials for demonstration
    var validCredentials = [
      { username: 'admin', password: 'admin123', role: 'admin' },
      { username: 'user', password: 'user123', role: 'user' },
      { username: 'soham', password: 'satpute', role: 'admin' }
    ];

    // Service methods
    var service = {
      /**
       * Authenticate user with provided credentials
       * @param {string} username - The username to authenticate
       * @param {string} password - The password to authenticate
       * @returns {Promise} - Promise resolving to authentication result
       */
      login: function(username, password) {
        // Create a deferred object to handle the async operation
        var deferred = $q.defer();

        // Simulate API call delay
        setTimeout(function() {
          // Find matching credentials
          var found = false;
```

```
            var user = null;

                        for (var i = 0; i < validCredentials.length; i++) {
                if (validCredentials[i].username === username &&
                    validCredentials[i].password === password) {
                    found = true;
                    user = validCredentials[i];
                    break;
                }
            }

            if (found) {
                // Set authentication state
                currentUser = {
                    username: user.username,
                    role: user.role
                };
                isAuth = true;

                    // Store in localStorage for persistence
                  localStorage.setItem('currentUser', JSON.stringify(currentUser));

                // Resolve with success
                deferred.resolve({
                    success: true,
                    username: user.username,
                    role: user.role
                });
            } else {
                // Resolve with error
                deferred.resolve({
                    success: false,
                    message: 'Invalid username or password'
                });
            }
        }, 500); // Simulate 500ms delay

        return deferred.promise;
    },

    /**
```

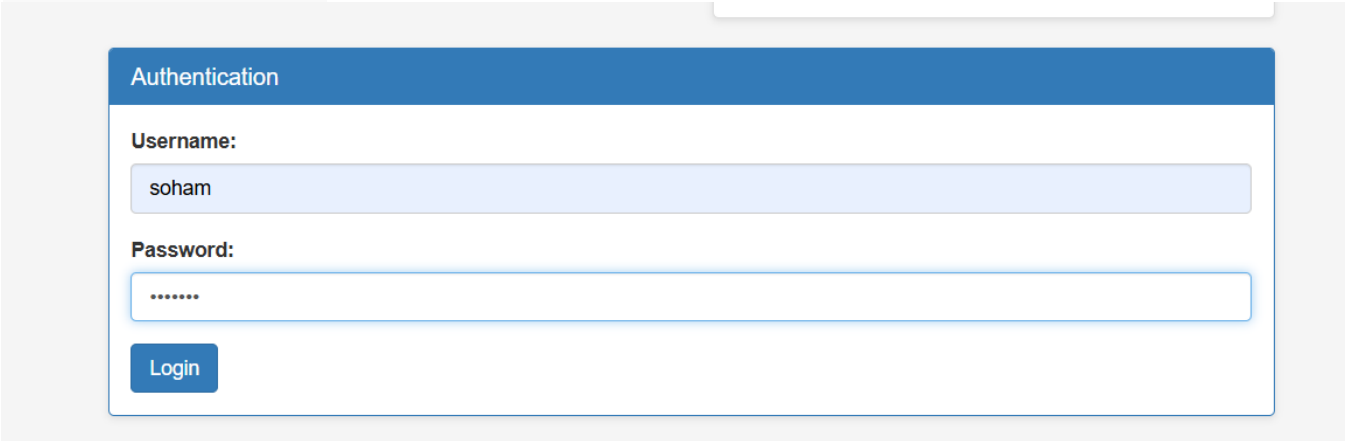
```
         * Log out the current user
         */
        logout: function() {
            currentUser = null;
            isAuth = false;
            localStorage.removeItem('currentU
ser');
        },

        /**
         * Check if user is authenticated
        * @returns {boolean} - Authentication
status
         */
        isAuthenticated: function() {
            return isAuth;
        },

        /**
         * Get current user information
           * @returns {Object|null} - Current
user or null if not authenticated
         */
        getCurrentUser: function() {
                        return currentUser ?
currentUser.username : null;
        },

        /**
         * Get current user role
         * @returns {string|null} - User role or
null if not authenticated
         */
        getUserRole: function() {
          return currentUser ? currentUser.role
: null;
        },

        /**
           * Check authentication status from
localStorage (for page refreshes)
         */
        checkAuthStatus: function() {
                            var storedUser =
localStorage.getItem('currentUser');
            if (storedUser) {
                try {
```

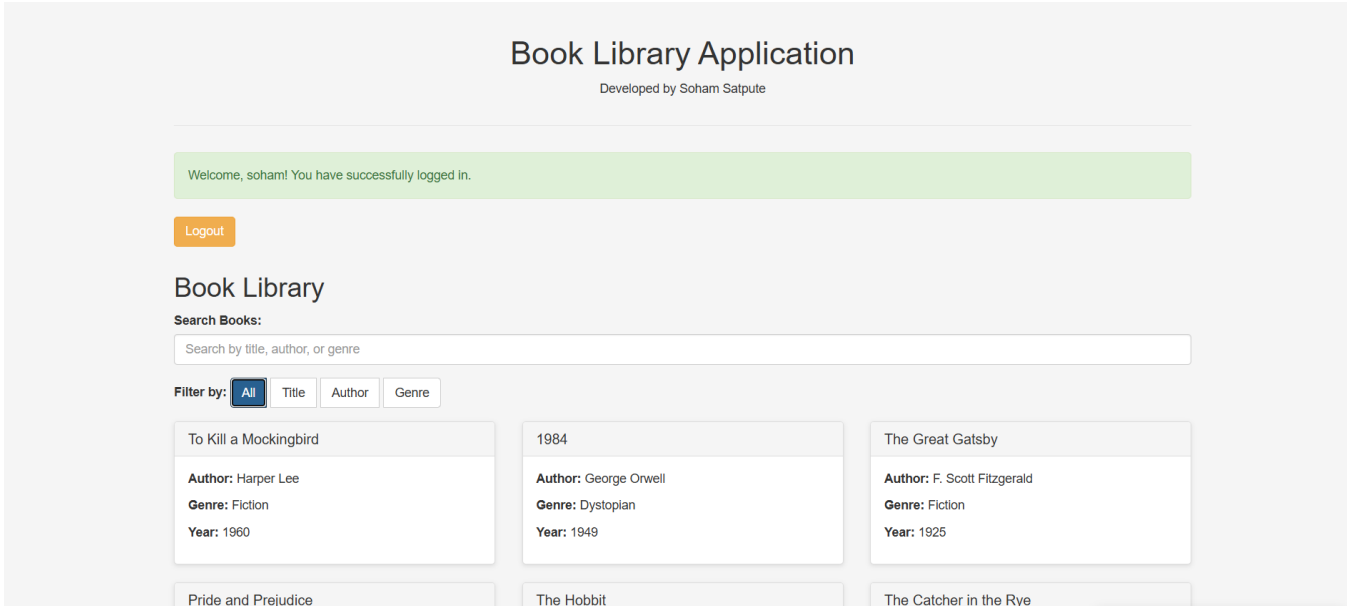
```
                                currentUser =
JSON.parse(storedUser);
                    isAuth = true;
                } catch (e) {
                    // Invalid stored data
                     localStorage.removeItem('cur
rentUser');
                }
            }
        }
    };

    return service;
}]);
```

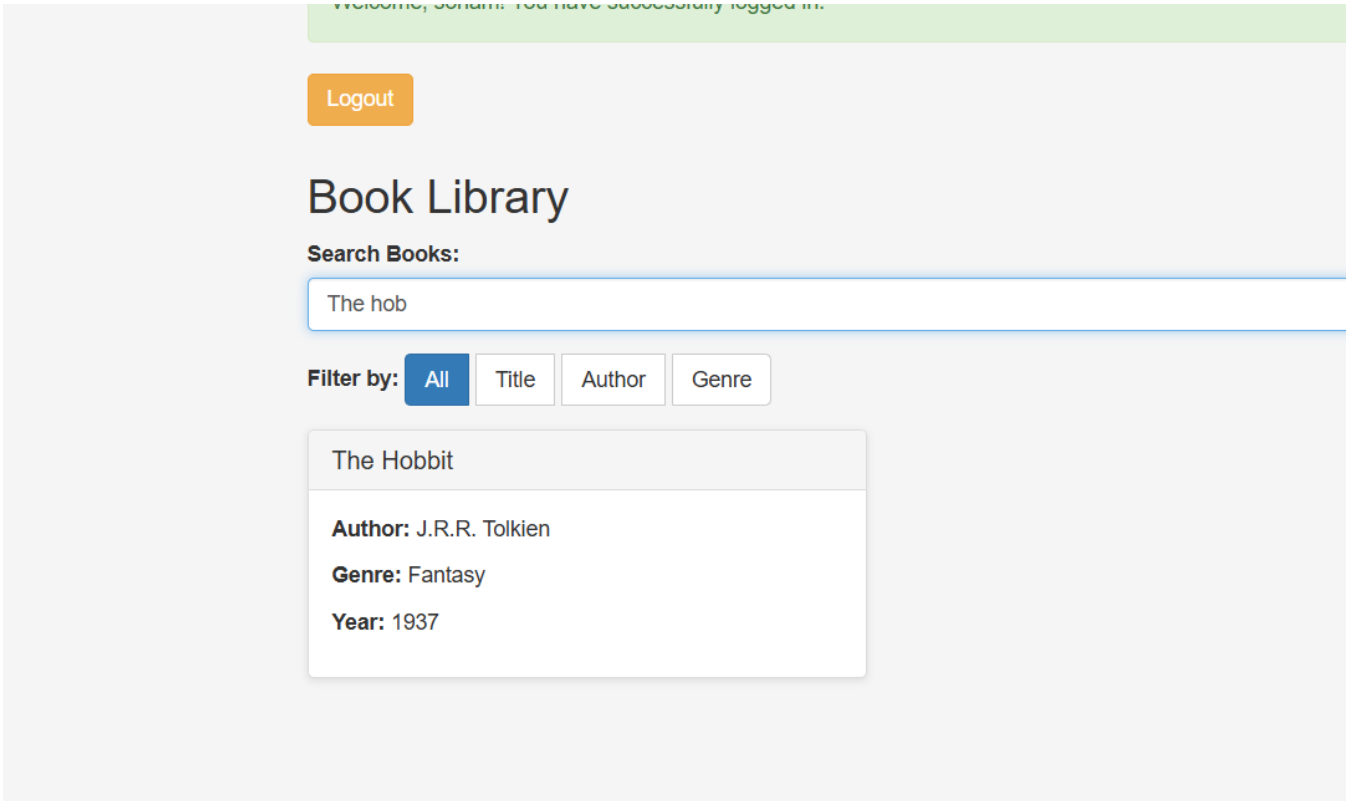
OUTPUT:

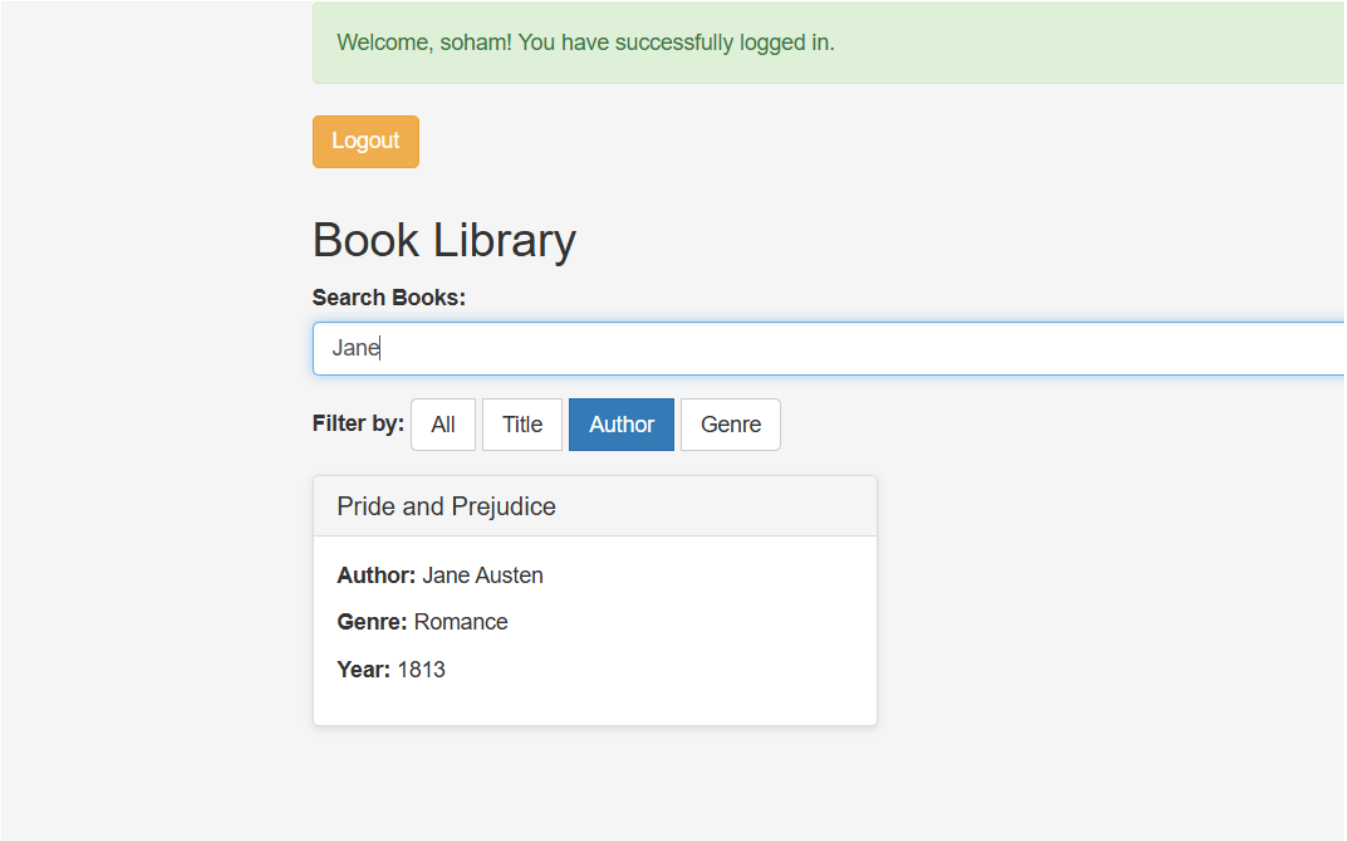


This screenshot illustrates the concepts of one-way data binding and one-way data binding with ng-bind in a web application




Book Library Application
Developed by Soham Satpute
Welcome, soham! You have successfully logged in.
Logout
Book Library
Search Books:
Search by title, author, or genre
Filter by: All Title Author Genre
To Kill a Mockingbird
Author: Harper Lee
Genre: Fiction
Year: 1960
1984
Author: George Orwell
Genre: Dystopian
Year: 1949
The Great Gatsby
Author: F. Scott Fitzgerald
Genre: Fiction
Year: 1925
Pride and Preiudice
The Hobbit
The Catcher in the Rye


Logout
Book Library
Search Books:
The hob
Filter by: All Title Author Genre
The Hobbit
Author: J.R.R. Tolkien
Genre: Fantasy
Year: 1937



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.