EXPERIMENT NO. 8

Name of Student	**Chirag Choudhary**
Class Roll No	**10**
D.O.P.	**01-04-2025**
D.O.S.	**08-04-2025**
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.

THEORY:

1) What are directives? Name some of the most commonly used directives in AngularJS application

   Directives in AngularJS are **special markers** (attributes, elements, or classes) that extend **HTML functionality** by attaching custom behaviors to DOM elements.
   They enable **dynamic content manipulation** and are essential in AngularJS applications for building reusable components.

   Commonly used Directives in Angular JS

   ng-app – Defines the root element of an AngularJS application.

   □ **ng-model –** Binds an input field to a variable in the scope (two-way data binding).

   ng-repeat – Loops through an array to display dynamic lists.

   ng-if – Conditionally renders elements based on an expression.

   ng-show / ng-hide – Shows or hides elements based on a condition.

   □ **ng-click –** Adds a click event listener to elements.

2) What is data binding in AngularJS?

   Data binding in AngularJS is the **automatic synchronization** of data between the **model (JavaScript variables)** and the **view (HTML UI elements)**. It helps in building **dynamic applications** without manually manipulating the DOM.

   Types of Data Binding in AngularJS

   1. **One-Way Data Binding (Interpolation & Expressions)**
      - Updates the view when the model changes but **not vice versa**.
      - Achieved using {{ expression }} (interpolation) or directives like ng-bind.

   ```
   <p>Hello, {{ username }}!</p>
   <p ng-bind="username"></p>
   ```

   2. **Two-Way Data Binding (ng-model)**
      - Synchronizes data **both ways**—when the user updates the UI, the model updates, and vice versa.

   ```
   <input type="text" ng-model="username">
   <p>Your name: {{ username }}</p>
   ```

   Data binding in AngularJS makes the application **responsive and interactive** by **automatically updating** the UI when the data changes, reducing the need for manual DOM manipulation

3. How is form validation done in angularJS

   AngularJS provides **built-in form validation** using directives and the ng-model directive to track user inputs. It helps ensure **data correctness** before submission.

   Key Features of Form Validation in AngularJS

   1. **Uses AngularJS directives** like ng-required, ng-minlength, ng-pattern, etc.
   2. **Real-time validation** – Errors appear as users type.
   3. **Built-in validation states** – $valid, $invalid, $dirty, $pristine track form status.
   4. **Custom validation** – Developers can define custom validation rules.

   AngularJS **simplifies form validation** with built-in directives, real-time error handling, and easy tracking of form states. This ensures **better user experience** and **data integrity**.

4. What is the use of AngularJS Controllers in the application?

   In AngularJS, **controllers** are used to manage the **application logic** and **data**. They act as an interface between the **view (HTML)** and the **model (data)**, making applications **dynamic and interactive**.

   Key Uses of AngularJS Controllers

   1. **Data Binding** – Controls how data is displayed in the view using $scope.
   2. **Business Logic** – Defines functions to handle user actions and process data.
   3. **Communication with Services** – Calls APIs or services to fetch/update data.
   4. **Event Handling** – Manages user interactions like button clicks and form submissions.
   5. **Separation of Concerns** – Keeps business logic separate from the view (HTML).

5. What is the use of AngularJS Filters in the application?

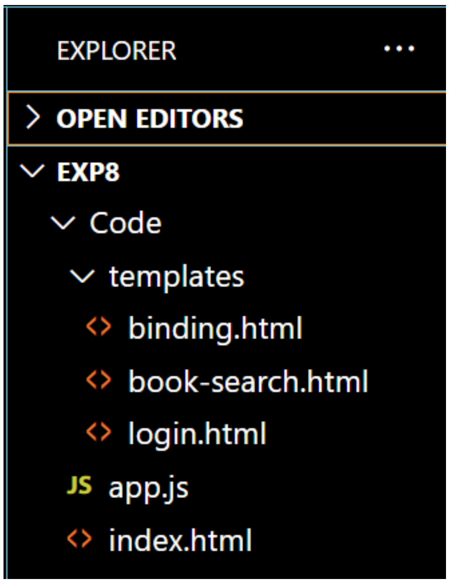
   In AngularJS, **filters** are used to **format, modify, or transform** data before displaying it in the view. They help in presenting data in a **more readable and user-friendly format** without changing the original data in the model.

   Key Uses of AngularJS Filters

   1. **Formatting Data** – Modify text, numbers, or dates for better readability.
   2. **Filtering Data** – Select specific data from a list (e.g., search results).
   3. **Sorting Data** – Arrange lists in ascending or descending order.
   4. **Currency & Number Formatting** – Display numbers in a currency format or with specific decimal places.
   5. **Custom Transformations** – Create custom filters for specific data manipulations.

Code : -

DIRECTORY STRUCTURE


```
EXPLORER ...
> OPEN EDITORS
v EXP8
  v Code
    v templates
        binding.html
        book-search.html
        login.html
      app.js
      index.html
```

Index.html

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
  <title>AngularJS Experiment</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/ang
ularjs/1.8.2/angular.min.js"></script>
  <script
src="https://ajax.googleapis.com/ajax/libs/ang
ularjs/1.8.2/angular-route.js"></script>
  <script src="app.js"></script>
</head>
<body>
  <h1>AngularJS Experiment</h1>
  <!-- Navigation Links -->
  <ul>
    <li><a href="#!/binding">One & Two Way
Binding</a></li>
    <li><a href="#!/login">Login
Authentication</a></li>
    <li><a href="#!/book-search">Book
Search</a></li>
  </ul>

  <!-- Content will load here based on route -->
  <div ng-view></div>
</body>
</html>
```

App.js

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

// Configure Routes
app.config(function($routeProvider) {
  $routeProvider
    .when('/binding', {
      templateUrl: 'templates/binding.html',
      controller: 'BindingController'
    })
    .when('/login', {
      templateUrl: 'templates/login.html',
      controller: 'LoginController'
    })
    .when('/book-search', {
      templateUrl: 'templates/book-search.html',
      controller: 'BookController'
    })
    .otherwise({
      redirectTo: '/binding'
    });
});

// Controllers
app.controller('BindingController',
function($scope) {
  $scope.message = "This is One-Way Data
Binding!";
  $scope.name = "John";
});

app.controller('LoginController',
function($scope) {
  $scope.username = '';
  $scope.password = '';
  $scope.message = '';

  $scope.login = function() {
    if ($scope.username === 'admin' &&
$scope.password === '1234') {
      $scope.message = 'Login Successful!';
    } else {
      $scope.message = 'Invalid Credentials!';
    }
  };
});

app.controller('BookController', function($scope)
{
  $scope.books = [
    { title: 'Book A', author: 'Author 1', genre:
'Fiction' },
    { title: 'Book B', author: 'Author 2', genre:
'Non-fiction' },
    { title: 'Book C', author: 'Author 3', genre:
'Science' }
  ];
  $scope.search = '';
});
```

Book-search.html

```
<h2>Book Search</h2>
<p>Search by Title, Author, or Genre:</p>
<input type="text" ng-model="search" placeholder="Search for books...">

<ul>
  <li ng-repeat="book in books | filter:search">
    <strong>{{ book.title }}</strong> by {{ book.author }} ({{ book.genre }})
  </li>
</ul>
```

binding.html

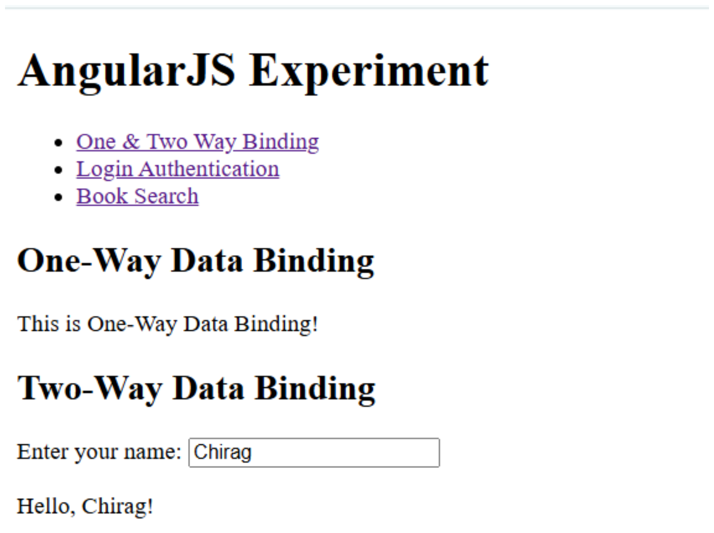
```
<h2>One-Way Data Binding</h2>

<p>{{ message }}</p>


<h2>Two-Way Data Binding</h2>

<p>Enter your name: <input type="text" ng-model="name"></p>

<p>Hello, {{ name }}!</p>
```

Login.html

```
<h2>Login Page</h2>

<form ng-submit="login()">

  <label>Username:</label>

  <input type="text" ng-model="username" required><br><br>


  <label>Password:</label>
  <input type="password" ng-model="password" required><br><br>


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


<p>{{ message }}</p>
```

OUTPUT: -

- One-way and Two way Binding



AngularJS Experiment

- One & Two Way Binding
- Login Authentication
- Book Search

One-Way Data Binding

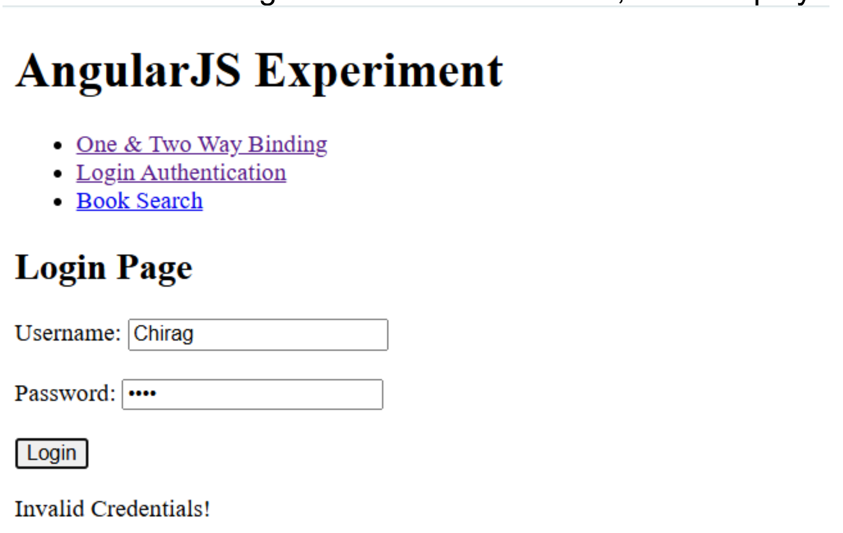
This is One-Way Data Binding!

Two-Way Data Binding

Enter your name: Chirag

Hello, Chirag!

- When user logins with incorrect details, it will display as “Invalid Credentials”



AngularJS Experiment

- One & Two Way Binding
- Login Authentication
- Book Search

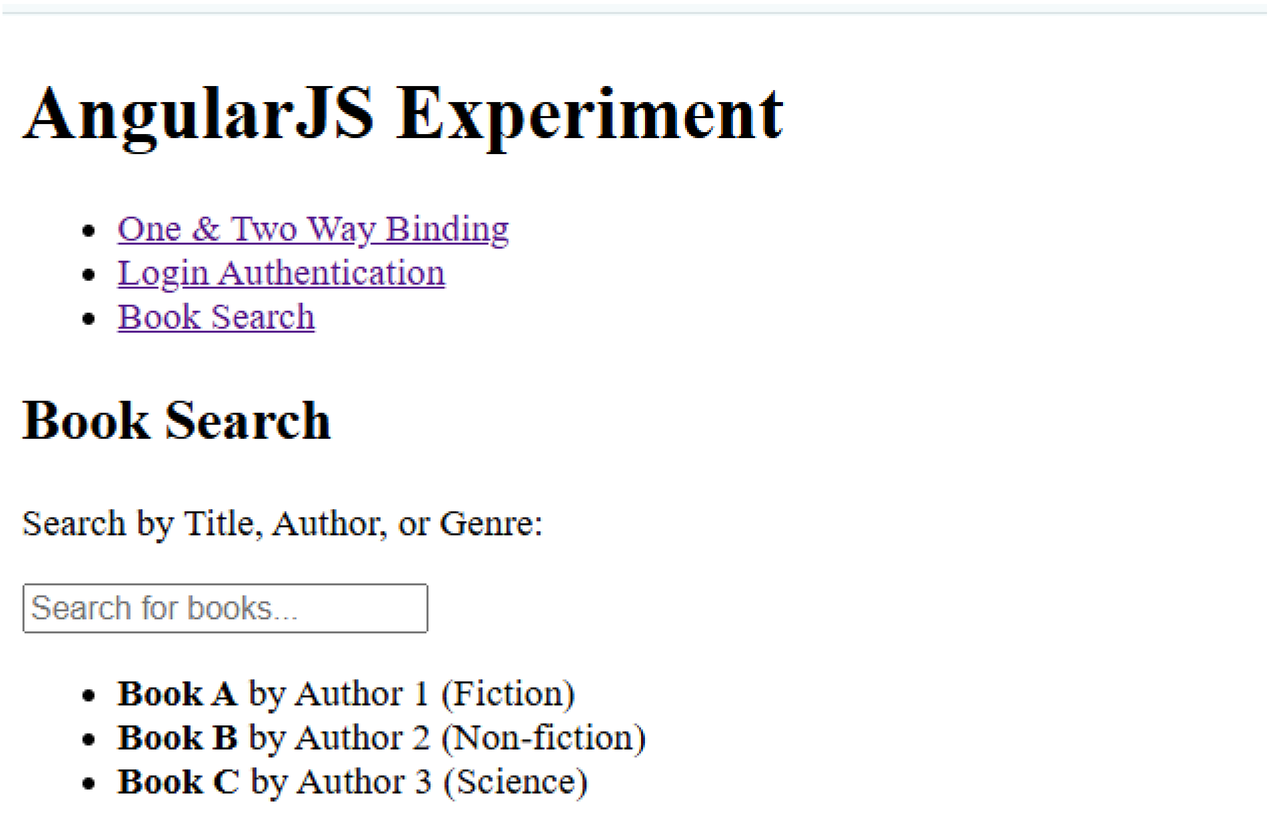
Login Page

Username: Chirag

Password: ••••

Invalid Credentials!

- After successful authentication, main page will be displayed



AngularJS Experiment

- One & Two Way Binding
- Login Authentication
- Book Search

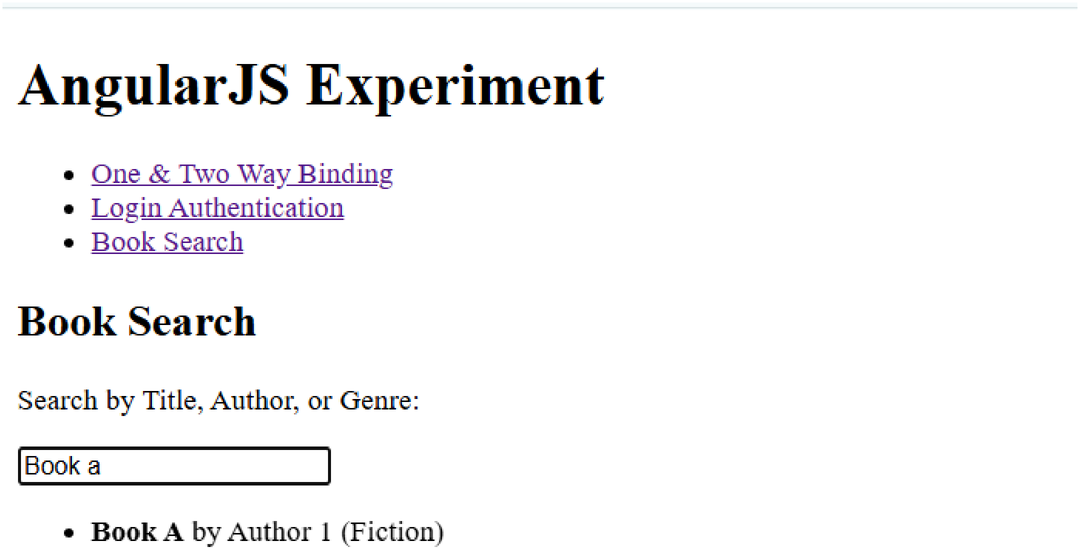
Book Search

Search by Title, Author, or Genre:

Search for books...

- **Book A** by Author 1 (Fiction)
- **Book B** by Author 2 (Non-fiction)
- **Book C** by Author 3 (Science)

- User can search by title, author and genre



AngularJS Experiment

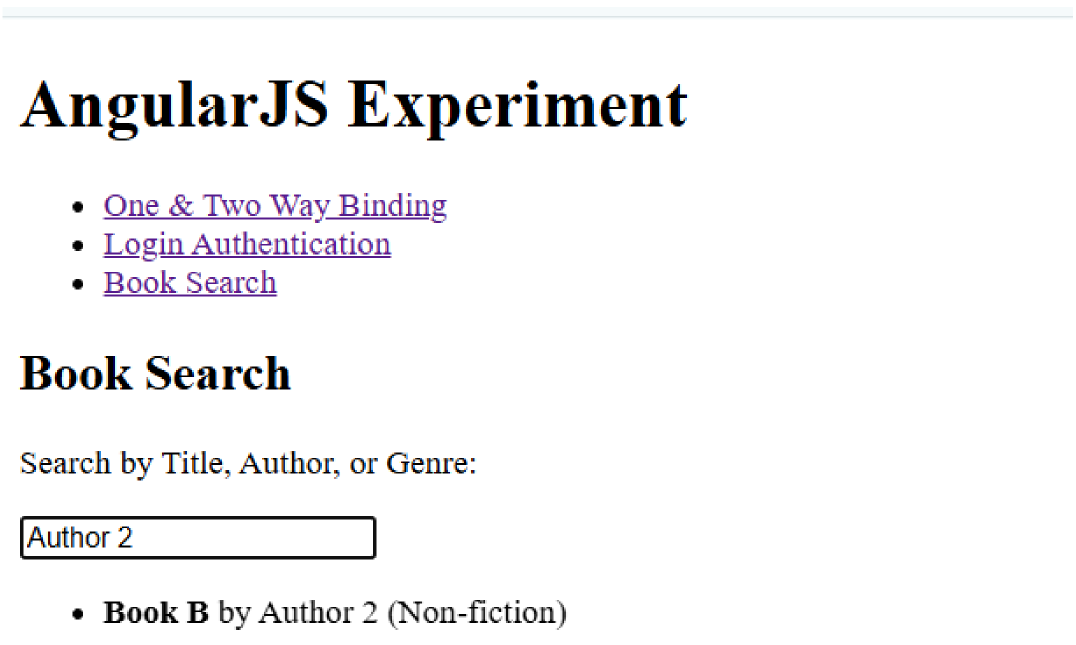
- One & Two Way Binding
- Login Authentication
- Book Search

Book Search

Search by Title, Author, or Genre:

Book a

- **Book A** by Author 1 (Fiction)



AngularJS Experiment

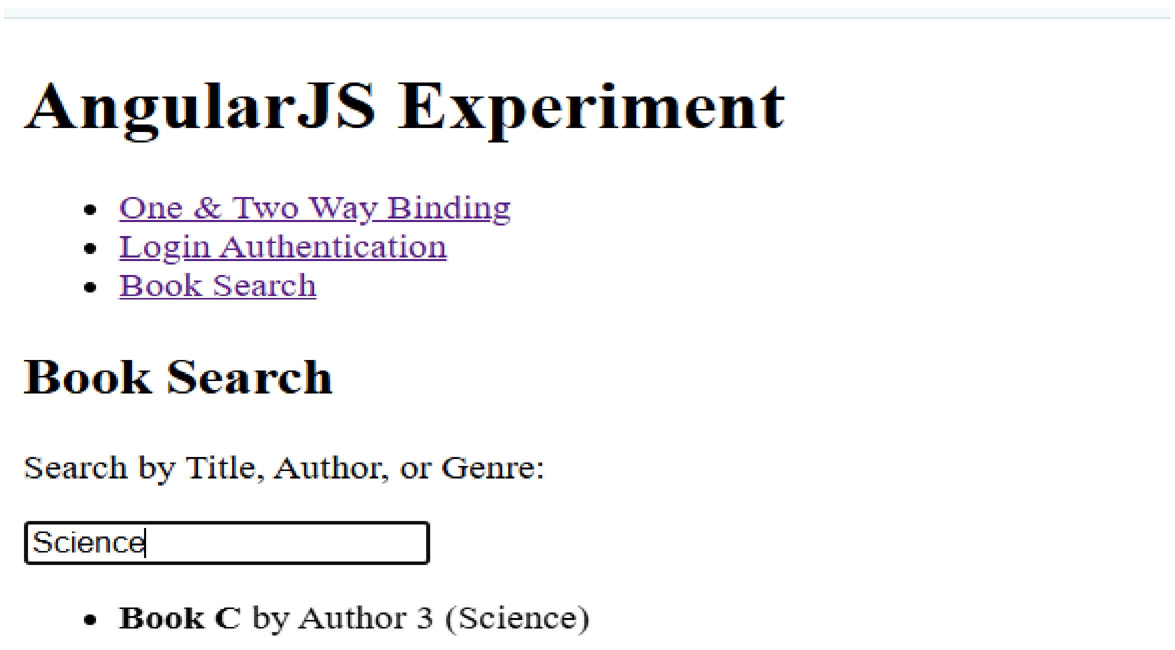
- One & Two Way Binding
- Login Authentication
- Book Search

Book Search

Search by Title, Author, or Genre:

Author 2

- **Book B** by Author 2 (Non-fiction)



AngularJS Experiment

- One & Two Way Binding
- Login Authentication
- Book Search

Book Search

Search by Title, Author, or Genre:

Science

- **Book C** by Author 3 (Science)

Conclusion: -

This practical explores key AngularJS concepts, including data binding, authentication, custom filters, and services. It demonstrates one-way and two-way data binding, showcasing how data flows between the model and view. A basic authentication system is implemented using AngularJS controllers, modules, and form directives. A custom filter (bookFilter) is created for searching books by title, author, or genre, while a modular authentication service ensures reusability and maintainability. These implementations highlight AngularJS's capabilities in building dynamic and interactive web applications.