EXPERIMENT NO. 8 : AngularJS

Name of Student	**Komal Milind Deolekar**
Class Roll No	**14**
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.

Github Link :

https://github.com/KomalDeolekar0607/Webx_Lab/tree/main/Webx_Lab_Exp_8

Theory:

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

In AngularJS, **directives** are special tokens in the markup that tell the library to do something to a DOM element (like hide it, repeat it, bind it, etc.).

They extend the HTML with **custom behavior** and can be either built-in or user-defined.

Commonly used AngularJS directives:

- ng-app – Initializes an AngularJS application.
- ng-model – Binds HTML input elements to application data.
- ng-bind – Binds data to HTML elements.
- ng-repeat – Loops through data (like arrays) and repeats elements.
- ng-if – Conditionally displays elements.
- ng-show / ng-hide – Shows or hides elements based on expressions.
- ng-submit – Binds a form submit action to a function

2. What is data binding in AngularJS?

Data binding in AngularJS is the synchronization between the model (JavaScript code) and the view (HTML).

AngularJS supports:

- **One-way binding**: From model to view ({{expression}}).
- **Two-way binding**: Between model and view using ng-model, where changes in the input reflect in the model and vice versa.

This eliminates the need for writing extra JavaScript code to update the DOM manually.

Example of One-Way Data Binding:

```
<!DOCTYPE html>
<html ng-app="app">
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="MainCtrl">

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

</body>
```

```
<script>
  angular.module('app', [])
    .controller('MainCtrl', function($scope) {
      $scope.message = "This is One-Way Binding";
    });
</script>
</html>
```

Example of Two-Way Data Binding:

```
<div ng-app="myApp" ng-controller="myCtrl">
  <input type="text" ng-model="name">
  <p>Hello {{name}}</p>
</div>

<script>
  angular.module('myApp', []).controller('myCtrl', function($scope) {
    $scope.name = "Komal";
  });
</script>
```

3. How is form validation done in angularJS

AngularJS provides **built-in validation** features to validate forms using:

- ng-model for two-way binding.
- Built-in directives like:
  - required – field must be filled
  - ng-minlength / ng-maxlength – input length
  - ng-pattern – regex pattern
  - type="email" – checks for valid email
- Form status properties like:
  - $valid
  - $invalid
  - $dirty
  - $pristine
  - $touched

You can use these in AngularJS to conditionally show error messages and prevent form submission.

Example:

```
<form name="myForm" novalidate>
  <label>Name:</label>
  <input type="text" name="username" ng-model="username" required>
  <span ng-show="myForm.username.$touched && myForm.username.$invalid">
    Name is required.
  </span>
  <br><br>
  <label>Email:</label>
  <input type="email" name="email" ng-model="email" required>
  <span ng-show="myForm.email.$touched && myForm.email.$invalid">
    Enter a valid email.
  </span>
  <br><br>
  <button ng-disabled="myForm.$invalid">Submit</button>
</form>
```

- myForm.username.$touched: Checks if the field has been interacted with.
- myForm.username.$invalid: Checks if it failed validation.
- ng-disabled="myForm.$invalid": Disables submit button if form is invalid.

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

AngularJS **controllers** are JavaScript functions used to build the business logic of an application.

They are responsible for:

- Initializing application data.
- Defining functions to be used in views.
- Interacting with services and models.
- Managing scope variables ($scope), which are then reflected in the view.

Example:

```
app.controller('MainController', function($scope) {
 $scope.message = "Hello from controller!";
});
```

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

AngularJS filters are used to format or transform data displayed in the view.

They can be used in templates with the pipe symbol |.

Common filters:

- uppercase, lowercase
- currency
- date
- filter – for custom search
- limitTo – to limit number of items
- orderBy – to sort data

Example:

```
<p>{{ name | uppercase }}</p>
```

```
<p ng-repeat="item in items | filter:searchText">{{ item }}</p>
```

Filters can also be **custom-defined** for specific formatting or search logic.

Code :

index.html

```
<!DOCTYPE html>

<html lang="en" ng-app="bookApp">

<head>

  <meta charset="UTF-8">

  <title>Stylish AngularJS App</title>

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

  <link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;600&display=swap" rel="stylesheet">

  <style>

    * {

      margin: 0;

      padding: 0;

      box-sizing: border-box;

      font-family: 'Inter', sans-serif;

    }

    body {

      background: linear-gradient(to right, #e0eafc, #cfdef3);

      min-height: 100vh;

      display: flex;

      align-items: center;

      justify-content: center;

    }

    .container {
```

```
  background: #fff;
  padding: 30px;
  border-radius: 16px;
  box-shadow: 0 10px 30px rgba(0, 0, 0, 0.1);
  width: 90%;
  max-width: 650px;
}
h2 {
  text-align: center;
  margin-bottom: 20px;
  color: #333;
}
input, button {
  width: 100%;
  padding: 12px;
  margin: 10px 0;
  border: 1px solid #ccc;
  border-radius: 10px;
  font-size: 16px;
}
button {
  background-color: #4e9af1;
  color: white;
  font-weight: bold;
  border: none;
  transition: background 0.3s ease;
}
button:hover {
  background-color: #3a84d8;
}
.message {
  text-align: center;
  margin-top: 10px;
}

.success { color: green; }
.error { color: red; }

.card-grid {
  display: flex;
  gap: 20px;
  flex-wrap: wrap;
  justify-content: center;
  margin-top: 20px;
}

.card {
  flex: 1 1 150px;
  background: #f3f9ff;
  padding: 20px;
  border-radius: 12px;
  text-align: center;
  cursor: pointer;
  box-shadow: 0 4px 8px rgba(0,0,0,0.1);
  transition: transform 0.2s ease, background 0.2s ease;
}

.card:hover {
  transform: translateY(-5px);
  background: #d8edff;
}

.section {
  margin-top: 30px;
}

.section h3 {
  text-align: center;
  margin-bottom: 15px;
  color: #333;
}
```

```
ul {
  list-style: none;
  padding: 0;
}

li {
  padding: 10px;
  background: #f7f7f7;
  margin-bottom: 8px;
  border-radius: 8px;
}

.logout-btn {
  margin-top: 20px;
  background: #ff5c5c;
}

.logout-btn:hover {
  background: #e04b4b;
}

@media(max-width: 500px) {
  .card-grid {
    flex-direction: column;
```

```
      }
    }
  </style>
</head>
<body ng-controller="MainController">

  <div class="container" ng-if="!isLoggedIn">
    <h2>🔐 Login to Continue</h2>
    <input type="text" ng-model="user.username" placeholder="Username">
    <input type="password" ng-model="user.password" placeholder="Password">
    <button ng-click="login()">Login</button>
    <div class="message">
      <p class="success" ng-if="loginSuccess">{{ loginSuccess }}</p>
      <p class="error" ng-if="loginError">{{ loginError }}</p>
    </div>
  </div>

  <div class="container" ng-if="isLoggedIn">
    <h2>🎉 Welcome, {{ user.username }}</h2>

    <div class="card-grid">
      <div class="card" ng-click="showSection('oneWay')">🔁 One-Way Binding</div>
      <div class="card" ng-click="showSection('twoWay')">🔄 Two-Way Binding</div>
      <div class="card" ng-click="showSection('books')">📚 Book Search</div>
```

```
</div>

<div class="section" ng-if="activeSection === 'oneWay'">
  <h3>🔁 One-Way Binding</h3>
  <p>Message: <strong>{{ oneWayMessage }}</strong></p>
</div>

<div class="section" ng-if="activeSection === 'twoWay'">
  <h3>🔄 Two-Way Binding</h3>
  <input type="text" ng-model="twoWayMessage" placeholder="Type something...">
  <p>You typed: <strong>{{ twoWayMessage }}</strong></p>
</div>

<div class="section" ng-if="activeSection === 'books'">
  <h3>📚 Book Search</h3>
  <input type="text" ng-model="searchText" placeholder="Search title, author, or genre">
  <ul>
    <li ng-repeat="book in books | bookFilter:searchText">
      <strong>{{ book.title }}</strong> by {{ book.author }} ({{ book.genre }})
    </li>
  </ul>
</div>

<button class="logout-btn" ng-click="logout()">Logout</button>
```

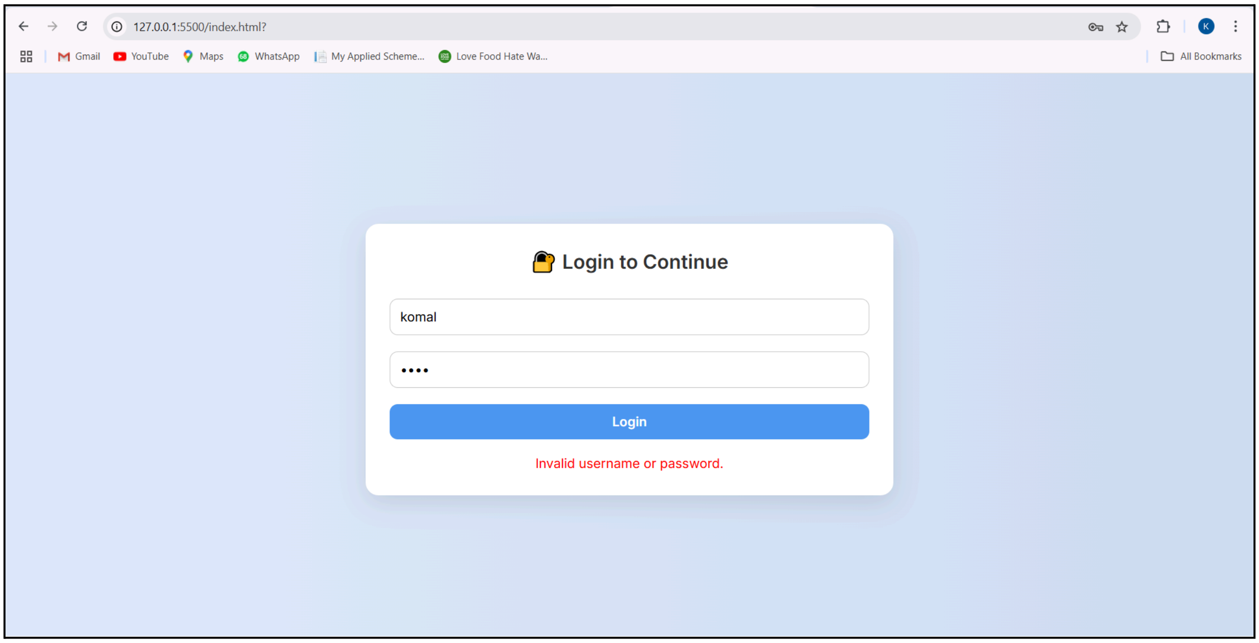
```
</div>


<script>
 var app = angular.module('bookApp', []);


 app.factory('AuthService', function () {
  var validUser = { username: 'admin', password: '1234' };
  return {
   login: function (user) {
    return user.username === validUser.username && user.password === validUser.password;
   }
  };
 });


 app.filter('bookFilter', function () {
  return function (books, searchText) {
   if (!searchText) return books;
   var filtered = [];
   searchText = searchText.toLowerCase();
   angular.forEach(books, function (book) {
    if (
     book.title.toLowerCase().includes(searchText) ||
     book.author.toLowerCase().includes(searchText) ||
     book.genre.toLowerCase().includes(searchText)
```

```
      ) {
        filtered.push(book);
      }
    });
    return filtered;
  };
});


app.controller('MainController', function ($scope, AuthService) {
  $scope.user = {};
  $scope.isLoggedIn = false;
  $scope.activeSection = null;


  $scope.login = function () {
    if (AuthService.login($scope.user)) {
      $scope.isLoggedIn = true;
      $scope.loginSuccess = "Login successful!";
      $scope.loginError = "";
    } else {
      $scope.loginError = "Invalid username or password.";
      $scope.loginSuccess = "";
    }
  };
```

```
      $scope.logout = function () {
        $scope.isLoggedIn = false;
        $scope.user = {};
        $scope.activeSection = null;
      };

      $scope.showSection = function (section) {
        $scope.activeSection = section;
      };

      $scope.oneWayMessage = "Welcome to One-Way Binding!";
      $scope.twoWayMessage = "";

      $scope.books = [
        { title: "The Hobbit", author: "J.R.R. Tolkien", genre: "Fantasy" },
        { title: "1984", author: "George Orwell", genre: "Dystopian" },
        { title: "The Alchemist", author: "Paulo Coelho", genre: "Fiction" },
        { title: "Sapiens", author: "Yuval Noah Harari", genre: "History" }
      ];
    });
  </script>

</body>
</html>
```

Output :

Login



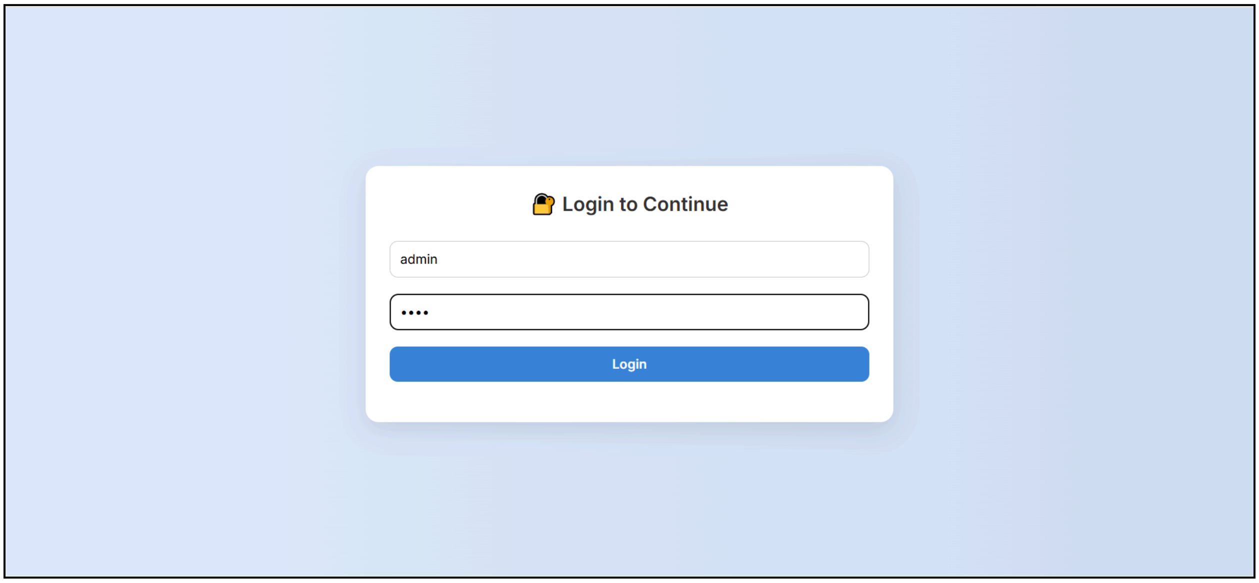
127.0.0.1:5500/index.html?

🔐 Login to Continue

komal

••••

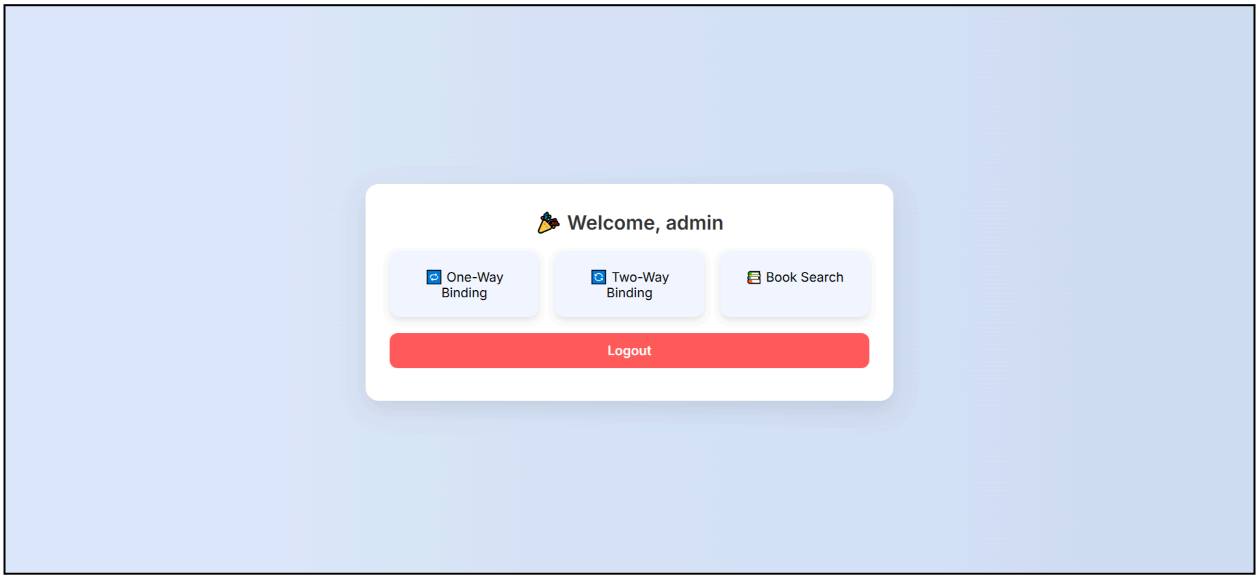
Invalid username or password.



🔐 Login to Continue

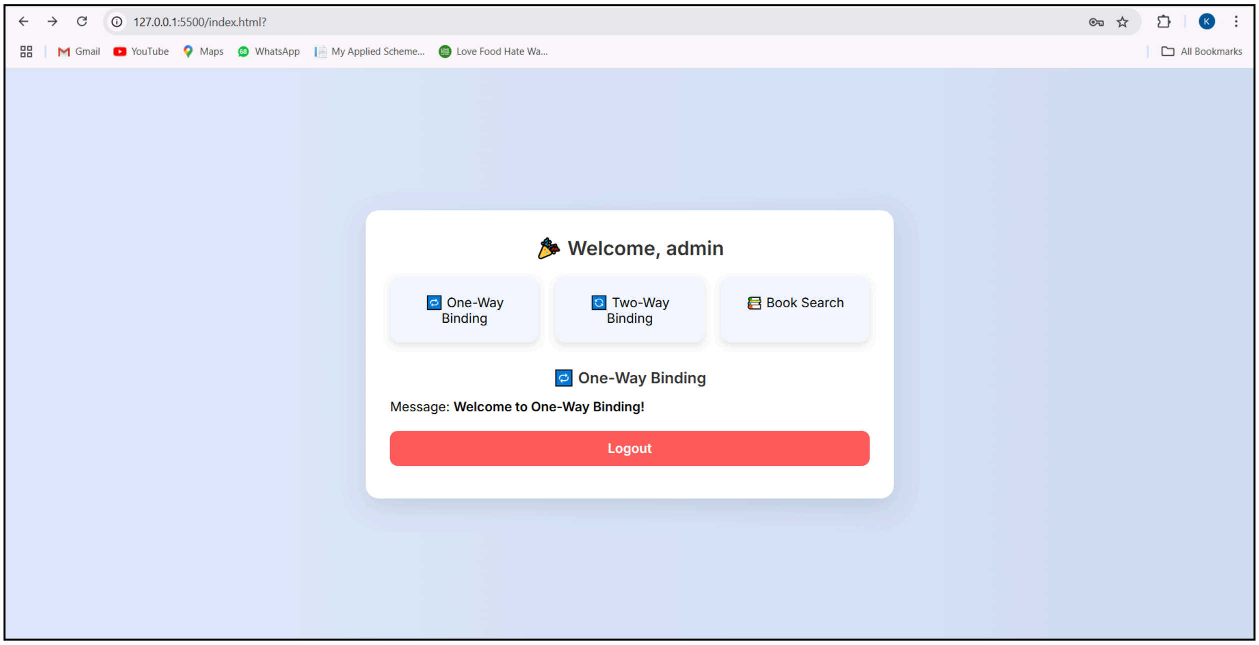
admin

••••



🎉 Welcome, admin

One-Way Binding

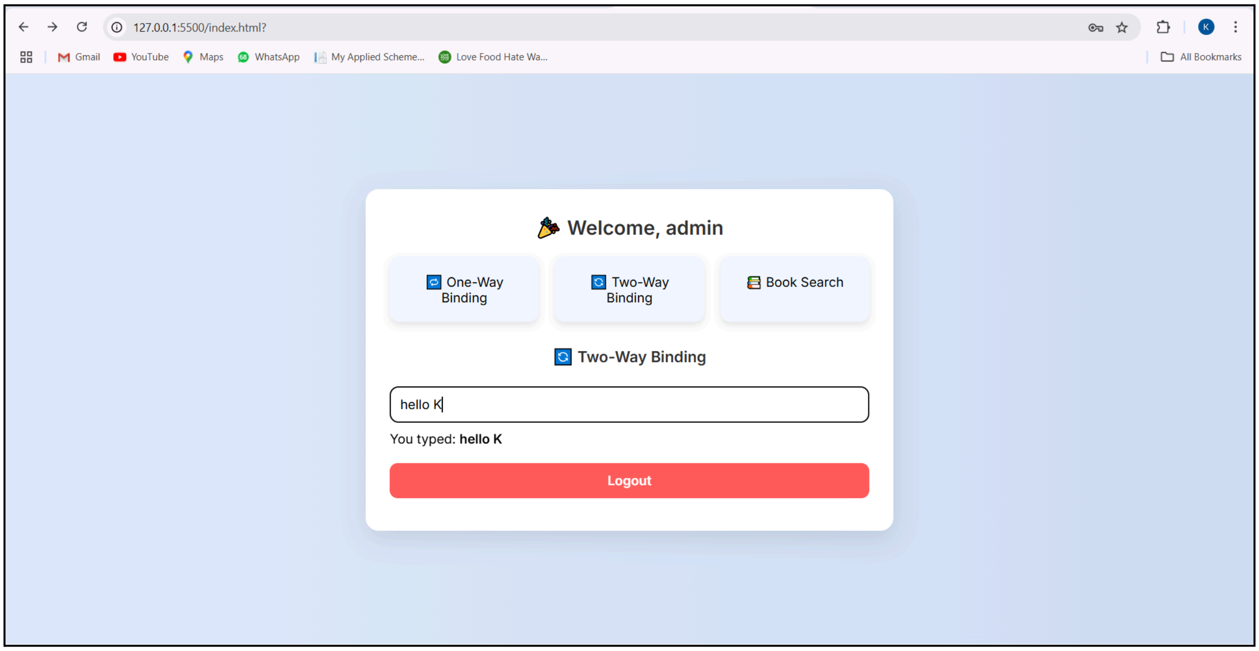


🎉 Welcome, admin

🔁 One-Way Binding

Message: **Welcome to One-Way Binding!**

Two-Way Binding



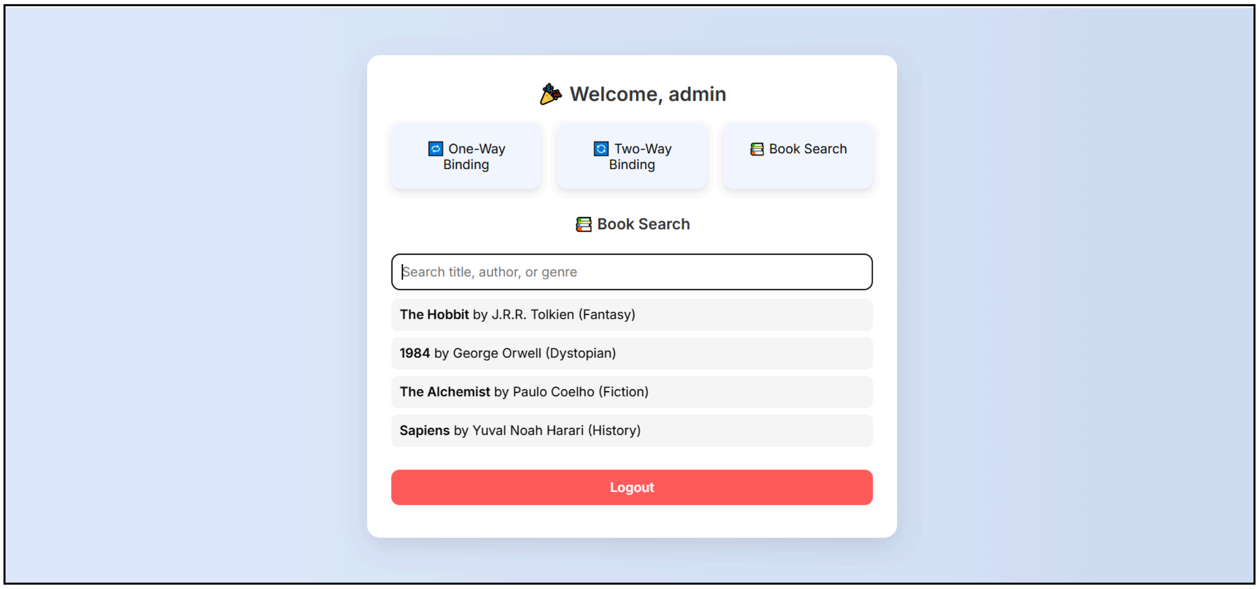
Welcome, admin

Two-Way Binding

hello K

You typed: **hello K**

Book Search



Welcome, admin

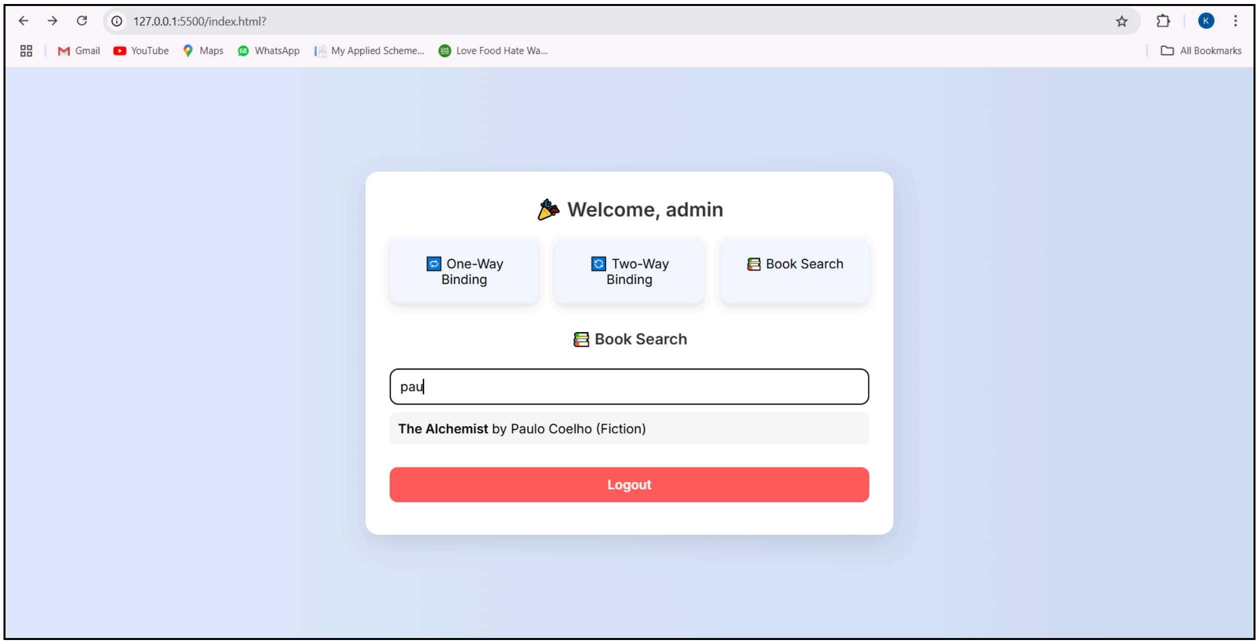
Book Search

The Hobbit by J.R.R. Tolkien (Fantasy)

1984 by George Orwell (Dystopian)

The Alchemist by Paulo Coelho (Fiction)

Sapiens by Yuval Noah Harari (History)

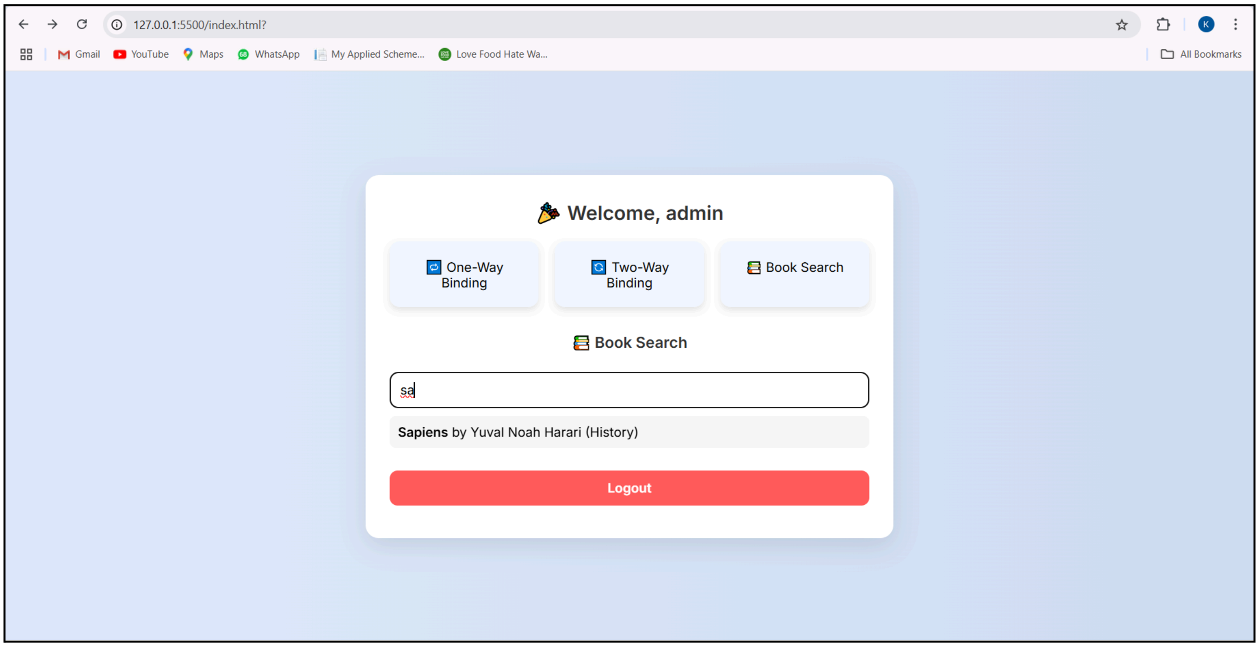


🎉 Welcome, admin

📚 Book Search

pau

The Alchemist by Paulo Coelho (Fiction)



🎉 Welcome, admin

📚 Book Search

sa

Sapiens by Yuval Noah Harari (History)

Conclusion :

AngularJS is a powerful JavaScript framework that simplifies the process of building dynamic and interactive web applications. It provides a robust structure using **modules**, **controllers**, **services**, and **directives**, allowing developers to organize their code efficiently.

Key features such as **data binding** (one-way and two-way), **form validation**, **custom filters**, and **dependency injection** make development more efficient and maintainable. The use of **directives** extends HTML capabilities, enabling clean and declarative UIs. With support for **client-side routing** and reusable components, AngularJS lays a solid foundation for scalable single-page applications (SPAs).

Overall, AngularJS is ideal for developing feature-rich applications with minimal effort, while maintaining code readability and structure.