Name: Diksha Utekar **D15A/63**

EXPERIMENT NO. 8 - AngularJS

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 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>
```

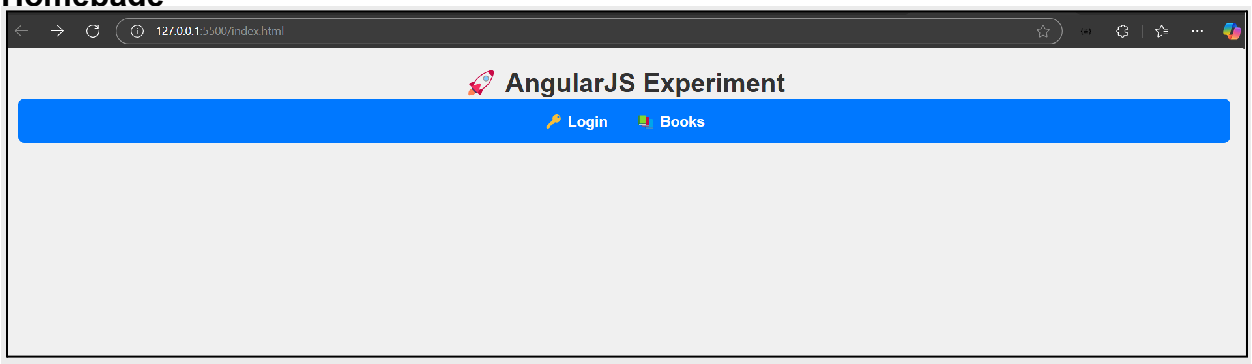
```
</form>
```

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`

OUTPUT

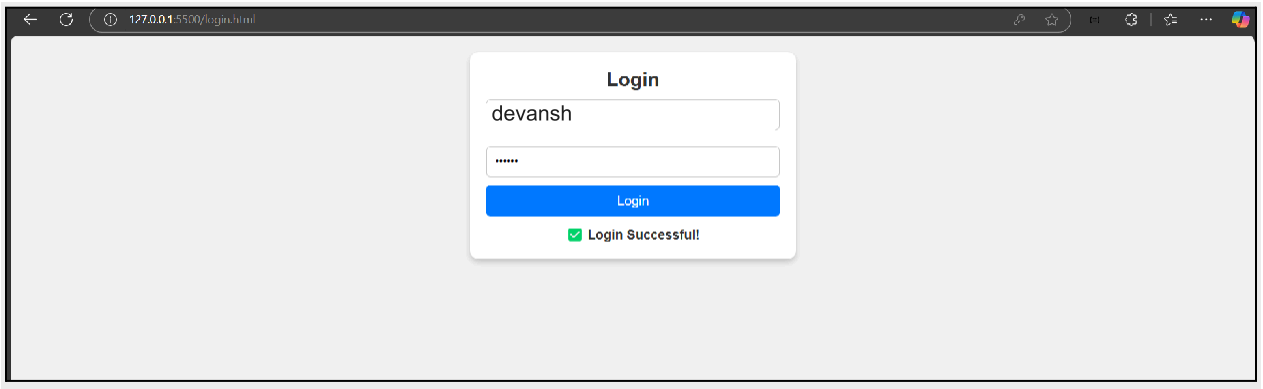
a) **Homepage**



This screenshot displays the homepage of the AngularJS application. It provides navigation links to different sections such as the login page and the book search

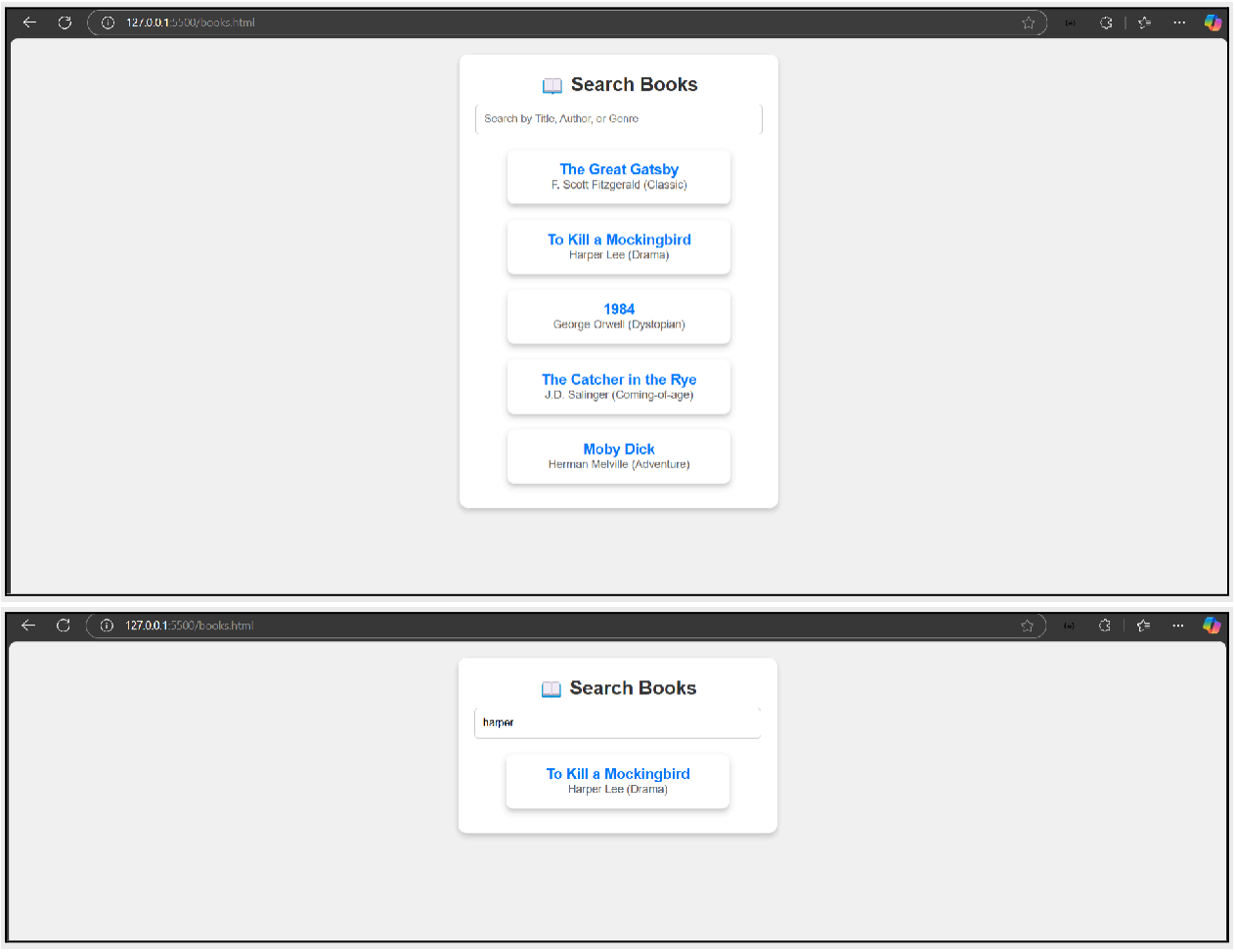
page. The page demonstrates AngularJS directives like `ng-app` for module initialization and `ng-controller` for managing application logic.

b) Login Page



This screenshot shows the login form where users enter their username and password. It demonstrates AngularJS form directives (`ng-model` for data binding, `ng-submit` for form submission) and a custom authentication service to verify credentials. Upon submitting the form, the system checks for hardcoded credentials. If correct, a success message appears; otherwise, an error message is displayed.

c) Book Search Page



This screenshot displays search functionality for books based on title, author, or genre.It demonstrates the implementation of a custom AngularJS filter (`bookFilter`) that dynamically filters book data as the user types in the search field. `ng-repeat` is used to loop through and display book data dynamically.

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.