

**EXPERIMENT NO. 8**

<b>Name of Student</b>	<b><u>Prajyot Shinde</u></b>
<b>Class Roll No</b>	<b><u>55</u></b>
<b>D.O.P.</b>	<b><u>01-04-25</u></b>
<b>D.O.S.</b>	<b><u>08-04-25</u></b>
<b>Sign and Grade</b>	

**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.

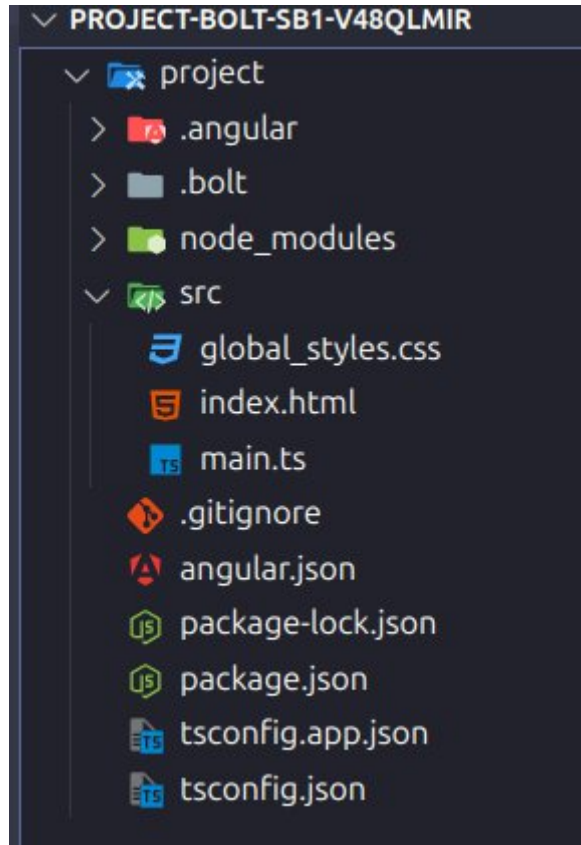
#### Key Uses of AngularJS Filters

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.

**GitHub Link :-** [https://github.com/tejasgunjal021/WEBX\\_EXP8](https://github.com/tejasgunjal021/WEBX_EXP8)

**Code :-**

## DIRECTORY STRUCTURE



### main.ts

```
import { Component, NgModule } from
'@angular/core';
import { bootstrapApplication } from
'@angular/platform-browser';
import { CommonModule } from
'@angular/common';
import { FormsModule } from
'@angular/forms';
import { BrowserModule } from
'@angular/platform-browser';
```

// Authentication Service

```
class AuthService {
  private validCredentials = {
    username: 'admin',
    password: 'password123'
  };
}
```

isAuthenticated = false;

```
login(username: string, password: string):
boolean {
```

```
  this.isAuthenticated = username ===
  this.validCredentials.username &&
  password ===
  this.validCredentials.password;
  return this.isAuthenticated;
}
```

```
logout(): void {
  this.isAuthenticated = false;
}
}
```

// Book Interface

```
interface Book {
  id: number;
  title: string;
  author: string;
  genre: string;
  language: string;
  price: number;
  rating: number;
  reviews: number;
```

```

description: string;
coverUrl: string;
}

// Book Filter Pipe
import { Pipe, PipeTransform } from
 '@angular/core';

@Pipe({
  name: 'bookFilter',
  standalone: true
})
class BookFilterPipe implements
 PipeTransform {
  transform(books: Book[], searchText:
 string, searchBy: string): Book[] {
    if (!books || !searchText) return books;

    searchText = searchText.toLowerCase();
    return books.filter(book => {
      switch(searchBy) {
        case 'title':
          return
book.title.toLowerCase().includes(searchTe
xt);
        case 'author':
          return
book.author.toLowerCase().includes(search
Text);
        case 'genre':
          return
book.genre.toLowerCase().includes(searchT
ext);
        default:
          return
book.title.toLowerCase().includes(searchTe
xt) ||

book.author.toLowerCase().includes(search
Text) ||

book.genre.toLowerCase().includes(searchT
ext);
      }
    });
  }
}

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [CommonModule, FormsModule,
 BookFilterPipe],
  template: `

```

```

<div class="nav-bar">
  <h1><img alt="PustakShala logo" data-bbox="605 105 625 120"/> PustakShala</h1>
  <a href="#" (click)="logout()"
*ngIf="isAuthenticated">Logout</a>
</div>

<div class="container">
  <!-- Login Form -->
  <div *ngIf="!isAuthenticated"
class="card">
    <h2>Welcome to PustakShala</h2>
    <p style="color: var(--text-
light);">Please login to explore our collection
of books</p>
    <form (ngSubmit)="onLogin()">
      <div class="form-group">
        <label>Username:</label>
        <input type="text" class="form-
control" [(ngModel)]="username"
name="username" placeholder="Enter your
username">
      </div>
      <div class="form-group">
        <label>Password:</label>
        <input type="password" class="form-
control" [(ngModel)]="password"
name="password" placeholder="Enter your
password">
      </div>
      <button type="submit" class="btn btn-
primary">Login to PustakShala</button>
      <div *ngIf="loginError" class="error-
message">
        <span>X</span> Invalid credentials.
Please try again.
      </div>
    </form>
  </div>

  <!-- Books Section -->
  <div *ngIf="isAuthenticated">
    <div class="card">
      <h2>Discover Books</h2>
      <div class="search-bar">
        <input
          type="text"
          class="form-control"
          [(ngModel)]="searchText"
          placeholder="Search for your next
favorite book...">
        <select class="form-control"
[(ngModel)]="searchBy">
          <option value="all">All
Categories</option>

```

```

      <option value="title">Title</option>
      <option
value="author">Author</option>
      <option
value="genre">Genre</option>
      </select>
    </div>

    <div class="books-grid">
      <div *ngFor="let book of books |
bookFilter:searchText:searchBy"
class="book-card card">
        <h3>{{ book.title }}</h3>
        <p>by {{ book.author }}</p>
        <p
class
s="desc
ription">{{
k.description }}</p>
        <div class="book-meta">
          <span
class="language-
tag">{{ book.language }}</span>
          <span
class="price">₹ {{ book.price }}</span>
          <span class="rating">
            ☆ {{ book.rating }}
          </span>
          <span>
            ({{ book.reviews }} reviews)
          </span>
        </div>
      </div>
    </div>
  </div>
</div>
,
})
export class App {
  username = "";
  password = "";
  loginError = false;
  isAuthenticated = false;
  searchText = "";
  searchBy = 'all';

  books: Book[] = [
    {
      id: 1,
      title: 'The White Tiger',
      author: 'Aravind Adiga',
      genre: 'Contemporary Fiction',
      language: 'English',
      price: 399,
      rating: 4.5,
      reviews: 2847,

```

```

      description: 'A darkly humorous
perspective of India\'s class struggle in a
globalized world.',
      coverUrl: 'white-tiger.jpg'
    },
    {
      id: 2,
      title: 'Train to Pakistan',
      author: 'Khushwant Singh',
      genre: 'Historical Fiction',
      language: 'English',
      price: 299,
      rating: 4.7,
      reviews: 1563,
      description: 'A historical novel recounting
the Partition of India in 1947.',
      coverUrl: 'train-to-pakistan.jpg'
    },
    {
      id: 3,
      title: 'The Guide',
      author: 'R.K. Narayan',
      genre: 'Classic',
      language: 'English',
      price: 250,
      rating: 4.6,
      reviews: 982,
      description: 'A spiritual journey through
the streets of Malgudi.',
      coverUrl: 'the-guide.jpg'
    },
    {
      id: 4,
      title: 'Midnight\'s Children',
      author: 'Salman Rushdie',
      genre: 'Magical Realism',
      language: 'English',
      price: 599,
      rating: 4.4,
      reviews: 3421,
      description: 'A magical realist chronicle
of India\'s journey from colonialism to
independence.',
      coverUrl: 'midnights-children.jpg'
    },
    {
      id: 5,
      title: 'The Immortals of Meluha',
      author: 'Amish Tripathi',
      genre: 'Mythology',
      language: 'English',
      price: 399,
      rating: 4.8,
      reviews: 5632,

```

```

    description: 'A reimagining of the Indian
god Shiva\'s journey from a tribal chief to a
divine being.',
    coverUrl: 'meluha.jpg'
  },
  {
    id: 6,
    title: 'God of Small Things',
    author: 'Arundhati Roy',
    genre: 'Literary Fiction',
    language: 'English',
    price: 450,
    rating: 4.6,
    reviews: 2156,
    description: 'A story of forbidden love
and social discrimination in Kerala.',
    coverUrl: 'small-things.jpg'
  }
];

```

```

constructor(private authService:
AuthService) {}

```

### Angular.json

```

{
  "$schema":
"./node_modules/@angular/cli/lib/config/sc
hema.json",
  "cli": {
    "analytics": "1e1de97b-a744-405a-8b5a-
0397bb3d01ce"
  },
  "newProjectRoot": "projects",
  "projects": {
    "demo": {
      "architect": {
        "build": {
          "builder":
"@angular/build:application",
          "configurations": {

```

```

onLogin() {
  this.isAuthenticated =
this.authService.login(this.username,
this.password);
  this.loginError = !this.isAuthenticated;
  if (this.isAuthenticated) {
    this.username = "";
    this.password = "";
  }
}

```

```

logout() {
  this.authService.logout();
  this.isAuthenticated = false;
}
}

```

```

bootstrapApplication(App, {
  providers: [AuthService]
});

```

```

"development": {
  "extractLicenses": false,
  "namedChunks": true,
  "optimization": false,
  "sourceMap": true
},
"production": {
  "aot": true,
  "extractLicenses": true,
  "namedChunks": false,
  "optimization": true,
  "outputHashing": "all",
  "sourceMap": false
},
"options": {


```

```

"assets": [],
"index": "src/index.html",
"browser": "src/main.ts",
"outputPath": "dist/demo",
"polyfills": ["zone.js"],
"scripts": [],
"styles": ["src/global_styles.css"],
"tsConfig": "tsconfig.app.json"
},
},
"serve": {
"builder": "@angular/build:dev-server",
"configurations": {
"development": {
"buildTarget":
"demo:build:development"
},
"production": {
"buildTarget":
"demo:build:production"
},
"defaultConfiguration": "development"
},
"prefix": "app",
"projectType": "application",
"root": "",
"schematics": {},
"sourceRoot": "src"
},
"version": 1
}

```

## OUTPUT: -


**PustakShala**

### Welcome to PustakShala

Please login to explore our collection of books

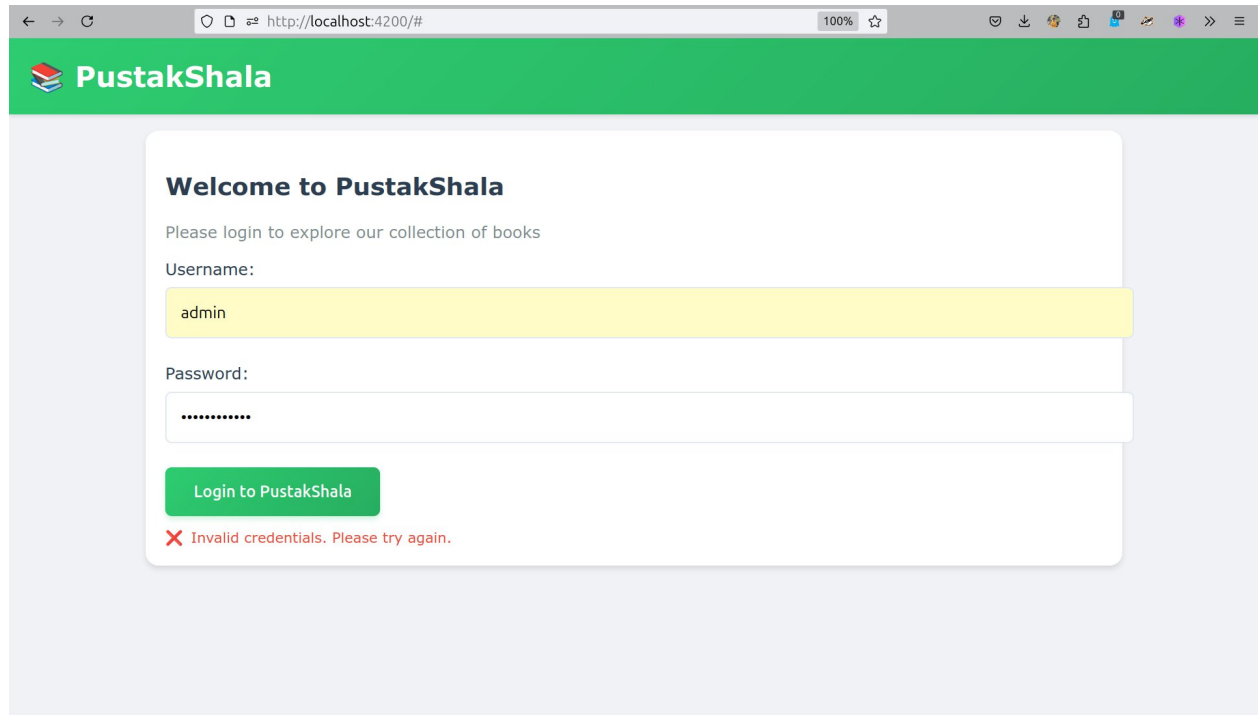
Username:

Password:

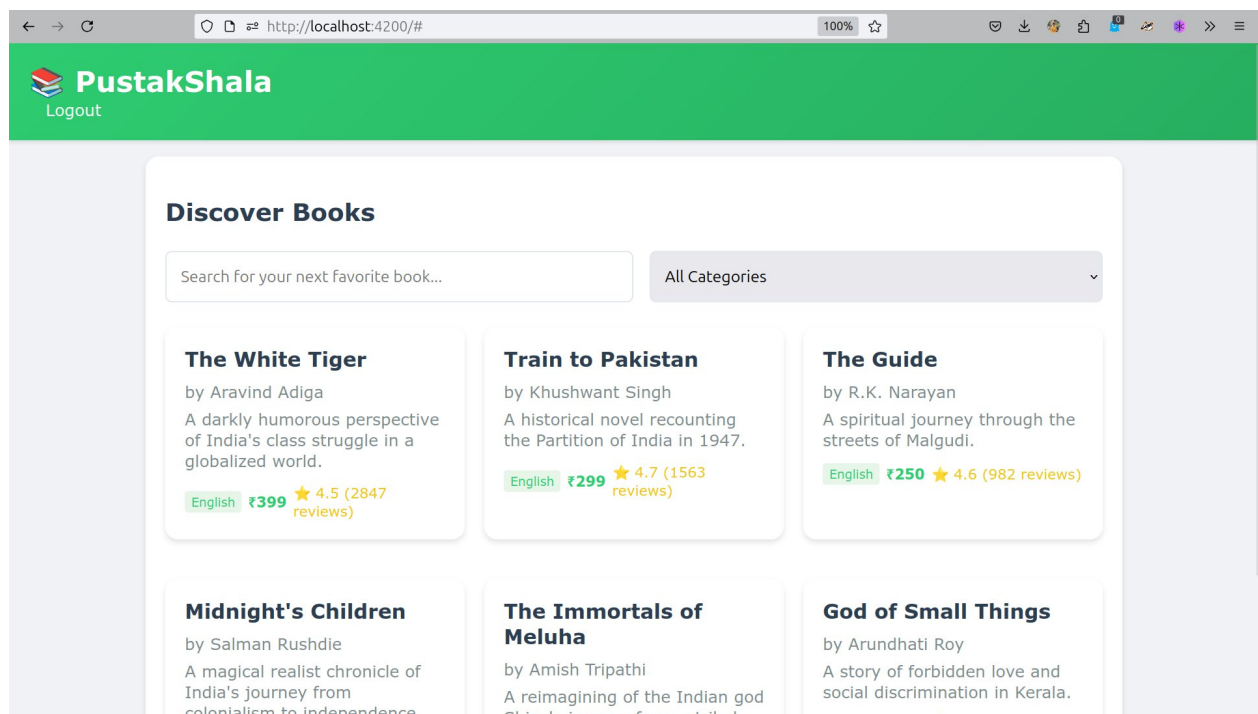
Login to PustakShala



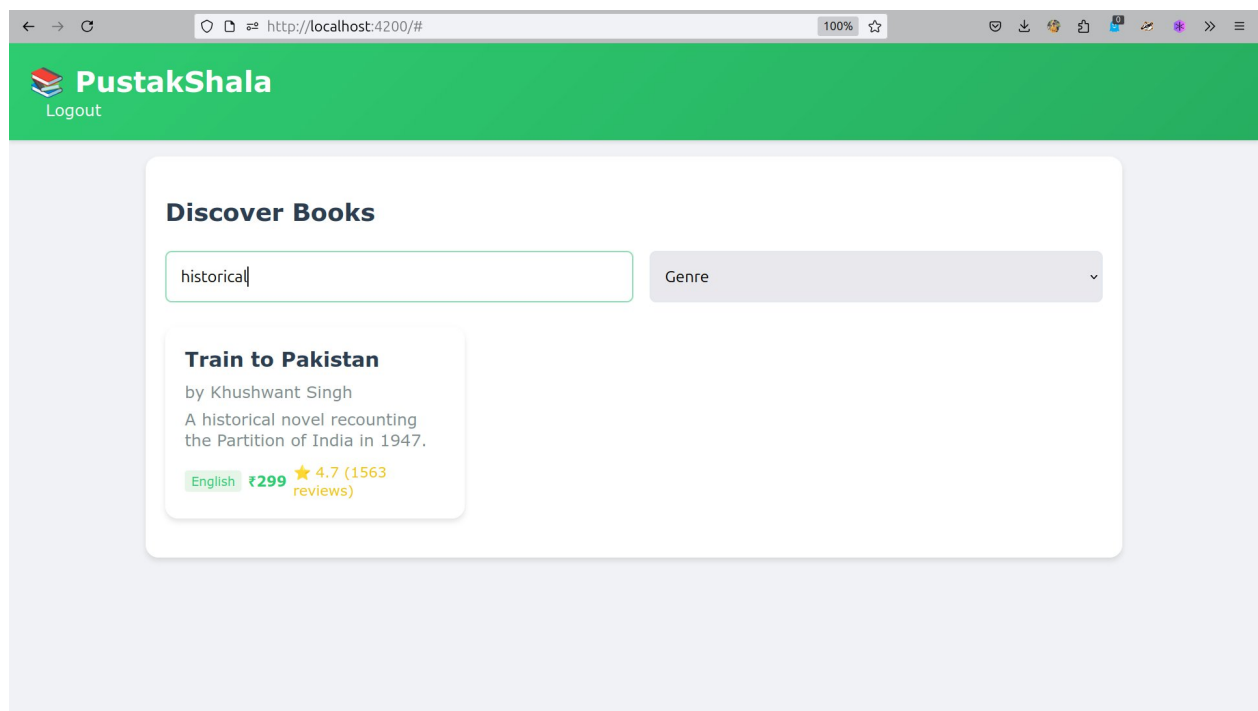
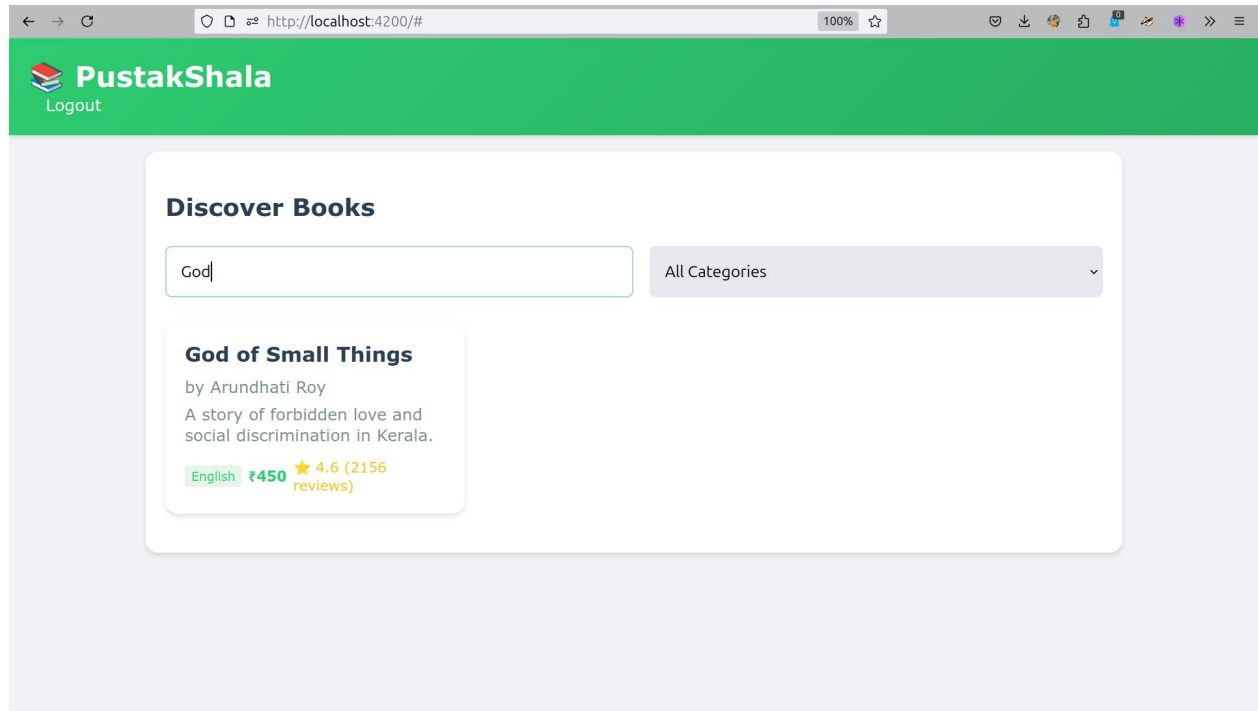
- When user logs in with incorrect details, it will display as “Invalid Credentials”

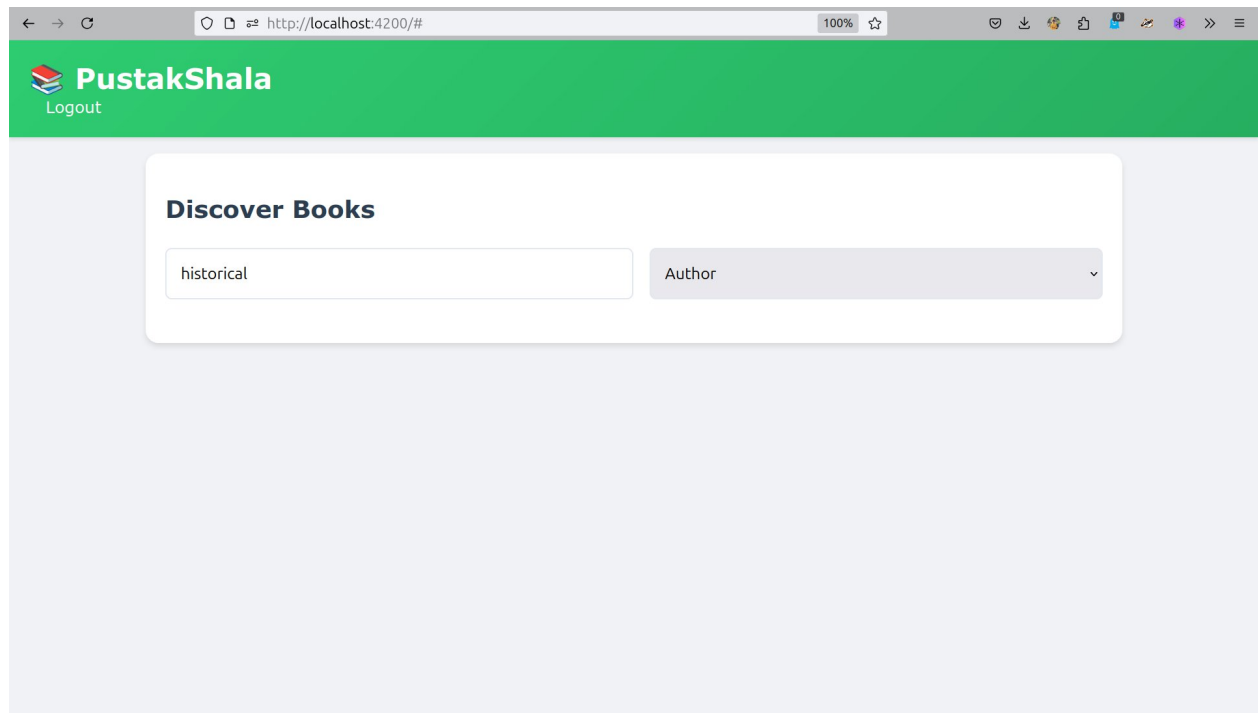


- After successful authentication, main page will be displayed



- User can search by title, author and genre



**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.