EXPERIMENT NO. 8

Name of Student	**Manorath Ital**
Class Roll No	**19**
D.O.P.	**01-04-25**
D.O.S.	**08-04-25**
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:

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 AngularJS include:

ng-app defines the root element of an AngularJS application.
ng-model binds an input field to a variable in the scope, enabling 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 and ng-hide show or hide elements based on a condition.
ng-click adds a click event listener to elements.

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.

There are two main types of data binding in AngularJS:

One-way data binding (Interpolation and Expressions) updates the view when the model changes but not vice versa. This is achieved using double curly braces like `{{ expression }}` or directives like ng-bind. For example:

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

Two-way data binding (ng-model) synchronizes data both ways — when the user updates the UI, the model updates, and vice versa. For example:

```
<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.

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.

AngularJS uses directives such as ng-required, ng-minlength, and ng-pattern to validate inputs. It supports real-time validation where errors appear as users type. AngularJS also uses built-in validation states like $valid, $invalid, $dirty, and $pristine to track the status of the form. Developers can define custom validation rules for specific needs. 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.

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.

Controllers in AngularJS are responsible for data binding by controlling how data is displayed in the view using the `$scope` object. They contain business logic to handle user actions and process data. Controllers also communicate with services to fetch or update data through APIs. Additionally, they manage user interactions such as button clicks and form submissions. By separating business logic from the view, controllers support clean and maintainable code architecture.

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.

Filters are commonly used to format text, numbers, or dates for better readability. They can filter data to show only relevant results (such as in search bars) or sort lists in ascending or descending order. AngularJS also provides built-in filters for currency and number formatting. Developers can create custom filters to implement specific data transformation logic. Filters enhance the user experience by delivering cleaner and more meaningful information in the UI.

GitHub Link : - https://github.com/MANO-RATH/WEBXEXP8.git

Input:
auth.service.ts

```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root',
})
export class AuthService {
  private isAuthenticated = false;

  login(username: string, password: string): boolean {
    // Implement your authentication logic here
    if (username === 'admin' && password === 'admin') {
      this.isAuthenticated = true;
      return true;
    }
    return false;
  }

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

  isLoggedIn(): boolean {
    return this.isAuthenticated;
  }
}
```

book.filter.ts

```
import { Pipe, PipeTransform } from '@angular/core';
import { Book } from './book.interface';

@Pipe({
  name: 'bookFilter',
})
export class BookFilterPipe implements PipeTransform {
  transform(books: Book[], searchText: string): Book[] {
    if (!books || !searchText) {
      return books;
    }
    return books.filter((book) =>
      book.title.toLowerCase().includes(searchText.toLowerCase())
    );
  }
}
```

book.interface.ts

```
export interface Book {
  title: string;
  author: string;
  genre: string;
}
```

config.json

```
{
  "apiUrl": "http://localhost:3000/api"
}
```

global_styles.css

```
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
}

nav {
  background-color: #333;
  color: #fff;
  padding: 1em;
}

nav a {
  color: #fff;
  margin-right: 1em;
  text-decoration: none;
}
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>AngularJS Project</title>
  <base href="/" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <link rel="stylesheet" href="global_styles.css" />
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

main.ts

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
```

```
import { AppModule } from './app/app.module';

platformBrowserDynamic()
  .bootstrapModule(AppModule)
  .catch((err) => console.error(err));
```

angular.json

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "projects": {
    "webxexp8": {
      "projectType": "application",
      "schematics": {},
      "root": "",
      "sourceRoot": "src",
      "prefix": "app",
      "architect": {
        "build": {
          "options": {
            "outputPath": "dist/webxexp8",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "tsconfig.app.json",
            "assets": ["src/favicon.ico", "src/assets"],
            "styles": ["src/global_styles.css"],
            "scripts": []
          }
        }
      }
    }
  }
}
```

package.json

```
{
  "name": "webxexp8",
  "version": "0.0.1",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build"
  },
  "dependencies": {
    "@angular/animations": "~12.0.0",
    "@angular/common": "~12.0.0",
    "@angular/compiler": "~12.0.0",
```

```
    "@angular/core": "~12.0.0",
    "@angular/forms": "~12.0.0",
    "@angular/platform-browser": "~12.0.0",
    "@angular/platform-browser-dynamic": "~12.0.0",
    "@angular/router": "~12.0.0",
    "rxjs": "~6.6.0",
    "tslib": "^2.0.0",
    "zone.js": "~0.11.4"
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "~12.0.0",
    "@angular/cli": "~12.0.0",
    "@angular/compiler-cli": "~12.0.0",
    "@types/node": "^12.11.1",
    "typescript": "~4.2.3"
  }
}
```

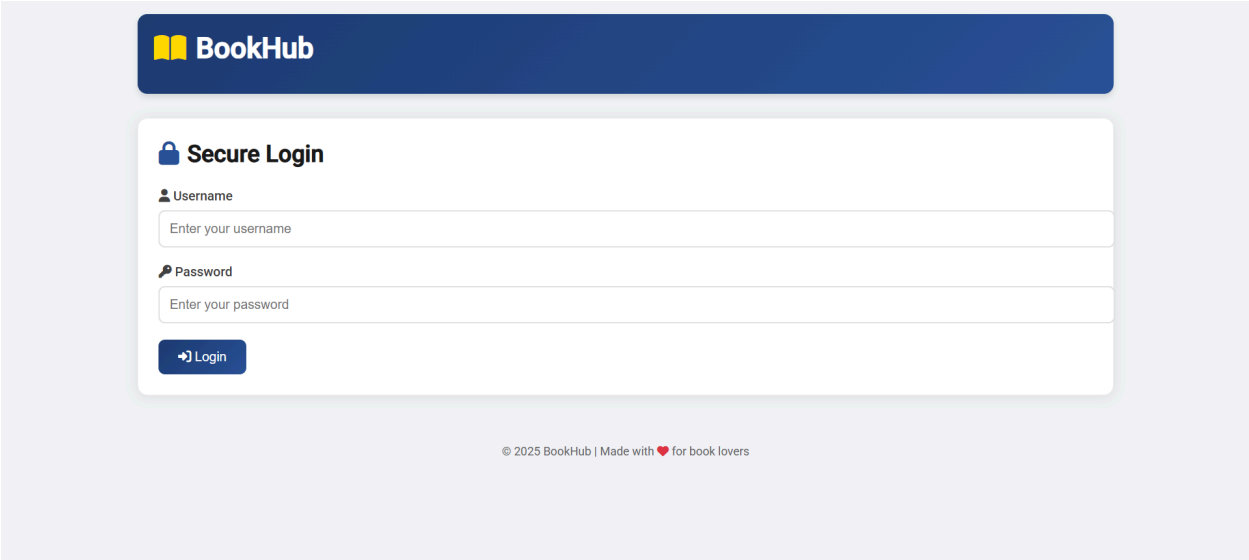
package-lock.json

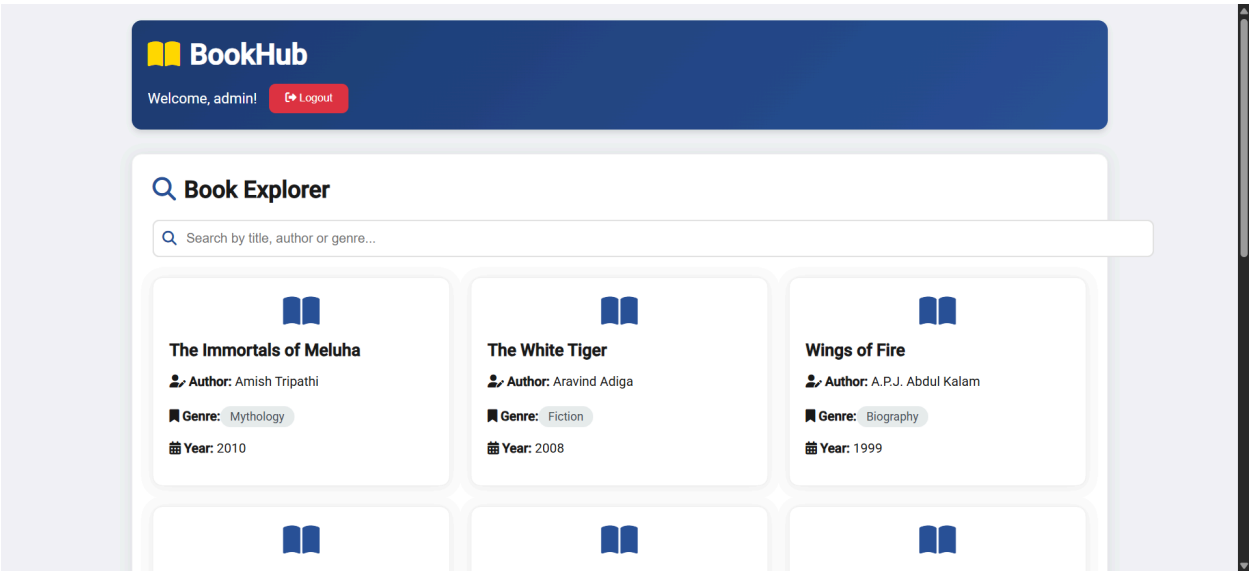
```
{
  "name": "webxexp8",
  "version": "0.0.1",
  "lockfileVersion": 1,
  "requires": true,
  "dependencies": {
    "@angular/animations": {
      "version": "12.0.0",
      "resolved": "https://registry.npmjs.org/@angular/animations/-/animations-12.0.0.tgz",
      "integrity": "sha512-...",
      "requires": {
        "tslib": "^2.0.0"
      }
    }
    // Additional dependencies...
  }
}
```

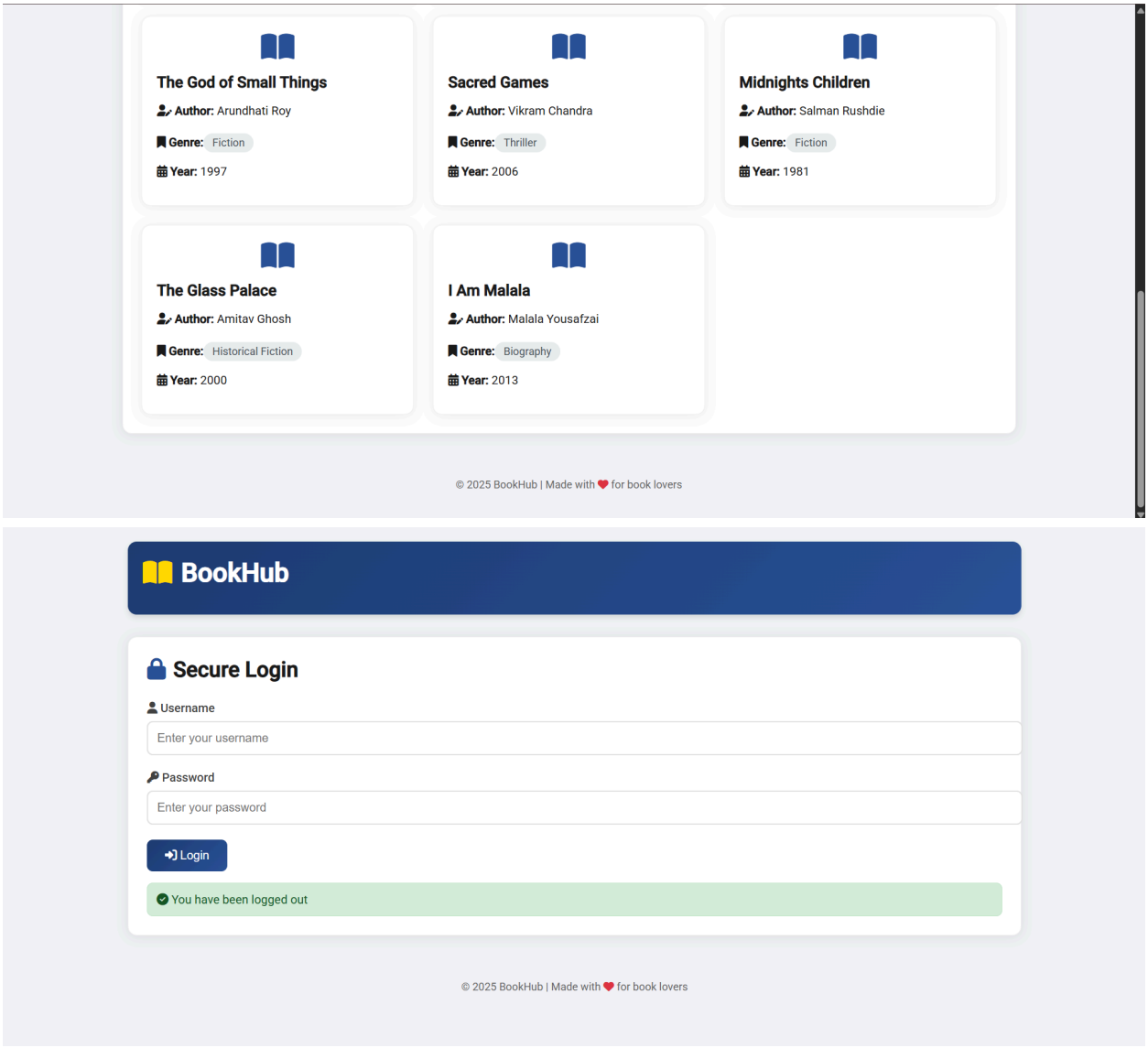
tsconfig.app.json

```
{
  "extends": "./tsconfig.json",
  "compilerOptions": {
    "outDir": "./out-tsc/app",
    "types": []
  },
  "files": ["src/main.ts", "src/polyfills.ts"],
  "include": ["src/**/*.d.ts"]
}
```

Output:


BookHub
Secure Login
Username
Enter your username
Password
Enter your password
Login
© 2025 BookHub | Made with for book lovers


BookHub
Welcome, admin!
Logout
Book Explorer
Search by title, author or genre...
The Immortals of Meluha
Author: Amish Tripathi
Genre: Mythology
Year: 2010
The White Tiger
Author: Aravind Adiga
Genre: Fiction
Year: 2008
Wings of Fire
Author: A.P.J. Abdul Kalam
Genre: Biography
Year: 1999



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.