Practical Journal

for

Advance Application Development

Submitted by: Raaj umesh Chaurasia

Roll no:23-375

Course: SYBsc Computer Science

Sem-IV

Submitted to: Ms. Ananya .Rao

Academic year-2024-2025

Department of computer science and information technology

PVG college of science and technology

INDEX

|  |  |
| --- | --- |
| Sr.no | Title |
| 1 | Write a program to implement MongoDB data models. |
| 2 | Write a program to implement CRUD operations on MongoDB. |
| 3 | Write a program to perform validation of a form using AngularJS. |
| 4 | Write a program to create and implement modules and controllers in AngularJS. |
| 5 | Create an application for Customer / Students records using AngularJS |
| 6 | Write a program to implement Error Handling in Angular JS |
| 7 | Create an app using Flutter to demonstrate the use of different layouts |
| 8 | Create a simple html “hello world” project using angular framework and apply ng-controller,ng-model and expressions. |

PRACTICAL 1

Aim: Write a program to implement MongoDB data models

Solution:

Step 1: Creating index.js and model.js file.

Step 2: Run the “npm init” command in the terminal to initialize the required

files.

Step 3: Installing “mongoose” package using “npm i mongoose” or “npm

install mongoose” in the project.

Step 4: Editing index.js file

Code:

const mongoose = require("mongoose");

mongoose.set('strictQuery', true);

mongoose.connect("mongodb://127.0.0.1:27017", {

dbName: "test", useUnifiedTopology: true, useNewUrlParser: true

}, err => err ? console.log(err) : console.log('Connected to database'));

// Creating Schema

const studentSchema = new mongoose.Schema(

{ name: String, rollNo: Number, class: String, age: Number, email:

String }

)

// Defining Student model

const Student = mongoose.model('Student', studentSchema);

// Create collection of Model

Student.createCollection().then(function () {

console.log('Collection is created!');

});

Step 5: Editing model.js file

Code:

const mongoose = require("mongoose");

//Scheme for collection

const studentSchema = new mongoose.Schema({

name: String,

rollNo: String,

class: String,

contactNo: String,

email: String

}, { collection: "students" });

//Exporting scheme

module.exports = mongoose.model("student", studentSchema);

Step 6: Executing “node index.js” command in terminal

PRACTICAL 2

Aim: Write a program to implement CRUD operations on MongoDB

Solution:

Step 1: Creating createCollection.js, insertOne.js, insertmany.js, getdata.js,

update.js, and delete.js file.

Step 2: Run the “npm init” command in the terminal to initialize the required

files.

Step 3: Installing “mongoose” package using “npm i mongoose” or “npm

install mongoose” in the project.

Step 4: Editing createCollection.js file

Code:

const mongoose = require("mongoose");

mongoose.connect("mongodb://127.0.0.1:27017", {

dbName: "test", useUnifiedTopology: true, useNewUrlParser: true

}, err => err ? console.log(err) : console.log('Connected to database'));

// Creating Schema

const studentSchema = new mongoose.Schema(

{ name: String, rollNo: Number, class: String, age: Number, email:

String }

)

// Defining Student model

const Student = mongoose.model('Student', studentSchema);

// get reference to database

var db = mongoose.connection;

// function to create collection of Model

Student.createCollection().then(function () {

console.log('Collection is created!');

});

// To Check error

db.on('error', console.error.bind(console, 'connection error:'));

Step 5: Editing insertOne.js file

Code:

const mongoose = require("mongoose");

mongoose.connect("mongodb://127.0.0.1:27017", {

dbName: "test", useUnifiedTopology: true, useNewUrlParser: true

}, err => err ? console.log(err) : console.log('Connected to database'));

// Creating Schema

const studentSchema = new mongoose.Schema(

{ name: String, rollNo: Number, class: String, age: Number, email:

String }

)

// Defining Student model

const Student = mongoose.model('Student', studentSchema);

// get reference to database

var Student1 = new Student({ name: 'VSatish', rollNo: 31, class: 'SyCs',

age: 19, email: '13VSatish@gmail.com' });

Student1.save(function(err,result){

if (err){

console.log(err);

}

else{

console.log(result)}});

Step 6: Editing insertmany.js file

Code:

const mongoose = require("mongoose");

mongoose.connect("mongodb://127.0.0.1:27017", {

dbName: "test", useUnifiedTopology: true, useNewUrlParser: true

}, err => err ? console.log(err) : console.log('Connected to database'));

// Creating Schema

const studentSchema = new mongoose.Schema(

{ name: String, rollNo: Number, class: String, age: Number, email:

String }

)

// Defining Student model

const Student = mongoose.model('Student', studentSchema);

// To insert Multi data in db

db.once('open', function () {

// save model to database

Student.insertMany([

{ name: 'test', rollNo: 31, class: 'SyCs', age: 20, email:

'testmail1@gmail.com' },

{ name: 'test1', rollNo: 32, class: 'SyCs', age: 18, email:

'testmail2@gmail.com' },

{ name: 'test2', rollNo: 33, class: 'SyCs', age: 25, email:

'testmail3@gmail.com' },

Prepared by Asst.Prof.Dnyaneshwar Deore

7

{ name: 'test3', rollNo: 34, class: 'SyCs', age: 21, email:

'testmail4@gmail.com' }

]).then(function(){

console.log("Data inserted") // Success

}).catch(function(error){

console.log(error) // Failure

});

});  
Step 7: Editing getdata.js file

Code:

const mongoose = require("mongoose");

mongoose.connect("mongodb://127.0.0.1:27017", {

dbName: "test", useUnifiedTopology: true, useNewUrlParser: true

}, err => err ? console.log(err) : console.log('Connected to database'));

// Creating Schema

const studentSchema = new mongoose.Schema(

{ name: String, rollNo: Number, class: String, age: Number, email:

String }

)

// Defining Student model

const Student = mongoose.model('Student', studentSchema);

// To get All data from db

Student.find({}).then(data => {

console.log("Data:")

console.log(data);

}).catch(error => {

console.log(error);

})

Step 8: Editing update.js file

Code:

const mongoose = require("mongoose");

mongoose.connect("mongodb://127.0.0.1:27017", {

dbName: "test", useUnifiedTopology: true, useNewUrlParser: true

}, err => err ? console.log(err) : console.log('Connected to database'));

// Creating Schema

const studentSchema = new mongoose.Schema(

{ name: String, rollNo: Number, class: String, age: Number, email:

String }

)

// Defining Student model

const Student = mongoose.model('Student', studentSchema);

// To update data in db

Student.updateOne({name:"test3",age:30}, function (err, result) {

if (err){

console.log(err)

}else{

console.log("Result :", result)

}

});

Step 9: Editing delete.js file

Code:

const mongoose = require("mongoose");

mongoose.connect("mongodb://127.0.0.1:27017", {

dbName: "test", useUnifiedTopology: true, useNewUrlParser: true

}, err => err ? console.log(err) : console.log('Connected to database'));

// Creating Schema

const studentSchema = new mongoose.Schema(

{ name: String, rollNo: Number, class: String, age: Number, email:

String }

)

// Defining Student model

const Student = mongoose.model('Student', studentSchema);

// get reference to database

var db = mongoose.connection;

// To update data in db

Student.deleteOne({ age: 20 }).then(function(){

console.log("Data deleted"); // Success

}).catch(function(error){

console.log(error); // Failure

});

PRACTICAL 3

Aim: Write a program to perform validation of a form using AngularJS

Solution:

Step 1: Create index.html and welcome.html files.

Step 2: Editing index.html file

Code:

<!DOCTYPE html>

<html>

<head>

<title>

AngularJs Form Validation

</title>

<script

src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<script>

var app = angular.module('formApp', []);

app.controller('formCtrl', function ($scope) {

$scope.sendForm = function () {

window.open("welcome.html");

$scope.msg='Form Submited Successfully';

};

$scope.getClass = function (color) {

return color.toString();

}

});

</script>

**<style>**

**.valid.false {**

**background: red;**

**}**

**.valid.true {**

**background: green;**

**}**

**.error {**

**color: red;**

**}**

**</style>**

**</head>**

**<body ng-app="formApp" ng-controller="formCtrl" bgcolor=”pink”>**

**<h3>Form validation demo app in AngularJs</h3>**

**<form name="personForm" ng-submit="sendForm()">**

**<label for="name">Name</label>**

**<input id="name" name="name" type="text" ng-model="person.name" required />**

**<span class="error" ng-show="personForm.name.$error.required"> Required!**

**</span>**

**<br /><br />**

**<label for="adress">Adress</label>**

**<input id="address" name="address" type="text" ng-model="person.address"**

**required />**

**<span class="error" ng-show="personForm.address.$error.required"> Required!**

**</span>**

**<br /><br />**

**<label for="contact">Contact No</label>**

**<input id="mobile" name="mobile" type="number" ng-model="person.mobile"**

**required />**

**<span class="error" ng-show="personForm.mobile.$error.required">Required**

**number!</span>**

**<span class="error" ng-show="personForm.mobile.$error.mobile">Invalid**

**mobile!</span>**

**<br /><br />**

**<label for="email">Email</label>**

**<input id="email" name="email" type="email" ng-model="person.email" required />**

**<span class="error" ng-show="personForm.email.$error.required">Required!</span>**

**<span class="error" ng-show="personForm.email.$error.email">Invalid**

**Email!</span>**

**<br /><br />**

**<input type="checkbox" ng-model="terms" name="terms" id="terms" required />**

**<label for="terms">I Agree to the terms.</label>**

**<span class="error" ng-show="personForm.terms.$error.required">You must agree**

**to the terms</span>**

**<br /><br />**

**<button type="submit">Submit Form</button>**

**<br /><br />**

**<span>{{msg}}</span>**

**</form>**

**</body>**

**</html>**

**Step 3: Editing welcome.html file**

**Code:**

**<html>**

**<head>**

**<title>Welcome Page</title>**

**</head>**

**<body bgcolor="yellow">**

**<h1>Record Successfully Submited............</h1>**

**</body>**

**</html>**

**A screenshot of a computer

Description automatically generated**

A screenshot of a computer

Description automatically generated

PRACTICAL 4

Aim: Write a program to create and implement modules and controllers in

AngularJS?

Solution:

Step 1: To create folder module and view page viewpage.html in root directory.

Step 2: Editing view page viewpage.html file

Code:

<html>

<head>

<title>Angular JS Modules</title>

<script

src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>

<script src="module/mainApp.js"></script>

<script src=" module/studentController.js"></script>

<style>

table, th , td {

border: 1px solid grey;

border-collapse: collapse;

padding: 5px;

}

table tr:nth-child(odd) {

background-color: #f2f2f2;

}

table tr:nth-child(even) {

background-color: #ffffff;

}

</style>

</head>

<body>

<h2>AngularJS Sample Application</h2>

<div ng-app="mainApp" ng-controller="studentController">

<table border="0">

<tr>

<td>Enter first name:</td>

<td><input type="text" ng-model="student.firstName"></td>

</tr>

<tr>

<td>Enter last name: </td>

<td><input type="text" ng-model="student.lastName"></td>

</tr>

<tr>

<td>Name: </td>

<td>{{student.fullName()}}</td>

</tr>

<tr>

<td>Subject:</td>

<td>

<table>

<tr>

<th>Name</th>

<th>Marks</th>

</tr>

<tr ng-repeat="subject in student.subjects">

<td>{{ subject.name }}</td>

<td>{{ subject.marks }}</td>

</tr>

</table>

</td>

</tr>

</table>

</div>

</body>

</html>

Step 3:To create mainApp.js file inside module folder and editing script file

Code:

var mainApp = angular.module("mainApp", []);

Step 4: To create studentController.js inside module folder and editing

script file

Code:

mainApp.controller("studCtrl", function($scope){

$scope.student = {

fname: "Satish",

lname: "Vishwakarma",

fees: 150,

subjects:[

{name:"Advance app devloipment", marks:80},

{name:"Advance app devloipment", marks:80},

{name:"Advance app devloipment", marks:80},

{name:"Advance app devloipment", marks:80},

{name:"Advance app devloipment", marks:80},

{name:"Advance app devloipment", marks:80},

{name:"Advance app devloipment", marks:80},

],

fullName: function(){

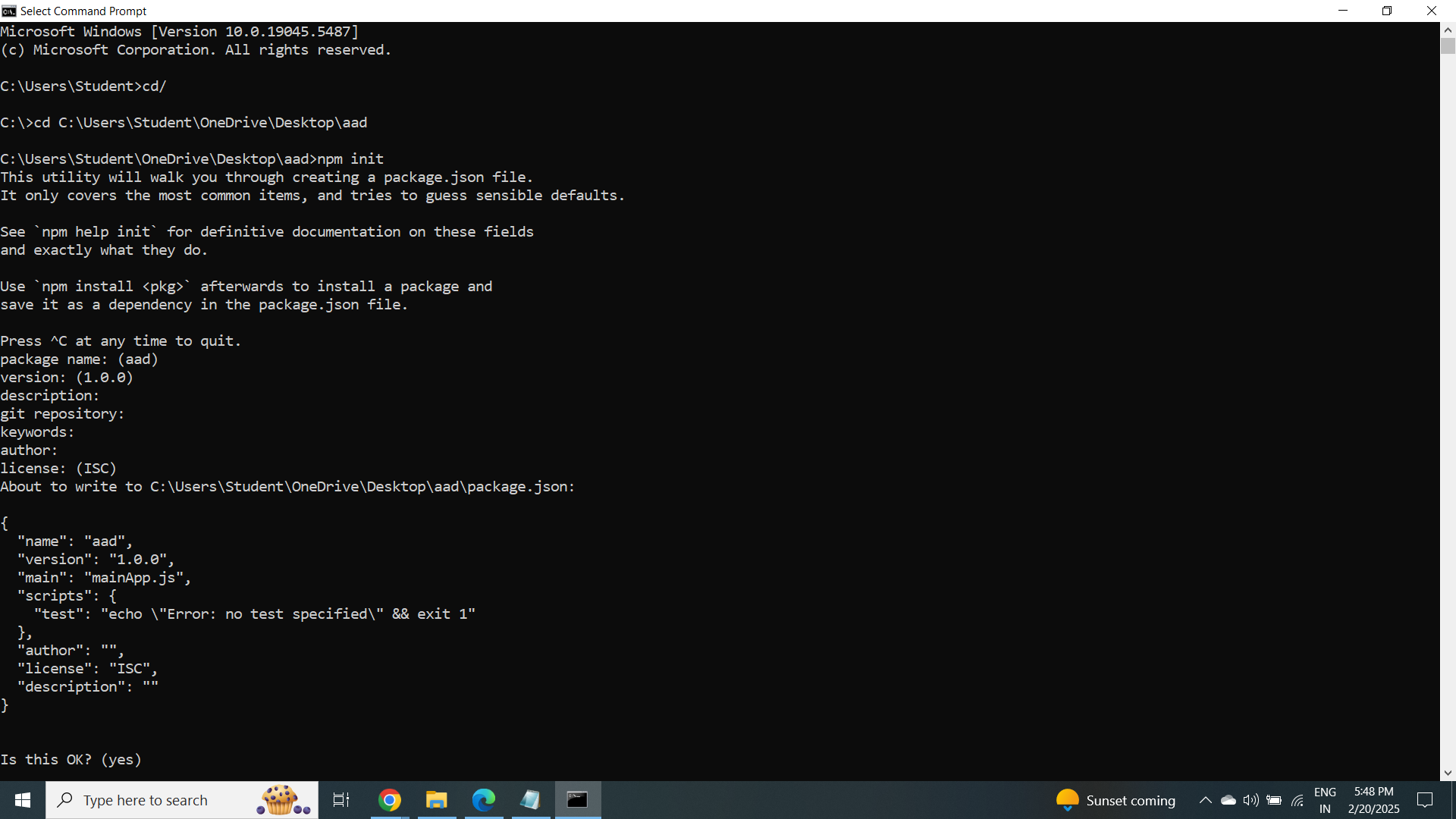
var stud = $scope.student;

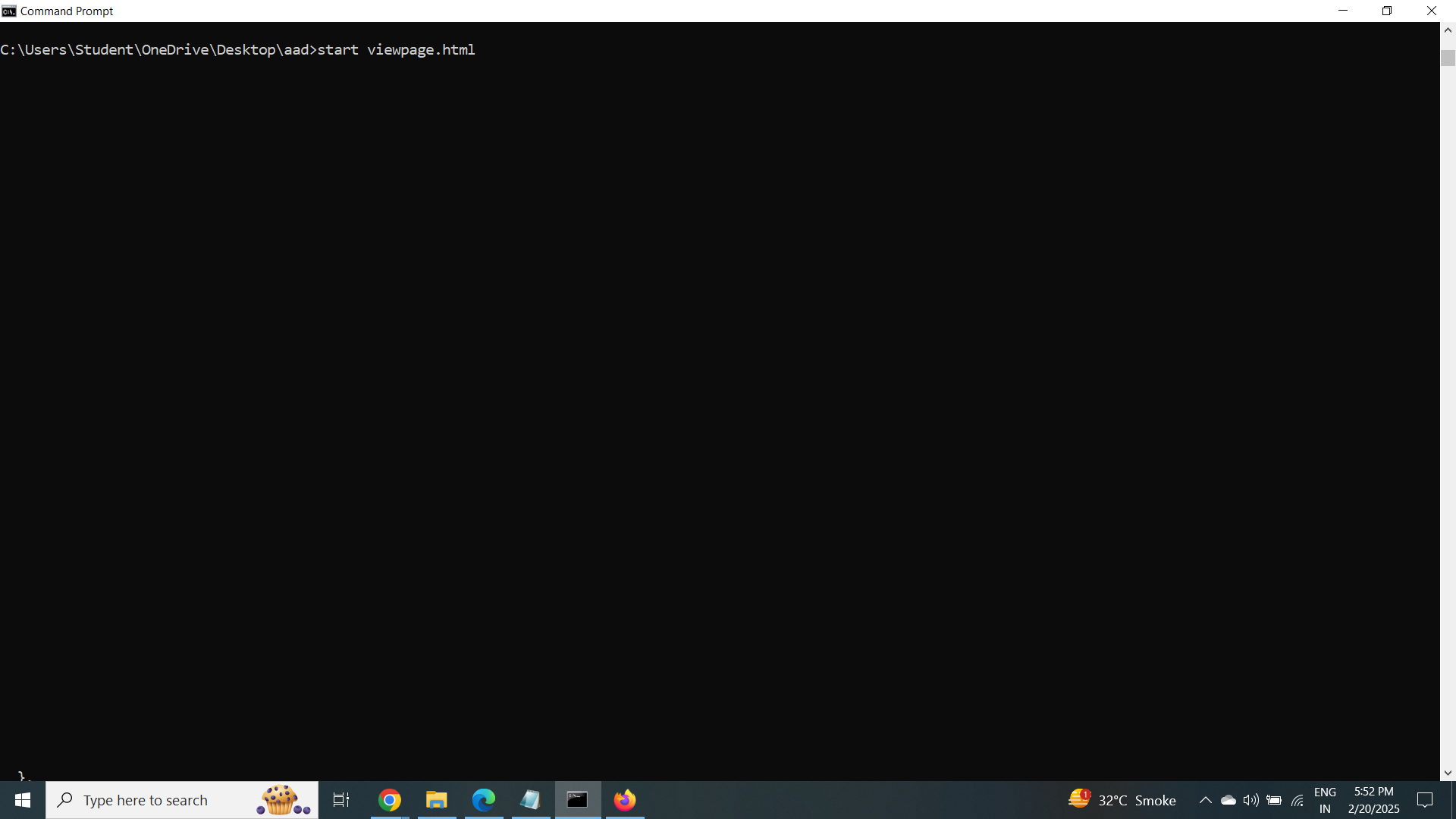
return stud.fname + " " + stud.lname;

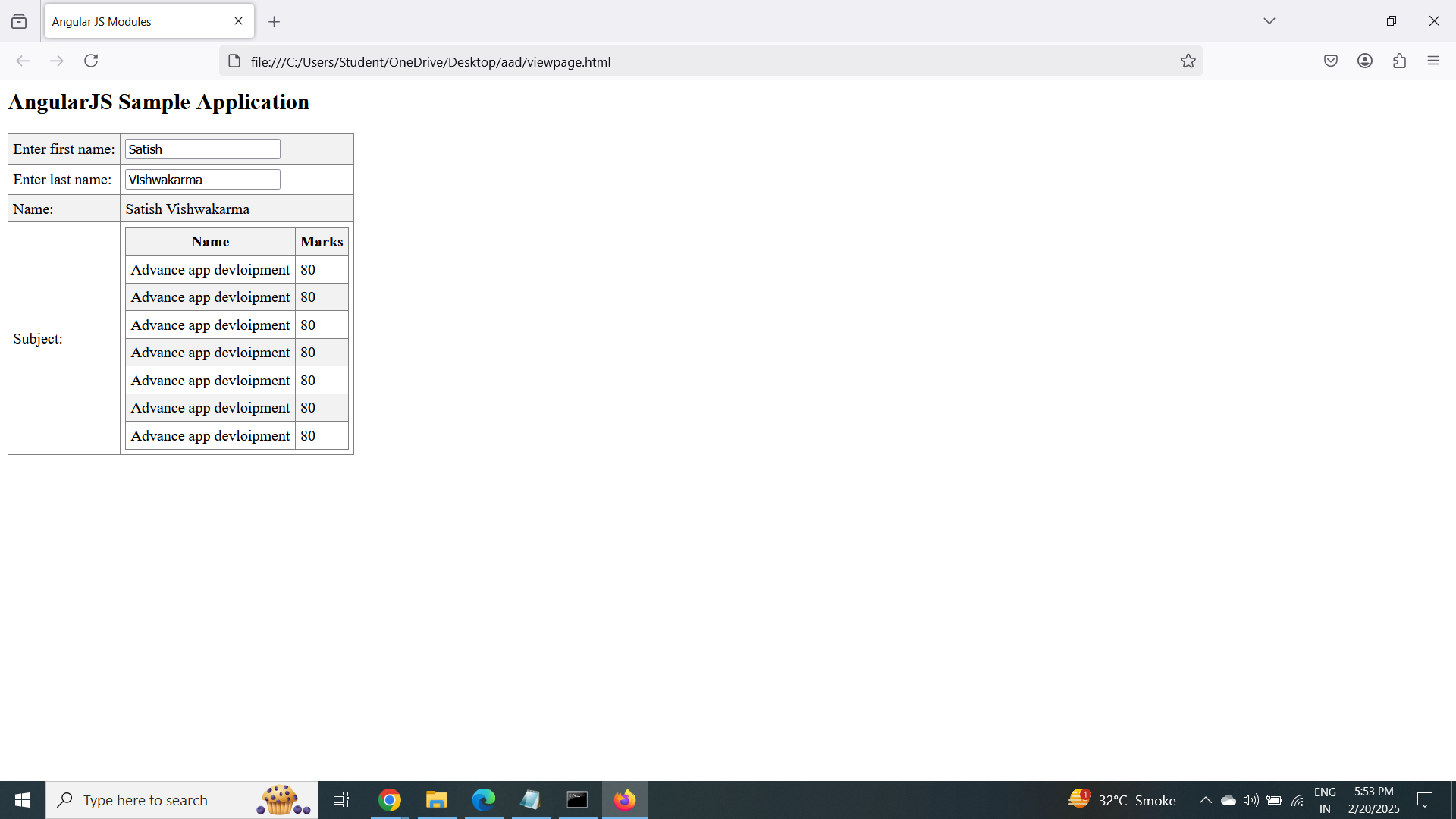
}

};

});







Practical no 5:

Aim:Creation of list for an application development

Create an application for Customer / Students records using AngularJS

1. Creation of List for an Application.

2. Adding elements in the List.

3. Removing elements from the List.

Make a Shopping List

<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

<body>

<script>

var app = angular.module("myShoppingList", []);

app.controller("myCtrl", function($scope) {

$scope.products = ["Milk", "Bread", "Cheese"];

});

</script>

<div ng-app="myShoppingList" ng-controller="myCtrl">

<ul>

<li ng-repeat="x in products">{{x}}</li>

</ul>

</div>

<p>So far we have made an HTML list based on the items of an array.</p>

</body>

</html>

Step 2. Adding Items:

<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

<body>

<script>

var app = angular.module("myShoppingList", []);

app.controller("myCtrl", function($scope) {

$scope.products = ["Milk", "Bread", "Cheese"];

$scope.addItem = function () {

$scope.products.push($scope.addMe);

}

});

</script>

<div ng-app="myShoppingList" ng-controller="myCtrl">

<ul>

<li ng-repeat="x in products">{{x}}</li>

</ul>

<input ng-model="addMe">

<button ng-click="addItem()">Add</button>

</div>

<p>Write in the input field to add items.</p>

</body>

</html>

Step 3. Removing Items:

<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

<body>

<script>

var app = angular.module("myShoppingList", []);

app.controller("myCtrl", function($scope) {

$scope.products = ["Milk", "Bread", "Cheese"];

$scope.addItem = function () {

$scope.products.push($scope.addMe);

}

$scope.removeItem = function (x) {

$scope.products.splice(x, 1);

}

});

</script>

<div ng-app="myShoppingList" ng-controller="myCtrl">

<ul>

<li ng-repeat="x in products">{{x}}<span ng-click="removeItem($index)">×</span></li>

</ul>

<input ng-model="addMe">

<button ng-click="addItem()">Add</button>

</div>

<p>Click the little x to remove an item from the shopping list.</p>

</body>

</html>

Create an application for Students records using AngularJS

"https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">

var app = angular.module("Subjects", []);

app.controller("my\_Ctrl", function($scope) {

$scope.name = [

"English", "Maths", "Economics"];

});

"https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">

var app = angular.module("Subjects", []);

app.controller("my\_Ctrl", function($scope) {

$scope.name = ["English", "Maths", "Economics"];

$scope.addingNewSubject = function () {

$scope.name.push($scope.addSubject);

}

});

"https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">

var app = angular.module("Subjects", []);

app.controller("my\_Ctrl", function($scope) {

$scope.name = ["English", "Maths", "Economics"];

$scope.addingNewSubject = function () {

$scope.name.push($scope.addSubject);

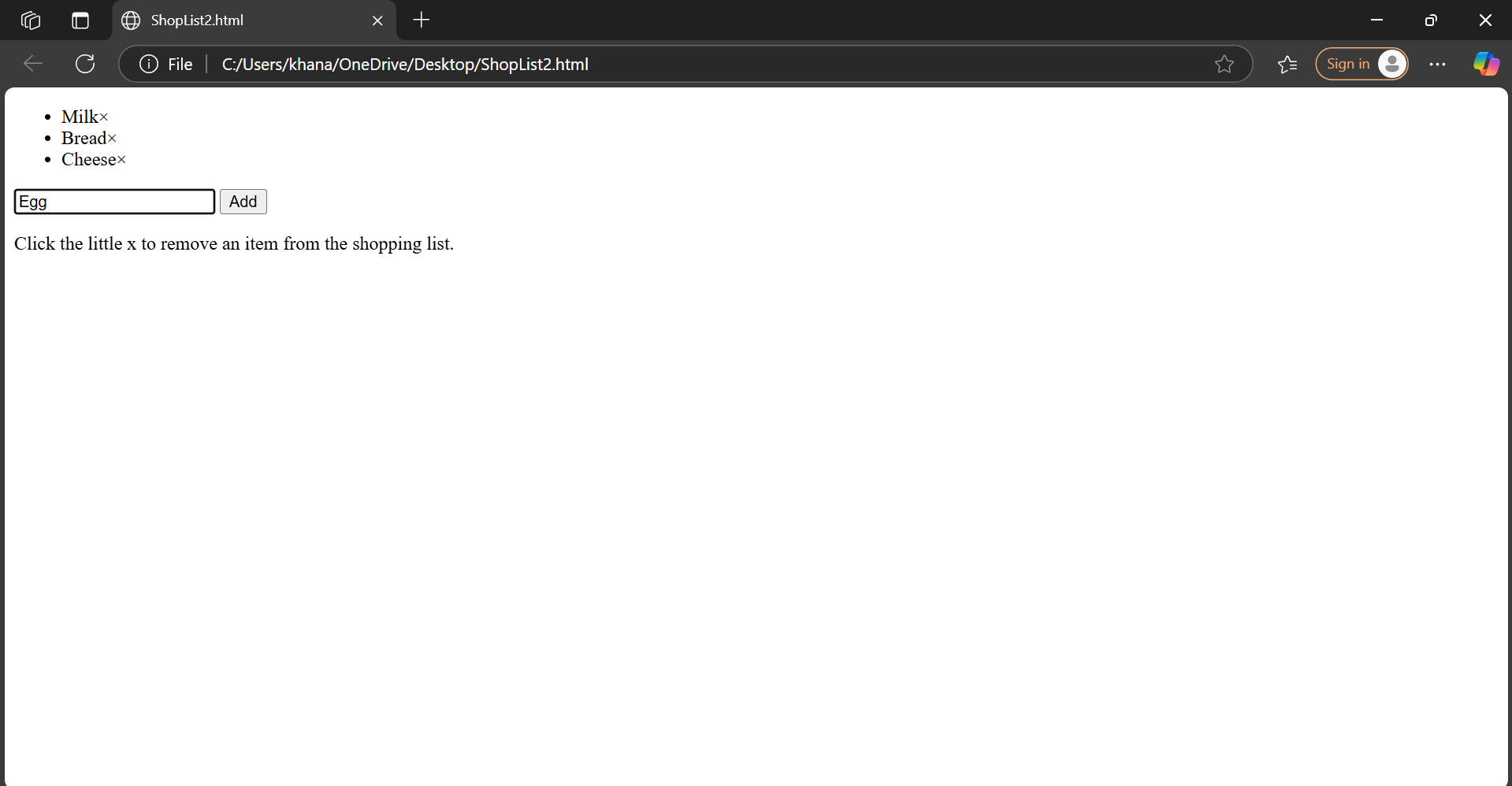
}

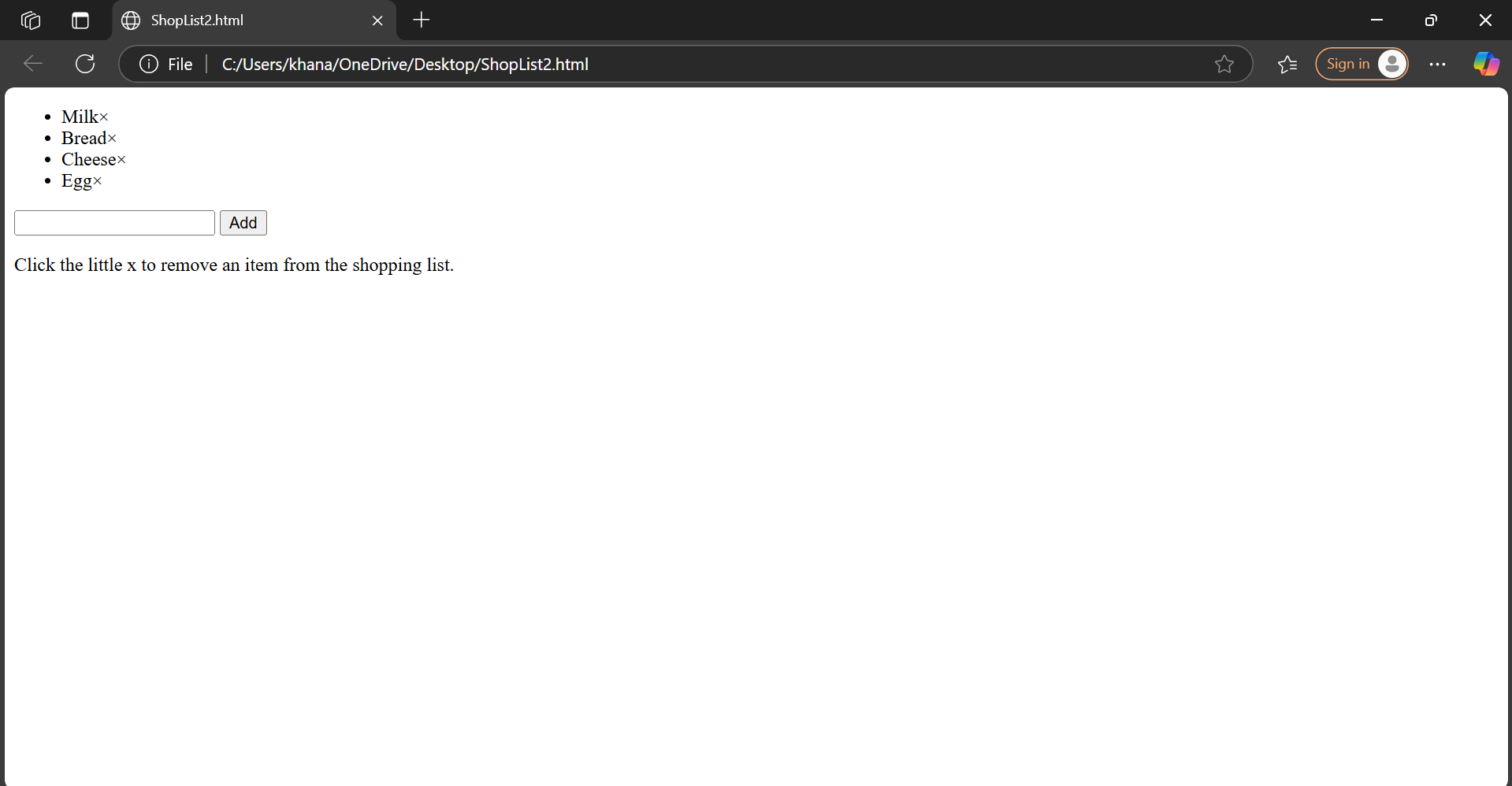
$scope.remove = function (x) {

$scope.name.splice(x, 1);

}

});





Practical no 6:

Aim:Error handling in angular.js

<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

<body>

<script>

var app = angular.module("myShoppingList", []);

app.controller("myCtrl", function($scope) {

$scope.products = ["Milk", "Bread", "Cheese"];

$scope.addItem = function () {

$scope.errortext = "";

if (!$scope.addMe) {return;}

if ($scope.products.indexOf($scope.addMe) == -1) {

$scope.products.push($scope.addMe);

} else {

$scope.errortext = "The item is already in your shopping list.";

}

}

$scope.removeItem = function (x) {

$scope.errortext = "";

$scope.products.splice(x, 1);

}

});

</script>

<div n

g-app="myShoppingList" ng-controller="myCtrl">

<ul>

<li ng-repeat="x in products">{{x}}<span ng-click="removeItem($index)">×</span></li>

</ul>

<input ng-model="addMe">

<button ng-click="addItem()">Add</button>

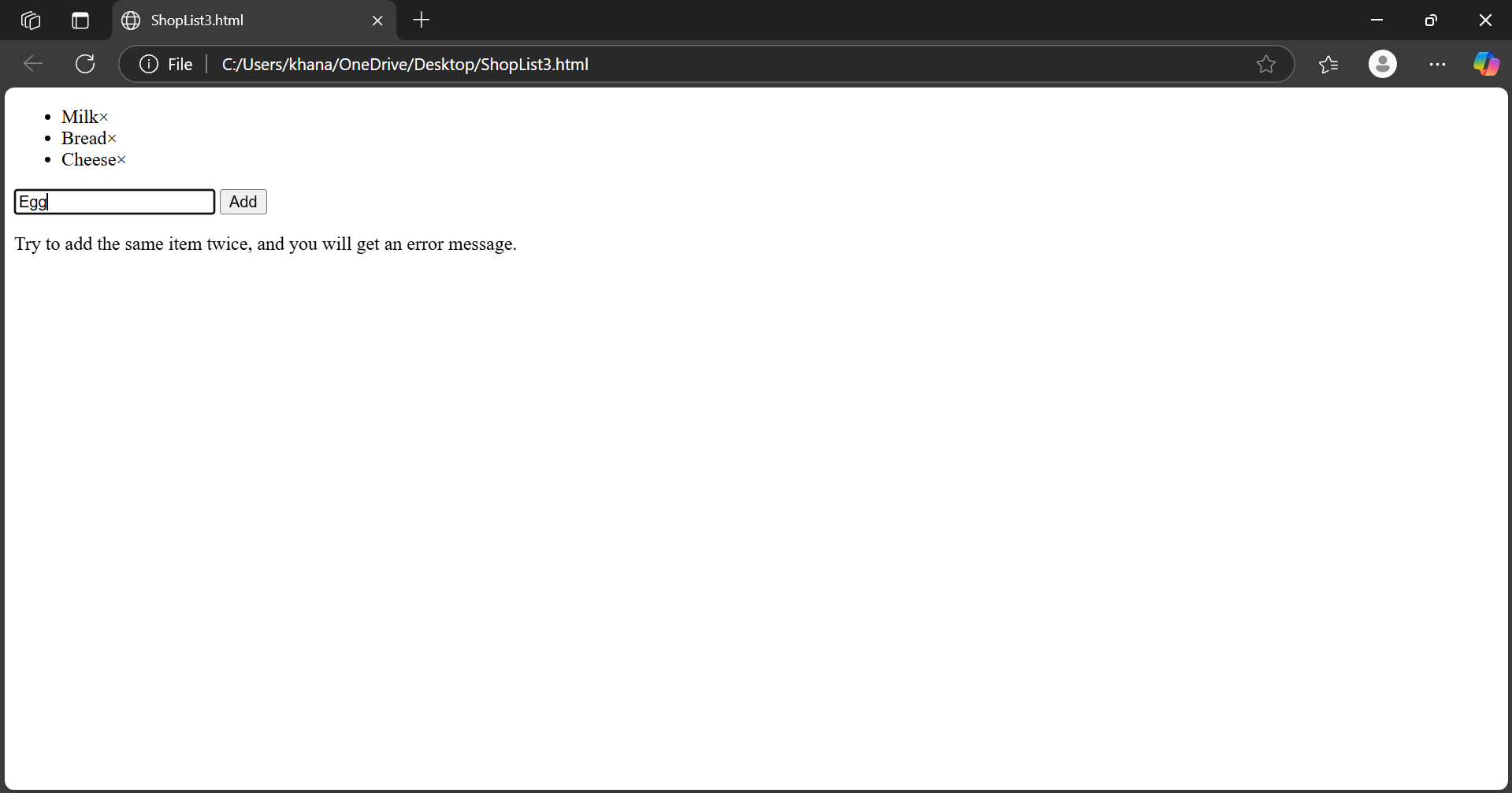
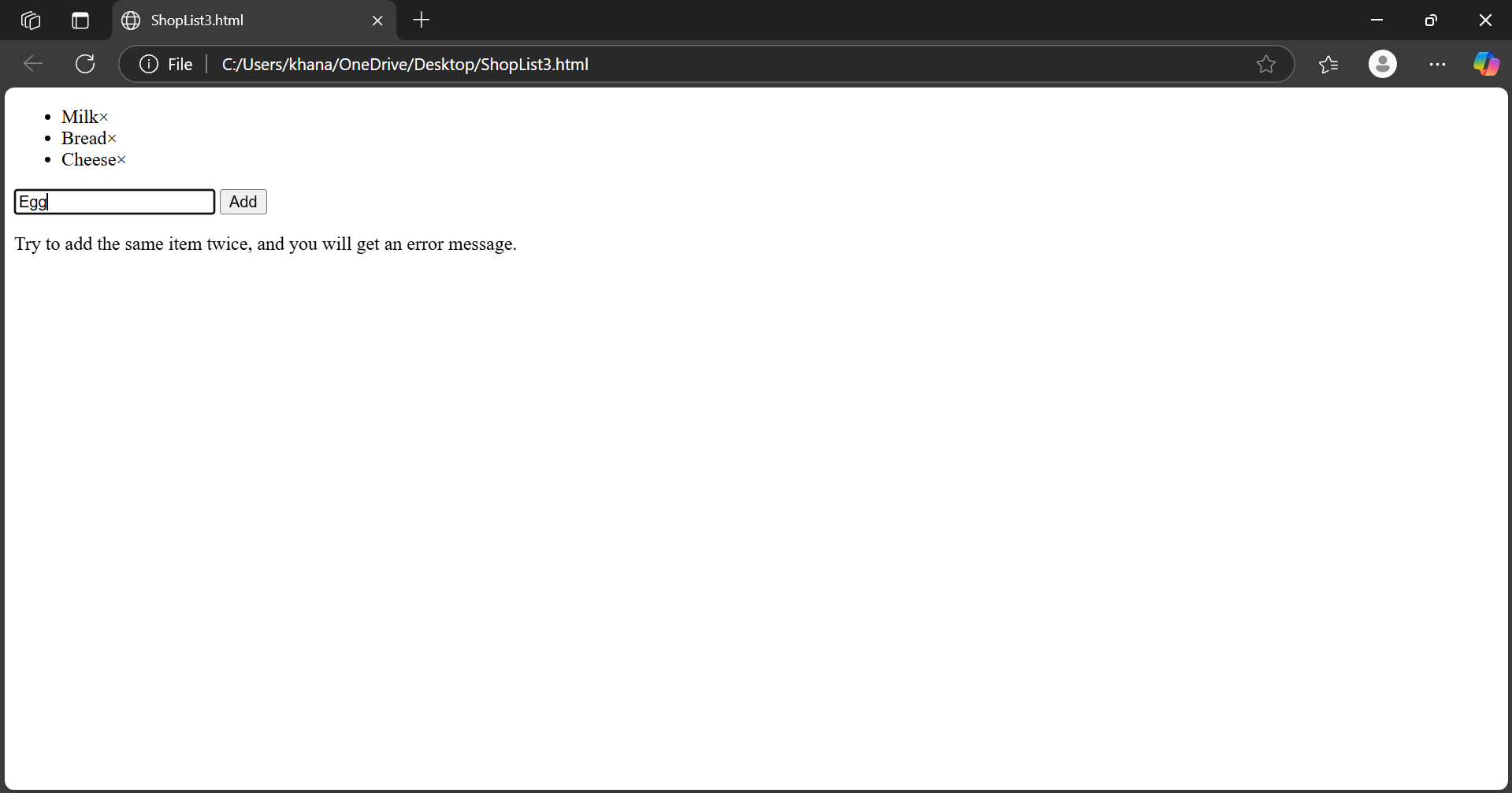
<p>{{errortext}}</p>

</div>

<p>Try to add the same item twice, and you will get an error message.</p>

</body>

</html>



Practical no 7

Aim-Implementation of layout in flutter

Code:

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) { return MaterialApp( home: Scaffold( appBar: AppBar(

title: Text('Layout Example'),

),

body: Column( children: [

Container(

height: 100, color: Colors.red,

),

Row(

children: [

Expanded( child: Container( height: 100, color: Colors.blue,

),

),

Expanded( child: Container( height: 100, color: Colors.green,

),

),

],

),

Padding(

padding: const EdgeInsets.all(20.0), child: Text('Hello, World!'),

),

],

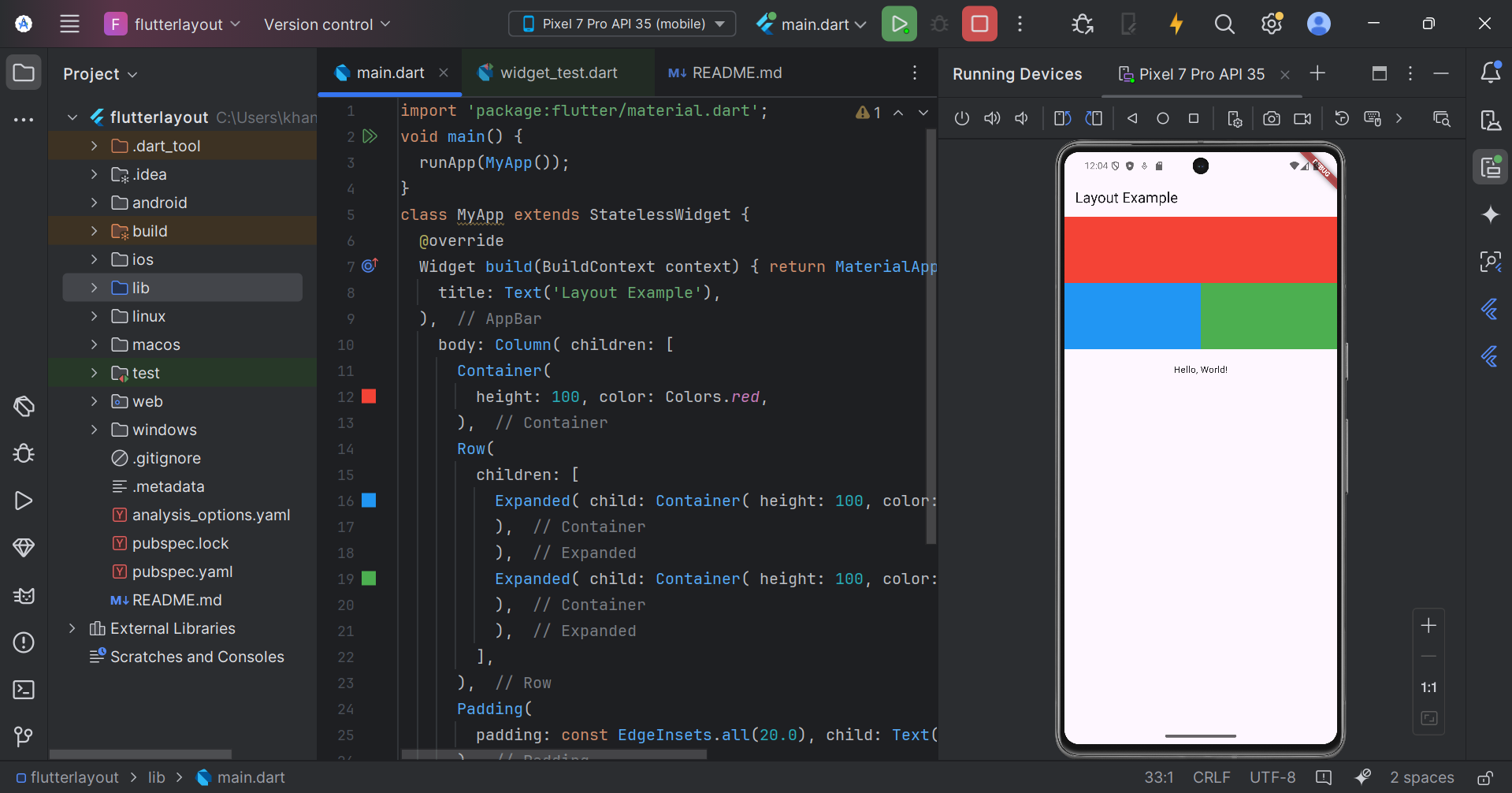
),

),

);

}

}



Practical 8

Create a simple html “hello world” project using angular framework and apply ng-controller,ng-model and expressions.

code:

<!DOCTYPE html>

<html ng-app="app">

<head>

<meta charset="utf 8">

<title>hello world</title>

</head>

<body>

<h1 ng-controller="HelloWorldCtrl">{{message}}</h1>

<script src="https://code.angularjs.org/1.6.9/angular.js"></script>

<script>

angular.module("app", []).controller("HelloWorldCtrl", function($scope) {

$scope.message="Hello World"

} )

</script>

</body>

</html>

PRACTICAL NO: 9

Create an app using flutter to implement an image gallery

import 'package:flutter/material.dart';

void main() {

runApp(const MyApp());

}

class MyApp extends StatelessWidget {

const MyApp({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Image Gallery',

theme: ThemeData(primarySwatch: Colors.blue),

home: const ImageGallery(),

);

}

}

class ImageGallery extends StatelessWidget {

const ImageGallery({Key? key}) : super(key: key);

// Change final to const

static const List<String> \_images = [

'https://picsum.photos/200/300?random=1',

'https://picsum.photos/200/300?random=2',

'https://picsum.photos/200/300?random=3',

'https://picsum.photos/200/300?random=4',

'https://picsum.photos/200/300?random=5',

'https://picsum.photos/200/300?random=6',

'https://picsum.photos/200/300?random=7',

'https://picsum.photos/200/300?random=8',

'https://picsum.photos/200/300?random=9',

];

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: const Text('Image Gallery'),

),

body: GridView.builder(

gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(

crossAxisCount: 3,

crossAxisSpacing: 4,

mainAxisSpacing: 4,

),

itemCount: \_images.length,

itemBuilder: (context, index) {

return GestureDetector(

onTap: () {

// You can implement a detailed view here

showDialog(

context: context,

builder: (context) => AlertDialog(

content: Image.network(\_images[index]),

),

);

},

child: Image.network(

\_images[index],

fit: BoxFit.cover,

),

);

},

),

);

}

}