A Laboratory Manual for

Web Development (3151606)

B.E. Semester 5 (Information Technology)

Institute logo

Directorate of Technical Education, Gandhinagar, Gujarat

Government Engineering College, Modasa

Certificate

This is to certify that Mr./Ms. **Malam Haribhai Devshibhai** Enrollment No. **210160116051** of B.E. Semester **6th** Information Technology of this Institute (GTU Code: 016) has satisfactorily completed the Practical / Tutorial work for the subject **Advanced Web Programming (3161611)** for the academic year 2022-23.

Place: ________

Date: ________

Name Faculty member **Head of the Department**

Index
(Progressive Assessment Sheet)

Sr. No.	Objective(s) of Experiment	Page No.	Date of performance	Date of sub-mission	Assessment Marks	Sign. of Teacher with date	Remarks
1	Design a webpage that reads a Text file using AJAX.						
2	Create a HTML form that will accept Enrolment No., Name, Semester, Branch, Mobile Number, Email, Address etc. from the student and display them on the page using Angular JS Directives and Expressions.						
3	Create a webpage using Angular JS that displays details of students' objects (such as Enrolment No., Name, Semester, Branch, Mobile Number, Email Address, Address etc.) in a tabular format with appropriate CSS effects.						
4	Modify Practical 3 and provide a search field to search records on the top of the page and also allow user to sort table according to the column values when user clicks on a column using filter.						
5	Write Angular JS code to read Customer's data in JSON format available in a Customers.php file using $http service and display the same on a webpage in tabular format.						
6	Create an Angular JS application for validation that will accept Email, Username and Password as required fields from the user. It will enable submit button only if all the entered data are valid.						
7	Create an example demonstrating the concept of Angular JS Routing.						
8	Study the installation of Node JS and installation of various packages in Node JS.						
9	Design a webpage with a file input control to browse appropriate file and four buttons Read File Synchronously, Read File Asynchronously, Compress File, Decompress File. Implement the functionality of all four buttons on the browsed file using Node JS.						
10	Create a Node JS application that will allow a user to browse and upload a file in localhost.						
11	Create a Node JS application that will allow a user to create new file, read file, write into a file and delete a file.						

Experiment No: 1

Aim : Design a webpage that reads a Text file using AJAX.

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>Read Text File with AJAX</title>
</head>
<body>
    <div class="content">
        <h1>About Me</h1>
        <div id="about-content"></div>
        <button onclick="readfile()">Read File</button>
        <script src="script.js"></script>
    </div>
</body>
</html>
```

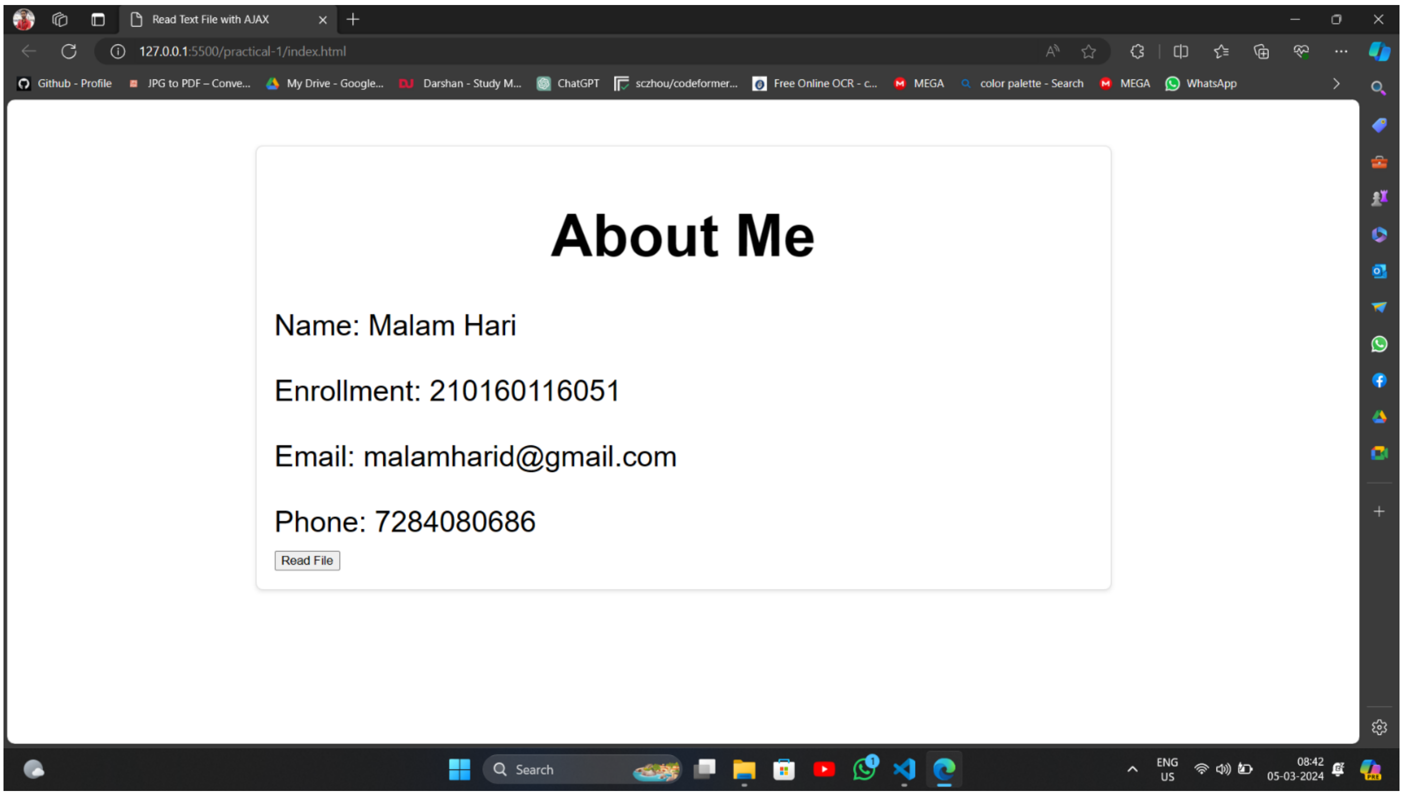
style.css

```
.content {
    font-family: Arial, sans-serif;
    max-width: 900px;
    margin: 50px auto;
    padding: 20px;
    background-color: #fff;
    border: 1px solid #ddd;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    font-size: xx-large;
}

h1{
    text-align: center;
}
```

script.js

```
const readTextFile = (file) => {
    let xhr = new XMLHttpRequest();
    xhr.open("GET", file, true);
    xhr.onreadystatechange = () => {
        if (xhr.readyState === 4 && xhr.status === 200) {
            document.getElementById("about-content").innerText = xhr.responseText;
        }
    };
    xhr.send();
}

const readfile = ()=>{
    readTextFile("about.txt");
}
```

Output



About Me

Name: Malam Hari

Enrollment: 210160116051

Email: malamharid@gmail.com

Phone: 7284080686

Quiz:

1. What is AJAX? Enlist the advantages of AJAX.

AJAX (Asynchronous JavaScript and XML) is a technique used in web development to send and receive data from a server asynchronously without interfering with the display and behavior of the existing web page.

The advantages of AJAX are:

1.Asynchronous Data Retrieval: AJAX enables web pages to fetch data from the server asynchronously without reloading the entire page.

2.Improved User Experience: By asynchronously loading data, AJAX can enhance the user experience by providing dynamic and interactive content.

3.Reduced Server Load: Since AJAX requests only retrieve the necessary data from the server, it reduces the overall server load

4.Bandwidth Conservation: AJAX allows web applications to transmit and receive data incrementally, minimizing the amount of data transferred between the client and the server.

5.Enhanced Interactivity: AJAX facilitates the creation of rich, interactive user interfaces by enabling real-time updates and interactions.

6.Support for Asynchronous Processing: AJAX supports asynchronous processing, allowing multiple tasks to be executed concurrently without blocking the user interface.

2. Which type of files can be read using AJAX?

The types of files that can be read using AJAX are:

1. Text files (e.g., .txt)
2. XML files (e.g., .xml)
3. JSON files (e.g., .json)
4. HTML files (e.g., .html)
5. CSV files (e.g., .csv)
6. Plain text files (e.g., .log)

Assessment :

Understanding ofProblem (3 marks)	Implementation ofProblem (4 marks)	Presentation and reportwriting (3 marks)	Total (10 marks)

Experiment No: 2

Aim : Create a HTML form that will accept Enrolment No., Name, Semester, Branch, Mobile Number, Email, Address etc. from the student and display them on the page using Angular JS Directives and Expressions.

Index.html

```
<!DOCTYPE html>
<html lang="en" ng-app="myApp">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Information Form</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <div ng-controller="StudentController">
        <div class="fo">
            <div class="container">
                <h2>Student Information Form</h2>
                <form id="myForm">
                    <div class="form-group">
                        <label for="enrolmentNo">Enrolment No.:</label>
                        <input type="text" id="enrolmentNo" ng-model="student.enrolmentNo"
required><br>
                    </div>
                    <div class="form-group">
                        <label for="name">Name:</label>
                        <input type="text" id="name" ng-model="student.name" required><br>
                    </div>
                    <div class="form-group">

                        <label for="semester">Semester:</label>
                        <input type="text" id="semester" ng-model="student.semester"
required><br>
                    </div>
                    <div class="form-group">

                        <label for="branch">Branch:</label>
                        <input type="text" id="branch" ng-model="student.branch"
required><br>
                    </div>
                    <div class="form-group">
                        <label for="mobile">Mobile Number:</label>
                        <input type="text" id="mobile" ng-model="student.mobile"
required><br>
                    </div>
                    <div class="form-group">

                        <label for="email">Email:</label>
                        <input type="email" id="email" ng-model="student.email"
required><br>
                    </div>
                    <div class="form-group">
                        <label for="address">Address:</label>
                        <textarea id="address" ng-model="student.address"
required></textarea><br>
                    </div>
                    <div class="form-group">
                        <button type="button" ng-click="submitForm()">Submit</button>
                    </div>
                </form>
            </div>
            <div class="container container-2">
                <h2>Student Information Preview</h2>

                <div class="form-group">
                    <p>Enrolment No.: <span ng-bind="student.enrolmentNo"></span></p>
                </div>
                <div class="form-group">
                    <p>Name: <span ng-bind="student.name"></span></p>
                </div>
                <div class="form-group">
                    <p>Semester: <span ng-bind="student.semester"></span></p>
                </div>
                <div class="form-group">
                    <p>Branch: <span ng-bind="student.branch"></span></p>
                </div>
                <div class="form-group">
                    <p>Mobile Number: <span ng-bind="student.mobile"></span></p>
                </div>
                <div class="form-group">
                    <p>Email: <span ng-bind="student.email"></span></p>
                </div>
                <div class="form-group">
                    <p>Address: <span ng-bind="student.address"></span></p>
                </div>
            </div>
        </div>
    </div>
    <script src="script.js"></script>
</body>

</html>
```

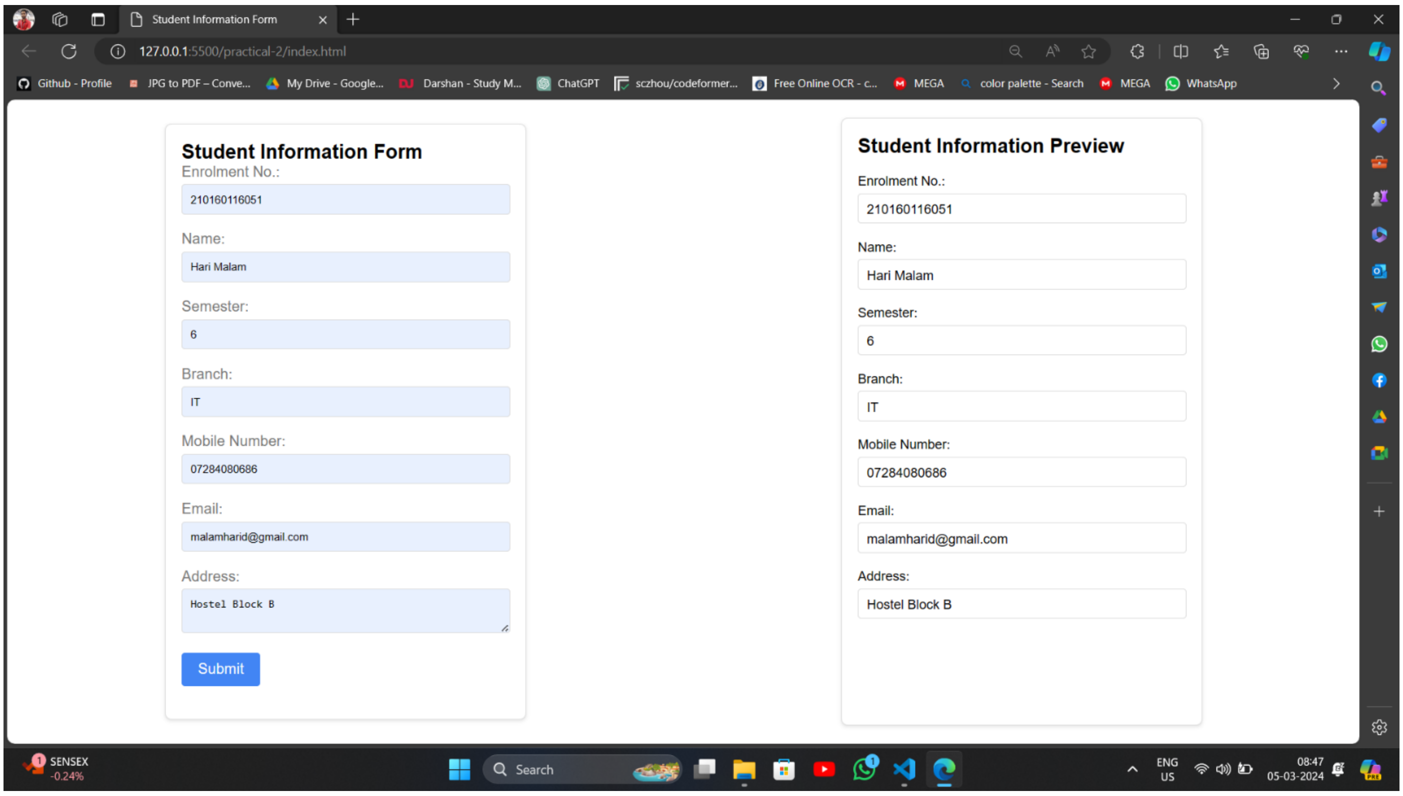
style.css

```
* {
    padding: 0;
    margin: 0;
}

.fo {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: space-around;
    align-items: center;
    height: 100vh;

}

.container {
    background-color: #fff;
    border: 1px solid #ddd;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    padding: 20px;
    width: 400px;
}

.form-group {
    margin-bottom: 20px;
}

label {
    font-size: 18px;
    color: #777;
}

input[type="text"],
input[type="email"],
textarea {
    width: calc(100% - 20px);
    padding: 10px;
    border: 1px solid #ddd;
    border-radius: 4px;
    margin-top: 5px;
}

button {
    background-color: #4285f4;
    color: #fff;
    padding: 10px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 18px;
}

button:hover {
    background-color: #357ae8;
}

.container-2 {
    height: 700px;
}

.container-2 h2 {
    margin-bottom: 20px;
}

.container-2 .form-group {
    margin-bottom: 20px;
}

span {
    display: block;
    width: calc(100% - 20px);
    padding: 10px;
    border: 1px solid #ddd;
    border-radius: 4px;
    margin-top: 5px;
    height: 15px;
}
```

script.js

```
const app = angular.module('myApp', []);
app.controller('StudentController', ()=>{

});
```

Output


Student Information Form

127.0.0.1:5500/practical-2/index.html

Student Information Form

Enrolment No.:	210160116051
Name:	Hari Malam
Semester:	6
Branch:	IT
Mobile Number:	07284080686
Email:	malamharid@gmail.com
Address:	Hostel Block B

Submit

Student Information Preview

Enrolment No.:	210160116051
Name:	Hari Malam
Semester:	6
Branch:	IT
Mobile Number:	07284080686
Email:	malamharid@gmail.com
Address:	Hostel Block B

Quiz:

1. **Enlist five most commonly used Angular JS directives and state their usages.**

Five commonly used AngularJS directives and their usages are:

1.**ng-model**: Binds the value of HTML controls (like input, select, textarea) to application data, enabling two-way data binding.

2.**ng-repeat**: Iterates over a collection (array or object) and repeats a set of HTML elements for each item in the collection.

3.**ng-click:** Attaches a click event handler to HTML elements, allowing them to trigger custom JavaScript functions defined in the AngularJS controller.
4.**ng-show/ng-hide**: Conditionally shows or hides HTML elements based on the evaluation of an expression.

5.**ng-if:** Conditionally renders HTML elements based on the evaluation of an expression. It adds or removes elements from the DOM based on the condition.

2. Explain various ways to display the values of the variables on the page with suitableexample.

There are several ways to display the values of variables on a webpage, depending on the context and requirements of your application. Here are some common methods with examples:

1. **Using JavaScript and HTML:** You can use JavaScript to manipulate the DOM and update HTML elements dynamically.

```
 <!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Display
   Variables</title>
</head>
<body>
   <p id="variableDisplay"></p>

   <script>
      // JavaScript code var myVariable = "Hello,
      world!";
      document.getElementById("variableDisplay").textContent = myVariable;
   </script>
</body>
</html>
```

2. **Using AngularJS Data Binding:** In AngularJS, you can use data binding to automatically update HTML elements when the associated model changes.

```
<!DOCTYPE html>
<html lang="en" ng-app="myApp">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initialscale=1.0">
<title>Display Variables</title>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.
min.js"></script>
</head>
<body>
<div ng-controller="MyController">
<p>{{ myVariable }}</p>
</div>
<script>
// AngularJS code
angular.module("myApp", []).controller("MyController", function($scope) {
$scope.myVariable = "Hello, world!";
});
</script>
</body>
</html>
```

3 .**Using Server-Side Technologies:** If you're using server-side technologies like PHP, you can embed variables directly into the
HTML.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta  name="viewport"     content="width=device-width, initial-scale=1.0">
<title>Display Variables</title>
</head>
<body>
<p><?php echo $myVariable; ?></p>
</body>
</html>
```

4. Using Template Engines: Template engines like Handlebars or EJS allow you to embed variables directly into HTML templates and render them on the client or server-side

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Display
Variables</title>
</head>
<body>
<p>{{ myVariable }}</p>
</body>
</html>
```

Assessment :

Understanding of Problem (3 marks)	Implementation of Problem (4 marks)	Presentation and report writing (3 marks)	Total (10 marks)

Experiment No: 3

Aim :Create a webpage using Angular JS that displays details of students' objects (such as Enrolment No., Name, Semester, Branch, Mobile Number, Email Address, Address etc.) in a tabular format with appropriate CSS effects.

Index.html

```
<!DOCTYPE html>
<html lang="en" ng-app="myApp">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Details</title>
    <link rel="stylesheet" href="style.css">
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>

<div ng-controller="StudentController">
    <h2>Student Details</h2>

    <table>
        <tr>
            <th>Enrolment No.</th>
            <th>Name</th>
            <th>Semester</th>
            <th>Branch</th>
        </tr>
        <tr ng-repeat="student in students">
            <td>{{ student.enrolment }}</td>
            <td>{{ student.name }}</td>
            <td>{{ student.semester }}</td>
            <td>{{ student.branch }}</td>
        </tr>
    </table>
</div>

<script src="script.js"></script>
</body>
</html>
```

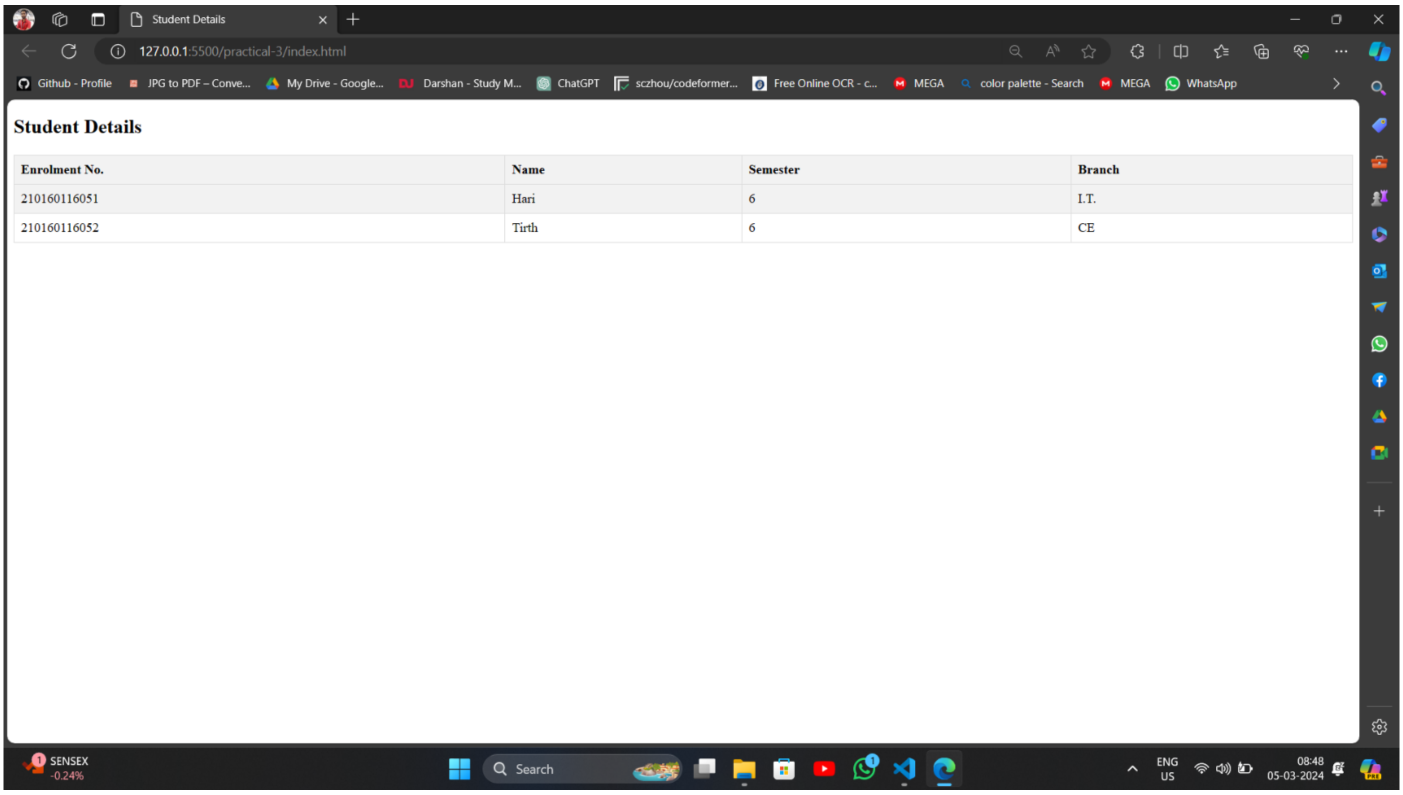
style.css

```
table {
    border-collapse: collapse;
    width: 100%;
}

th, td {
    border: 1px solid #dddddd;
    text-align: left;
    padding: 8px;
}

th {
    background-color: #f2f2f2;
}

tr:nth-child(even) {
    background-color: #f2f2f2;
}
```

script.js

```
const app = angular.module('myApp', []);
app.controller('StudentController', ($scope) => {
    $scope.students = [
        { enrolment: '210160116051', name: 'Hari', semester: '6', branch: 'I.T.' },
        { enrolment: '210160116052', name: 'Tirth', semester: '6', branch: 'CE' },
    ];
});
```

Output



Student Details

Enrolment No.	Name	Semester	Branch
210160116051	Hari	6	I.T.
210160116052	Tirth	6	CE

Quiz:

1. **Write a note on : MVC Architecture.**

MVC (Model-View-Controller) is an architectural design pattern used in software development to structure applications. It divides an application into three interconnected components:

1. **Model:** Represents the data and business logic of the application. It manages the data, responds to requests for information, and interacts with the database or external services.
2. **View:** Represents the presentation layer of the application. It displays information to the user and communicates user inputs to the controller. Views can be web pages, templates, or any user interface element.
3. **Controller:** Acts as an intermediary between the model and the view. It receives input from the user via the view, processes it using the model, and updates the view with the results. Controllers handle user interactions, make decisions, and invoke changes in the model or view.

- Advantages of the MVC architecture include:

1. **Modularity:** Each component has a specific responsibility, making it easier to understand, maintain, and modify the code.
2. **Separation of Concerns:** The separation of components allows developers to focus on individual aspects of the application without impacting others.
3. **Reusability:** Components can be reused in different parts of the application or in other projects.
   Parallel Development: Different developers or teams can work on different components simultaneously, improving productivity and reducing development time.
4. **Testability:** Components can be tested independently, facilitating unit testing, integration testing, and automated testing.

2. **Explain the usage of ng-repeat, ng-controller directives with suitable example.**

1. **<u>ng-repeat Directive:</u>**

The **ng-repeat directive** is used to iterate over a collection of items and generate HTML elements for each item in the collection. It is commonly used to display lists of data dynamically.
Example:

```
<ul>
   <li ng-repeat="item in items">{{ item }}</li> </ul>
```

2. <u>ng-controller Directive:</u>

The **ng-controller directive** is used to define a controller for a specific portion of the HTML document. It binds the scope of the controller to the HTML element and allows the controller to interact with the view.
Example:

```
<div ng-controller="UserController as userCtrl">
   <h1>Welcome, {{ userCtrl.username }}</h1> </div>
```

Assessment :

Understanding of Problem (3 marks)	Implementation of Problem (4 marks)	Presentation and report writing (3 marks)	Total (10 marks)

Experiment No: 4

Aim : Modify Practical 3 and provide a search field to search records on the top ofthe page and also allow user to sort table according to the column values when user clicks on a column using filter.

Index.html

```
<!DOCTYPE html>
<html lang="en" ng-app="myApp">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>Student Details</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>

<div ng-controller="StudentController">
    <h2>Student Details</h2>
    <input type="text" ng-model="search" placeholder="Search...">

    <table>
        <tr>
            <th ng-click="sortBy('enrolment')">Enrolment No.</th>
            <th ng-click="sortBy('name')">Name</th>
            <th ng-click="sortBy('semester')">Semester</th>
            <th ng-click="sortBy('branch')">Branch</th>
        </tr>

        <tr ng-repeat="student in students | filter:search |
orderBy:sortColumn:reverseSort">
            <td>{{ student.enrolment }}</td>
            <td>{{ student.name }}</td>
            <td>{{ student.semester }}</td>
            <td>{{ student.branch }}</td>
        </tr>
    </table>
</div>

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

</body>
</html>
```

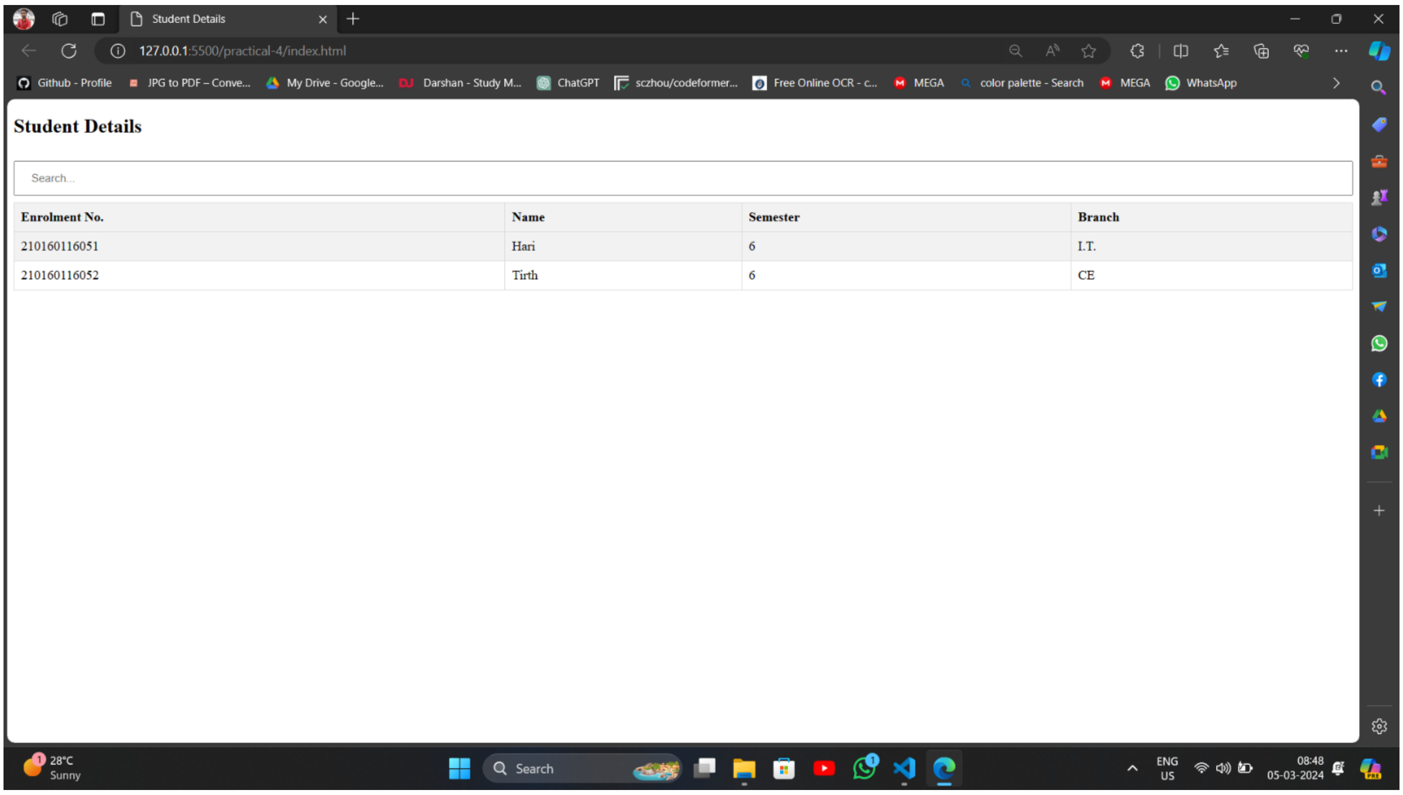
style.css

```css
table {
    border-collapse: collapse;
    width: 100%;
}

th, td {
    border: 1px solid #dddddd;
    text-align: left;
    padding: 8px;
}

th {
    background-color: #f2f2f2;
    cursor: pointer;
}

tr:nth-child(even) {
    background-color: #f2f2f2;
}

input[type=text] {
    width: 100%;
    padding: 12px 20px;
    margin: 8px 0;
    box-sizing: border-box;
}
```

script.js

```javascript
var app = angular.module('myApp', []);
app.controller('StudentController', function($scope) {
    $scope.students = [
        { enrolment: '210160116051', name: 'Hari', semester: '6', branch: 'I.T.' },
        { enrolment: '210160116052', name: 'Tirth', semester: '6', branch: 'CE' },
    ];

    $scope.sortColumn = '';
    $scope.reverseSort = false;

    $scope.sortBy = function(column) {
        if ($scope.sortColumn === column) {
            $scope.reverseSort = !$scope.reverseSort;
        } else {
            $scope.reverseSort = false;
        }
        $scope.sortColumn = column;
    };
});
```

Output


Student Details

Search...

Enrolment No.	Name	Semester	Branch
210160116051	Hari	6	I.T.
210160116052	Tirth	6	CE

Quiz:

1. **Discuss various filters present in Angular JS with suitable examples.**

   AngularJS provides several built-in filters that allow developers to format and manipulate data displayed in the view. Below are some commonly used filters along with examples:

   {{ expression | filterName: parameter }}:

   1. **currency:** Formats a number to a currency format.

   ```
   <p>{{ price | currency }}</p>
   ```

   2. **date:** Formats a date object to a specified format.

   ```
   <p>{{ today | date:'yyyy-MM-dd' }}</p>
   ```

   3. **uppercase/lowercase:** Converts a string to uppercase or lowercase.

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

4. **orderBy:** Orders an array by the value of a specified property.

```
<div ng-repeat="item in items | orderBy:'name'">
{{ item.name }}
</div>
```

5. **filter:** Filters an array based on a specified criteria.

```
<div ng-repeat="user in users | filter:'John'">
   {{ user.name }} </div>
```

6. **limitTo:** Limits the number of items displayed in an array.

```
<ul>
   <li ng-repeat="item in items | limitTo:5">{{ item }}</li> </ul>
```

7. **number:** Formats a number to a specified number of decimal places.

```
<p>{{ number | number:2 }}</p>
```

8. **json** :Formats an object to a JSON string.

```
<p>{{ object | json }}</p>
```

9. **custom filters**: You can also create custom filters by defining them in your AngularJS module.

```
angular.module('myApp').filter('myCustomFilter',
   function() { return function(input) { // filter logic
   };
});
```

2. Write Angular JS code to display numbers in Rupee format with Rupee symbol.

```
  <!DOCTYPE html>
<html lang="en" ng-app="myApp">
<head>
   <meta charset="UTF-8">
   <title>AngularJS Rupee Format</title>
   <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>

<div ng-controller="MyController as ctrl">
   <p>{{ ctrl.price | currency:"₹" }}</p> </div>

<script> angular.module('myA
   pp',[])
   .controller('MyController', function() {
   this.price = 1000; });
</script>

</body>
</html>
```

Assessment :

Understanding of Problem (3 marks)	Implementation of Problem (4 marks)	Presentation and report writing (3 marks)	Total (10 marks)

Experiment No: 5

Aim : Write Angular JS code to read Customer's data in JSON format available in aCustomers.php file using $http service and display the same on a webpage in tabular format.

Index.html

```
<!-- for running this folder need to run in apache server -->
<!-- kali linux : php -S localhost:8000 -->

<!DOCTYPE html>
<html ng-app="myApp">

<head>
    <title>Customer Data</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
    <link rel="stylesheet" href="style.css">
</head>

<body ng-controller="myCtrl">
    <table>
        <tr>
            <th>Name</th>
            <th>Email</th>
            <th>Phone</th>
        </tr>
        <tr ng-repeat="customer in customers">
            <td>{{customer.name}}</td>
            <td>{{customer.email}}</td>
            <td>{{customer.phone}}</td>
        </tr>
    </table>
    <script src="script.js"></script>
</body>

</html>
```

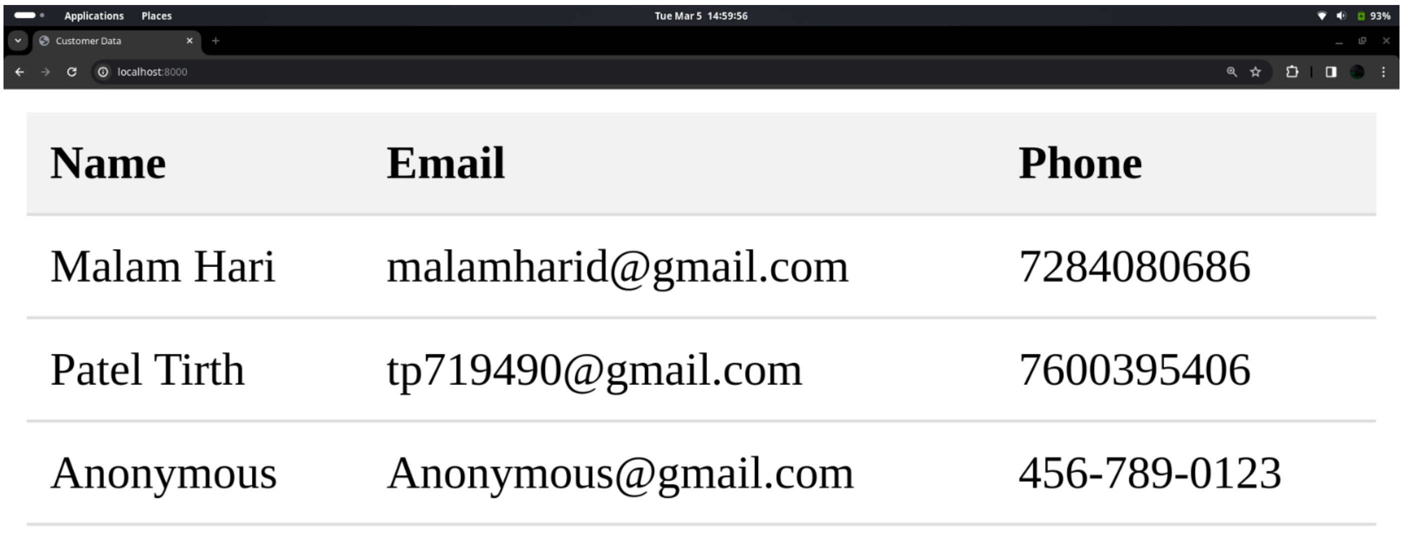
style.css

```
table {
    width: 100%;
    border-collapse: collapse;
}
th, td {
    padding: 8px;
    text-align: left;
    border-bottom: 1px solid #ddd;
}
th {
    background-color: #f2f2f2;
}
```

script.js

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

app.controller('myCtrl', function ($scope, $http) {
    $http.get("Customers.php").then(function (response) {
        $scope.customers = response.data;
    });
});
```

Output



Name	Email	Phone
Malam Hari	malamharid@gmail.com	7284080686
Patel Tirth	tp719490@gmail.com	7600395406
Anonymous	Anonymous@gmail.com	456-789-0123

Quiz:

1. Explain the usage of any five services available in Angular JS with suitable examples.

The usage of five services in AngularJS with suitable examples are:

a. **$http:** Facilitates HTTP communication with remote servers.

```
$http.get('https://api.example.com/data')
.then(function(response) { console.log('Data:', response.data);
})
.catch(function(error)
   {
   console.error('Error
   :', error); });
```

b. **$scope:** Binds controllers to views, enabling interaction.

```
$scope.message = 'Hello, World!';
```

c. **$location:** Provides access to and manipulation of the browser URL.

```
console.log('Current URL:', $location.absUrl());
```

d. **$timeout:** Executes functions after a specified delay.

```
$timeout(function() { console.log('Delayed action executed!'); },
   2000); // Delay of 2000 milliseconds (2 seconds)
```

e. **$filter:** Formats and filters data in AngularJS applications.

```
var formattedDate = $filter('date')(new Date(), 'yyyy-MM-dd'); console.log('Formatted
Date:',          formattedDate);
```

2. Write Angular JS code to display clock on the web page.

```
<!DOCTYPE html>
<html lang="en" ng-app="ClockApp">
<head>
   <meta charset="UTF-8">
   <title>AngularJS Clock</title>
   <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></sc ript>
</head>
<body ng-controller="ClockController">
   <h1>{{ currentTime }}</h1>

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

      app.controller('ClockController', function($scope, $interval) {
         // Update the current time every second
         $interval(function() {
            $scope.currentTime = new Date().toLocaleTimeString(); }, 1000);
      });
   </script>
</body>
</html>
```

Assessment :

Understanding of Problem (3 marks)	Implementation of Problem (4 marks)	Presentation and report writing (3 marks)	Total (10 marks)

Experiment No: 6

Aim : Create an Angular JS application for validation that will accept Email, Username and Password as required fields from the user. It will enable submit button only if all the entered data are valid.

Index.html

```
<!DOCTYPE html>
<html ng-app="validationApp">

<head>
    <title>AngularJS Form Validation</title>
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <div class="container" ng-controller="FormController">
        <form name="myForm" ng-submit="submitForm()" novalidate>
            <div class="form-group">
                <label>Email:</label>
                <input type="email" name="email" ng-model="formData.email" required>
                <span class="error" ng-show="myForm.email.$dirty &&
myForm.email.$invalid">Invalid Email</span>
            </div>
            <div class="form-group">
                <label>Username:</label>
                <input type="text" name="username" ng-model="formData.username" required>
                <span class="error" ng-show="myForm.username.$dirty &&
myForm.username.$invalid">Username is
                    required</span>
            </div>
            <div class="form-group">
                <label>Password:</label>
                <input type="password" name="password" ng-model="formData.password"
required>
                <span class="error" ng-show="myForm.password.$dirty &&
myForm.password.$invalid">Password is
                    required</span>
            </div>
            <button type="submit" class="submit-btn" ng-
disabled="myForm.$invalid">Submit</button>
        </form>
    </div>

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

```
</html>
```

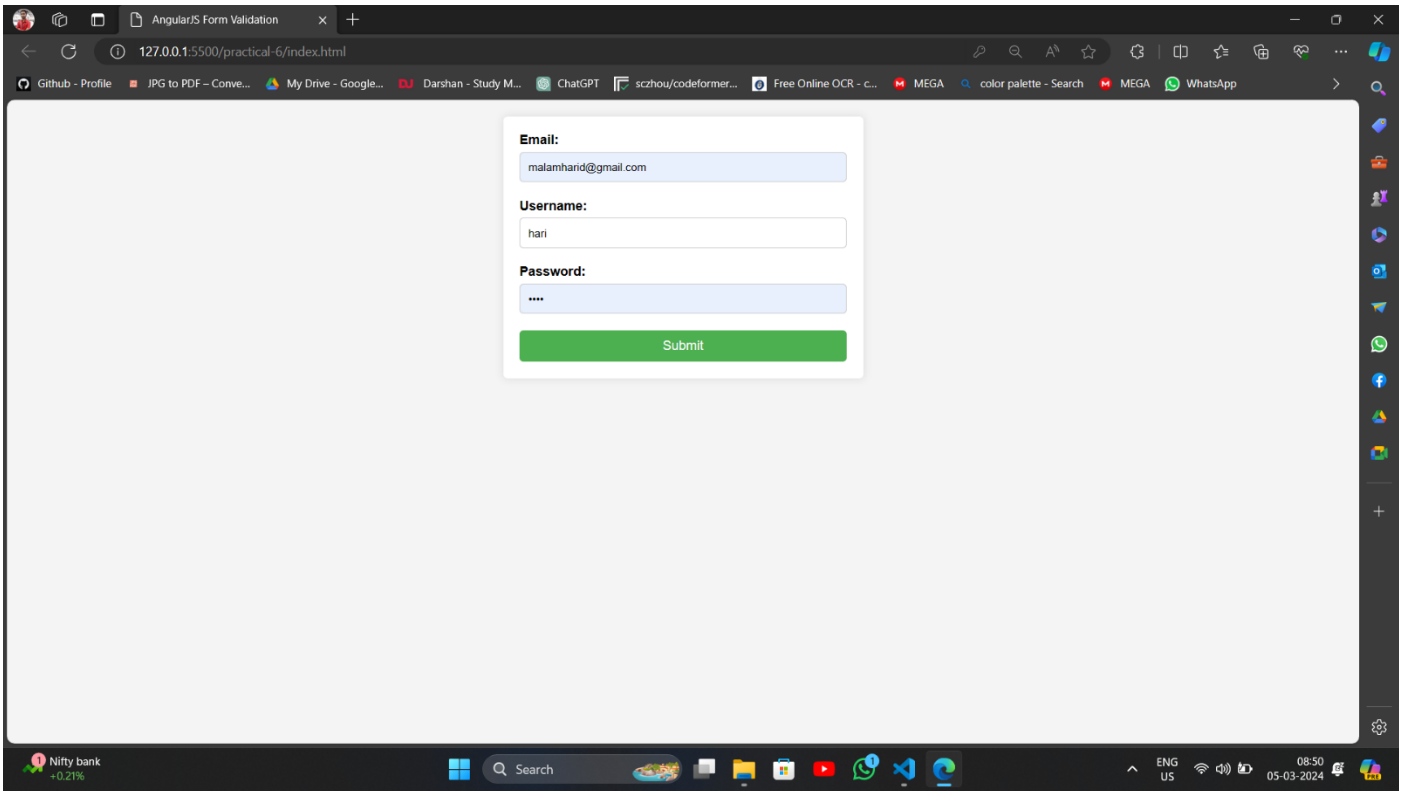
style.css

```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 20px;
}

.container {
    max-width: 400px;
    margin: 0 auto;
    background-color: #fff;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

.form-group {
    margin-bottom: 20px;
}

.form-group label {
    display: block;
    margin-bottom: 5px;
    font-weight: bold;
}

.form-group input {
    width: 100%;
    padding: 10px;
    border-radius: 5px;
    border: 1px solid #ccc;
    box-sizing: border-box;
}

.error {
    color: red;
}

.submit-btn {
    width: 100%;
    padding: 10px;
    border-radius: 5px;
    border: none;
    background-color: #4CAF50;
    color: #fff;
    cursor: pointer;
    font-size: 16px;
}

.submit-btn:hover {
    background-color: #45a049;
}
.submit-btn {
    width: 100%;
    padding: 10px;
    border-radius: 5px;
    border: none;
    background-color: #4CAF50;
    color: #fff;
    cursor: pointer;
    font-size: 16px;
}
.submit-btn:hover {
    background-color: #45a049;
}
.submit-btn[disabled] {
    background-color: #ccc;
    cursor: not-allowed;
}
```

script.js

```
angular.module('validationApp', [])
.controller('FormController', function ($scope) {
    $scope.submitForm = function () {
        if ($scope.myForm.$valid) {
            console.log('Form submitted successfully!');
        }
    };
});
```

Output


AngularJS Form Validation
127.0.0.1:5500/practical-6/index.html
Submit

Email:
malamharid@gmail.com

Username:
hari

Password:
••••

Quiz:

1. **Briefly discuss the usage of all the Input states and Form states with suitable examples.**

In AngularJS, input states and form states are important concepts used to manage user input and form validation.

- **Input States:**

i. **Untouched**: The input field has not been interacted with yet.

ii. **Touched**: The input field has been interacted with by the user.

iii. **Pristine:** The input field has not been modified by the user.

iv. **Dirty:** The input field has been modified by the user.

v. **Valid**: The input field data is valid according to the defined validation rules.

vi. **Invalid:** The input field data is invalid according to the defined validation rules.

- Form States:

i. **$untouched:** The form has not been interacted with yet.

ii. **$touched:** The form has been interacted with by the user.

iii. **$pristine:** None of the form fields have been modified by the user.

iv. **$dirty:** At least one of the form fields has been modified by the user.

v. **$valid:** All the form fields have valid data according to the defined validation rules.

vi. **$invalid:** At least one of the form fields has invalid data according to the defined validation rules

.

Example:

```
<form name="myForm" ng-app>
<input type="text" name="username" ng-model="user.username" required>

<div ng-if="myForm.username.$untouched">Username is untouched.</div>
<div ng-if="myForm.username.$touched">Username is touched.</div>
<div ng-if="myForm.username.$pristine">Username is pristine.</div>
<div ng-if="myForm.username.$dirty">Username is dirty.</div>

<div ng-if="myForm.username.$valid">Username is valid.</div>
<div ng-if="myForm.username.$invalid">Username is invalid.</div>

<button type="submit" ng-disabled="myForm.$invalid">Submit</button> </form>
```

2. Discuss various CSS classes present in Angular JS used for these validation states.

In AngularJS, there are several CSS classes that are dynamically applied to form elements to represent their validation states. These classes help in styling form elements based on their validity and interaction states. Some of the commonly used CSS classes:

1. **ng-valid:** Applied to form elements when their input data is valid.
2. **ng-invalid:** Applied to form elements when their input data is invalid.
3. **ng-pristine:** Applied to form elements when they have not been interacted with by the user.
4. **ng-dirty:** Applied to form elements when they have been interacted with and their value has been changed.
5. **ng-touched:** Applied to form elements when they have been interacted with by the user.
6. **ng-untouched:** Applied to form elements when they have not been interacted with by the user.

Assessment :

Understanding of Problem (3 marks)	Implementation of Problem (4 marks)	Presentation and report writing (3 marks)	Total (10 marks)

Experiment No: 7

Aim : Create an example demonstrating the concept of Angular JS Routing.

Index.html

```
<!DOCTYPE html>
<html ng-app="myApp">

<head>
    <title>AngularJS Routing Example</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular-
route.min.js"></script>
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <div class="container">
        <h1>AngularJS Routing Example</h1>
        <ul class="navigation">
            <li><a href="#/">Home</a></li>
            <li><a href="#/about">About</a></li>
            <li><a href="#/contact">Contact</a></li>
        </ul>
        <div ng-view></div>
    </div>
    <script src="script.js"></script>
</body>

</html>
```

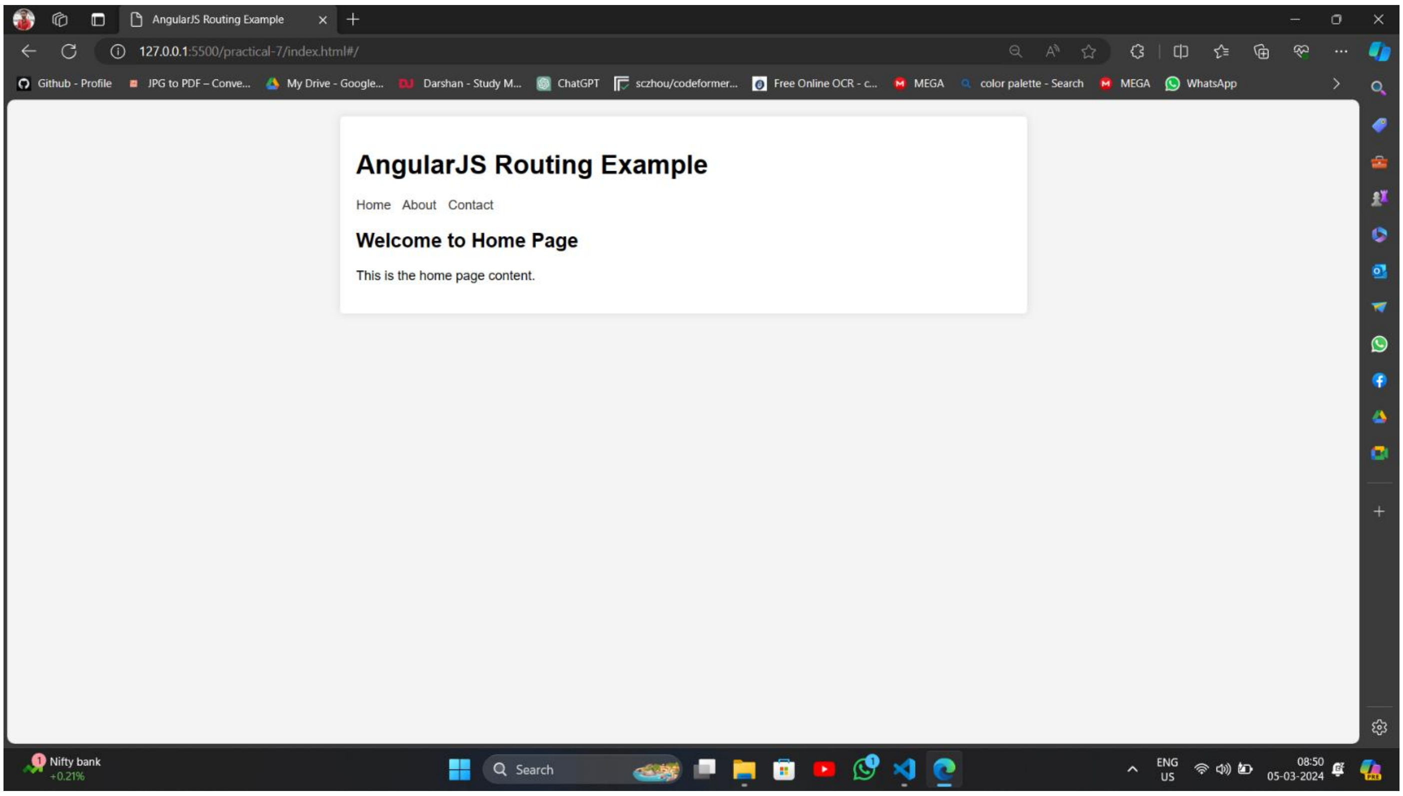
style.css

```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 20px;
}

.container {
    max-width: 800px;
    margin: 0 auto;
    background-color: #fff;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

.navigation {
    list-style-type: none;
    padding: 0;
}

.navigation li {
    display: inline;
    margin-right: 10px;
}

.navigation li a {
    text-decoration: none;
    color: #333;
}

.navigation li a:hover {
    color: #007bff;
    text-decoration: underline;
}

.ng-view {
    margin-top: 20px;
}
```

script.js

```
var app = angular.module('myApp', ['ngRoute']);

app.config(function ($routeProvider, $locationProvider) {
    $locationProvider.hashPrefix('');
    $routeProvider
        .when('/', {
            templateUrl: 'views/home.html'
        })
        .when('/about', {
            templateUrl: 'views/about.html'
        })
        .when('/contact', {
            templateUrl: 'views/contact.html'
        })
        .otherwise({
            redirectTo: '/'
        });
});
```

Output


AngularJS Routing Example
127.0.0.1:5500/practical-7/index.html#/

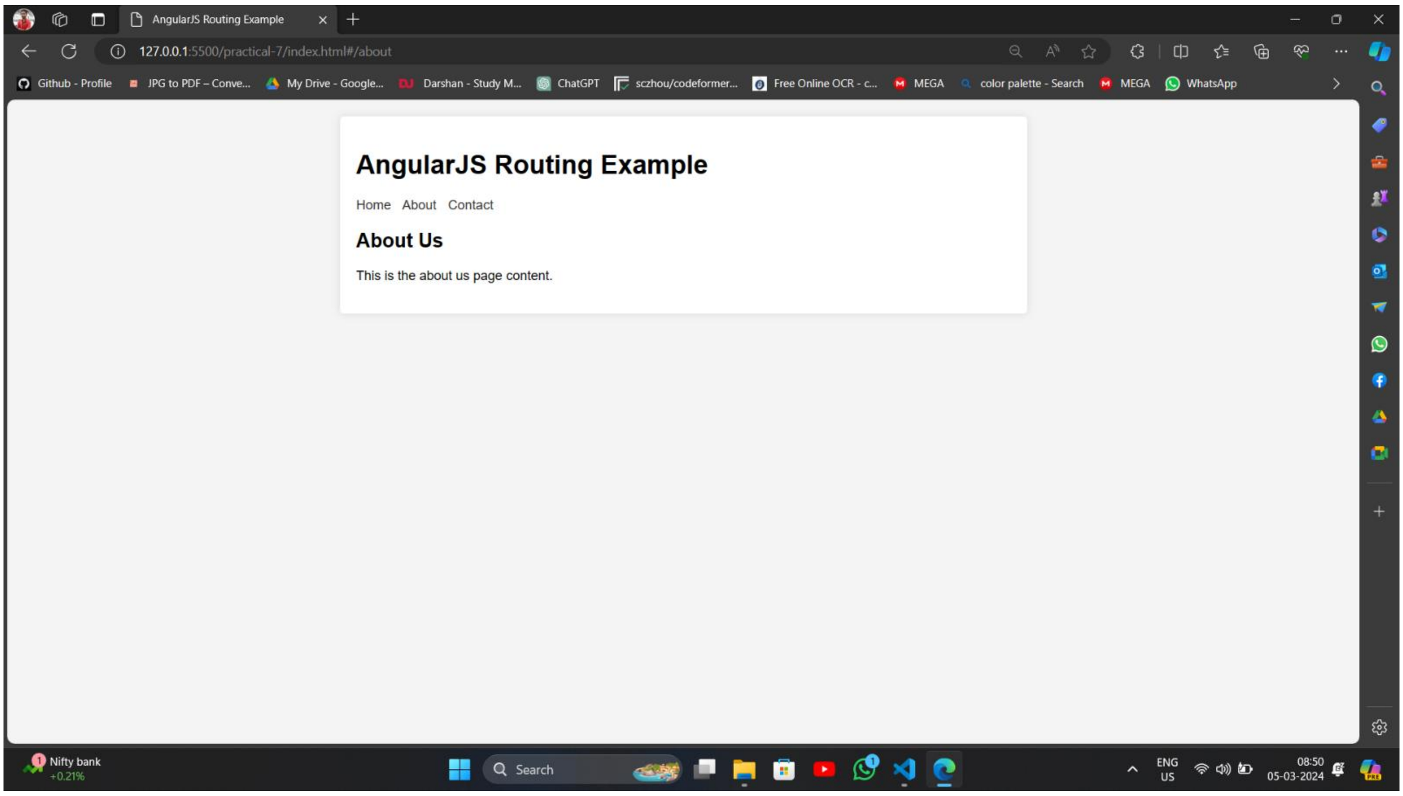
AngularJS Routing Example

Home About Contact

Welcome to Home Page

This is the home page content.

A3-210160116051

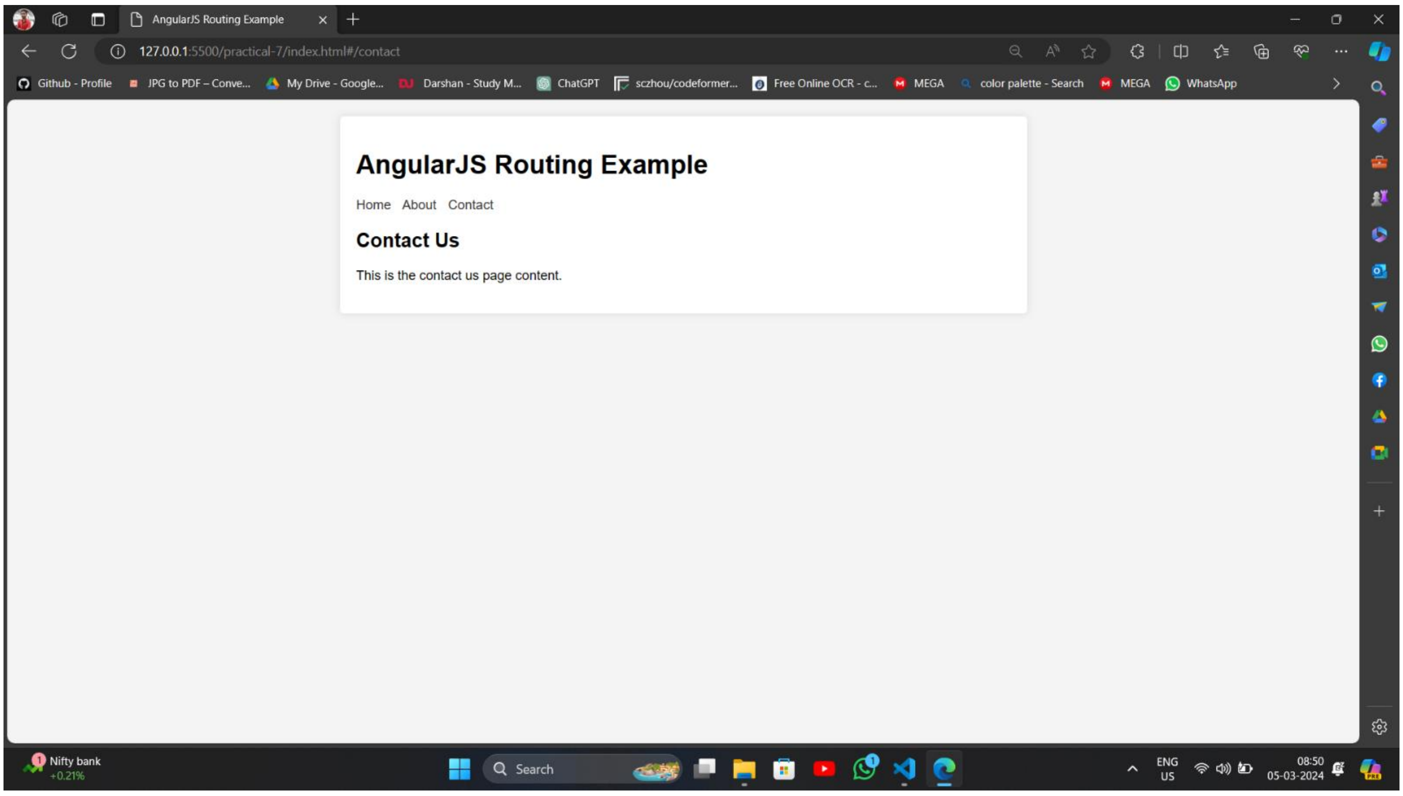

127.0.0.1:5500/practical-7/index.html#/about

AngularJS Routing Example

Home About Contact

About Us

This is the about us page content.


127.0.0.1:5500/practical-7/index.html#/contact

AngularJS Routing Example

Home About Contact

Contact Us

This is the contact us page content.

Quiz:

1. Write a note on : Routing in AngularJS

   Routing in AngularJS: Creating a Seamless Single-Page Application

   - Routing is a crucial concept in building Single-Page Applications (SPAs) with AngularJS. It allows you to manage navigation between different views within the same application without full page reloads, enhancing user experience and improving application performance.

   Key Points:

   - **ngRoute Module:** Enables routing functionality by providing services like **$routeProvider** for
     configuring routes.
   - **Route Configuration:** Use **$routeProvider** to define mappings between URLs and views. Each route typically specifies:
     - **URL Pattern:** Defines the URL path that triggers the route. You can use wildcards (***)** and named parameters (**/:paramName**) for flexibility.
     - **Template:** The HTML template associated with the route, specifying the content to be displayed.
     - **Controller:** The JavaScript controller responsible for handling the view's logic and data
   - **Hashbang URLs:** By default, AngularJS uses hash-based URLs (e.g.,**http://example.com/#/about**). You can configure server-side routing for cleaner URLs.
   - **Navigation:** Use **ng-href** directive in templates or programmatic navigation using **$location** service to navigate between views.

   Benefits:

   - **Improved User Experience:** Seamless navigation without page reloads feels more responsive and fluid.
   - **SPA Structure:** Enables building complex applications with multiple views, enhancing maintainability and organization.
   - **Deep Linking:** Allows users to bookmark specific views and share them with others using URLs.

2. State the usefulness of ng-view directive. Explain various ways to use it.

   The ng-view directive in AngularJS is essential for implementing routing in Single-Page Applications. The reason why it's useful and how to use it in various ways:

   Usefulness:

   - **Dynamic Content Placeholder:** The **ng-view** directive marks a spot within your HTML where the content of different views (templates) will be dynamically inserted based on the current route.
   - **Central to Routing:** It works in tandem with the **$routeProvider** service to manage the display of views as the user navigates different sections of your application.

   Ways to Use ng-view:

1. Basic Single View Routing

```
<body ng-app="myApp" ng-controller="mainCtrl">
<h1>Welcome to My App!</h1>
<div ng-view></div>
<script>
var app = angular.module('myApp', ['ngRoute']);

app.config(function($routeProvider) {
$routeProvider .when('/', { templateUrl:
'home.html', controller: 'homeCtrl' })
.otherwise({ redirectTo: '/'
});
});
</script>
</body>
```

2. Using Nested Views

```
<div ng-view></div>

<h3>Profile Details</h3>
<div ng-view></div>
```

Assessment :

Understanding of Problem (3 marks)	Implementation of Problem (4 marks)	Presentation and report writing (3 marks)	Total (10 marks)