**PART A**

(Part A: TO BE REFFERED BY STUDENTS)

**Experiment No. 08**

**A.1 AIM:**

Implement Table and Form validation through Angular JS by using directive, controller, expression, modules etc.

**A.2 Pre requisite:**

HTML, CSS, JavaScript

**A.3 Outcome:**

After successful completion of this experiment students will be able to:

1. Understand and implement directives, expressions, controllers, scope in Angular JS.
2. Understand the principles behind data binding in Angular JS.

**A.4 Theory:**

**Tables**

Table data is generally repeatable. The ng-repeat directive can be used to draw table easily. The following example shows the use of ng-repeat directive to draw a table –

<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>

**Example**

The following example shows the use of all the above-mentioned directives.

<html>

<head>

<title>Angular JS Table</title>

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

<script>

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

mainApp.controller('studentController', function($scope) {

$scope.student = {

firstName: "Mahesh",

lastName: "Parashar",

fees:500,

subjects:[

{name:'Physics',marks:70},

{name:'Chemistry',marks:80},

{name:'Math',marks:65},

{name:'English',marks:75},

{name:'Hindi',marks:67}

],

fullName: function() {

var studentObject;

studentObject = $scope.student;

return studentObject.firstName + " " + studentObject.lastName;

}

};

});

</script>

</body>

</html>

**Form Validation:** AngularJS performs form validation on the client side. AngularJS monitors the state of the form and input fields (input, text-area, select), and notify the user about the current state. AngularJS also holds information about whether the input fields have been touched, modified, or not. Form input fields have the following states:

* $untouched: It shows that field has not been touched yet.
* $touched: It shows that field has been touched.
* $pristine: It represents that the field has not been modified yet.
* $dirty: It illustrates that the field has been modified.
* $invalid: It specifies that the field content is not valid.
* $valid: It specifies that the field content is valid.

**AngularJS includes the following validation directives.**

* ng-required: Sets required attribute on an input field.
* ng-minlength: Sets minlength attribute on an input field.
* ng-maxlength: Sets maxlength attribute on an input field. Setting the attribute to a negative or non-numeric value, allows view values of any length.
* ng-pattern: Sets pattern validation error key if the ngModel value does not match the specified RegEx expression.

These all are the properties of the input field which can be either true or false. Forms have the following states:

* $pristine: It represents that the fields have not been modified yet.
* $dirty: It illustrates that one or more fields have been modified.
* $invalid: It specifies that the form content is not valid.
* $valid: It specifies that the form content is valid.
* $submitted: It specifies that the form is submitted.

These all are the properties of the form which can be either true or false. These states can be used to show meaningful messages to the user.

**ng-click**

Reset data of a form using on-click directive of a button.

<input name = "firstname" type = "text" ng-model = "firstName" required>

<input name = "lastname" type = "text" ng-model = "lastName" required>

<input name = "email" type = "email" ng-model = "email" required>

<button ng-click = "reset()">Reset</button>

<script>

function studentController($scope) {

$scope.reset = function() {

$scope.firstName = "Mahesh";

$scope.lastName = "Parashar";

$scope.email = "MaheshParashar@tutorialspoint.com";

}

$scope.reset();

}

</script>

**Example**

The following example will showcase all the above-mentioned directives.

**testAngularJS.htm**

<html>

<head>

<title>Angular JS Forms</title>

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

<form name = "studentForm" novalidate>

<table border = "0">

<tr>

<td>Enter first name:</td>

<td><input name = "firstname" type = "text" ng-model = "firstName" required>

<span style = "color:red" ng-show = "studentForm.firstname.$dirty && studentForm.firstname.$invalid">

<span ng-show = "studentForm.firstname.$error.required">First Name is required.</span>

</span>

</td>

</tr>

<tr>

<td>Enter last name: </td>

<td><input name = "lastname" type = "text" ng-model = "lastName" required>

<span style = "color:red" ng-show = "studentForm.lastname.$dirty && studentForm.lastname.$invalid">

<span ng-show = "studentForm.lastname.$error.required">Last Name is required.</span>

</span>

</td>

</tr>

<tr>

<td>Email: </td><td><input name = "email" type = "email" ng-model = "email" length = "100" required>

<span style = "color:red" ng-show = "studentForm.email.$dirty && studentForm.email.$invalid">

<span ng-show = "studentForm.email.$error.required">Email is required.</span>

<span ng-show = "studentForm.email.$error.email">Invalid email address.</span>

</span>

</td>

</tr>

<tr>

<td>

<button ng-click = "reset()">Reset</button>

</td>

<td>

<button ng-disabled = "studentForm.firstname.$dirty &&

studentForm.firstname.$invalid || studentForm.lastname.$dirty &&

studentForm.lastname.$invalid || studentForm.email.$dirty &&

studentForm.email.$invalid" ng-click="submit()">Submit</button>

</td>

</tr>

</table>

</form>

</div>

<script>

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

mainApp.controller('studentController', function($scope) {

$scope.reset = function() {

$scope.firstName = "Mahesh";

$scope.lastName = "Parashar";

$scope.email = "MaheshParashar@tutorialspoint.com";

}

$scope.reset();

});

</script>

</body>

</html>

**A.5 Procedure/Task:**

1. Create an Angular JS application to implement Angular JS table to calculate total pay of order placed.

2. Design client-side validation for student registration form using angular JS. The angular JS validation should apply at input control level and form control level with following validations.

* + User Name is RequiredField with Maxlength of 15-character Minlength of 05 character.
  + Email is valid and can’t be blank
  + Age cannot be less than 18
  + If form is not modified the background color is aqua else if one of field gets modified, the background color is lawn green.
  + The submit button is disabled if form data is invalid.

3. To implement registration page form and form validation in Angular JS.

Create a form with the fields given below and validation is to be done for

1. Must not be empty username, start with capital letter, only have alphabets.
2. Must not be empty password, 5 and 15 characters long, must be alphanumeric.
3. Confirm password.
4. Email field and its basic validation.
5. Must not be empty checkbox. When the page loads, one checkbox must always be checked.
6. Atleast one Radio button must be selected.
7. Dynamic dropdown from controller. (i.e dropdown items must be populated from within the controller). Atleast one dropdown item must be selected.
8. Phone Number- 10 digits
9. Pan Card Validation

3. Prepare the document. Save and close the file and name it as **EXP08\_Name of Student**

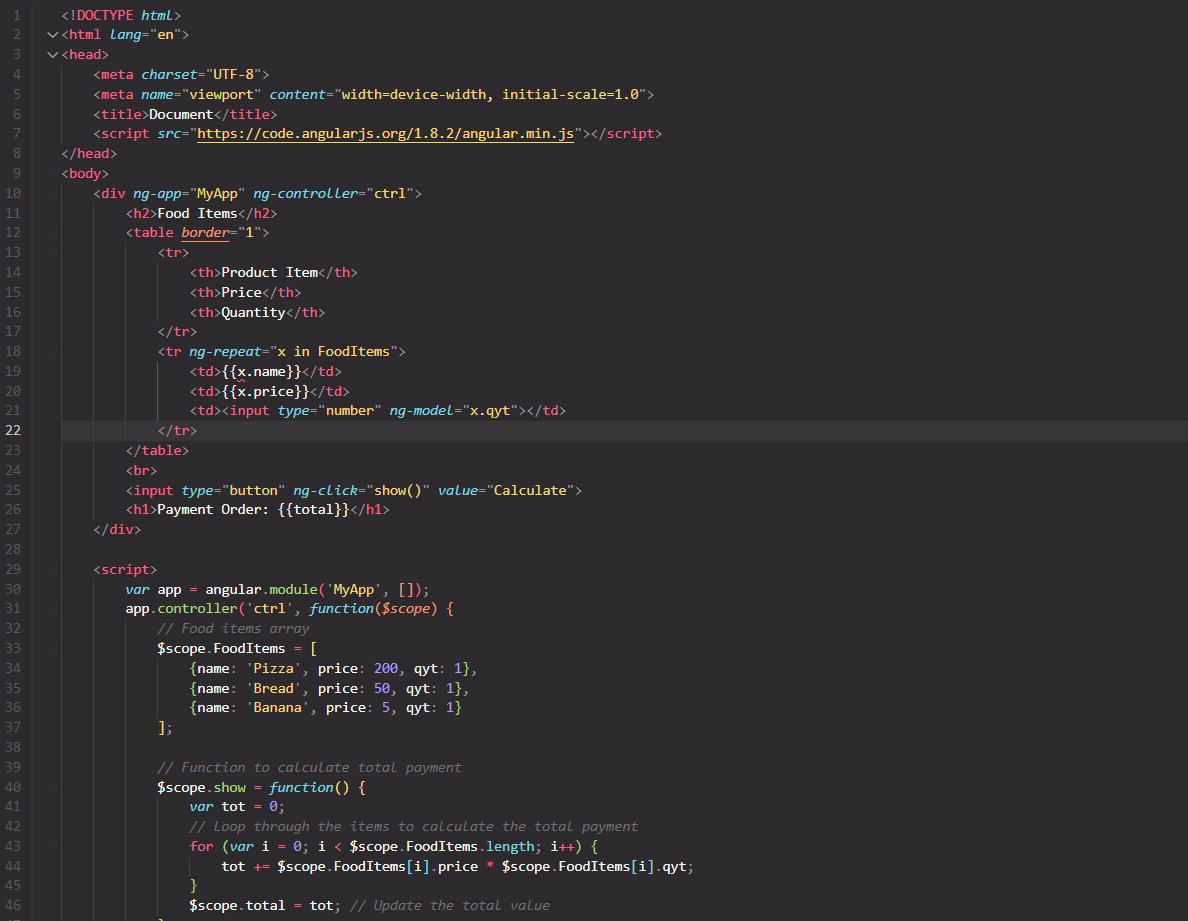
**PART B**

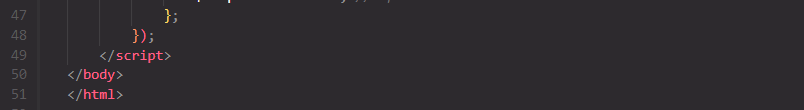
(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)

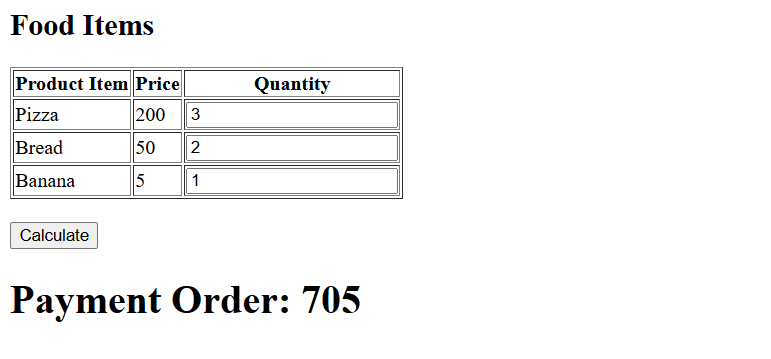
|  |  |
| --- | --- |
| Roll No. :E258 | Name: Kundan S. Patil |
| Class :BTech CS | Batch :A3 |
| Date of Experiment :18-03-2025 | Date/Time of Submission :19-03-2025 |
| Grade : |  |

**B.1 Code:**

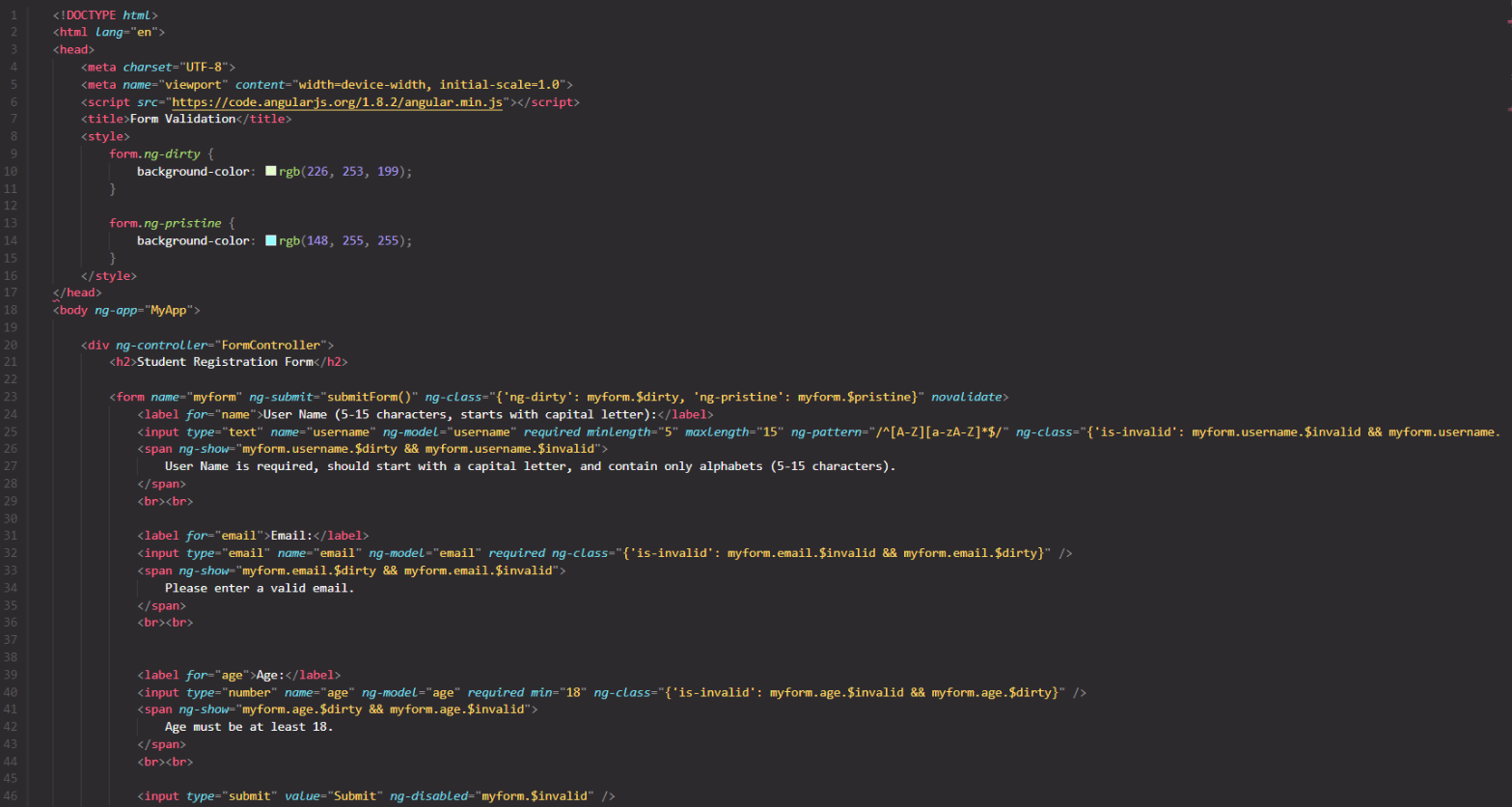
****

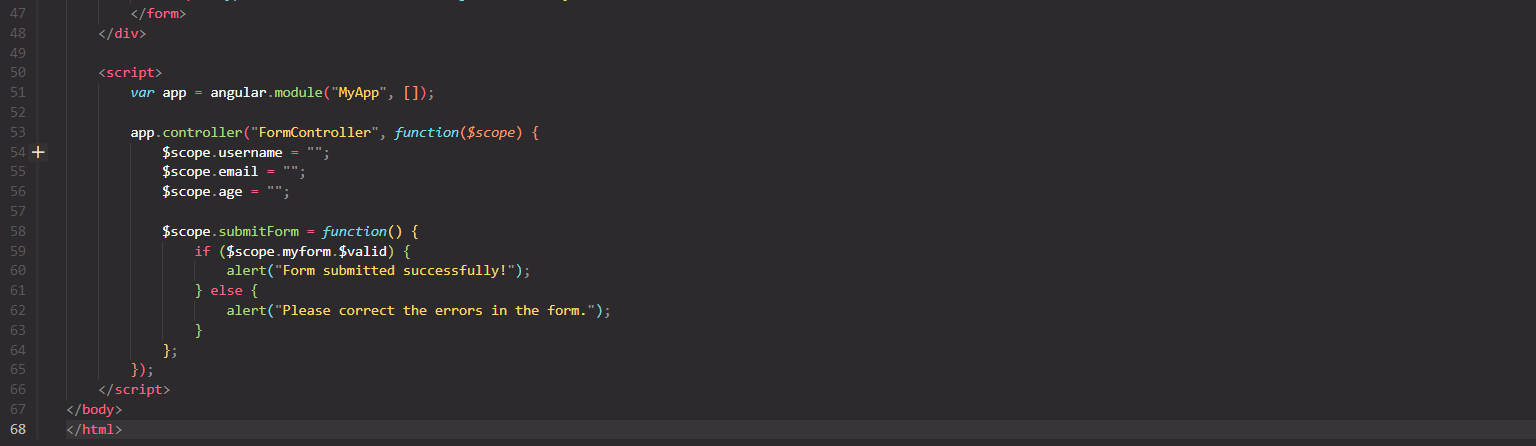
****

**B.2 Output:**

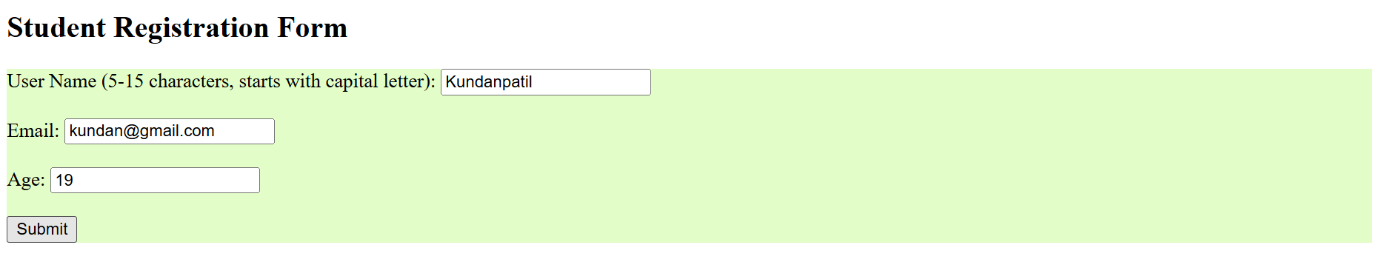
****

**B.1 Code:**

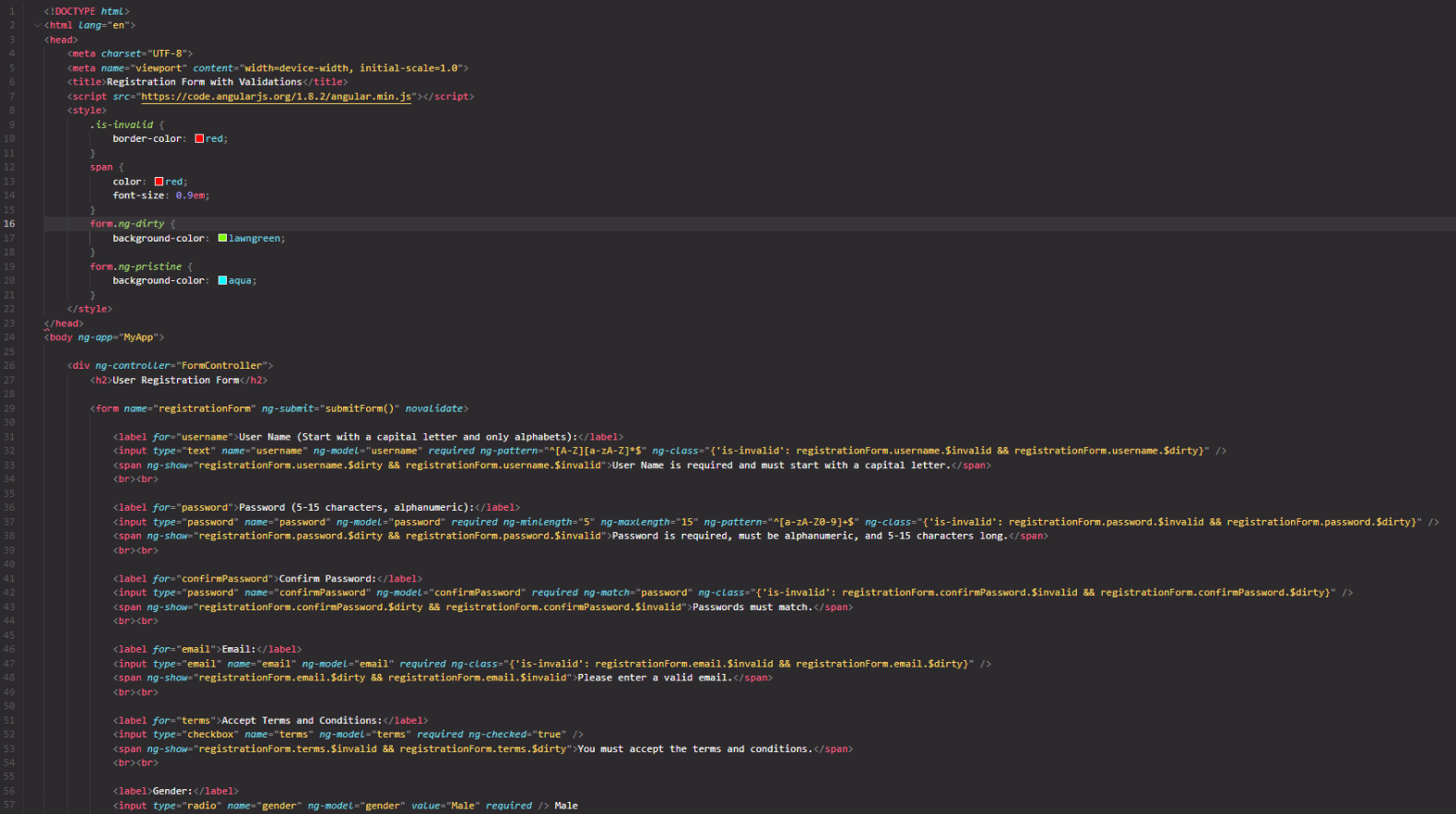
****

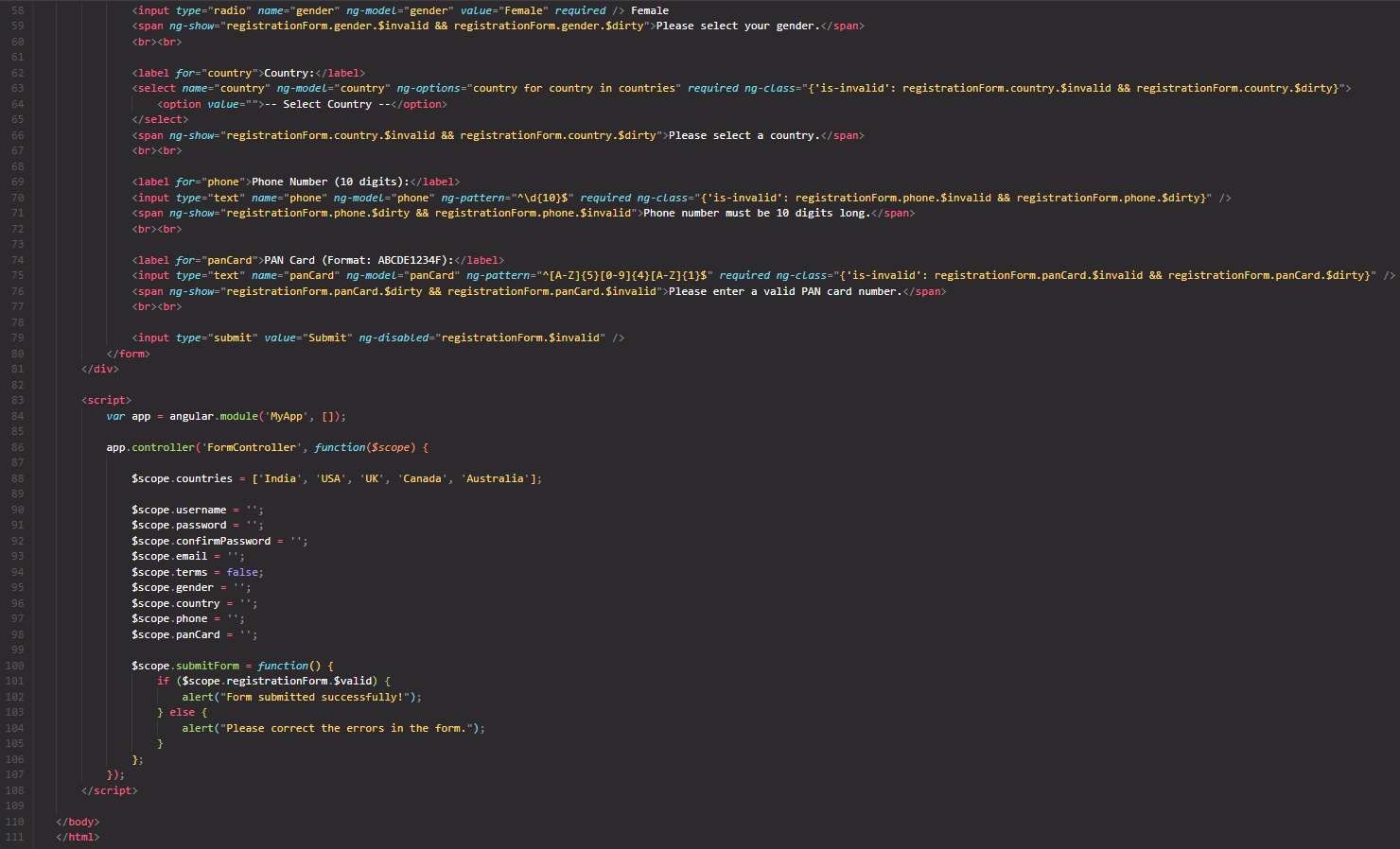
****

**B.2 Output:**

****

**B.1 Code:**

****

****

**B.2 Output:**

****

**B.3 Conclusion:**

In the context of all the AngularJS implementations above, we have explored how to build a **student registration form** with complex validations, a **food order calculator**, and **dynamic form handling**. The key objective was to ensure that forms collect accurate data while providing users with an interactive and responsive interface.

### **Key Points:**

1. **Form Validation with AngularJS:**
   * We utilized AngularJS’s built-in directives such as ng-model, ng-required, ng-pattern, ng-minlength, and ng-maxlength for both **field-level** and **form-level validation**. This allows for real-time feedback, ensuring that users input valid data before submitting the form.
2. **Dynamic Content Handling:**
   * For dynamic dropdowns (like the country list) and calculation-based outputs (such as the total price of items), AngularJS’s **two-way data binding** and **directives** such as ng-repeat and ng-options made it easy to update and reflect data without needing manual DOM manipulation.
3. **User Feedback and Experience:**
   * AngularJS provided **real-time validation feedback**, changing the form’s background color and displaying error messages when data was invalid. The **submit button** was disabled until all required fields passed validation, enhancing user experience and preventing incorrect data submission.
4. **Efficient Form Submission and Data Integrity:**
   * With **dynamic validation rules** and **clear error messages**, users were guided step-by-step in correctly filling out the form. The use of **ng-disabled** for the submit button ensured that the form could not be submitted until all validation checks were passed.

**B.3 Observations and Learning:**

### Learnings:

* **AngularJS Makes Form Validation Easy**: The use of AngularJS’s directives simplifies form validation and handling, making it easier to manage complex forms with multiple validation rules.
* **Two-Way Data Binding is Powerful**: AngularJS’s two-way data binding allows dynamic updates to both the form and the model, which ensures that the UI and underlying data are always synchronized.
* **User Experience is Key**: Real-time feedback, color changes, and error messages ensure that users can easily fill out the form, reducing frustration and error rates.

### Observations:

* **Maintainability**: The modular approach with AngularJS makes the code easy to maintain and extend for future features.
* **Dynamic Forms**: AngularJS’s flexibility allows for dynamically generated content and easy management of forms with different input types.
* **Validation Flexibility**: With AngularJS, handling complex validations (like custom email formats, phone number lengths, etc.) was straightforward.