ASSIGNMENT 6

AIM: Implement student registration form in Angular JS

Theory:

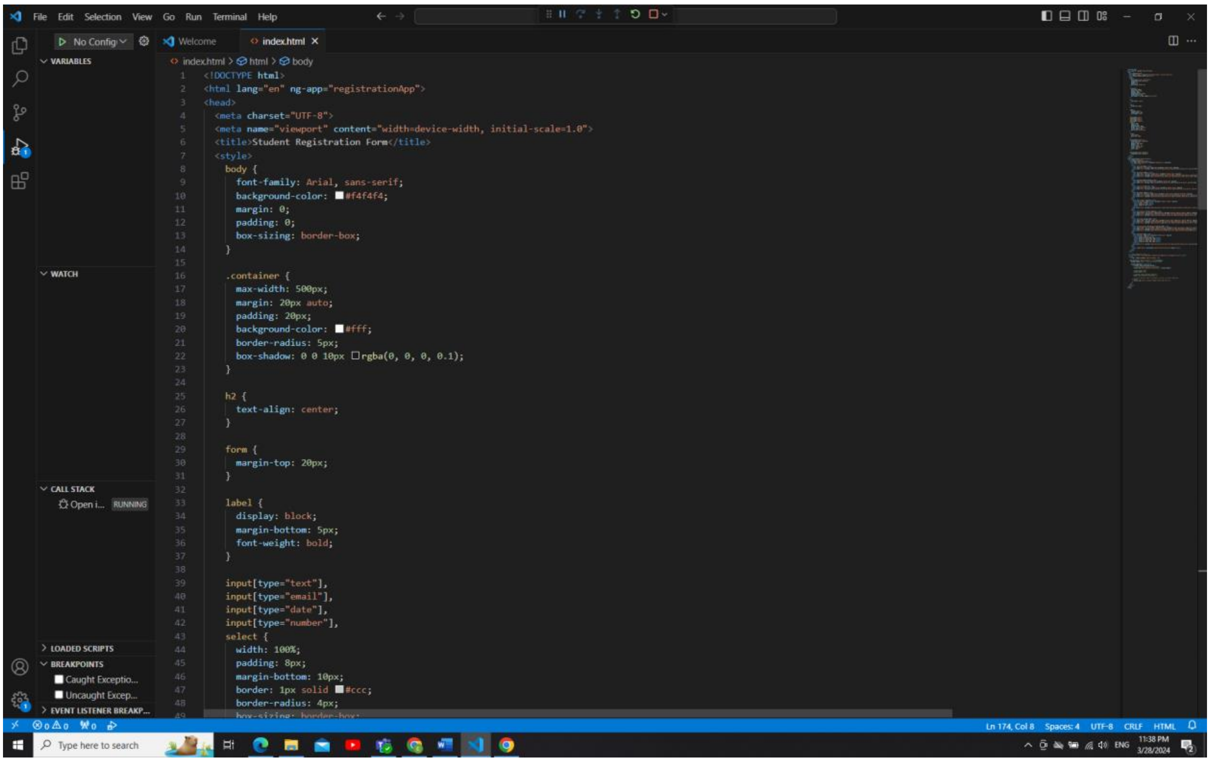
Form validation in Angular allows you to ensure that user input meets certain requirements before it is submitted. Angular provides both template-driven and reactive approaches to form validation. Here's some theory about form validation in Angular:

1. **Template-Driven Forms**:
    - In template-driven forms, validation is performed using directives such as **ngModel** and template reference variables.
    - Angular provides several built-in validators like **required**, **minLength**, **maxLength**, **pattern**, etc.
    - You can also create custom validators by writing your own functions.
2. **Reactive Forms**:
    - Reactive forms are model-driven and defined programmatically using TypeScript.
    - Validators are applied directly to form controls using functions provided by Angular's **Validators** class.
    - Reactive forms offer more flexibility and control compared to template-driven forms, especially for complex validation scenarios.
3. **Validation Directives**:
    - Angular provides various validation directives such as **ngModel**, **ngModelGroup**, **ngForm**, and **formControl** for handling form validation.
    - These directives enable you to apply validation rules and display error messages based on the state of form controls.
4. **Displaying Validation Errors**:
    - Angular provides several ways to display validation errors to users.
    - You can use Angular's built-in directives like **ngIf** and **ngClass** to conditionally show error messages based on the validity state of form controls.
    - You can also customize the appearance of error messages using CSS and Angular's template syntax.
5. **Cross-Field Validation**:
    - Angular allows you to perform validation that depends on the values of multiple form controls.
    - You can implement cross-field validation by creating custom validators or by using built-in validators like **Validators** class methods.
6. **Asynchronous Validation**:

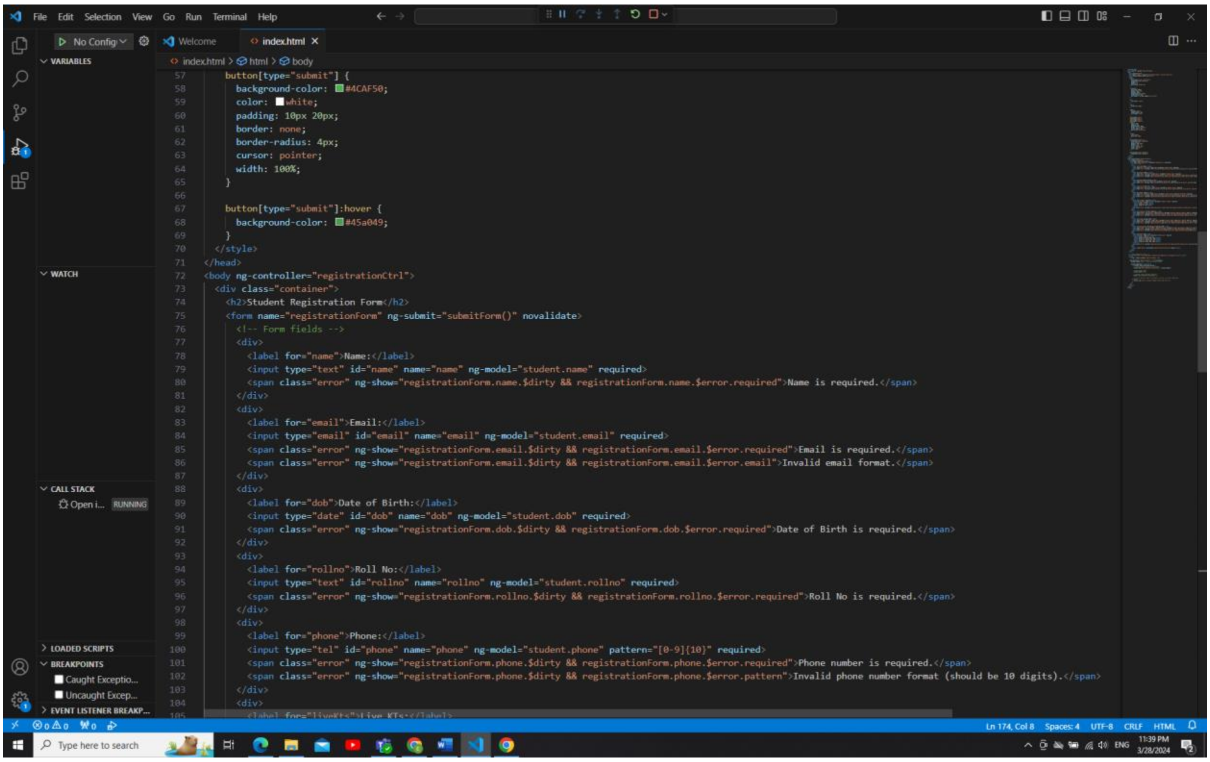
- In addition to synchronous validation, Angular supports asynchronous validation for cases where validation rules depend on data fetched from external sources.
- Asynchronous validators return **Observable** or **Promise** objects and are applied using the **AsyncValidatorFn** function.

7. **Handling Form Submission**:

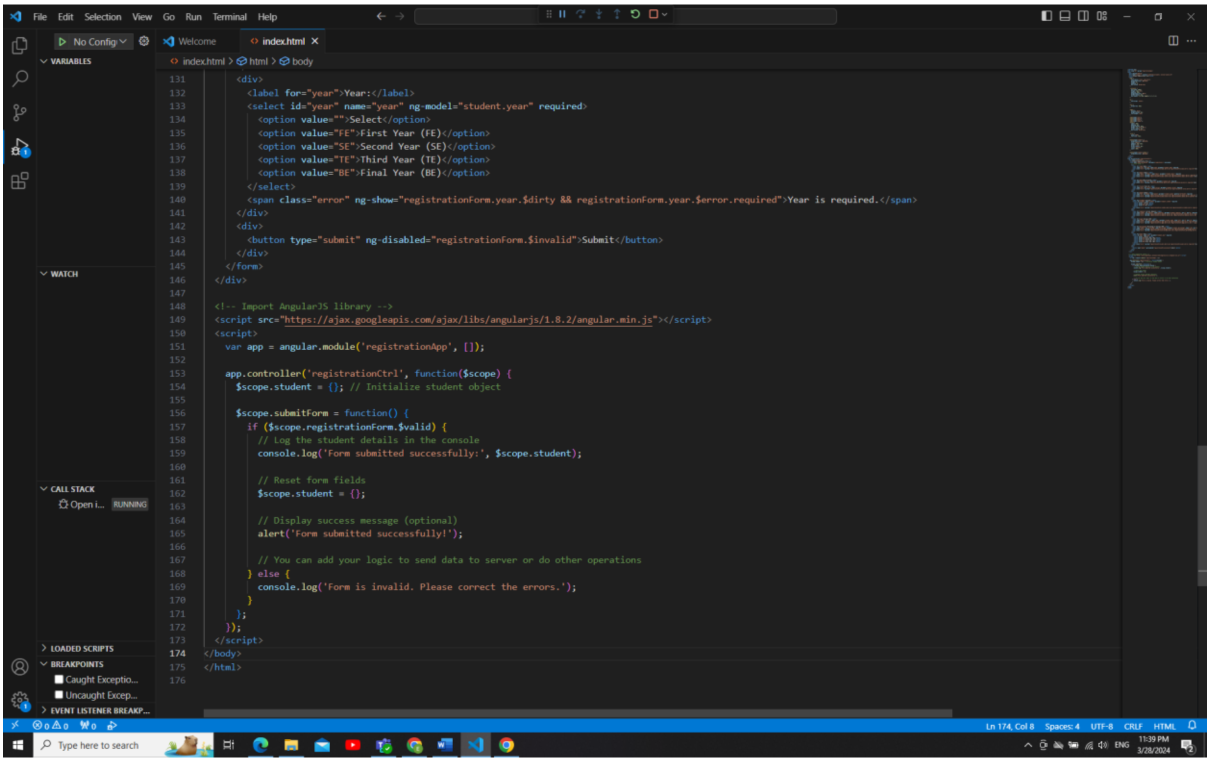
- Once form validation is successful, you can handle form submission by calling methods or functions defined in your component.
- You can prevent the default form submission behavior by calling **event.preventDefault()** in the submit handler function.



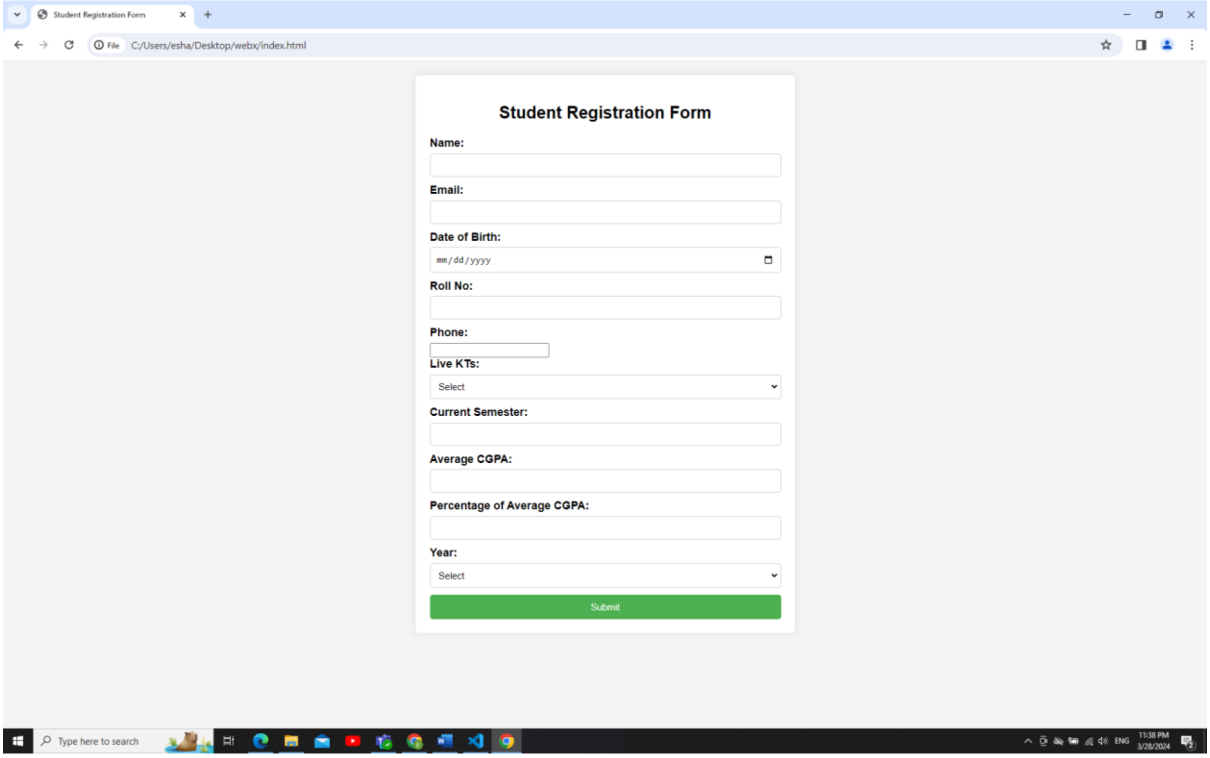
```
<!DOCTYPE html>
<html lang="en" ng-app="registrationApp">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Student Registration Form</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    .container {
      max-width: 500px;
      margin: 20px auto;
      padding: 20px;
      background-color: #fff;
      border-radius: 5px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }

    h2 {
      text-align: center;
    }

    form {
      margin-top: 20px;
    }

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

    input[type="text"],
    input[type="email"],
    input[type="date"],
    input[type="number"],
    select {
      width: 100%;
      padding: 8px;
      margin-bottom: 10px;
      border: 1px solid #ccc;
      border-radius: 4px;
      box-sizing: border-box;
```



```
    button[type="submit"] {
      background-color: #4CAF50;
      color: white;
      padding: 10px 20px;
      border: none;
      border-radius: 4px;
      cursor: pointer;
      width: 100%;
    }

    button[type="submit"]:hover {
      background-color: #45a049;
    }
  </style>
</head>
<body ng-controller="registrationCtrl">
  <div class="container">
    <h2>Student Registration Form</h2>
    <form name="registrationForm" ng-submit="submitForm()" novalidate>
      <!-- Form fields -->
      <div>
        <label for="name">Name:</label>
        <input type="text" id="name" name="name" ng-model="student.name" required>
        <span class="error" ng-show="registrationForm.name.$dirty && registrationForm.name.$error.required">Name is required.</span>
      </div>
      <div>
        <label for="email">Email:</label>
        <input type="email" id="email" name="email" ng-model="student.email" required>
        <span class="error" ng-show="registrationForm.email.$dirty && registrationForm.email.$error.required">Email is required.</span>
        <span class="error" ng-show="registrationForm.email.$dirty && registrationForm.email.$error.email">Invalid email format.</span>
      </div>
      <div>
        <label for="dob">Date of Birth:</label>
        <input type="date" id="dob" name="dob" ng-model="student.dob" required>
        <span class="error" ng-show="registrationForm.dob.$dirty && registrationForm.dob.$error.required">Date of Birth is required.</span>
      </div>
      <div>
        <label for="rollno">Roll No:</label>
        <input type="text" id="rollno" name="rollno" ng-model="student.rollno" required>
        <span class="error" ng-show="registrationForm.rollno.$dirty && registrationForm.rollno.$error.required">Roll No is required.</span>
      </div>
      <div>
        <label for="phone">Phone:</label>
        <input type="tel" id="phone" name="phone" ng-model="student.phone" pattern="[0-9]{10}" required>
        <span class="error" ng-show="registrationForm.phone.$dirty && registrationForm.phone.$error.required">Phone number is required.</span>
        <span class="error" ng-show="registrationForm.phone.$dirty && registrationForm.phone.$error.pattern">Invalid phone number format (should be 10 digits).</span>
      </div>
      <div>
        <label for="liveKts">Live KTs:</label>
```



```
      <div>
        <label for="year">Year:</label>
        <select id="year" name="year" ng-model="student.year" required>
          <option value="">Select</option>
          <option value="FE">First Year (FE)</option>
          <option value="SE">Second Year (SE)</option>
          <option value="TE">Third Year (TE)</option>
          <option value="BE">Final Year (BE)</option>
        </select>
        <span class="error" ng-show="registrationForm.year.$dirty && registrationForm.year.$error.required">Year is required.</span>
      </div>
      <div>
        <button type="submit" ng-disabled="registrationForm.$invalid">Submit</button>
      </div>
    </form>
  </div>

  <!-- Import AngularJS library -->
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
  <script>
    var app = angular.module('registrationApp', []);

    app.controller('registrationCtrl', function($scope) {
      $scope.student = {}; // Initialize student object

      $scope.submitForm = function() {
        if ($scope.registrationForm.$valid) {
          // Log the student details in the console
          console.log('Form submitted successfully:', $scope.student);

          // Reset form fields
          $scope.student = {};

          // Display success message (optional)
          alert('Form submitted successfully!');

          // You can add your logic to send data to server or do other operations
        } else {
          console.log('Form is invalid. Please correct the errors.');
        }
      };
    });
  </script>
</body>
</html>
```



Student Registration Form
File C:/Users/esha/Desktop/webx/index.html
Type here to search
ENG
11:38 PM
3/28/2024

Student Registration Form

Name:

Email:

Date of Birth: mm/dd/yyyy

Roll No:

Phone:

Live KTs: Select

Current Semester:

Average CGPA:

Percentage of Average CGPA:

Year: Select