| **Experiment No. – 5** | | | | |
| --- | --- | --- | --- | --- |
| **Date of Performance:** | **5/2/25** | | | |
| **Date of Submission:** | **12/2/25** | | | |
| Program Execution/  formation/  correction/  ethical practices  (06) | Timely  Submission  (01) | Viva  (03) | Experiment  Total (10) | Sign with Date |
|  |  |  |  |  |

**Experiment No. 5**

**Title:** Design Web services using AngularJS Framework.

**1.1 Aim:** Design Web services using AngularJS Framework

**1.2 Course Outcome:** Understand how TypeScript and AngularJS framework can build dynamic, responsive single-page web applications

**1.3 Learning Objectives:** To understand the concept and implementation of AngularJS services for modular code design and to demonstrate how services help in sharing data and logic across controllers efficiently.

**1.4 Requirement:**

**Software Requirements:**

* Web Browser (Google Chrome / Firefox)
* Text Editor or IDE (Visual Studio Code / Sublime Text / Atom)
* AngularJS (included via CDN or downloaded)
* Local server (optional, e.g., XAMPP or Live Server extension in VS Code)

**Hardware Requirements:**

* Processor: Intel i3 or higher
* RAM: Minimum 4 GB
* Hard Disk: Minimum 500 MB free space
* Stable Internet Connection (for accessing CDN and APIs)

1.5 Related Theory:

#### What is AngularJS?

AngularJS is an open-source JavaScript-based front-end web framework mainly maintained by Google. It allows developers to create dynamic, single-page applications (SPAs) where content is updated dynamically without the need to reload the entire webpage.

AngularJS extends HTML with new attributes and binds data to HTML using expressions. It follows the Model-View-Controller (MVC) architecture, which helps in separating concerns and managing large-scale web applications effectively.

#### Key Concepts in AngularJS for Web Services:

1. **Model-View-Controller (MVC):**
   * **Model:** Represents the data of the application. It is the single source of truth.
   * **View:** Represents the UI (User Interface) of the application.
   * **Controller:** Acts as a bridge between the Model and the View. It processes input, manipulates data, and updates the View.
2. **Two-way Data Binding:** AngularJS synchronizes data between the model and the view. Any change in the UI reflects in the model and vice versa. This makes applications more responsive and reduces boilerplate code.
3. **Directives:** AngularJS comes with a set of built-in directives like ng-model, ng-bind, ng-repeat, and custom directives. Directives are special tokens in the markup that tell the library to do something to a DOM element (e.g., repeat it, bind it, or attach behavior).
4. **Dependency Injection (DI):** AngularJS has a built-in dependency injection subsystem that helps in creating, understanding, and testing applications easily. It allows you to inject services such as $http, $route, $scope, etc., into your controller functions.
5. **$http Service:** $http is an AngularJS service used to communicate with remote servers via the browser’s XMLHttpRequest object or via JSONP. It is commonly used to consume RESTful APIs in web applications.
6. **Routing:** AngularJS can handle multiple views and pages using routing, which enables the creation of SPAs. The $routeProvider service is used to define different routes (URLs) and map them to specific templates and controllers.

#### Why Use AngularJS for Web Services?

* **Dynamic Content Management:** AngularJS helps build real-time, dynamic content applications where the content on the page updates without refreshing.
* **API Consumption:** AngularJS makes it easy to consume RESTful APIs and handle data asynchronously.
* **Separation of Concerns:** The MVC structure separates logic from the presentation, making code more maintainable.
* **Reusable Components:** AngularJS allows for modular development by breaking applications into reusable components and services.

#### Common Use Cases of AngularJS in Web Services:

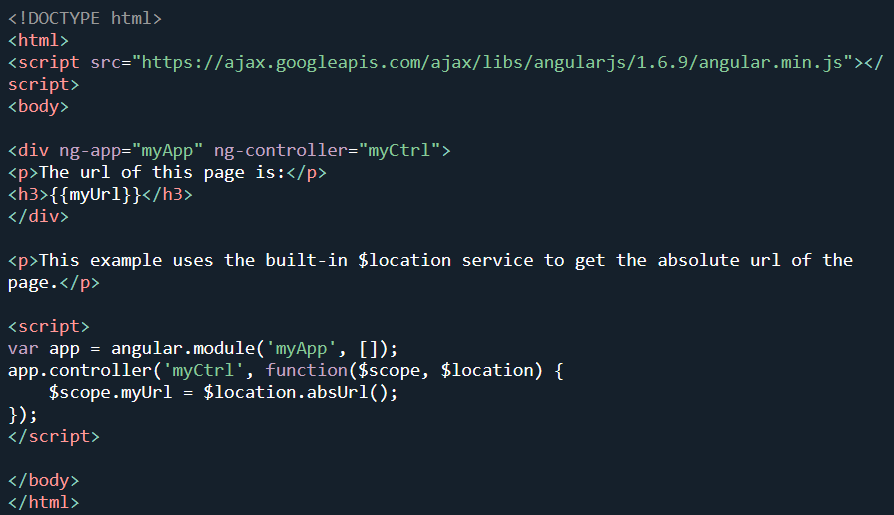
* Creating dashboards with dynamic content
* Building real-time data monitoring tools
* Interacting with backend APIs for CRUD operations
* Designing web forms with validation and live feedback
* Creating SPAs with multiple views without reloading the page

**1.6 Procedure:**

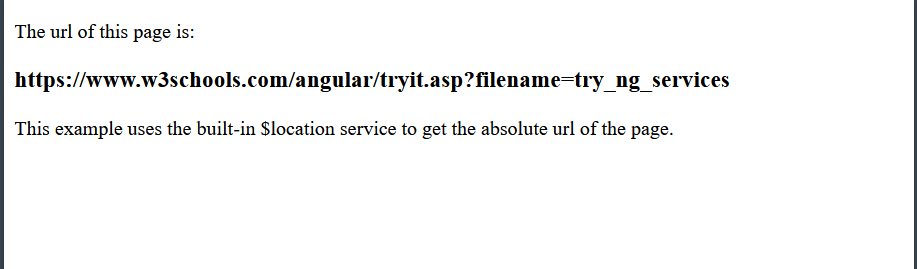
1. Create a New HTML File:  
   * Open a text editor (like VS Code or Notepad++).
   * Create a new file and save it as location-example.html.
2. Include AngularJS Library:

Add the AngularJS CDN in the <head> or top of the <html> document:

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

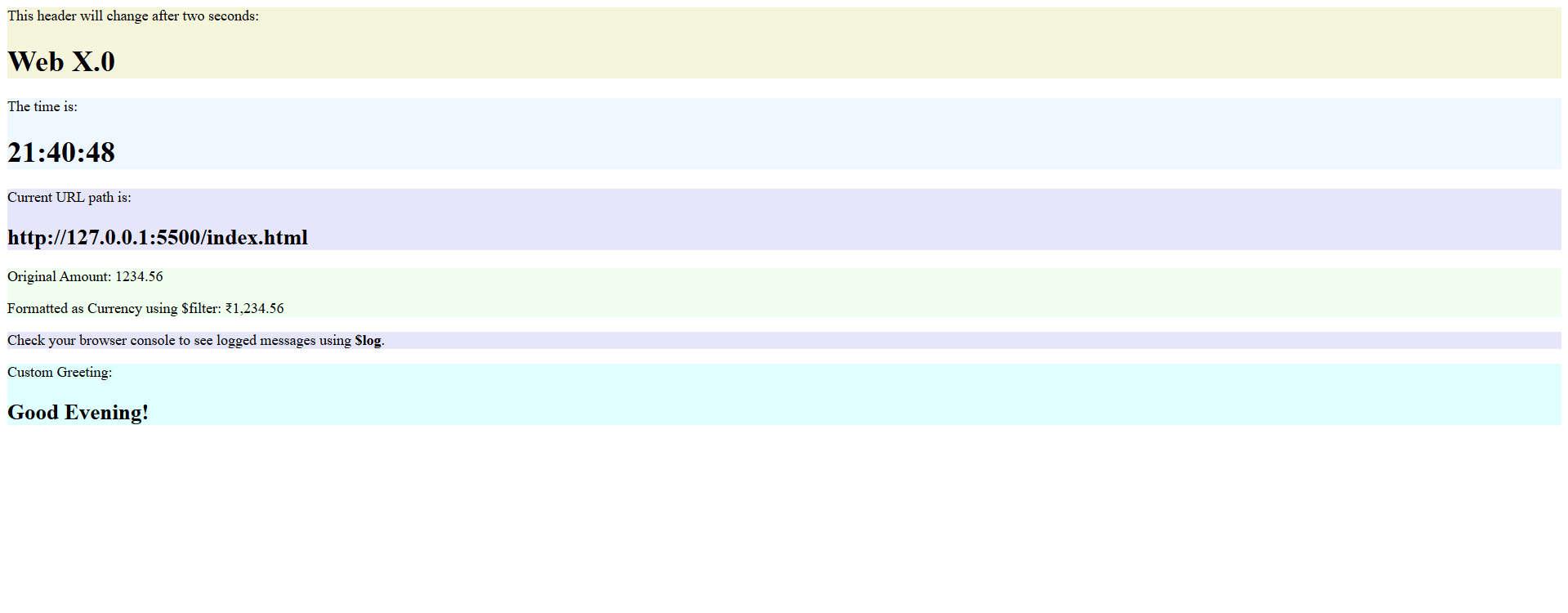
1. Define AngularJS Application and Controller:  
   * Use the ng-app directive to initialize the AngularJS application (myApp).
   * Use the ng-controller directive to bind the controller (myCtrl) to a <div> element.
2. Use $location Service:  
   * In the controller function, inject the $location service along with $scope.
   * Use $location.absUrl() to get the absolute URL of the current page and bind it to a scope variable (myUrl).
3. Display the URL:  
   * Use AngularJS expression {{myUrl}} inside an HTML tag (like <h3>) to display the value of myUrl on the page.

1. Save and Run:  
   * Save the file and open it in a web browser.
   * The absolute URL of the page will be displayed dynamically using AngularJS.



**1.7 Program and Output:**

| <!DOCTYPE html> <html lang="en">  <head>  <meta charset="UTF-8" />  <meta name="viewport" content="width=device-width, initial-scale=1.0" />  <title>AngularJS Services Example</title>  <script src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.8.3/angular.min.js"></script>  </head>  <body ng-app="myApp" ng-controller="MainController">    <div class="timeout" style="background-color: beige">  <p>This header will change after two seconds:</p>  <h1>{{myHeader}}</h1>  </div>   <div class="interval" style="background-color: aliceblue">  <p>The time is:</p>  <h1>{{theTime}}</h1>  </div>   <div class="location" style="background-color: lavender">  <p>Current URL path is:</p>  <h2>{{currentPath}}</h2>  </div>   <div class="filter" style="background-color: honeydew;">  <p>Original Amount: {{amount}}</p>  <p>Formatted as Currency using $filter: {{formattedAmount}}</p>  </div>   <div class="log-output" style="background-color: lavender;">  <p>Check your browser console to see logged messages using <strong>$log</strong>.</p>  </div>   <div class="custom-service" style="background-color: lightcyan;">  <p>Custom Greeting:</p>  <h2>{{greetingMessage}}</h2>  </div>   <script>  var app = angular.module("myApp", []);   app.service("GreetingService", function() {  this.getGreeting = function() {  var hour = new Date().getHours();  if (hour < 12) {  return "Good Morning!";  } else if (hour < 18) {  return "Good Afternoon!";  } else {  return "Good Evening!";  }  };  });   app.controller("MainController", function ($scope, $timeout, $interval, $location, $filter, $log,GreetingService) {  $scope.myHeader = "Web";  $timeout(function () {  $scope.myHeader = "Web X.0";  }, 2000);   $scope.theTime = new Date().toLocaleTimeString();  $interval(function () {  $scope.theTime = new Date().toLocaleTimeString();  }, 1000);   $scope.currentPath = $location.absUrl();   $scope.amount = 1234.56;  $scope.formattedAmount = $filter('currency')($scope.amount, '₹');   $log.info("MainController initialized");  $log.debug("This is a debug message");  $log.warn("This is a warning message");  $log.error("This is an error message");   $scope.greetingMessage = GreetingService.getGreeting();  });  </script>  </body> </html> |
| --- |



**1.8 Conclusion:**

In this experiment, we successfully designed a simple web service using the AngularJS framework and demonstrated the use of the built-in service and custom services. This helped us understand the concept of AngularJS services, two-way data binding, and the modular structure of an AngularJS application. Through this practical, we gained hands-on experience in using AngularJS to create dynamic and interactive web applications.

**1.9 Questions:**

### **1. What is a custom service in AngularJS and why is it used?**

**Answer:** A custom service in AngularJS is a user-defined reusable component that contains application logic and can be injected into controllers or other services. Custom services are used to organize and encapsulate code, especially reusable functions like data processing, validation, or API communication, to make the application modular and maintainable.

**2. What is two‑way data binding in AngularJS, and how does it help when designing web services?**

**Answer:** Two‑way data binding in AngularJS automatically synchronizes data between the model (JavaScript objects) and the view (HTML). When your code updates a model property, the view reflects that change instantly; when the user updates an input in the view, the corresponding model property is updated. This reduces boilerplate code for DOM manipulation and makes it straightforward to display data retrieved from a web service (via $http), and to have user edits propagate back to your application logic.

### **3. What is the difference between a factory and a service in AngularJS?**

**Answer:** In AngularJS, both factory and service are used to create reusable components. The difference lies in how they are implemented:

* A **service** is a constructor function that is instantiated with the new keyword. The properties and methods are attached to this.
* A **factory** is a function that returns an object. It offers more flexibility as it allows private variables and functions. Use a service when you want to use OOP-style methods and a factory for more functional or modular code patterns.