Lifecycle Hooks for Component Views and Content

IIHT

Time To Complete: 10 to 12 hr

Contents

1 Project Abstract

Angular's component lifecycle hooks play a crucial role in controlling component initialization, updates, and destruction. This assignment focuses on understanding and implementing **Lifecycle Hooks** like `ngAfterViewInit` and `ngAfterContentInit` to dynamically manipulate and project content within components.

The objective is to create a **Dynamic Form Single Page Application (SPA)** that allows adding form fields dynamically while utilizing lifecycle hooks to manage projected content and view initialization behavior.

2 Problem Statement

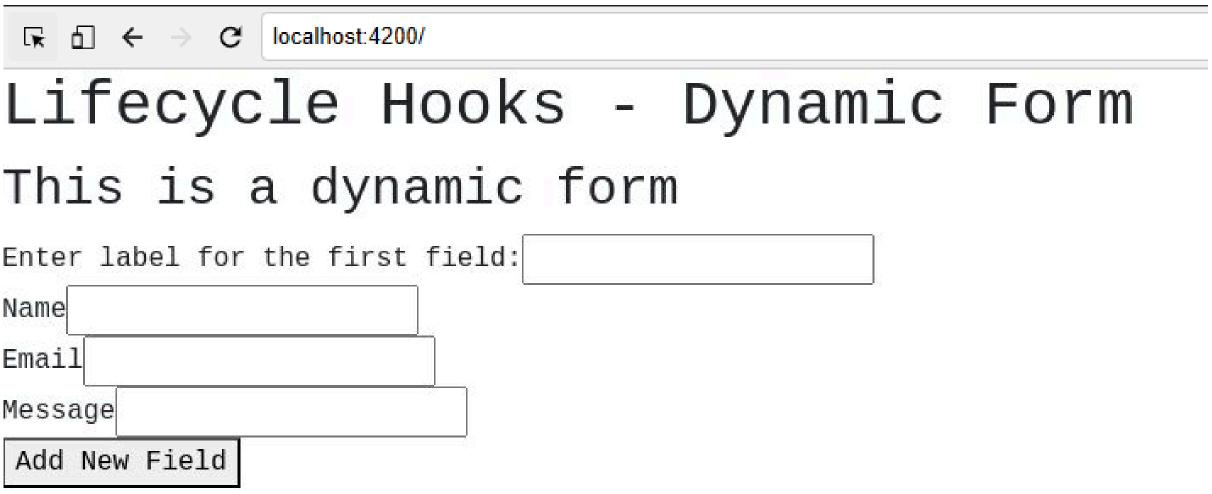
You are tasked with building a **Dynamic Form SPA** using Angular 15, where:

- The form has pre-defined fields.
- Users can add new fields dynamically.
- Content projection is used to inject custom content into the form.
- Angular lifecycle hooks (`ngAfterViewInit`, `ngAfterContentInit`) are used to handle view and content initialization.

3 Proposed Lifecycle Hooks for Component Views and Content Application Wireframe

UI needs improvisation and modification as per given use case and to make test cases passed.

3.1 ScreenShots

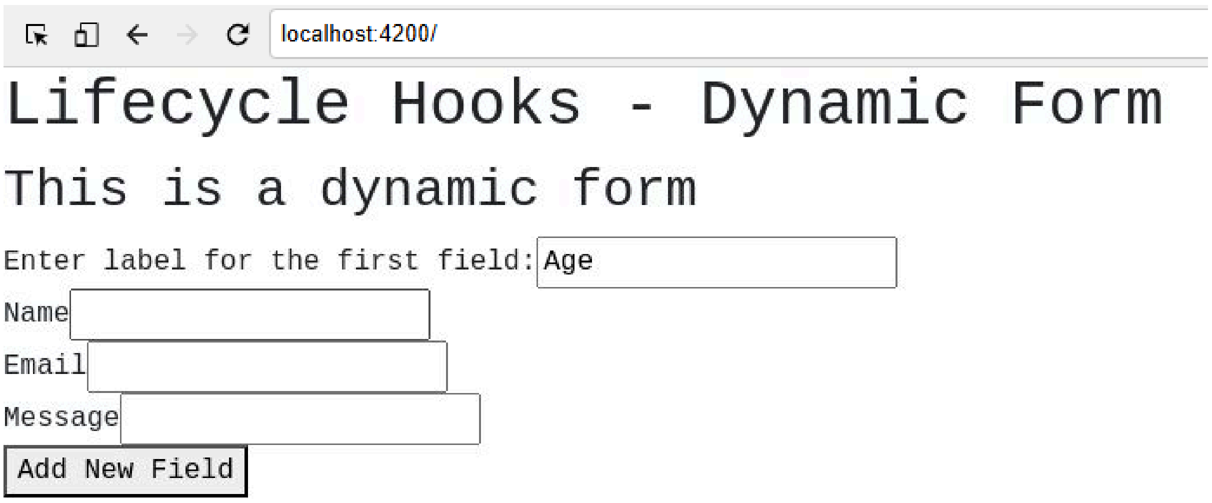

localhost:4200/

Lifecycle Hooks - Dynamic Form

This is a dynamic form

Enter label for the first field:	
Name	
Email	
Message	

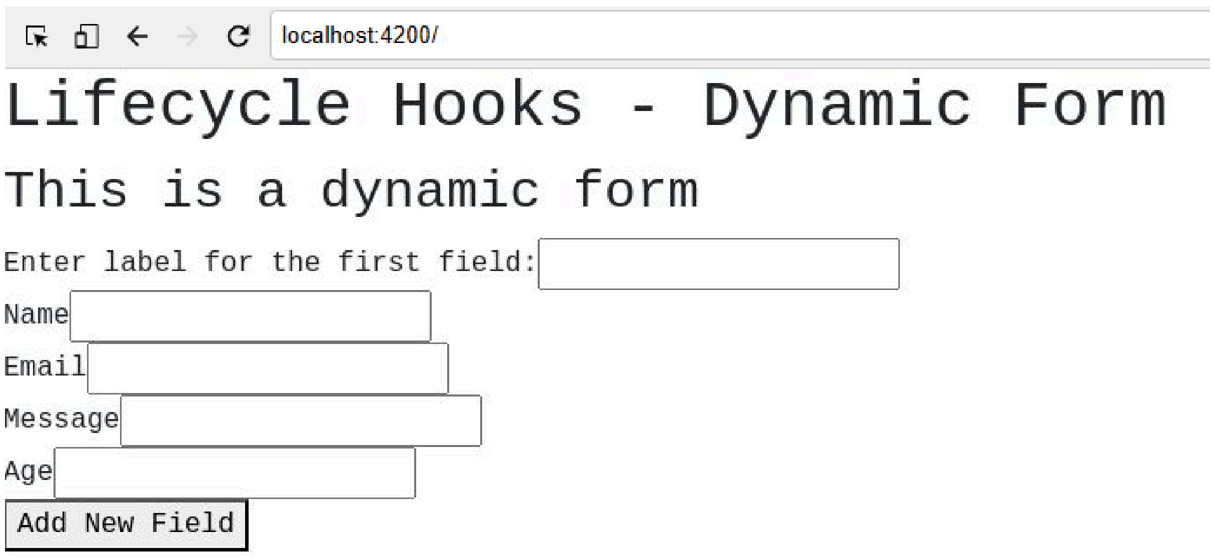
*****Add Field*****


localhost:4200/

Lifecycle Hooks - Dynamic Form

This is a dynamic form

Enter label for the first field:	Age
Name	
Email	
Message	



localhost:4200/

Lifecycle Hooks - Dynamic Form

This is a dynamic form

Enter label for the first field:

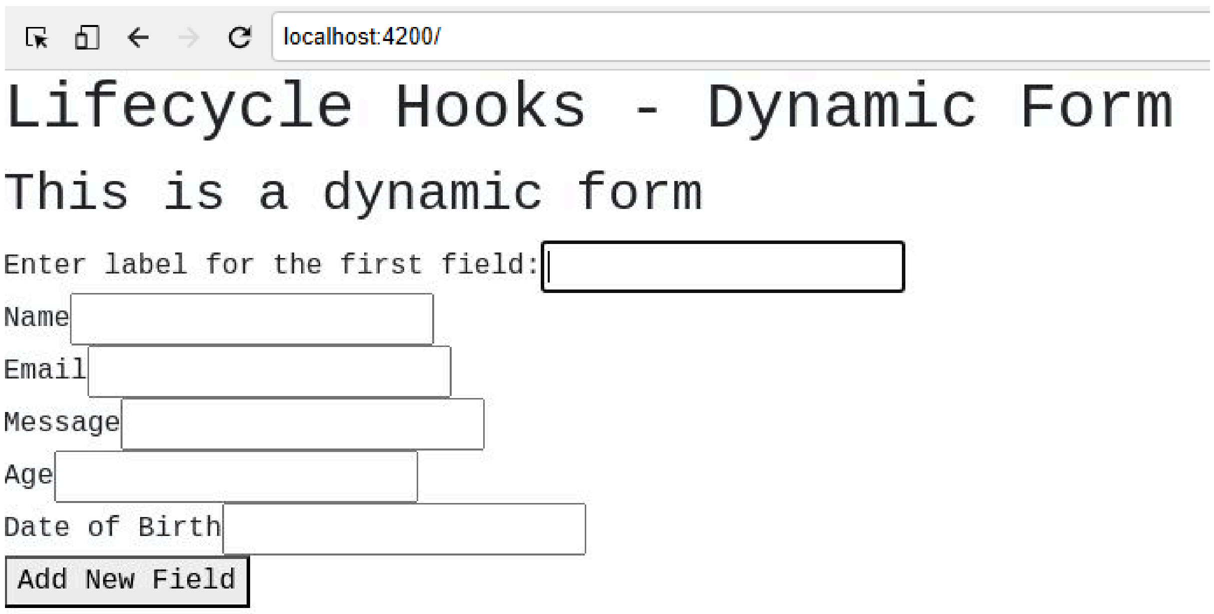
Name

Email

Message

Age

Add New Field



localhost:4200/

Lifecycle Hooks - Dynamic Form

This is a dynamic form

Enter label for the first field:

Name

Email

Message

Age

Date of Birth

Add New Field

4 Business-Requirement:

As an application developer, develop the Lifecycle Hooks for Component Views and Content (Single Page App) with below guidelines:

User Story #	User Story Name	User Story
US_01	Welcome Page	As a user I should be able to visit the welcome page as the default page. Acceptance criteria: **AppComponent:** 1. Display the main heading **"Lifecycle Hooks - Dynamic Form"** in an **h1** tag. Define a property named **`title`** in the component class: ● Type: `string` ● Value: `"Lifecycle Hooks - Dynamic Form"` (or similar) Implement the `ngOnInit()` lifecycle hook: ● Log a message to the console: `"AppComponent initialized"` Inside the `app-dynamic-form` opening and closing tags: ● Project a heading element (e.g., `<h3>`) as custom content. ● Use a template reference `#customLabel` with content as “Custom Label for the Dynamic Form”. **DynamicFormComponent:** 2. Display the subheading projected from `AppComponent`. 3. Display a default message **"This is a dynamic form"** in an **h2** tag using a component property. **Create a formFields variable as:** ● Type: Array of strings ● Purpose: To store the labels of dynamic input fields. ● Initialize with 3 default field labels: `"Name"`, `"Email"`, `"Message"`. **Create a dynamicMessage variable as:** ● Type: `string` ● Purpose: Display a heading message for the form (e.g., `"This is a dynamic form"`). **Create a userLabel variable as:**

- Type: `string`
- Purpose: Stores the label entered by the user for a new form field.
- This value is updated in real time based on user input in the first field.

Define two properties using the `@ViewChild` decorator:

1. firstInput

- Targets the first input element using a template reference (`#firstInput`).
- Used to automatically focus this input after the view is initialized.

2. firstLabel

- Targets the label for the first input using a template reference (`#firstLabel`).
- Optionally used to capture an initial label value.

Create and define ngAfterViewInit as:

- Logs: `"ngAfterViewInit: View initialized."`
- Automatically focuses the first input field using `firstInput.nativeElement.focus()`.
- Optionally reads the text content or value from the first label and assigns it to `userLabel`.

Create and define ngAfterContentInit as:

- Logs: `"ngAfterContentInit: Content initialized."`
- Used to indicate that projected content from the parent is ready.

Create and define ngAfterViewChecked as:

- Logs: `"ngAfterViewChecked: View checked."`
- Used to demonstrate that the view is checked on every change detection cycle.

Create a method named **`addNewField()`**:

1. If `userLabel` is not empty:
    - Add it to the `formFields` array.
2. If `userLabel` is empty:
    - Add a default field label like `"New Field 4"`, `"New Field 5"`, etc.
3. This dynamically updates the form with new input fields.

		In the HTML template: 1. Display the `dynamicMessage` in a heading (`<h2>`). 2. Include the **first label and input field**: ○ Add `#firstLabel` to the label and `#firstInput` to the input. ○ Bind the `userLabel` value to the input's `(input)` event. 3. Use `*ngFor` to render all fields in the `formFields` array: ○ For each label, show an associated input box. 4. Add a button labeled **"Add New Field"**: ○ On click, it should call the `addNewField()` method. **** Kindly refer to the screenshots for any clarifications. ****

5 Constraints

1. You should be able to press the "TAB" key and "SHIFT + TAB" to navigate from top field to bottom field and vice-versa.

6 Mandatory Assessment Guidelines

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.**
3. **This editor Auto Saves the code.**
4. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**
5. **This is a web-based application, to run the application on a browser, use the internal browser in the workspace. Click on the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.**
   Note: The application will not run in the local browser

6. **You can follow series of command to setup Angular environment once you are in your project-name folder:**
    a. **npm install -> Will install all dependencies -> takes 10 to 15 min.**
    b. **npm run start -> To compile and deploy the project in browser. You can press the <Ctrl> key while clicking on localhost:4200 to open the project in the browser -> takes 2 to 3 min.**
    c. **npm run test -> to run all test cases. It is mandatory to run this command before submission of workspace -> takes 5 to 6 min.**
7. **Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on "Submit Assessment" after you are done with code.**