Route Parameters and Route Guard Basics

IIHT

Time To Complete: 10 to 12 hr

CONTENTS

1 Project Abstract

Routing is essential for navigation in single-page applications. This assignment focuses on two core concepts of Angular routing: **Route Parameters** and **Route Guards**. The goal is to learn how to pass dynamic parameters via routes and secure specific routes using basic route guarding mechanisms.

The objective is to create a simple SPA where the user logs in and navigates to their profile page, which uses dynamic route parameters and is protected by a route guard.

2 Problem Statement

You are tasked with developing a **Route Parameter and Route Guard SPA** using Angular.

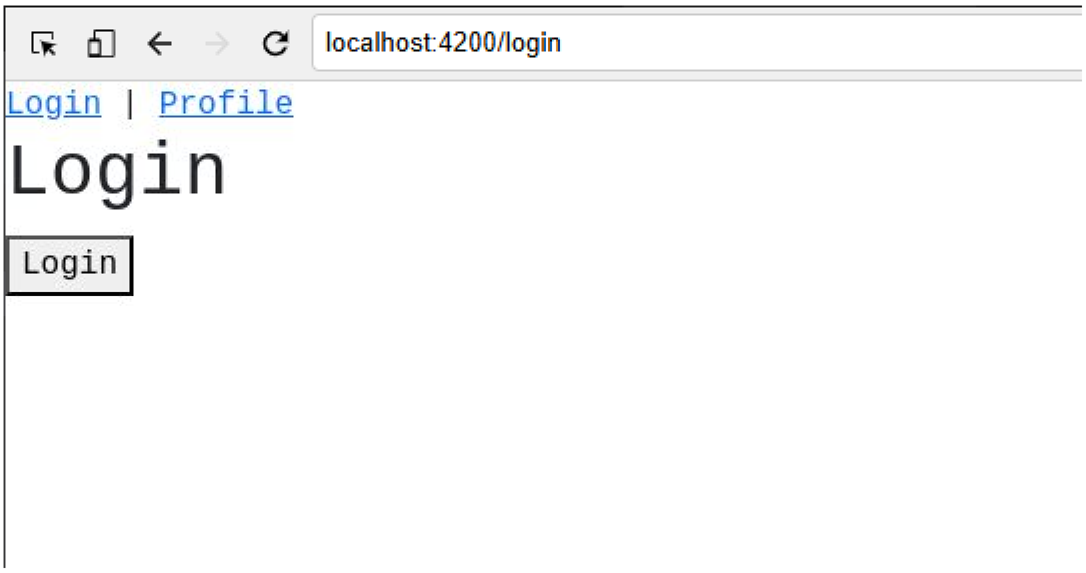
The application should:

- Include two pages: **Login** and **Profile**.
- Use dynamic route parameters to display the logged-in user’s ID in the Profile page.
- Protect the Profile page using a basic route guard.
- Implement a simple authentication service and guard logic to control access.

3 Proposed Route Parameters and Route Guard Basics Application Wireframe

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

3.1 ScreenShots

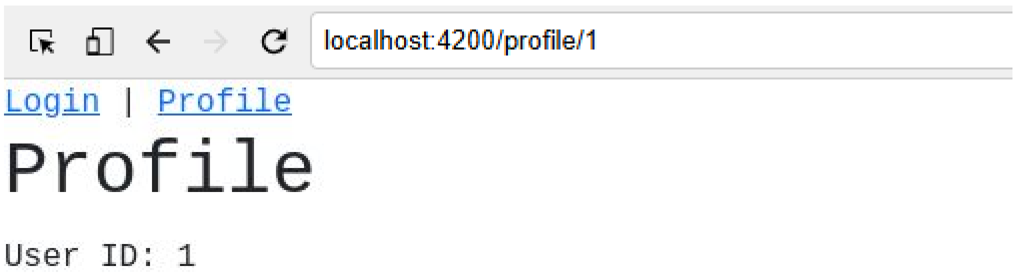

localhost:4200/login

Login | Profile

Login

Login

*****Profile*****


localhost:4200/profile/1

Login | Profile

Profile

User ID: 1

4 Business-Requirement:

As an application developer, develop the Route Parameters and Route Guard Basics (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. Should display navigation links for **Login** as “/login” and **Profile as “/profile/1”,** with each link wrapped inside an **<a> tag** using `routerLink`.
2. Clicking on **Login** should navigate to `/login`.
3. Clicking on **Profile** should navigate to `/profile/1`, where the user ID is dynamically set.
4. Include the `<router-outlet>` directive to render routed components.

AppRoutingModule:

5. Redirect the empty path ('') to /login using redirectTo and pathMatch: 'full'.
6. /login ⟶ Loads the LoginComponent.
7. /profile/:id ⟶ Loads the ProfileComponent. This route should be protected by AuthGuard, meaning it should only be accessible if the guard allows it.
8. The route to the profile should be restricted using canActivate with AuthGuard.

LoginComponent:

9. Should display the heading **"Login"** in an **h1** tag.
10. Provide a **Login** button.
11. On clicking the Login button, user authentication should be triggered using invoking login() of AuthService file and the user should be redirected to the profile page with links as “/profile/1”.

ProfileComponent:

12. Should display the heading **"Profile"** in an **h1** tag.
13. Below it, display **User ID: {userId}** dynamically fetched from route parameters in a **p tag**.

AuthGuard:

14. If `authService.isLoggedIn()` returns `true`, allow access by returning `true`.
15. If not, redirect the user to `/login` and return `false`.

**** Kindly refer to the screenshots for any clarifications. ****

AuthService:

16. Maintain a private boolean property loggedIn to track the user's login state (initially false).
17. login() ⟶ Sets loggedIn to true.
18. logout() ⟶ Sets loggedIn to false.

		19. isLoggedIn() ⟶ Returns the current value of loggedIn as a boolean.

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.**