EVENT MANAGEMENT-ROUTING

IIHT

Time To Complete: 10 to 12 hr

CONTENTS

1 Project Abstract

In today's digital era, managing events efficiently demands interactive and real-time web applications. **Event Management - Routing** is developed using Angular that focuses on organizing events with ease and structure.

The project demonstrates the use of Angular routing to navigate between different components such as event creation, guest management, and feedback collection. By leveraging Angular's powerful routing module, this application ensures seamless navigation and a user experience. The system helps users create and track events, manage attendees, and collect feedback — all from a unified interface.

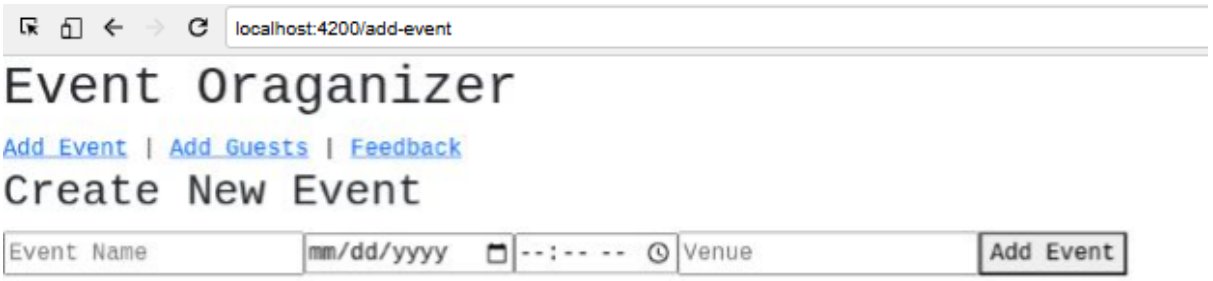
2 Problem Statement

"Event Management-Routing" application demonstrates how Angular routing enhances navigation between modules like `Add Event`, `Add Guests`, and `Feedback`. It serves as a practical example of applying Angular concepts to solve real-world event organization challenges.

3 Proposed Event Management-Routing Application Wireframe

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

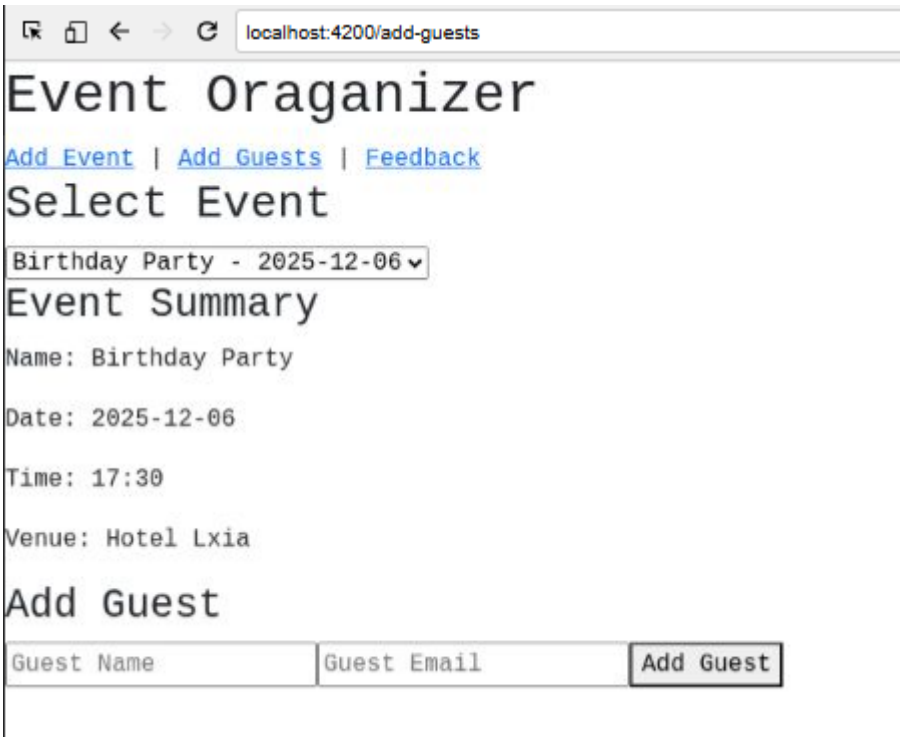
3.1 Welcome Page



3.2 ScreenShots

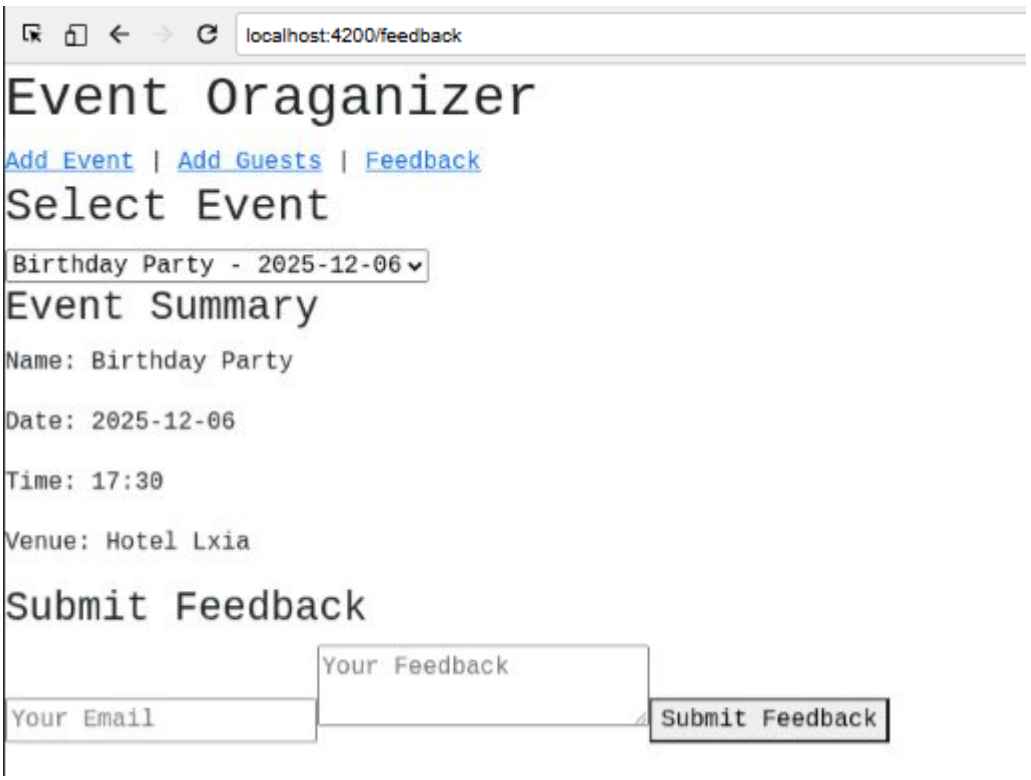
Add Event

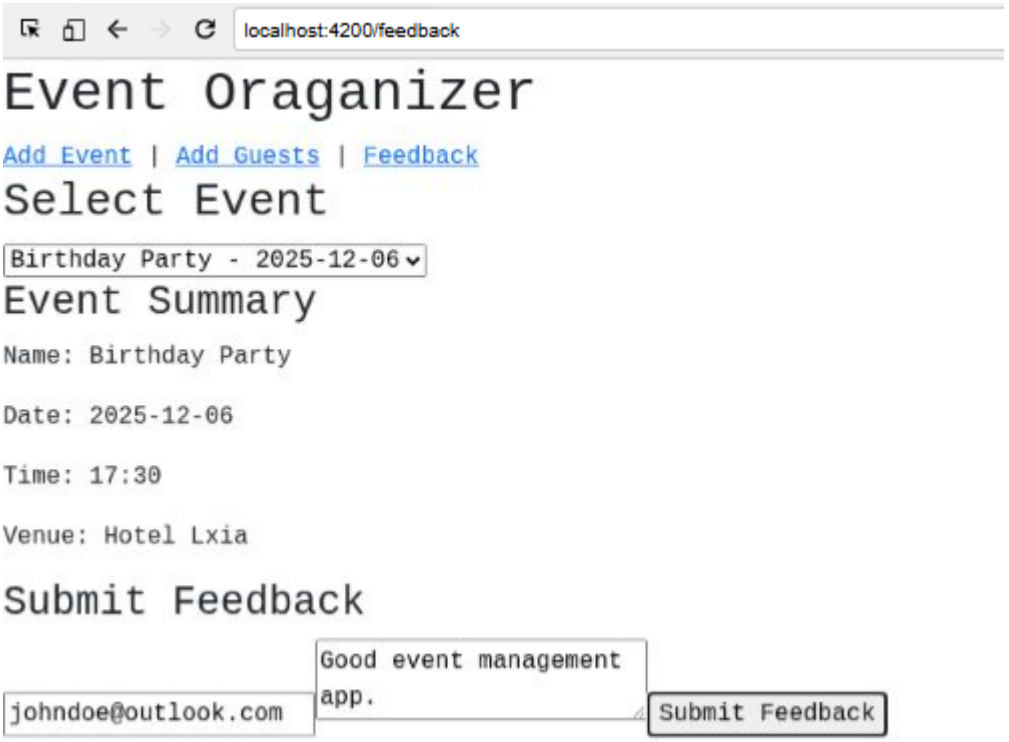
Add Guest




localhost:4200/add-guests
Event Oraganizer
Add Event | Add Guests | Feedback
Select Event
Birthday Party - 2025-12-06
Event Summary
Name: Birthday Party
Date: 2025-12-06
Time: 17:30
Venue: Hotel Lxia
Add Guest
Shubman Gill
shubgill@gmail.com
Add Guest

Feedback


localhost:4200/feedback
Event Oraganizer
Add Event | Add Guests | Feedback
Select Event
Birthday Party - 2025-12-06
Event Summary
Name: Birthday Party
Date: 2025-12-06
Time: 17:30
Venue: Hotel Lxia
Submit Feedback
Your Feedback
Your Email
Submit Feedback



4 BUSINESS-REQUIREMENT:

As an application developer, develop the Event Management-Routing Application 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:** **Purpose** **AppComponent** ● Set up a navigation layout for an event organizer application. ● Route users between different modules: **Add Event**, **Add Guests**, and **Feedback**.

HTML Structure

1. **Top-level heading:**
   - Displays an `<h1>` element:
     "Event Organizer"

2. **Navigation section using `<nav>` tag:**
   - Three anchor links using `routerLink`:
     - `Add Event` → Navigates to `/add-event`
     - `Add Guests` → Navigates to `/add-guests`
     - `Feedback` → Navigates to `/feedback`

3. **Router outlet:**
   - `<router-outlet>` is used to dynamically render routed components.

Functions & Responsibilities

- Root component of the Angular application.
- Sets up layout and contains navigation.
- Uses `routerLink` to allow users to navigate to different parts of the app without reloading the page.
- Hosts the `<router-outlet>` for displaying child routed components.

EventFormComponent

Purpose

Allows the user to create a new event by submitting a simple form.

HTML Structure

1. Display a heading: "Create New Event" in `<h2>`.
2. Provide a form using `(ngSubmit)="addEvent()"`.
   - Input for Event Name (required)
   - Input for Date (type: date, required)
   - Input for Time (type: time, required)
   - Input for Venue (required)
   - Submit button labeled "Add Event"

Functions & Responsibilities

addEvent()

- Uses `eventService.addEvent(this.event)` to store the event.

- Resets the form fields after submission.

GuestFormComponent

Purpose

Allows users to:

- Select an existing event.
- View basic event summary.
- Add guest details tied to the selected event.

HTML Structure

1. **Dropdown to Select Event**
   - Label: “Select Event” (`<h2>`)
   - Populated via `*ngFor` from the `events` array.
   - Uses two-way binding `[(ngModel)]="selectedEventId"` and triggers `(ngModelChange)="selectEvent($event)"`.

2. **Conditional Block: If an Event Is Selected**
   - Displays Event Summary (`<h3>Event Summary`)
     - Name
     - Date
     - Time
     - Venue
   - Displays a Guest Form (`<h3>Add Guest`)
     - Input for Guest Name (required)
     - Input for Guest Email (required)
     - Submit button: “Add Guest”

Component Logic & Responsibilities

`constructor(...)`

- Initializes `events` and `selectedEventId` from `EventService`.

`selectEvent(eventId)`

- Called when an event is selected from the dropdown.
- Updates `selectedEventId` locally and in the service via `selectEvent(...)`.

`addGuest()`

- Checks if an event is selected.

- Adds a guest by calling `eventService.addGuest(...)`.
- Resets `newGuest` to default state after successful addition.

FeedbackFormComponent

Purpose

Allows a user to:

- Select an event.
- View the selected event's details.
- Submit feedback for the selected event including their email and comments.

HTML Structure

1. **Heading**:
   - `<h2>` — "Select Event"
2. **Dropdown to Select Event**:
   - Bound to `[(ngModel)]="selectedEventId"`
   - Populated using `*ngFor="let event of events"`
   - Triggers `(ngModelChange)="selectEvent($event)"`
3. **Conditional Block (`*ngIf="selectedEvent"`)**
   If an event is selected:
   - **Section: Event Summary**
     - Heading: `<h3>Event Summary`
     - Shows: `name`, `date`, `time`, `venue` from `selectedEvent`.
   - **Section: Submit Feedback**
     - Heading: `<h3>Submit Feedback`
     - Input field: "Your Email" (bound to `guestEmail`, `required`)
     - Textarea: "Your Feedback" (bound to `comments`, `required`)
     - Button: "Submit Feedback"

Component Responsibilities

`constructor(...)`

- Loads the list of events via `eventService.getEvents()`.
- Gets any previously selected event ID using `eventService.getSelectedEventId()`.

		`selectEvent(eventId)`
		• Sets `selectedEventId` and updates it in the service using `eventService.selectEvent(...)`.
		
		`submitFeedback()`
		• Checks if an event is selected.
		• If selected:
		  ○ Submits feedback using `eventService.addFeedback(...)`.
		  ○ Clears the input fields (`guestEmail`, `comments`) after submission.
		
		--- 	
		
		Note: Service file is already implemented.
		
		**** 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.**