Module 8

What to learn

Routing
Generate an app with routing enabled
Importing your new components into AppRoutingModule
Defining a basic route
Define your routes in your Routes array.
Add your routes to your application.
Template with routerLink and router-outlet
Getting route information
Router
ActivatedRoute
ParamMap
Setting up redirects

Practice Exercise

Practice 1
Do the hands for tours of heroes Application from the following link. (Cover section milestone1) https://angular.io/guide/router-tutorial-toh
Practice 2
Use Case:
When a user logs in, navigate them to:

/welcome if it's their first login.
/dashboard for subsequent logins.

Key Steps:

Check a flag in local storage or an API (isFirstLogin).
Redirect to the appropriate route based on the flag.

Practice 3

Order Management with Filtered List

Use Case:
Create a route /orders with query parameters ?status=pending or ?status=completed to filter orders.

Display orders dynamically based on the status query parameter.

Key Steps:

Use ActivatedRoute to read query parameters.
Call a service to fetch filtered orders.

Practice 4

E-commerce Product Pagination

Use Case:
Build a product list page with pagination using routes like /products?page=1.

Fetch and display products for the specified page.

Key Steps:

Read the page query parameter and dynamically load products for the given page.
Add navigation buttons to switch pages.

Practice 5

Dynamic Tab Navigation in a Profile Page

Use Case:
Implement a user profile with tabs like "Details", "Orders", and "Settings".

Use child routes such as /profile/details, /profile/orders, and /profile/settings.
Highlight the active tab dynamically.

Key Steps:

Create child routes for tabs.
Use routerLinkActive to indicate the active tab.

Practice 6

Admin Panel with Restricted Access by Role

Use Case:
In an admin panel, restrict access to specific sections:

/admin/users for Admin users.
/admin/reports for Manager users.
Redirect unauthorized users to an "Access Denied" page.

Key Steps:

Use a route guard to check user roles dynamically.
Configure routes based on role.

Practice 7

Checkout Flow with Step Validation

Use Case:
Implement a multi-step checkout flow:

/checkout/cart
/checkout/shipping
/checkout/payment
Ensure users can’t proceed to /checkout/payment without completing /checkout/shipping.

Key Steps:

Use route guards to enforce step validation.
Pass data between steps using a shared service.

Practice 8

Event Calendar with Date Navigation

Use Case:
Create a calendar with routes like /events?date=2024-12-01 to display events for a specific date.

Add "Next Day" and "Previous Day" buttons to navigate dates.

Key Steps:

Use query parameters for the date.
Fetch events for the given date and update the calendar dynamically.

Practice 9

Favorite Items with State Persistence

Use Case:
Build a route /favorites to display a user's favorite items.

Allow toggling of items between "Favorite" and "Not Favorite".
Persist the state locally or via an API.

Key Steps:

Fetch favorite items on route load.
Update the list dynamically based on user actions.

Practice 10

Content Preview with Secure Access

Use Case:
Create a route /preview/:id that displays restricted content previews for specific users.

Ensure only users with valid access tokens can view the preview.

Key Steps:

Implement a route guard that validates tokens.
Redirect unauthorized users to /login.

Practice 11

Dynamic Redirect Based on User Action

Use Case:
For an application with a "Contact Us" form, after submission:

Redirect to /thank-you if the submission is successful.
Redirect to /form-error if it fails.

Key Steps:

Use the Router to programmatically navigate based on the form response.

Assignment Exercise

Assignment 1
Tutorial site task site needs to be implemented. Application should have topnav bar which contains html/CSS/javascript Respective daywise assignment should be loaded in the router outlet.

Assignment 2

Use Case: Employee Management System with Filter Logic

Scenario:

You are tasked with creating an **Employee Management System** where users can view, add, edit, and delete employee data, as well as manage the projects they are assigned to. The system includes **nested routes** and **filter functionality** to manage employees based on their **status** and **position**.

Routes:

Employee List
Route: /employees
Description: Displays a list of all employees.
Actions:
Add, Edit, View, and Delete Employee.
Filter Functionality:
Filters by **Employee Status** (e.g., Active, Inactive).
Filters by **Position** (e.g., Developer, Manager).
Business Logic:

Display employees based on selected filters.
Automatically sort employees by **Joining Date**.

Employee Details with Nested Tabs

Route: /employees/:id

Description: Displays employee details with nested tabs for Personal Info and Assigned Projects.

Tabs:

Personal Info: /employees/:id/personal

Show employee's personal information such as name, email, phone number, and position.

Assigned Projects: /employees/:id/projects

Display all projects assigned to the employee, with options to **add** or **remove** projects.

Business Logic:

Disable the **Assigned Projects** tab if the employee has no assigned projects yet.

Add New Employee

Route: /employees/new

Description: Opens a form to add a new employee.

Business Logic:

Field Validation:

Ensure that the **email** is unique across the system before submission.

Ensure that **position** is selected from a predefined list (e.g., Manager, Developer).

Ensure **phone number** is unique.

Redirect to the employee's details page upon successful creation (/employees/:id).

Edit Employee

Route: /employees/:id/edit

Description: Opens a form to edit employee details.

Business Logic:

Validate the **email** and **phone number** fields before submitting.

Disable the **email** and **position** fields for employees with "Manager" status.

Display a **confirmation** prompt before allowing an edit.
Redirect to the employee details page after successful edit (/employees/:id).

Assign Project to Employee

Route: /employees/:id/projects/assign

Description: Opens a form to assign a project to the employee.

Business Logic:

Ensure the employee does not already have the project assigned.
Prevent assigning the same project multiple times.
Filter projects by status (e.g., Active, Completed) before displaying them in the form.

Additional Features & Business Logic:

Filter Logic on Employee List:

Filter by Status: Allow users to filter employees based on their status (Active, Inactive).
Filter by Position: Allow users to filter employees by their position (e.g., Developer, Manager).
Sort Employees: Automatically sort the employee list by their **Joining Date**, with the most recent joiners at the top.
Pagination: Implement pagination to limit the number of employees displayed per page, especially useful when the number of employees is large.
Search: Allow users to search for employees by **name** or **email**.

Confirmation Dialog for Deleting Employees:

When a user attempts to delete an employee, show a **confirmation dialog**:

"Are you sure you want to delete this employee?"
Only allow deletion if the employee is **Active**.
If the employee is assigned to projects, ask whether to remove the employee from those projects before deletion.

Redirect Logic:

After adding or editing an employee, redirect the user to the **Employee Details** page:

For new employees: /employees/:id.
For edited employees: /employees/:id.

Handling Errors:

If a submission fails (for adding or editing an employee), display a meaningful error message explaining why the operation failed (e.g., "Email already exists" or "Invalid phone number").

Expected User Flow:

Employee List Page (/employees):

The user lands on the employee list page, where they can filter by status or position, search, and paginate through employees.
The user can click on an employee to view or edit details.

Employee Details Page (/employees/:id):

When the user clicks on a specific employee, they are directed to the details page.
If the employee has no assigned projects, the "Assigned Projects" tab is disabled.
The user can navigate to the "Personal Info" or "Assigned Projects" tabs, where they can modify or view the employee's data.

Add/Edit Employee Page (/employees/new or /employees/:id/edit):

The user fills out a form to either add a new employee or edit an existing one.
The form includes validation for uniqueness of email and phone number, and appropriate field disabling (e.g., for managers).

Assign Project Page (/employees/:id/projects/assign):

The user can assign a project to an employee. Only active, non-duplicate projects are shown.
After assigning, the employee's assigned projects are updated in the database, and the user is redirected to the employee details page.

Business Logic Summary:

Filtering: Allows users to filter employees by status and position.
Field Validation: Ensures data integrity, such as unique email, phone, and position.
Tab Disabling: The "Assigned Projects" tab is disabled if no projects are assigned.
Conditional Deletion: Employees can only be deleted if they are **Active**, and the system will check for project assignments before allowing deletion.
Redirecting: After adding or editing an employee, the user is redirected to the details page.

Online Reference

No online Reference

Introduction

the basics

course project-basics

debugging

components & databinding deep dive

course project – components & databinding

directives deep dive

Using Services & Dependency Injection

Course Project – Services & Dependency Injection

Changing Pages with Routing

Course Project – Routing

Handling Forms in Angular Apps

Course Project-Forms

Using Pipes to Transform Output

Making Http Requests