Challenge

Implement Interactions Between Angular Components

Challenge

- Challenge: Create Keep-Note Application with multiple interacting components

Menu

00:02 00:00:21 02/ 21

Points to Remember

- Ensure components are created as per the given component hierarchy.
- Angular CLI command should be used to create the components.
- CSS classes defined in the.css files of the respective components should be used to style the components.
- The color, background color, font, and other CSS properties can be defined to create pleasing aesthetics.

Instructions for Challenge

- Click here for the boilerplate.
- Read the README.md file in the boilerplate for further instructions about the challenge.
- Fork the boilerplate into your own workspace.
- Clone the boilerplate into your local system.
- Open command terminal and set the path to the folder containing the cloned boilerplate code.
- Run the command `npm install` to install the dependencies.
- Open the folder containing the boilerplate code in VS Code.
- Complete the solution in the given partial code provided in the boilerplate.

Notes:

The solution of this challenge will undergo an automated evaluation on `hobbes`.
(Local testing is recommended prior to `hobbes` testing)

The test cases are available in the boilerplate.

Context

As you are aware, `Keep-Note` is a web application that allows users to maintain notes. It is developed as a single-page application using multiple components.

Note: The stages through which the development process will be carried out are shown below:

Stage 1: Create basic `Keep-Note` application to add and view notes.

Stage 2: Implement unit testing for the `Keep-Note` application.

Stage 3: Create Keep-Note application with multiple interacting components to add, view and search notes.

Stage 4: Implement persistence in the Keep-Note application.

Stage 5: Style the `Keep-Note` application using Material design.

Stage 6: Create simple form with validation in the `Keep-Note` application.

Stage 7: Create complex form with validation in the `Keep-Note` application.

Stage 8: Enable navigation in the `Keep-Note` application.

Stage 9: Secure routes in the `Keep-Note` application

Context (Cont'd)

In this sprint, we are at Stage 3.

In this development stage, the application should add and read notes from an array that is declared in a separate file. For convenience, the application should allow users to search for notes by the title of the note.

Create `Keep-Note` Application with Multiple Interacting Components

Design the Keep-Note application as an SPA with multiple components. The application should display notes, allow searching for a note by its title and allow adding a new note.

Note: Tasks to complete the challenge are given in the upcoming slide.

CHALLENGE

Tasks

- To develop the solution for the `Keep-Note` application, following tasks need to be completed:
  - Task 1: Create data models
  - Task 2: Create components
  - Task 3: Design application header
  - Task 4: Display notes
  - Task 5: Search notes
  - Task 6: Add a new note

Note: The details of the tasks listed above are provided in the upcoming slides

Task 1: Create Data Models

- Inside boilerplate code, create data models under the folder with the name `models`.
- Create a type `Note` in the **`note.ts`** file in the `models` folder with the following type properties:
  - `id` (number)
  - `title` (string)
  - `content` (string)

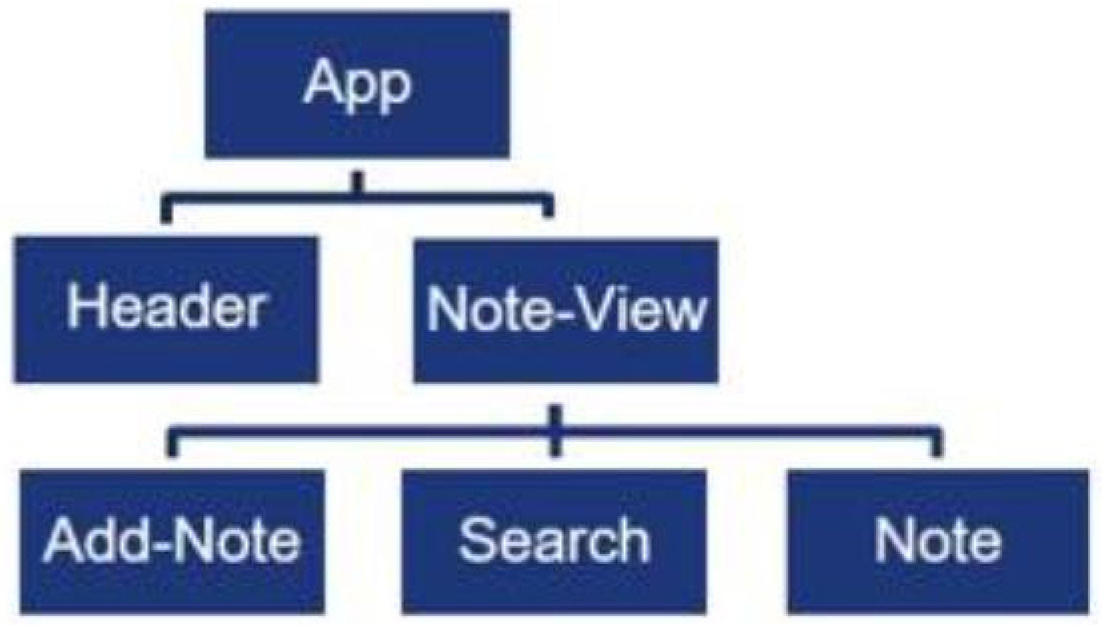
Task 1: Create Data Models (Cont'd.)

- Create constant `NOTES` in the **`notes.ts`** file in the **`models`** folder.
    - The `NOTES` constant should be an array with the following notes data:

id	title	content
1	Sample Note	This is a sample note added for testing purpose
2	Practice Exercise: Typescript	To add a function to calculate final bill amount
3	Challenge: Typescript	To add a function to fetch notes
4	Refactor Practice Exercise	Code needs to be well-indented
5	Refactor Challenge	Make the design responsive and add comments in the code

Task 2: Create Components

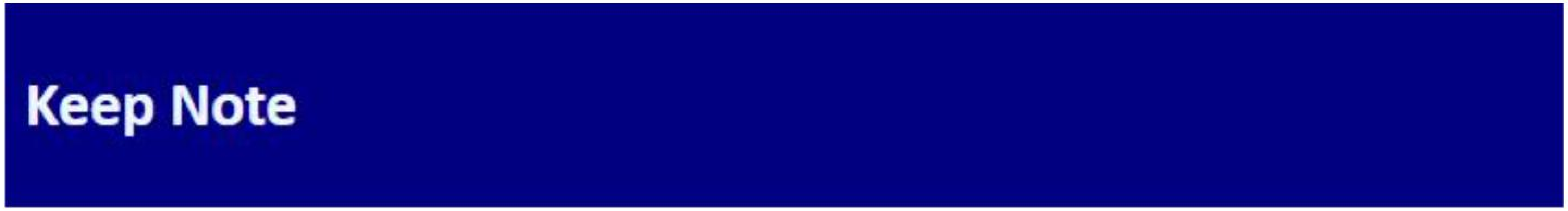
- Create components as shown in the component hierarchy diagram below:



- Use Angular CLI command to create components.
- Render the components using component selector as per the hierarchy.

Task 3: Design Application Header

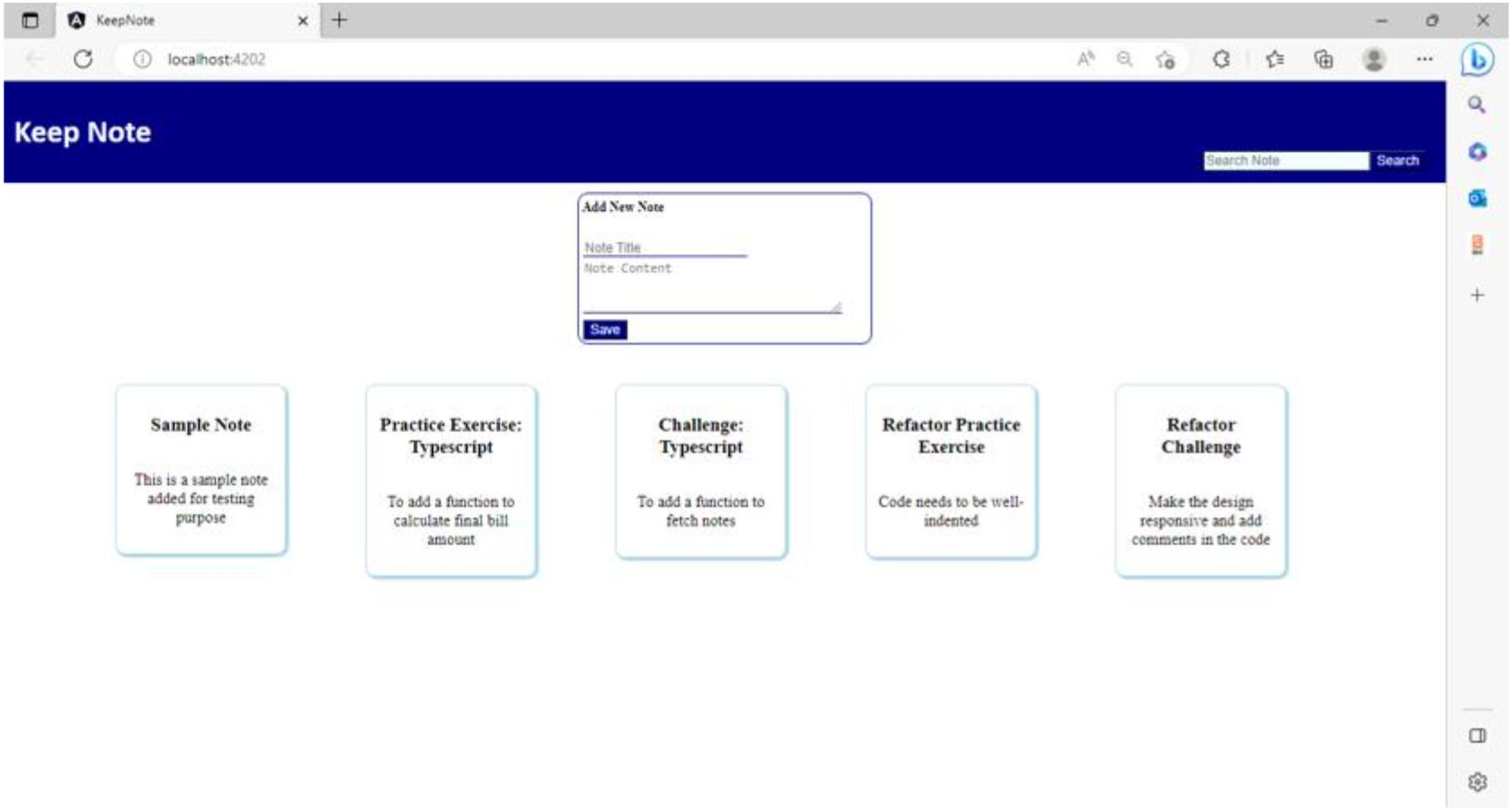
- Modify the `Header` component to display the application title `Keep Note`.
- The below image shows a sample layout of the header.
  - Background color can be different but should be aesthetically pleasing.



Task 4: Display Notes

- The `Note-View` component should handle the responsibility of reading notes from the **`Notes.ts`** file.
- For each note read, the `Note-View` component should render the `Note` component.
- The `Note` component should accept note data as the input from the `Note-View` component and display the `title` and `content` values of the note on the UI.

Task 4: Expected Output



Sample Note
This is a sample note added for testing purpose

Practice Exercise: Typescript
To add a function to calculate final bill amount

Challenge: Typescript
To add a function to fetch notes

Refactor Practice Exercise
Code needs to be well-indented

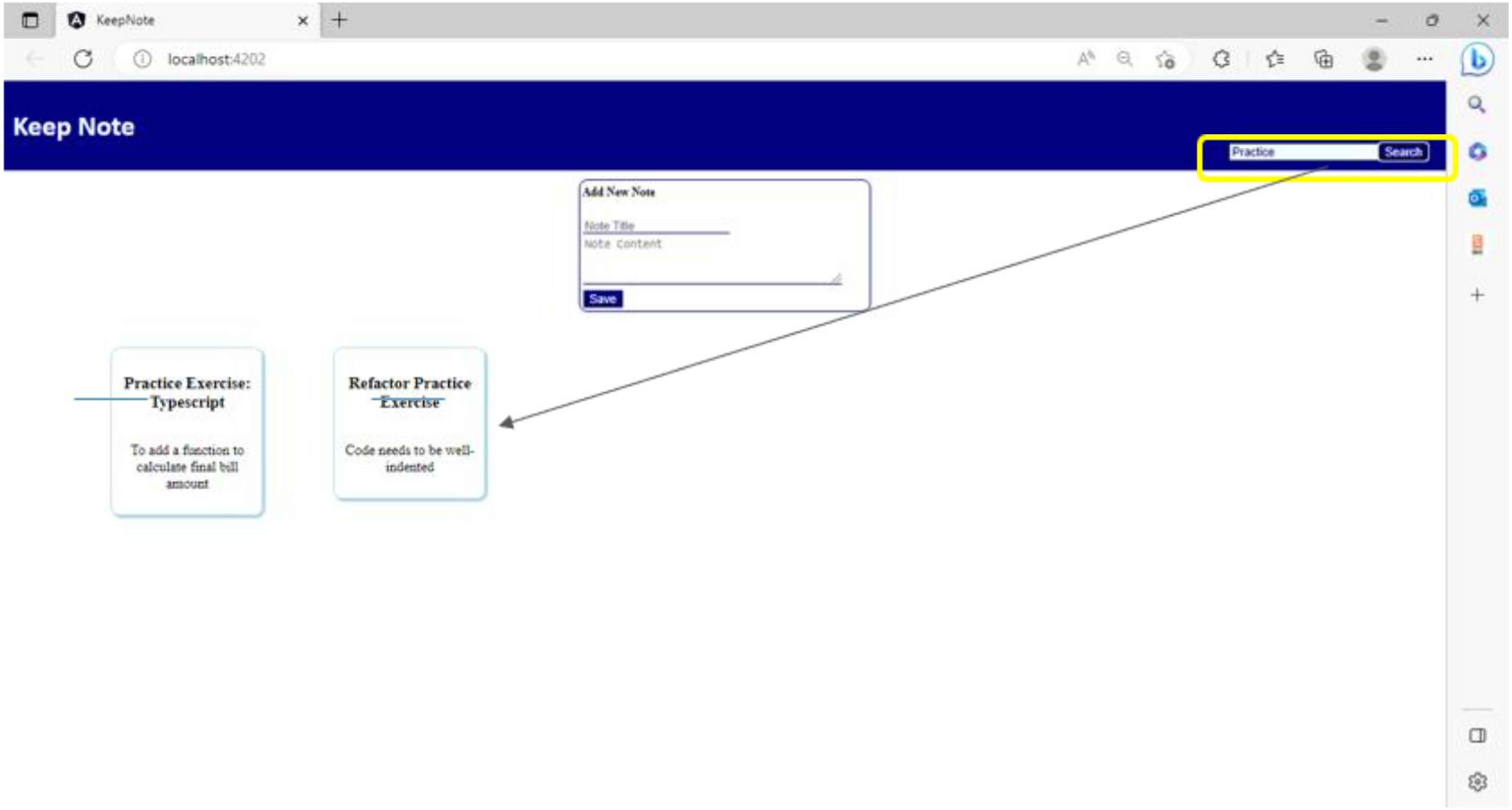
Refactor Challenge
Make the design responsive and add comments in the code

Task 5: Search Note

- The `Search` component should allow the user to search a note by its title.
- When the user clicks the search button, the `Search` component should emit an event with the search input.
- The `Note-View` component should listen to the event fired by the `Search` component and search for the note in the notes data.
  - If found, the note data should be displayed on the UI.
  - If the search input is empty, all the notes should be displayed on the UI.

Note: The search data is case-sensitive.

Task 5: Expected Output



KeepNote

localhost:4202

Keep Note

Practice

Search

Add New Note

Note Title

Note Content

Save

Practice Exercise: Typescript

To add a function to calculate final bill amount

Refactor Practice Exercise

Code needs to be well-indented

Task 6: Add a New Note

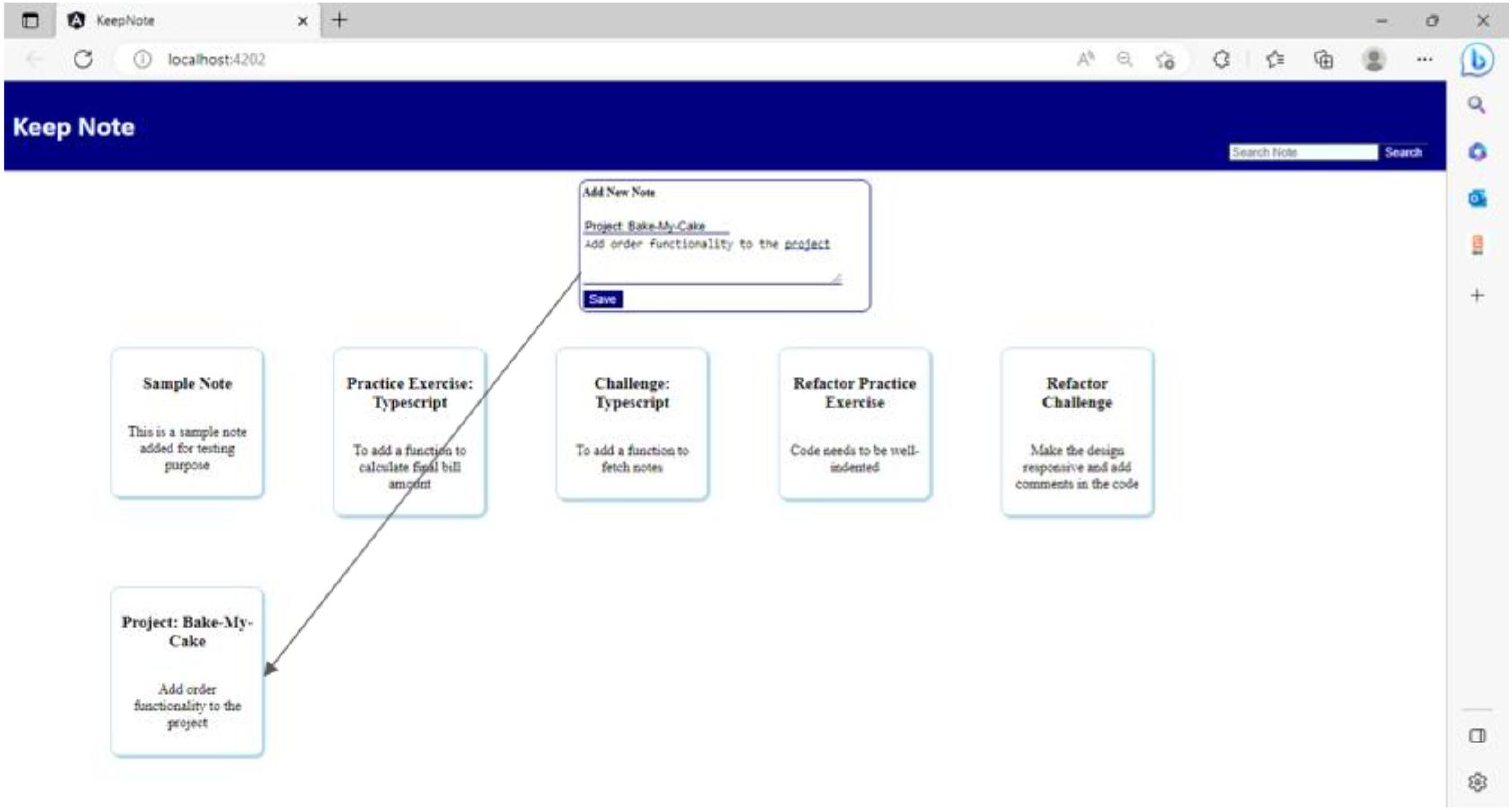
- The `Add-Note` component should handle the responsibility of adding a new note to the notes array.
- The component should allow a user to input the note details: `title` and `content` values.
- The details entered should be saved to the array in the **`notes.ts`** file.
- The newly added note should be displayed along with the existing notes, by the `Note` component.

Notes:

Since persistence is not implemented, the newly added note data will be removed upon refreshing the page.

The field `id` of the note will not be stored.

Task 6: Expected Output



Project: Bake-My-Cake
Add order functionality to the project

Sample Note
This is a sample note added for testing purpose

Practice Exercise: Typescript
To add a function to calculate final bill amount

Challenge: Typescript
To add a function to fetch notes

Refactor Practice Exercise
Code needs to be well-indented

Refactor Challenge
Make the design responsive and add comments in the code

Project: Bake-My-Cake
Add order functionality to the project

Test the Solution Locally

Test the solution first locally and then on `hobbes`. Steps to test the code locally are:

- From the command line terminal, set the path to the folder containing cloned boilerplate code.
- Run the command `ng test` or `npm run test` to test the solution locally and ensure all the test cases pass.
- Refactor the solution code if the test cases are failing and do a re-run.
- Finally, push the solution to git for automated testing on `hobbes`.