



Challenge **Guard Routes in an SPA**



Challenge

- Challenge: Guard routes in the Keep-Note application.

Points to Remember

- Only the authenticated users should be allowed to view, add, edit or delete notes.
 - The unauthenticated users should be redirected to the login view for authentication, if they try to navigate to the views related to the notes data.
- The code should be well indented for better readability.
- Appropriate comments should be added to increase readability.

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.
- Copy the files from the app folder of the Keep-Note solution developed for the challenge of the previous sprint - Sprint 4: Implement Navigation using Angular Routing .
 - Paste these files in the app folder of the boilerplate code.

Notes:

1. The solution of this challenge will undergo an automated evaluation on the CodeReview platform. (Local testing is recommended prior to testing on the CodeReview platform).
2. 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 9.
- In this stage, the routes of the Keep-Note application need to be guarded using `CanActivate` and `CanDeactivate` guards.

Guard Routes in the Keep-Note Application

In this final stage of development of the Keep-Note application, the user should first log in and get the credentials validated. Upon successful validation, the user should be allowed to access views that permit interactions with the notes data.

The application should also seek confirmation from the user before leaving the edit note view with unsaved changes.

Note: The activities to develop the above solution are given in the upcoming slide.

CHALLENGE



Activities

Following activities need to be completed, to develop the solution for the Keep-Note application:

- Activity 1: Guard routes with restricted access.
- Activity 2: Prevent losing unsaved changes using guard.

Notes:

1. Each activity will have its own tasks.
2. The partial instructions to complete these activities are given in the upcoming slides.

Activity 1: Guard Routes With Restricted Access

- In the Keep-Note application, add a service with the name `AuthService`. The details about this service are given below:
 - The service class should be in the `services` folder along with the other service classes.
 - Inside the `AuthService`:
 - ❖ Define a method `login()` that validates the users' login credentials (`email` and `password`) and sets the users' login status accordingly.
 - For this challenge, you can use dummy values for login credentials.
 - ❖ Define a method `logout()` that allows users to logout.
 - ❖ Define a method `isLoggedIn()` that checks for user's logged in status.

Note: The service class name and the method names mentioned above are used in testing, so you must use the same names while coding.

Activity 1: Guard Routes With Restricted Access (cont'd)

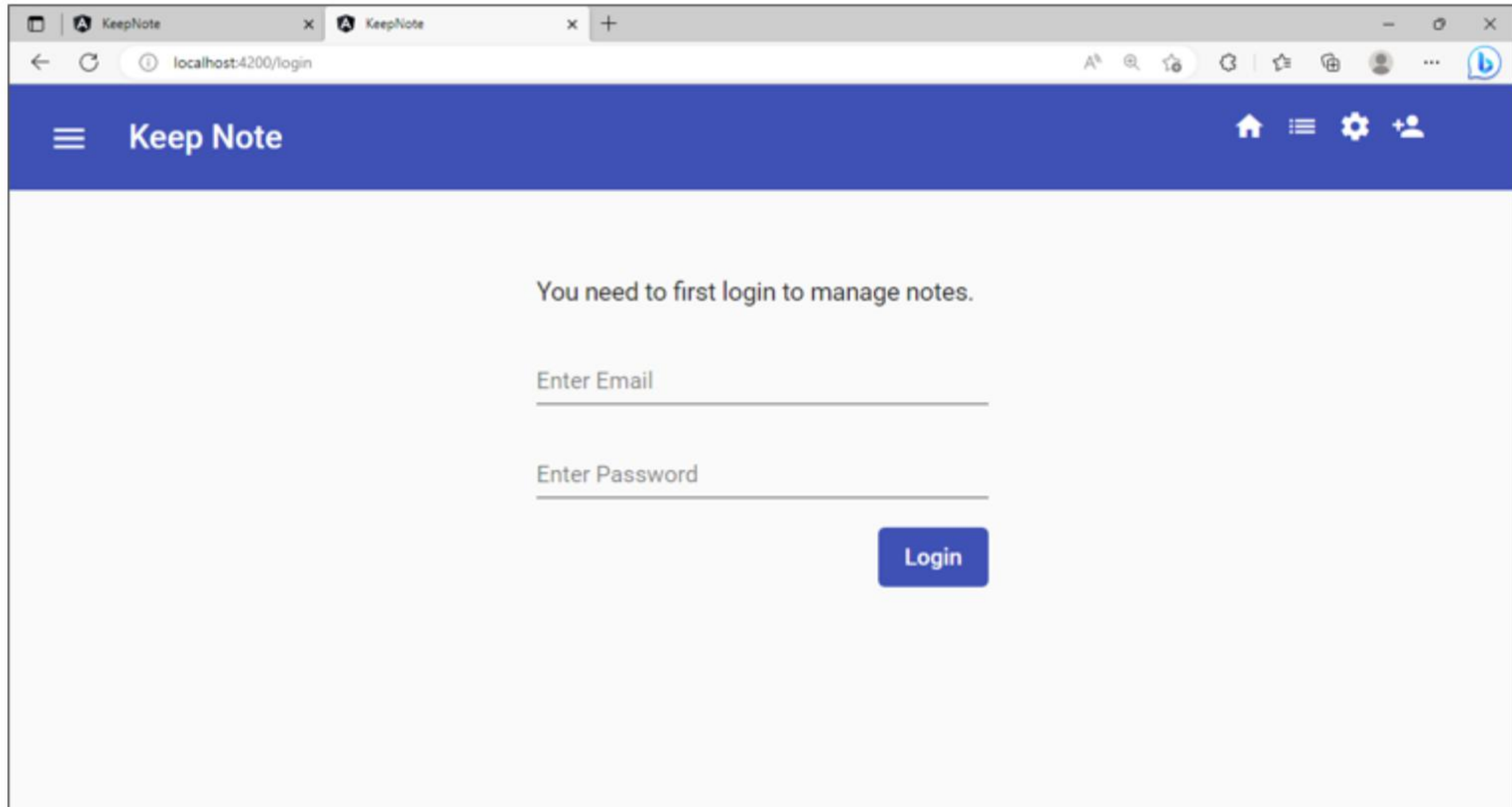
- Add a Login Component that handles the following responsibilities:
 - The login view should allow the users to enter their email and password as login credentials.
 - It should have a button to submit and get the login credentials validated by the `login()` method of `AuthService`.
- Define the route to navigate to the Login component
- Enable programmatic navigation to the Login component by defining the method `navigateToLoginView()` in the `RouterService`.

Note: The component name and the method names mentioned above are used in testing, so you must use the same names while coding.

Activity 1: Guard Routes With Restricted Access (cont'd)

- Define `CanActivate` route guard with the name `AuthGuard`. The details about the guard are given below:
 - The guard should return value `true` for logged in users otherwise redirect users to the `login` view and return value `false`.
 - ❖ Call the appropriate method from the `AuthService` to check the logged in status of the user.
 - ❖ Call the appropriate method from the `RouterService` to enable navigation to the `login` view
- Apply guard to routes with restricted access based on the details given below:
 - The access to all the views handling notes data should be restricted.
 - When the application gets launched, the `login` view should be the default view.
 - Post successful login, the application should navigate the users to the notes view.
 - The notes view should have provision to allow the users to logout.
 - Once logged out, the users should again be navigated to the `login` view.
 - Without logging in, users should not be provided access to the notes view.

Expected Output – Login View



The screenshot shows a web browser window with two tabs, both labeled 'KeepNote'. The address bar displays 'localhost:4200/login'. The page features a blue header with a hamburger menu icon, the text 'Keep Note', and navigation icons for home, list, settings, and user. The main content area has a light gray background and contains the following text and form elements:

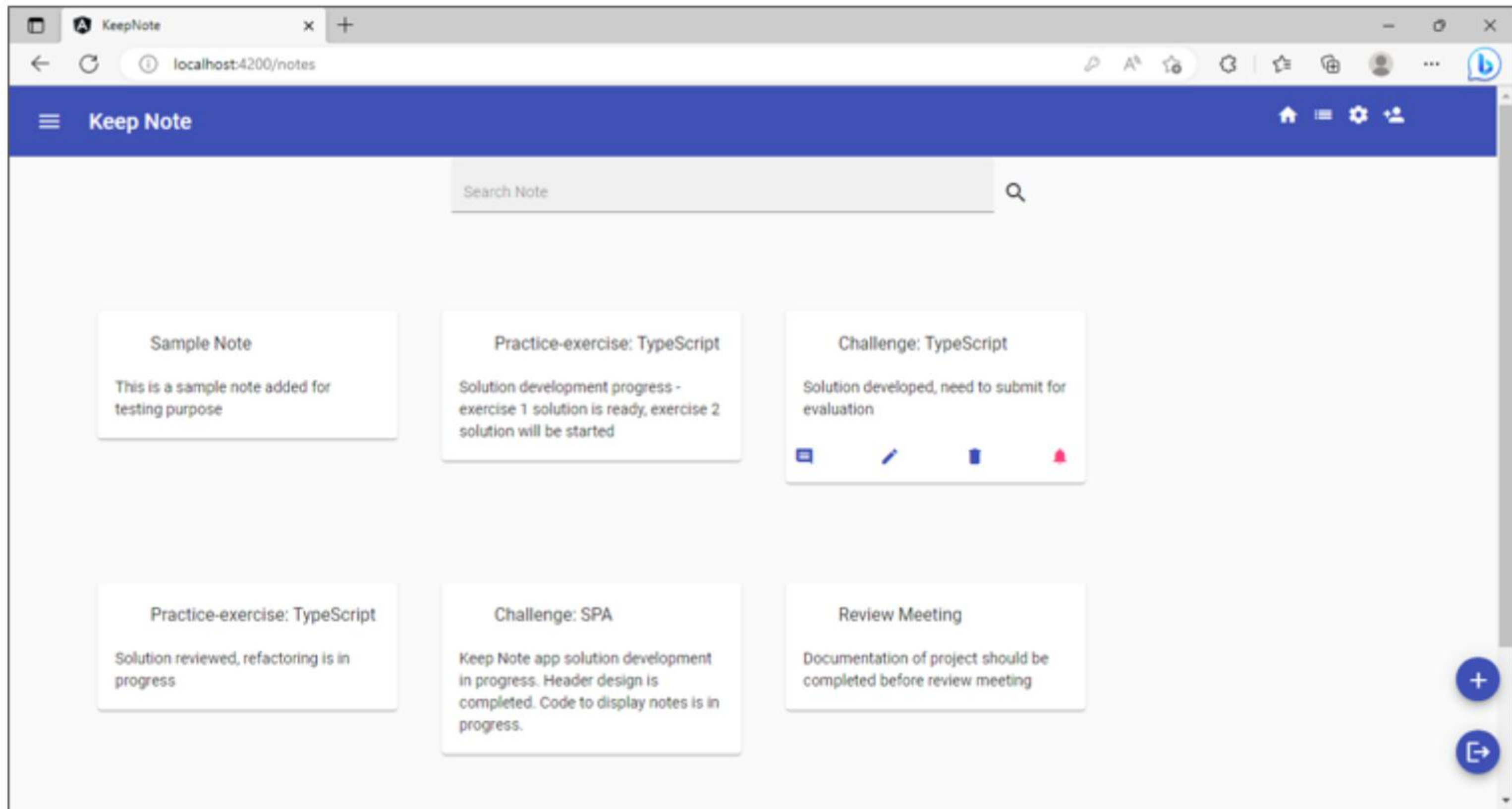
You need to first login to manage notes.

Enter Email

Enter Password

Login

Expected Output – Notes View (After Successful Login)



Activity 2: Prevent Losing Unsaved Changes Using Guard

- Add `CanDeactivate` route guard with the name `CanDeactivate`. The details about the guard are given below:
 - The guard should call the method `canDeactivate()` of the target component and return the value returned by that method.
- Refactor the component that edits note to implement a confirmation workflow for unsaved edit changes. The details to refactor are given below:
 - Add `canDeactivate()` method that seeks confirmation from the user if changes are not saved and accordingly returns a `boolean` value.

Note: The guard name and the method name mentioned above are used in testing, so you must use the same names while coding.

Expected Output – Edit View (Leaving without Saving)

