Challenge

Developing Interactive Template-Driven Forms Inside SPA

Challenge

- Challenge: Develop a template-driven form to add a note in the Keep Note application

Points to Remember

- The form control elements must be created using Angular material components.
- All the required modules to work with Angular forms and material components should be imported in the application root module.
- Custom styles should be added while designing the form and displaying the notes as cards.
- HTML5 validation attributes should be used to validate form input values.
- The form should be reset once the new note is successfully added.
- Run the `json-server` to add a note data to the `notes.json` file in the **`keep-note-data`** folder.

Instructions for Challenge

- Click here for the boilerplate.
- Please read the README.md file provided in the boilerplate for further instructions about the challenge.
- Fork the boilerplate into your own workspace.
- Clone the boilerplate into your local system.
- Run the command `npm install` to install the dependencies.
- Use the solution code of the Keep Note application developed of the challenge of the sprint: `Style a single page application using Angular Material`

Notes:

1. The solution to this challenge will undergo an automated evaluation on `CodeReview` platform. (Local testing is recommended prior to `CodeReview` testing).
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 a simple form with validation in the `Keep-Note` application.

Stage 7: Create a 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 6.

In the previous stage, for pleasing aesthetics and a rich user experience, the looks of the `Keep-Note` application had been enhanced by using the Angular Material library.

In this stage, a template-driven form should be created to add a new note with additional form fields in the `Keep-Note` application. The form controls should be validated according to the requirements specified.

Develop a Template-Driven Form to Add a Note in the Keep Note Application

Develop a template-driven form to add a new note. The data model for a note should include the following properties: title, content, reminder date, category, and priority level. Priority levels can be low, medium, high, or critical.

Note: The tasks to develop the form are given in the upcoming slides.

CHALLENGE

Tasks

- The following tasks need to be completed to develop the solution for the Keep-Note application:
  - Task 1: Modify the data model
  - Task 2: Include the required modules
  - Task 3: Modify the `NoteAddComponent`
  - Task 4: Define the form layout inside the template
  - Task 5: Handle form validation
  - Task 6: Display a notification message on successful form submission
  - Task 7: View the newly added note

Note: Details related to a few tasks are given in the upcoming slides.

Task 1: Modify the Data Model

- Modify the data model **`Note`** that reflect the form data model.
- Modify type `Note` in the `note.ts` file in the **`models`** folder, which should have the following type properties:
  - id (number)
  - title (string)
  - content (string)
  - reminderDate (string)
  - category (number)
  - priority (string)

Task 5: Handle Form Validation

- The following are the add note form validation requirements:

Form Control	Validation	Error Messages
Note Title	Should not be blank	- Note title is required
Note Content	Should not be blank and have minimum length of 5 characters	- Note content is required - Note content should have minimum 5 characters
Reminder Date	Should not be blank and date value should be greater than or equal to today's date	- Reminder date is required
Category	No Validation	Nil
Priority Level	Value should be one among the 4 values - Low, Medium, High or Critical	Nil

Add **`name`** attribute with values matching the model properties to each form control, which Angular uses to register the element with the parent `<form>`.

Notes:

1. The component name (`NoteAddComponent`), and the form controls with `name` properties are used in testing, and hence must use the same name while coding.
2. Error messages text should be used as mentioned above as these texts are used in testing.

Task 7: View the Newly Added Note

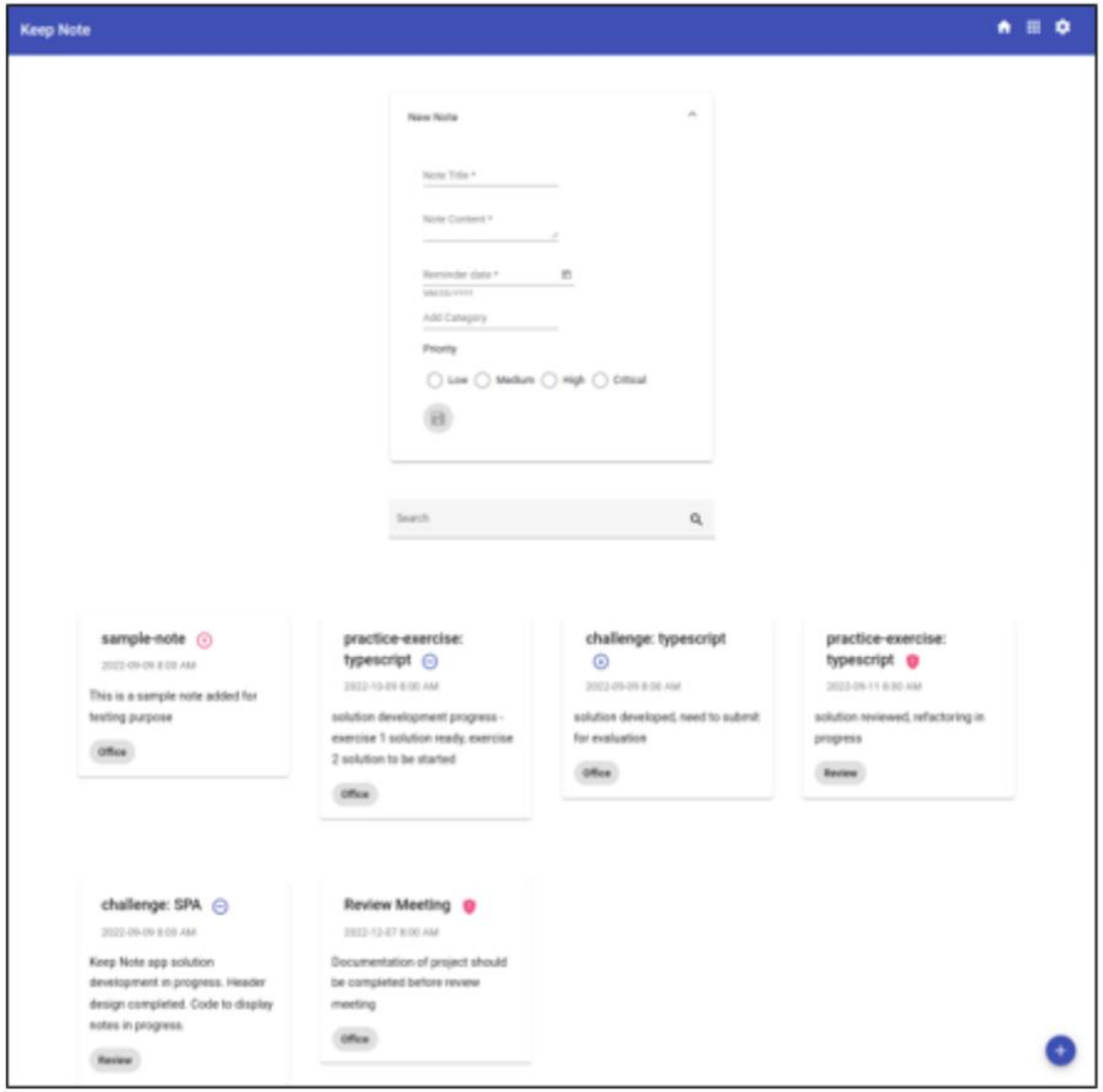
- The following mat-icons should be used for displaying the priority levels while adding the note to a card.

Priority Level	Mat-Icon
Critical	gpp_maybe
High	arrow_circle_up
Medium	remove_circle_outline
Low	arrow_circle_down

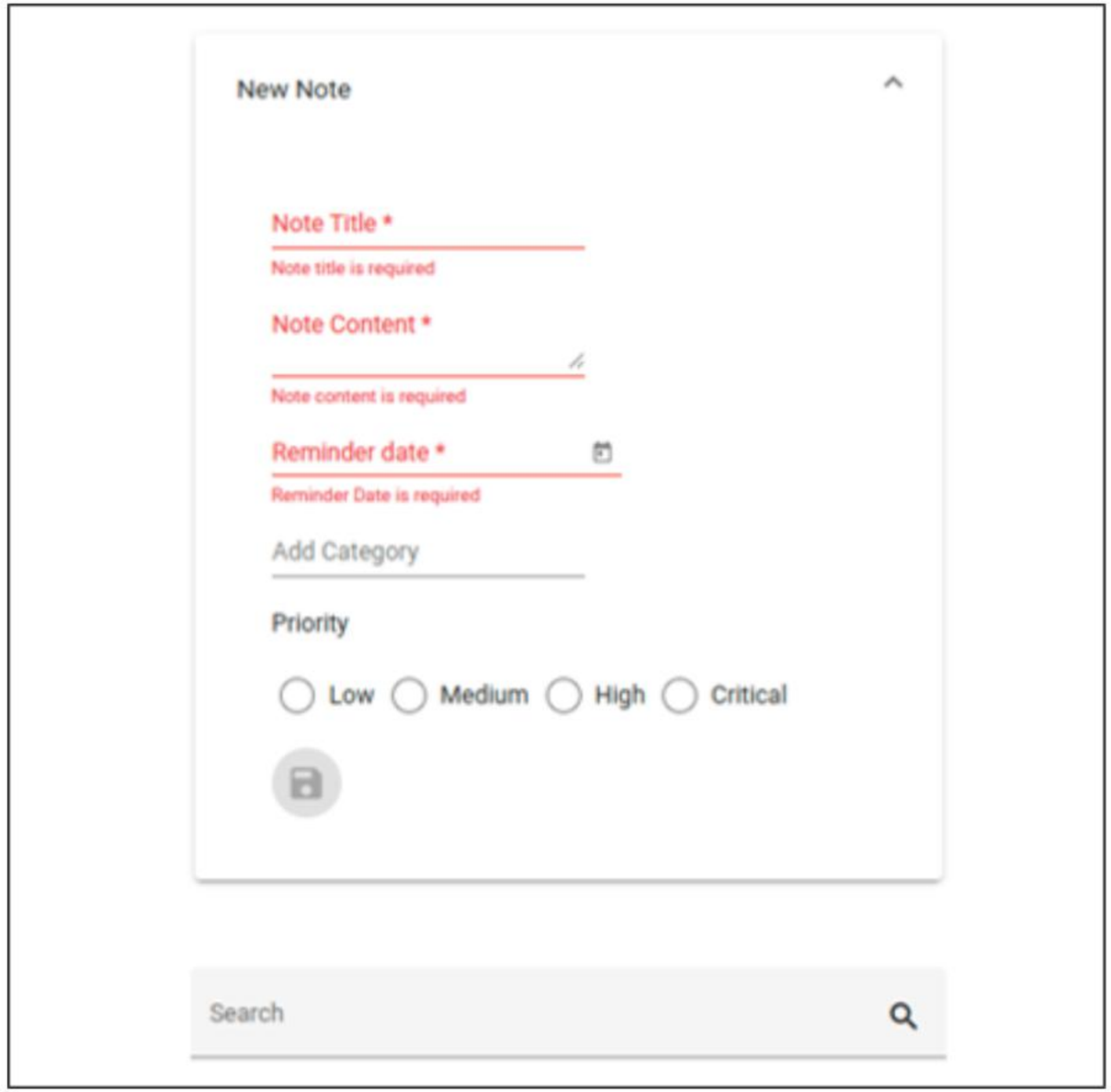
- In `note.component.html`, priority levels should be displayed next to the note title using these mat-icons and the `*ngIf` directive.

```
<button *ngIf="note?.priority=='Critical'" color="accent"
mat-icon-button>
   <mat-icon>gpp_maybe</mat-icon>
</button>
```

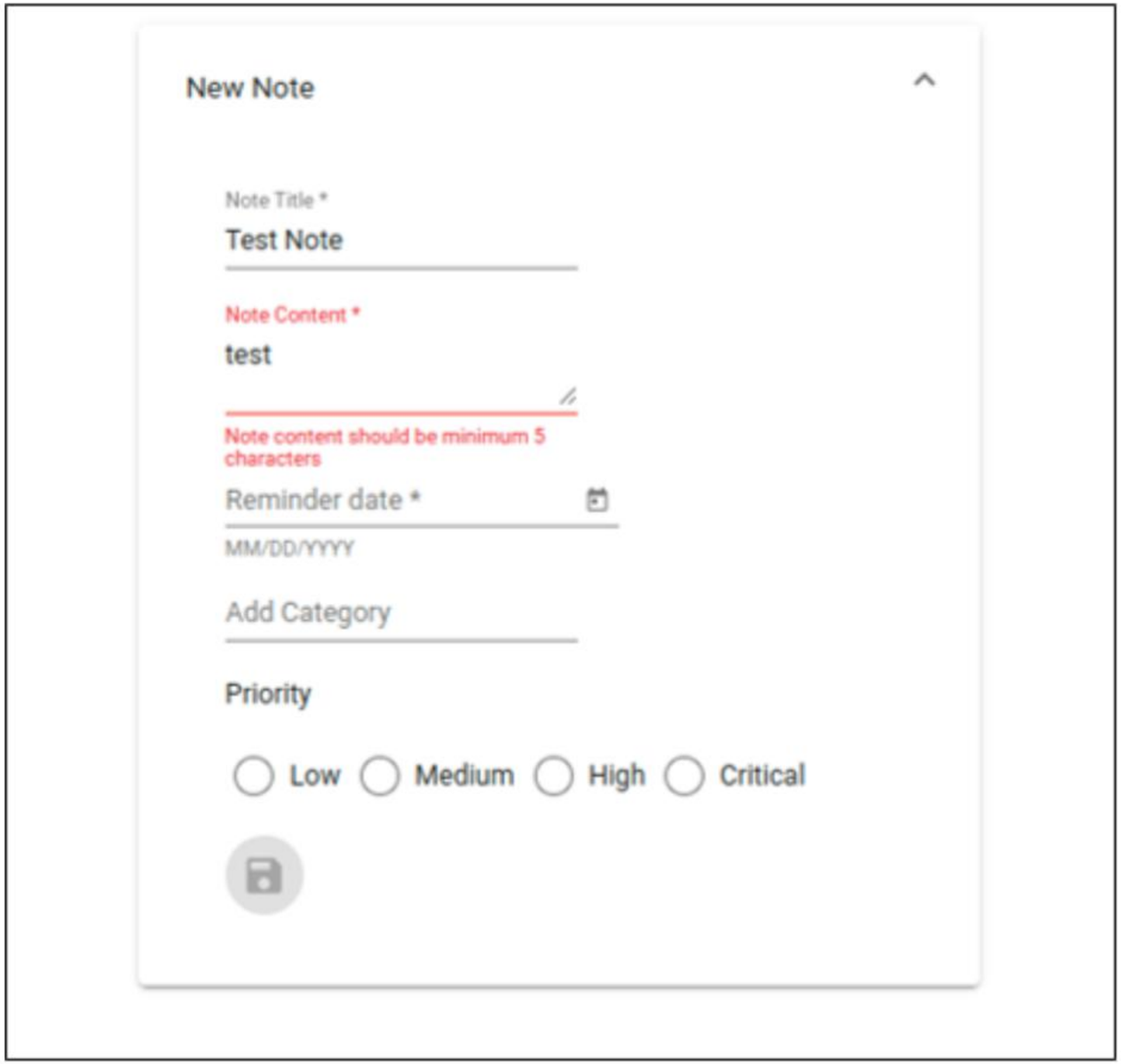
Expected Output After Task 4 (Add Note Form)



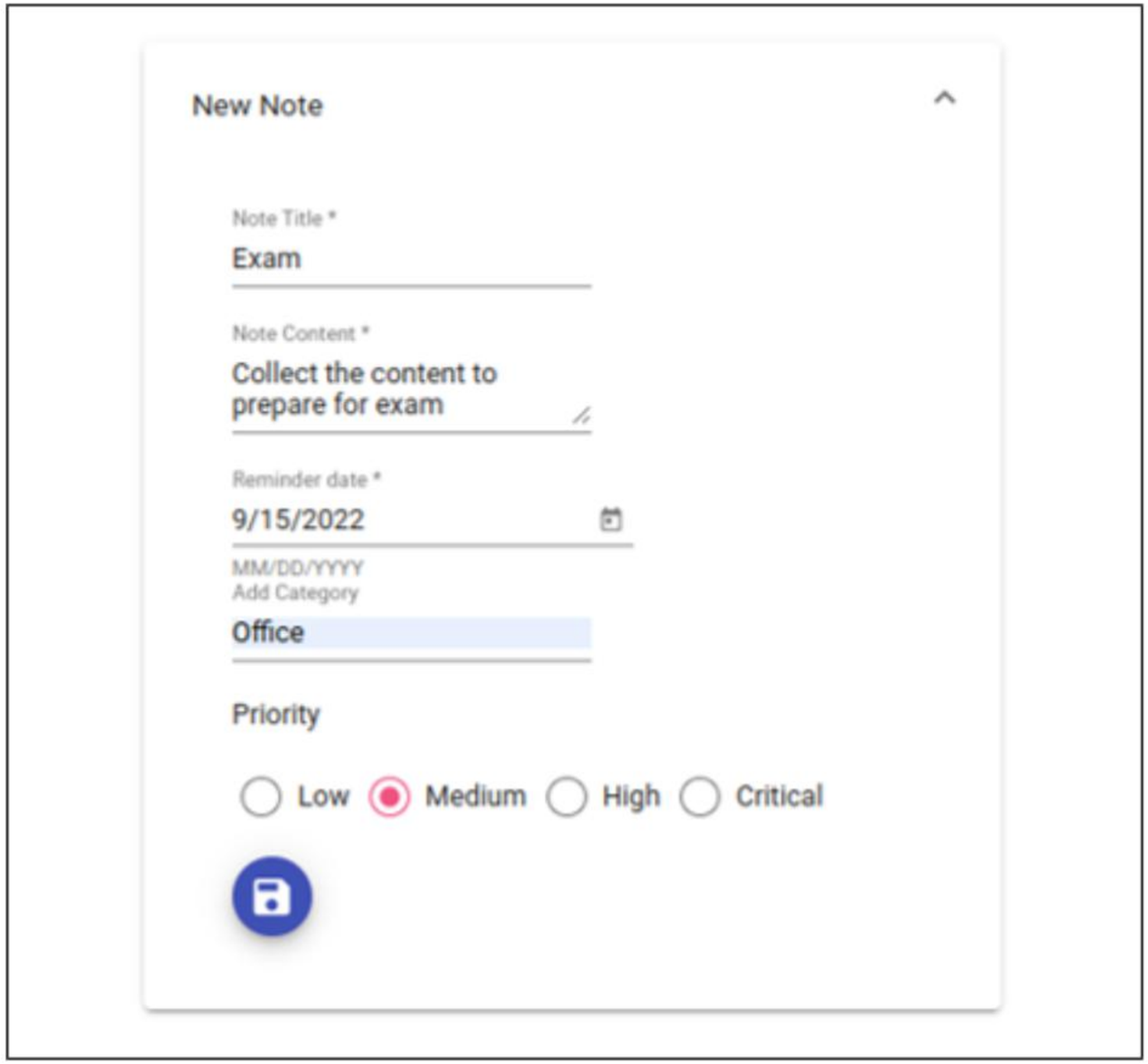
Expected Output After Task 5 (Form With Validation Errors)



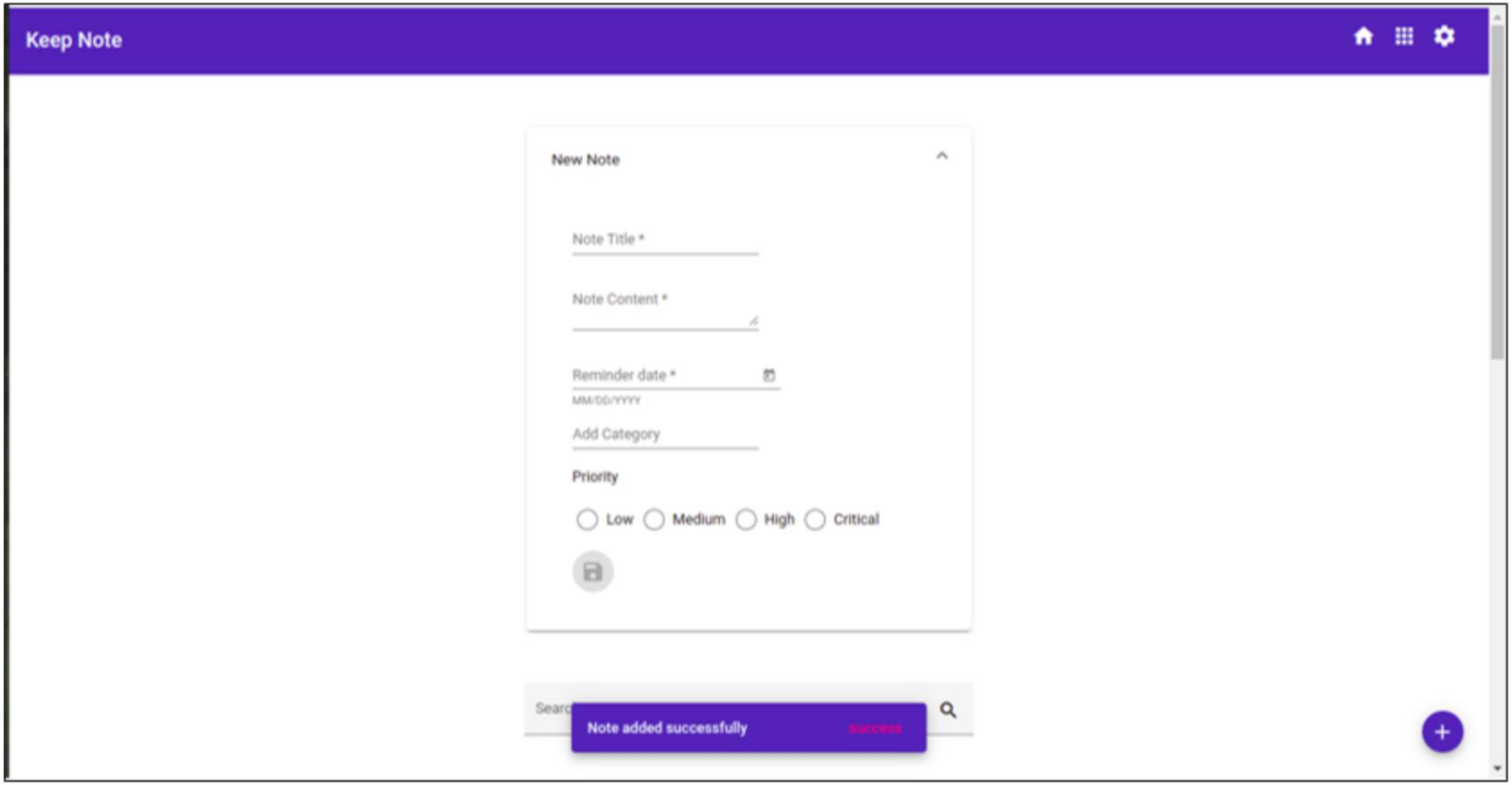
Expected Output After Task 5 (Form With Validation Errors)



Expected Output After Task 5 (Form With Valid Values)



Expected Output After Task 6 (Successful Form Submission)



Expected Output After Task 7(Newly Added Note)