

GROUP 3

June 15, 2024

Mobile Programming



Tran Anh Vu
Pham Ngoc Thuy Nhung
Doan Thi Thuy Linh

Q Table of content

1

INTRODUCTION
REQUIREMENT

3

ISSUES AND OVERCOME


2

PROCESS MAKING

4

RESULTS

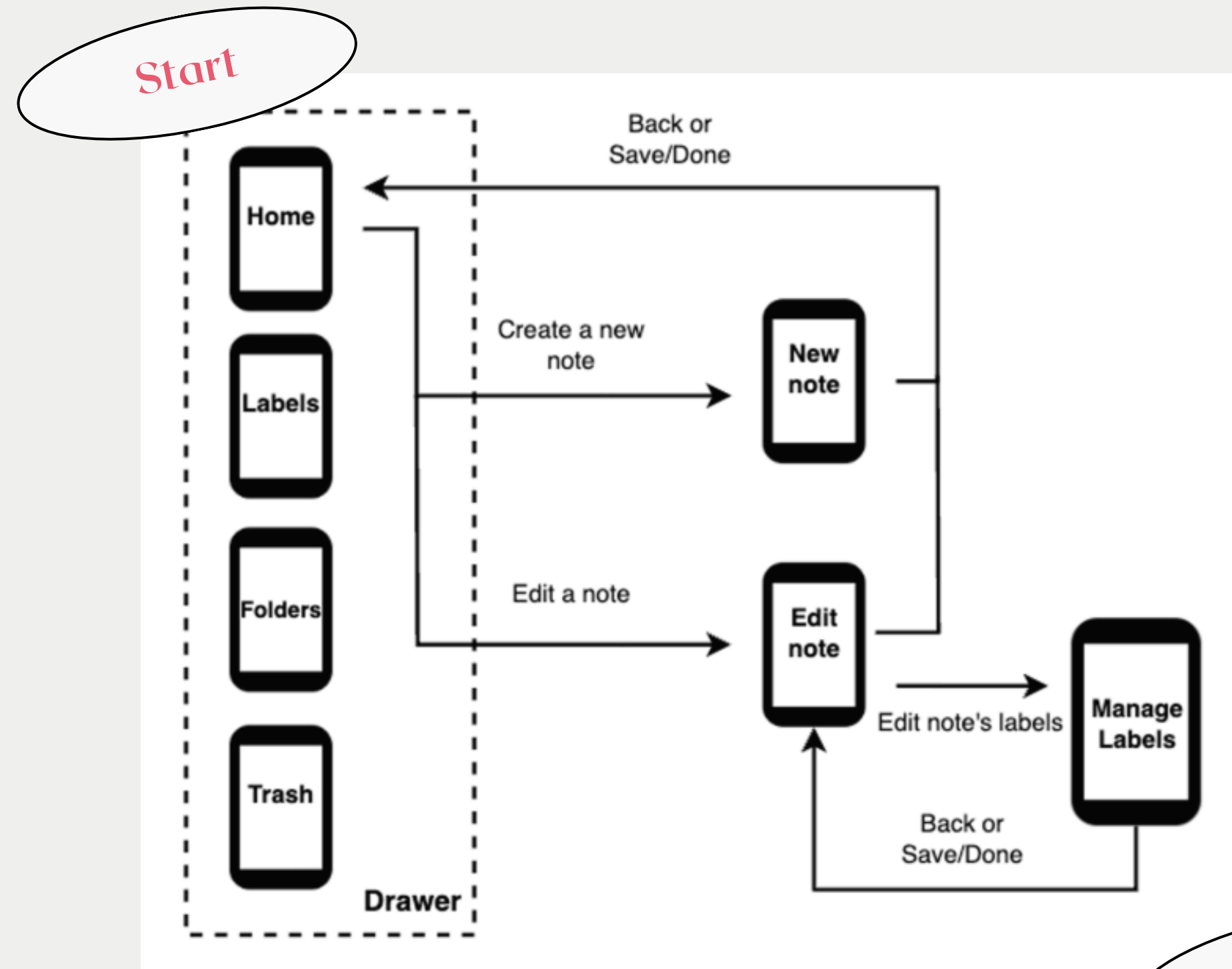
INTRODUCTION REQUIREMENT



Are you
ready?

Navigation of apps

Analysis requirement



7 core components

6 mandatory, 1 optional

INTRODUCTION REQUIREMENT

Home Screen

- Fr1** Display all notes (scrollable list), handle empty state.
- Fr2** Navigate to New Note screen.
- Fr4** Search notes by content.

New note screen

Input for note content

Edit note screen

- Fr3.1** Display and update note content.
- Fr3.2** Toggle bookmark status.
- Fr3.3** Bottom sheet menu for color, labels, and delete.
- Fr3.4** Navigate to Manage Labels screen.
- Fr3.5** Move note to Trash.

Manage Labels Screen

- Display available labels as buttons.
- Link/unlink labels to the current note.

Labels Screen

- Fr5** Search labels, create new labels.
- Fr6** Edit (update/delete) existing labels

Trash Screen

- Fr7** Display trashed notes.
- Fr8** Restore or permanently delete individual notes.
- Fr9** Restore all or empty Trash.

Folders Screen (Optional)

INTRODUCTION REQUIREMENT

UI Design

These are UI requirements



PROCESS MAKING



Follow
me

UI Design

Home Screen

This is our version android User Interface

Home Screen

- Fr1** Display all notes (scrollable list), handle empty state.
- Fr2** Navigate to New Note screen.
- Fr4** Search notes by content.

Navigation bar

Fr1: List all existing notes

Fr1: Handle empty state

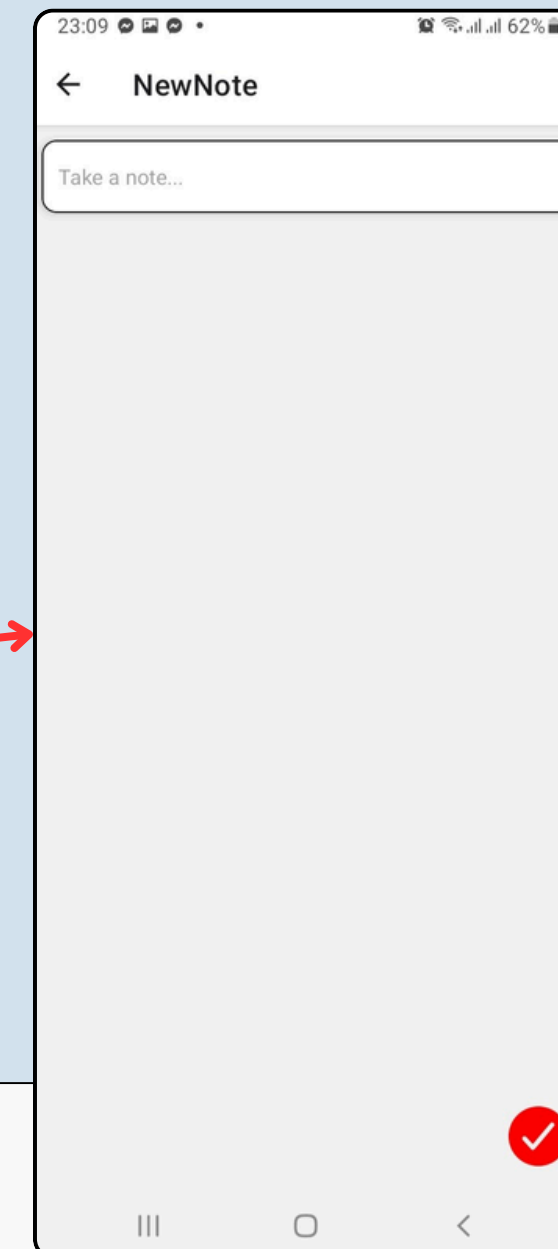
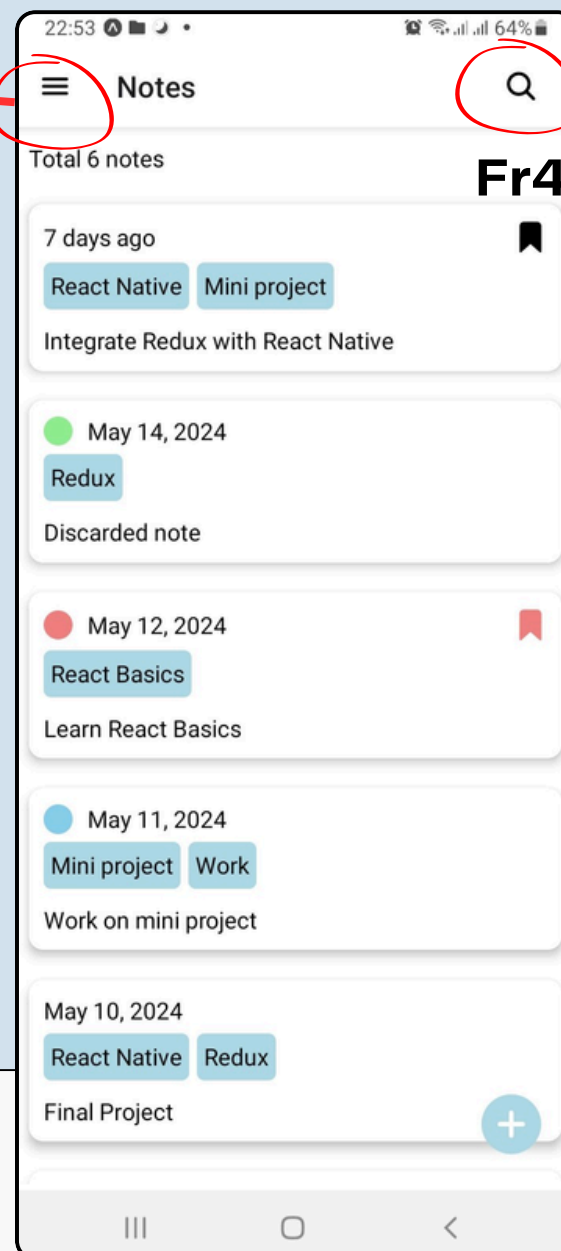
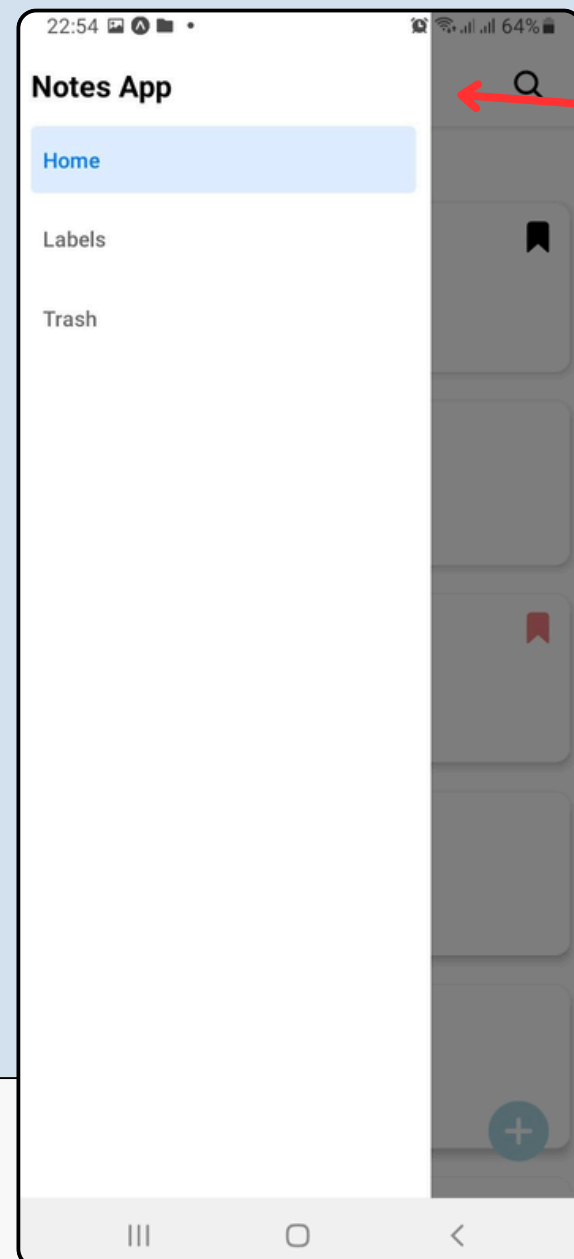
Fr2: Navigate to New Note screen

Fr4: Search

Input text and an action button to allow the user to add a new note

New Note screen

PROCESS MAKING



UI Design

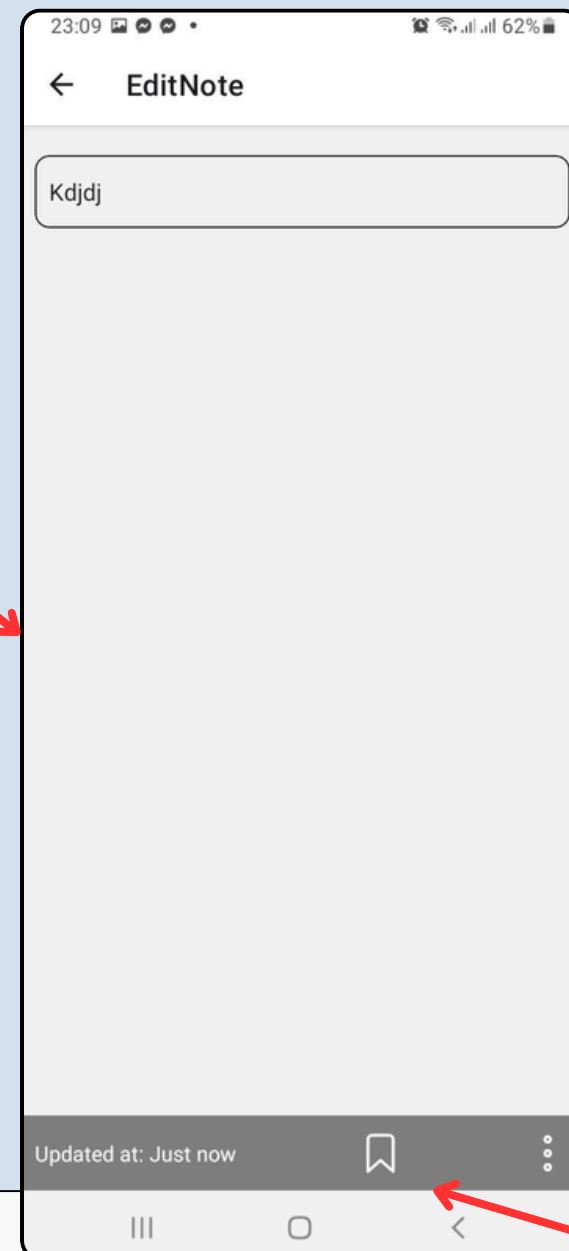
This is our version android User Interface

Edit Note screen

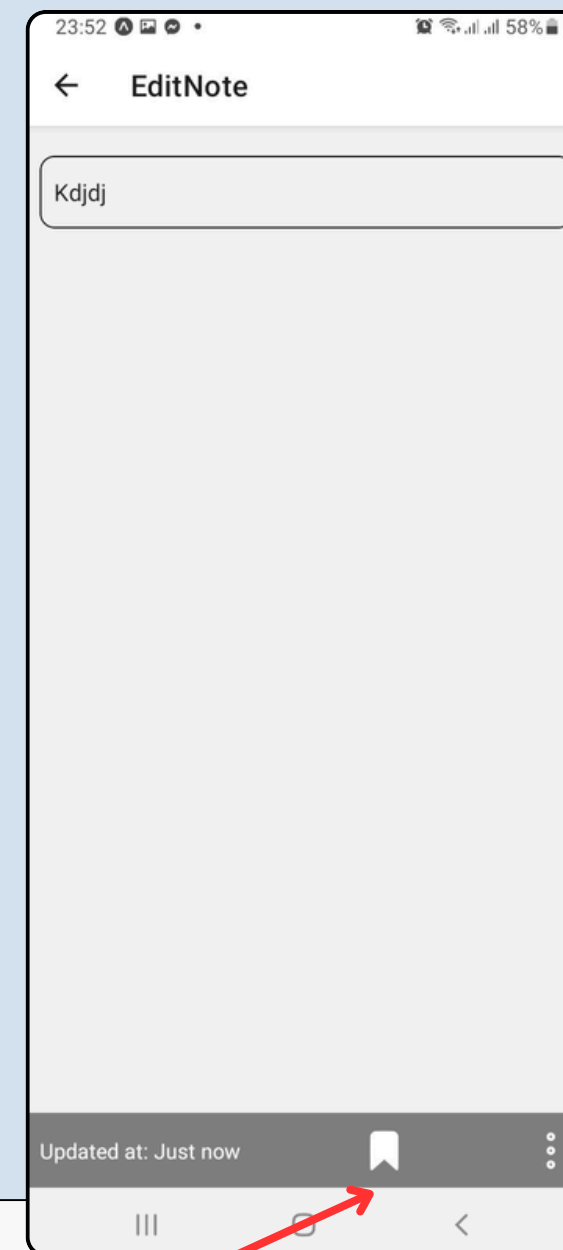
Fr3.1: Display and Update
Note content



Click on note



Fr3.2: Toggle bookmark status



Edit note screen

- Fr3.1** Display and update note content.
- Fr3.2** Toggle bookmark status.
- Fr3.3** Bottom sheet menu for color, labels, and delete.
- Fr3.4** Navigate to Manage Labels screen.
- Fr3.5** Move note to Trash.

PROCESS MAKING

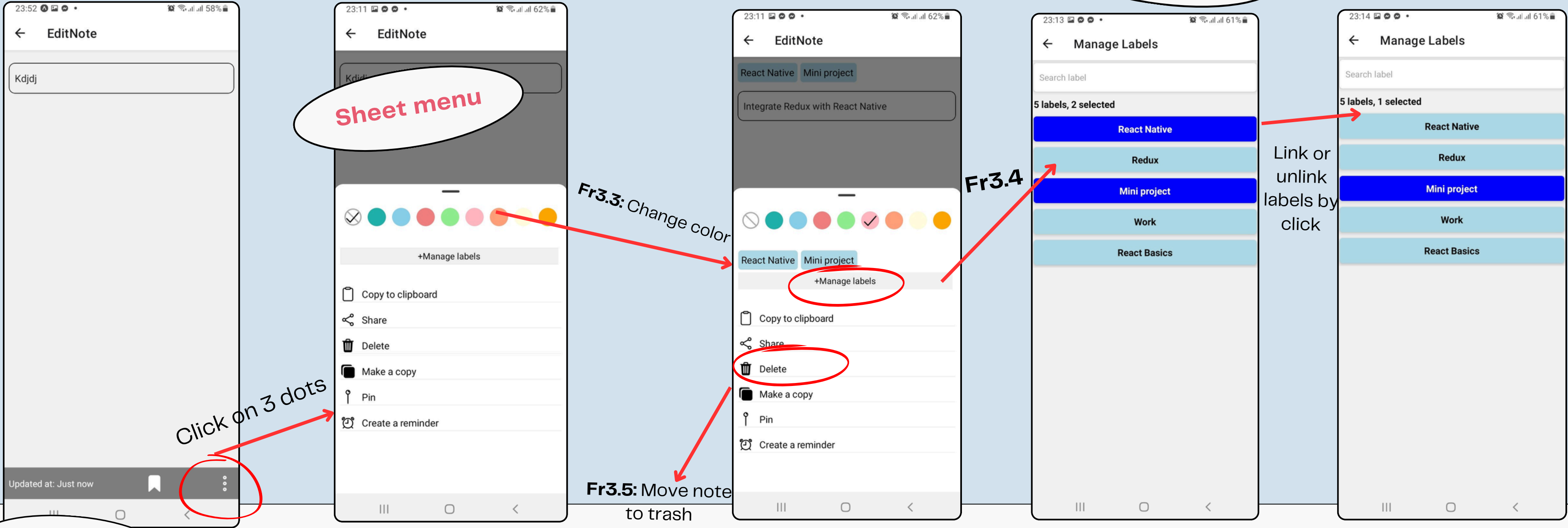
UI Design

This is our version android User Interface

Edit note screen

- Fr3.1 Display and update note content.
- Fr3.2 Toggle bookmark status.
- Fr3.3 Bottom sheet menu for color, labels, and delete.
- Fr3.4 Navigate to Manage Labels screen.
- Fr3.5 Move note to Trash.

Manage labels screen



PROCESS MAKING

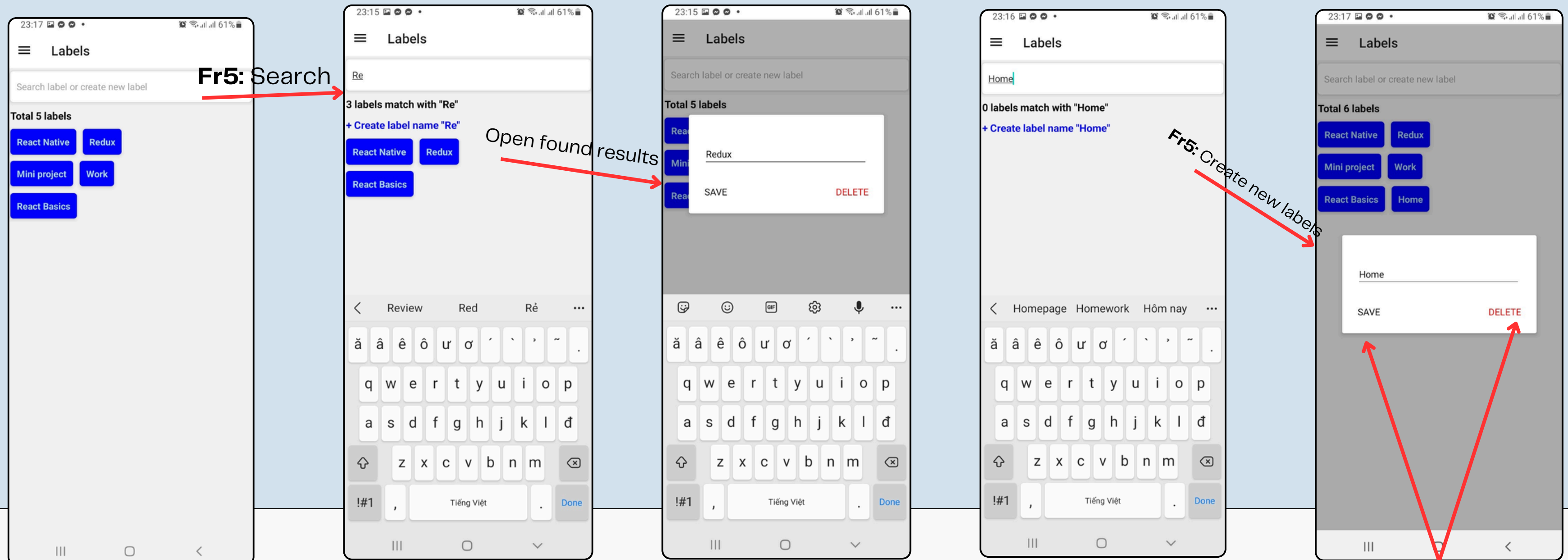
UI Design

This is our version android User Interface

Labels Screen

Labels Screen

- Fr5 Search labels, create new labels.
- Fr6 Edit (update/delete) existing labels



Fr6: Edit (update/delete) existing labels

UI Design

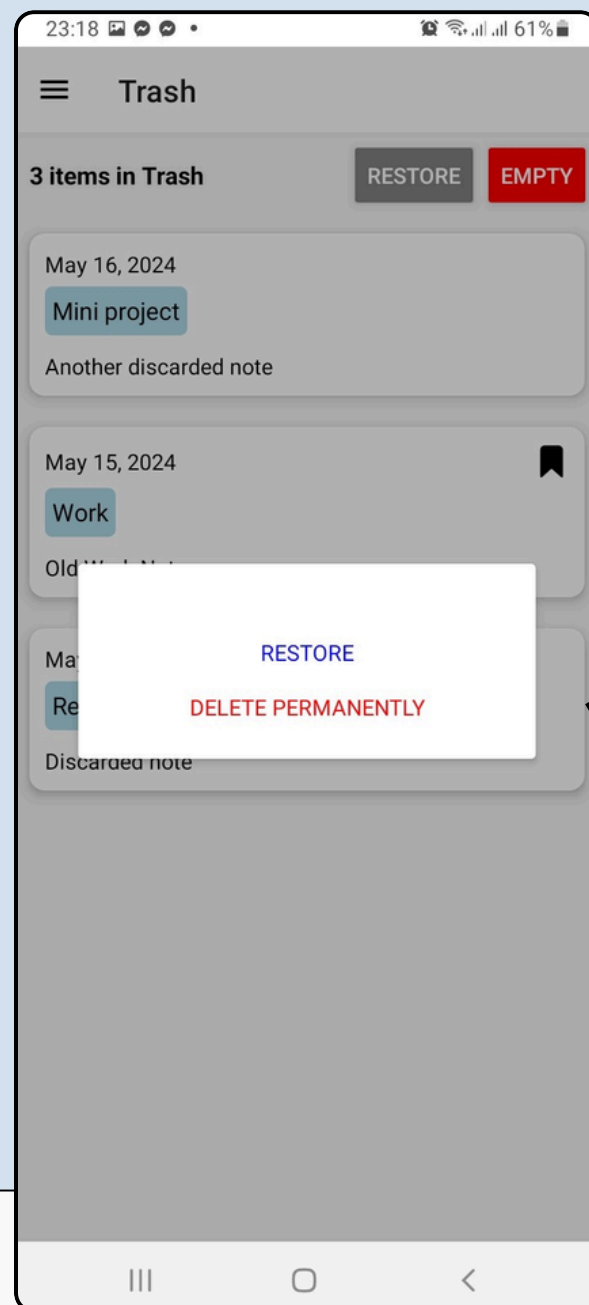
This is our version android User Interface

Trash screen

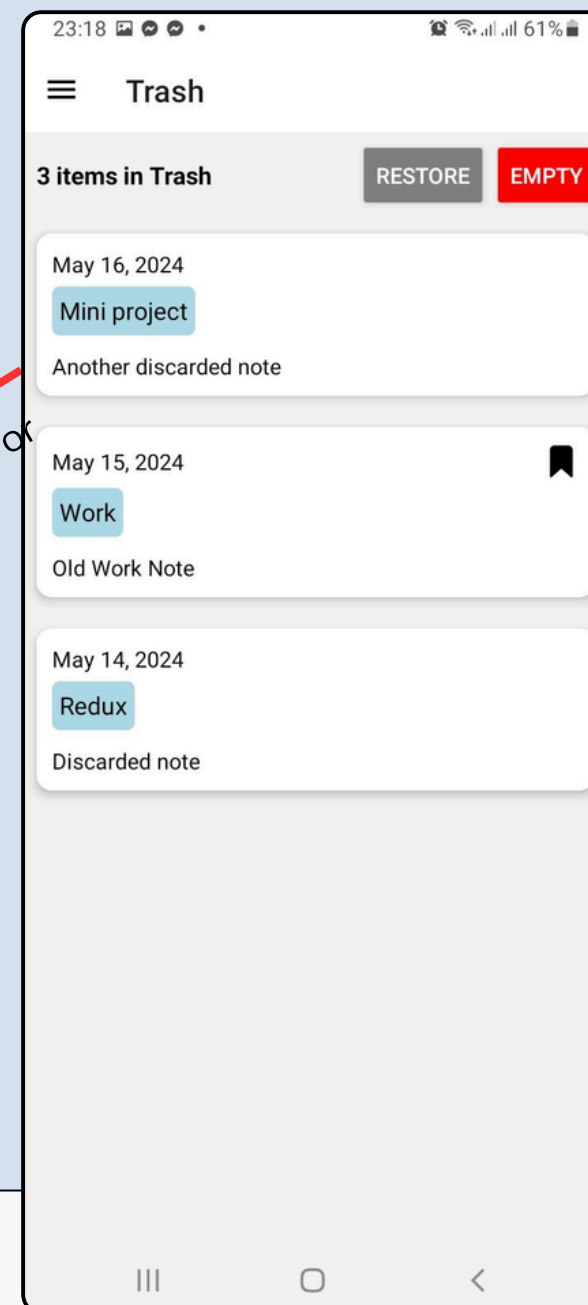
Trash Screen

- Fr7 Display trashed notes.
- Fr8 Restore or permanently delete individual notes.
- Fr9 Restore all or empty Trash.

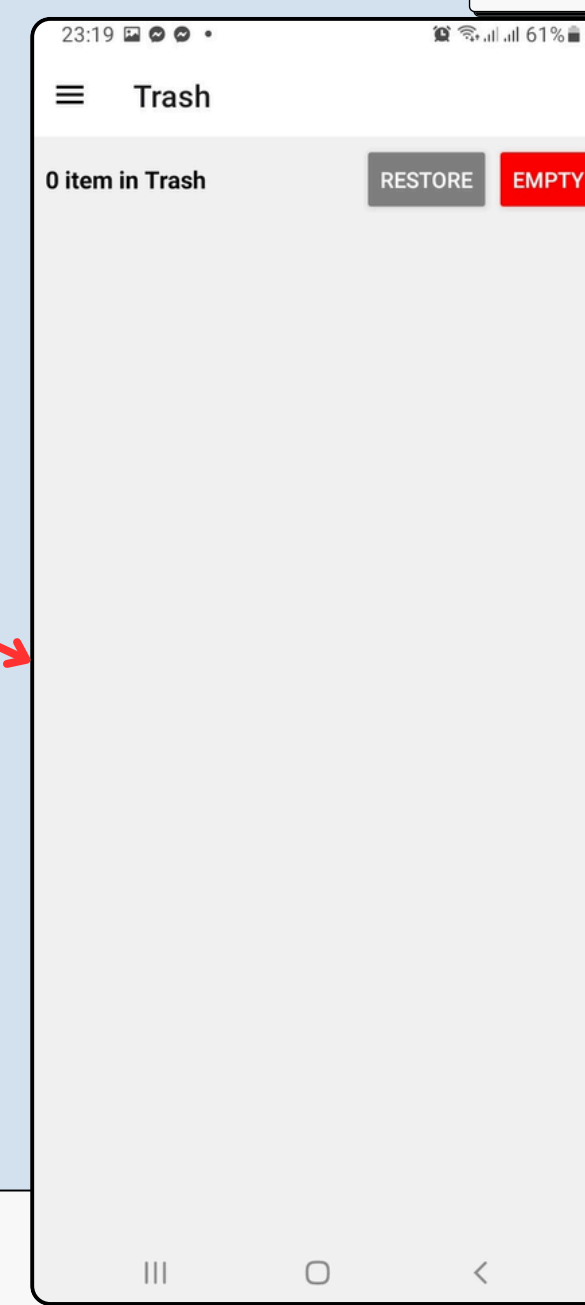
Fr7: Display trashed notes



Fr8: Click on a note to restore or delete permanently

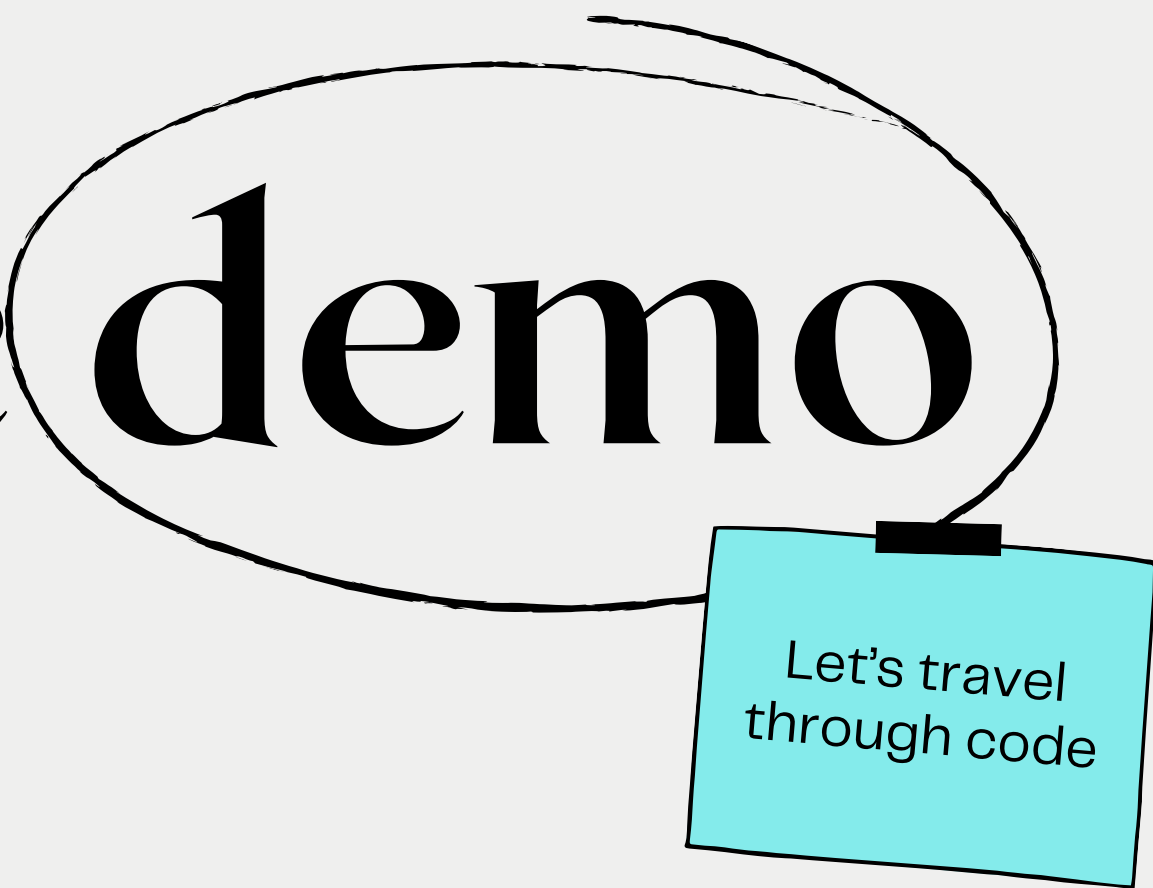


Fr9: Restore all or empty trash



PROCESS MAKING

Code demo



Let's travel
through code

Overall folder

June 10, 2024

- | | |
|-------------|----------|
| 1 component | 4 screen |
| 2 database | 5 utils |
| 3 models | 6 App.js |

```
✓ 62FIT3MPR_FINAL_GROUP3
  > .expo
  > assets
  > components
  > database
  > models
  > node_modules
  > screen
  > utils
  JS App.js M
  {} app.json
  B babel.config.js
  {} package-lock.json M
  {} package.json M
```

Code Design

Q component

This is the function of each file code

✓ components

JS CustomDrawer.js

JS ExistedNotes.js

JS LabelList.js

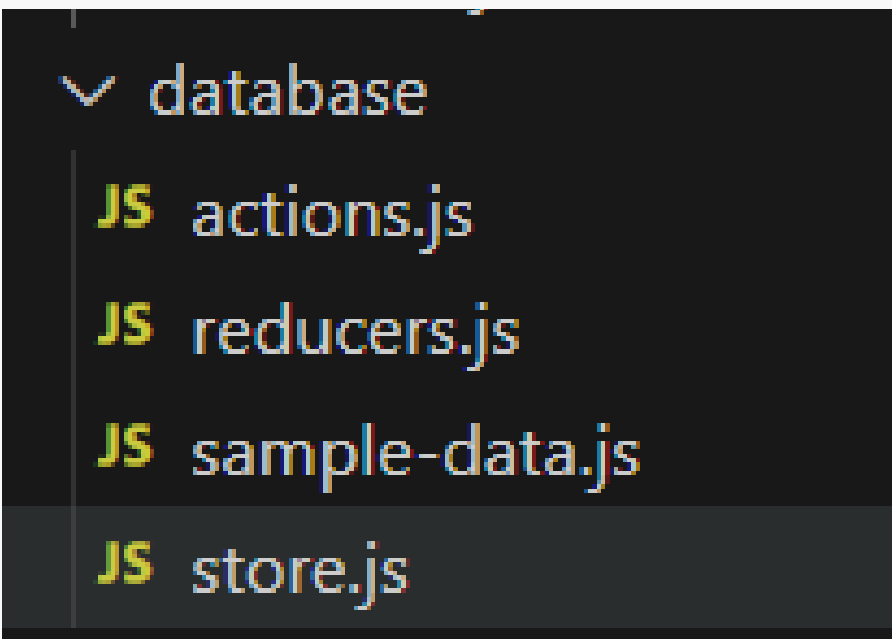
PROCESS MAKING

- **CustomDrawer.js:** Contains the DrawerComponent to display the application's navigation drawer with the title "Notes App" and a list of navigation items.
- **ExistedNotes.js:** Contains the ExistedNotes component to display a list of saved notes, each note can be selected to navigate to the editing screen. It displays information such as the update date, label, and content of the note.
- **LabelList.js:** Contains the LabelList component to display a list of labels related to a note.

Code Design

Q database

This is the function of each file code

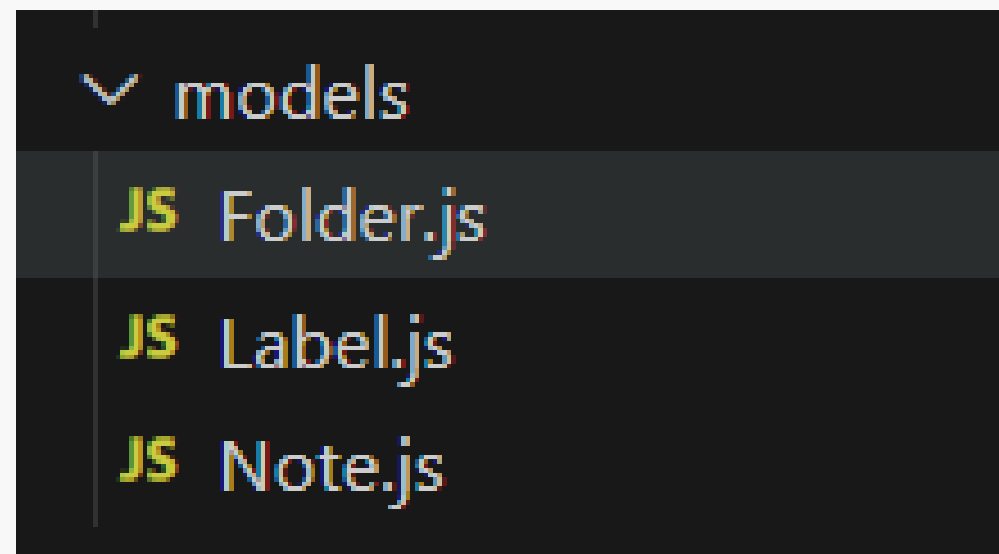


- **actions.js:** Contains action creators defining the types of actions used in the application, including adding, updating, deleting notes and labels, restoring notes, permanently deleting notes, and cleaning up the trash.
- **reducers.js:** Contains reducers handling actions defined in actions.js and updating the application's state. It includes reducers for notes, labels, and the trash.
- **sample-data.js:** Contains sample data for labels, colors, folders, and notes used to initialize the application's initial state.
- **store.js:** Contains the Redux store configuration, initializes the initial state based on sample data from sample-data.js, and combines reducers from reducers.js

Code Design

Q models

This is the function of each file code



- **Folder.js:** Defines the Folder class with attributes such as **id**, **folderName**, and **updateTime**. Folder objects are used to store information about folders in the application.
- **Label.js:** Defines the Label class with attributes such as **id** and **label**. Label objects are used to store information about labels in the application.
- **Note.js:** Defines the Note class with attributes such as **id**, **content**, **color**, **labelIds**, **updatedAt**, **isBookmarked**, and **folderId**. Note objects are used to store information about notes in the application.

Code Design

Q screen

This is the function of each file code

✓ screen

JS EditNoteScreen.js

JS HomeScreen.js

JS LabelsScreen.js

JS ManageLabelsScreen.js

JS NewNoteScreen.js

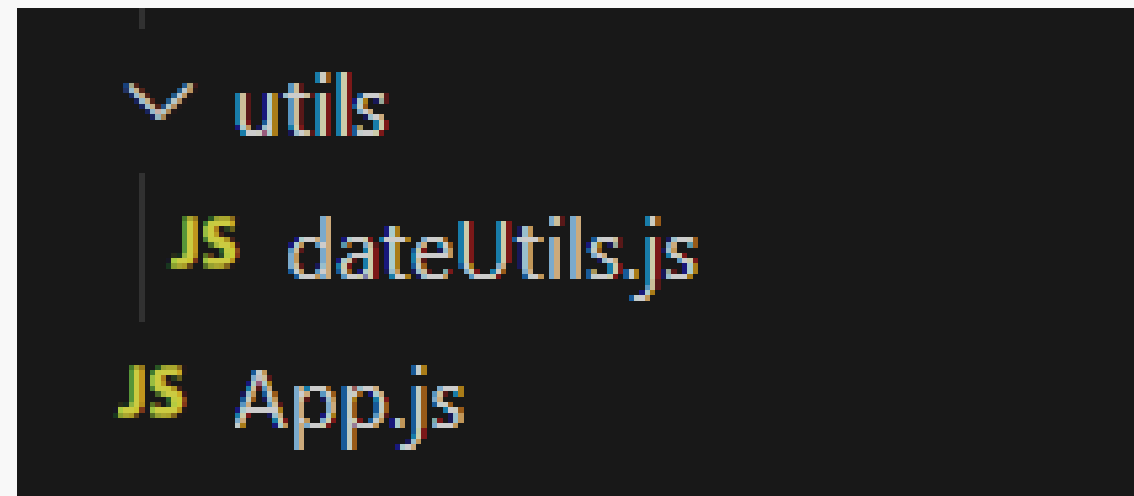
JS TrashScreen.js

- **EditNoteScreen.js:** Contains a component to display and edit information of a note.
- **HomeScreen.js:** Contains a component to display the home screen of the application, including saved notes.
- **LabelsScreen.js:** Contains a component to display and manage labels in the application.
- **ManagementLabelsScreen.js:** Contains a component to manage labels, such as adding, editing, and deleting labels.
- **NewNoteScreen.js:** Contains a component to create a new note.
- **TrashScreen.js:** Contains a component to display and manage deleted notes (trash).

Code Design

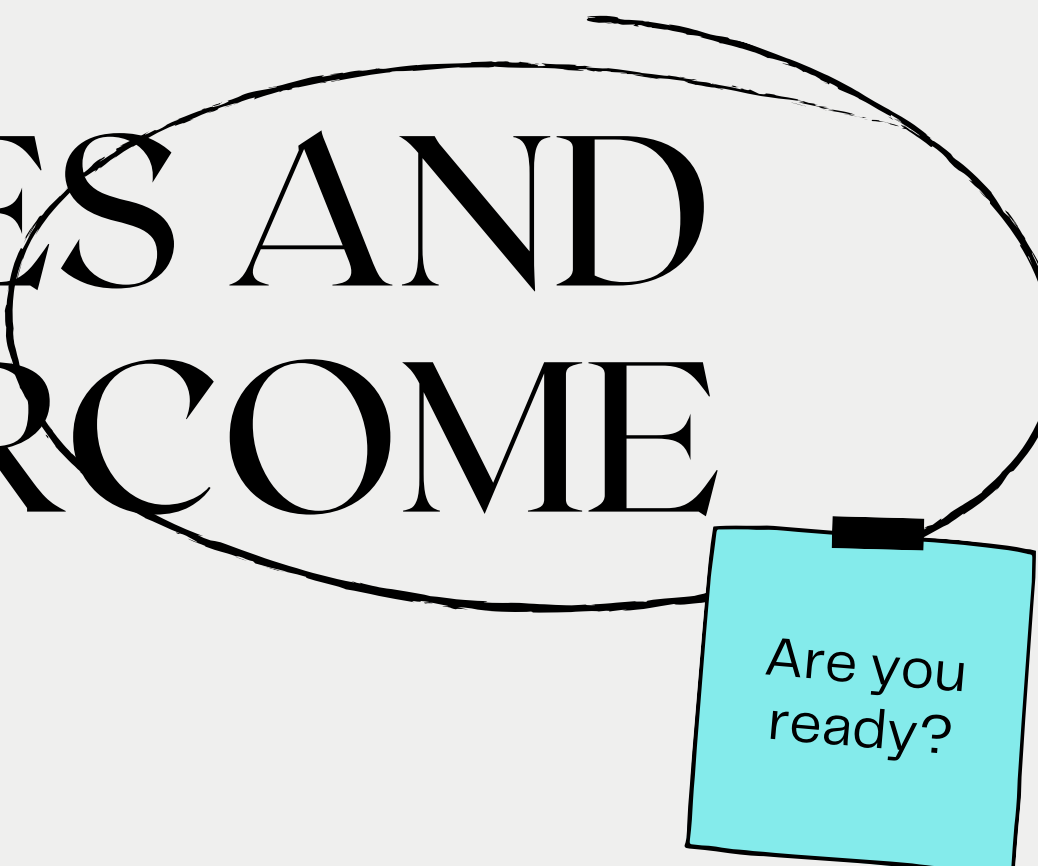
This is the function of each file code

Q utils & App.js



- **dateUtils.js:** Contains utility functions for formatting and manipulating dates, such as `formatUpdatedAt`.
- **App.js:** This is the entry point file of the application, initializing and configuring the React Native application, including: Setting up navigation between screens. Configuring the Redux store and linking it with reducers. Setting up providers (e.g., `NavigationContainer`, `ReduxProvider`) to provide services throughout the application.

ISSUES AND OVERCOME



Are you
ready?

```
export default function App() {
  return (
    <Provider store={store}>
      <NavigationContainer>
        <StatusBar style="auto" />
        <Drawer.Navigator
          initialRouteName="Home"
          drawerContent={(props) => <DrawerComponent {...props} />}
        >
          <Drawer.Screen
            name="Home"
            component={HomeStack}
            options={{
              headerShown: false,
            }}
          />
          <Drawer.Screen name="Labels" component={LabelsScreen} />
          <Drawer.Screen name="Trash" component={TrashScreen} />
        </Drawer.Navigator>
      </NavigationContainer>
    </Provider>
  );
}
```

Home page and navigation

1

Challenge: Encountering initial difficulty in moving back and forth between pages when starting the project

2

Solution – Filtering Issue: .Searching documentation and video tutorials about components “NavigationContainer” and “Drawer” to complete page transitions and pop up navigation bars

State Management for Labels in TrashScreen

```
useEffect(() => {  
  if (search === "") {  
    setFilteredLabels(labels);  
  } else {  
    const filtered = labels.filter((label) =>  
      label.label.toLowerCase().includes(search.toLowerCase())  
    );  
    setFilteredLabels(filtered);  
  }  
}, [search, labels]);
```

1

Challenge: Label Management Screen – Filtering labels based on search input might not update correctly.

2

Solution – Filtering Issue: Ensure that the filteredLabels state updates correctly when search state changes. Modify the 'useEffect' hook to update filteredLabels based on the search state.

```

<Icons
  name="checkmark-circle"
  size={50}
  color={content ? "green" : "red"}
  style={{
    position: "absolute",
    bottom: 10,
    right: 10,
  }}
  onPress={() => {
    if (!content) {
      return;
    }
    dispatch(
      addNote({
        content,
      })
    );
    setContent("");
    navigation.goBack();
  }}
/>
</View>
);
}

```

- 1 Challenge: The checkmark icon ('Ionicons') color in 'NewNoteScreen' is not updating dynamically based on the 'content' input. It currently checks for content existence but does not change color when content is entered or deleted.
- 2 Solution: Modify the 'Ionicons' component to dynamically update its color based on the 'content' state. This ensures visual feedback to the user regarding the validity of note creation.

Update the current notes when user is searching

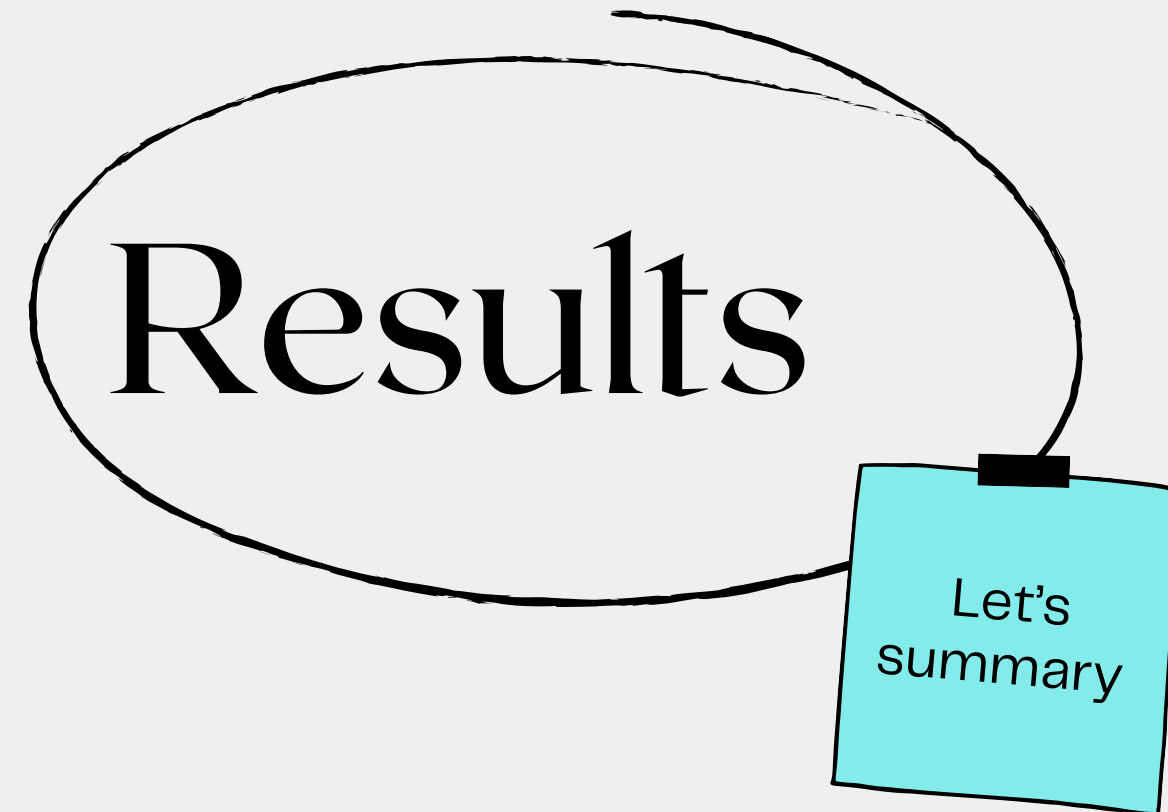
1

Challenge: Handling add cases when user searching

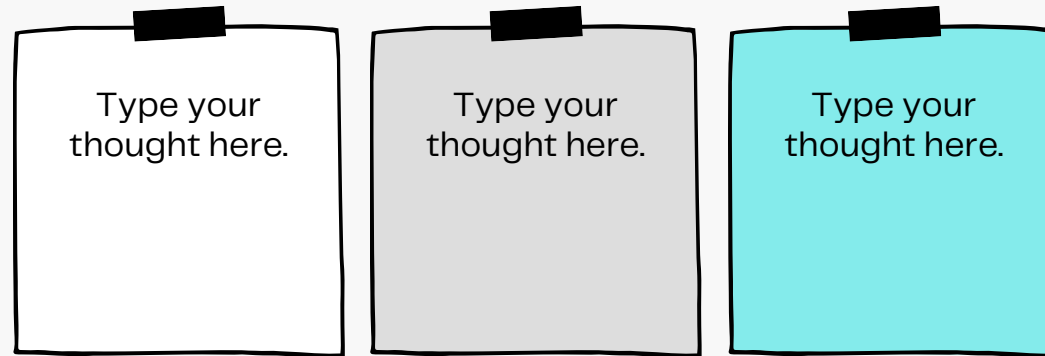
2

Solution: Modifying the 'status' arrow functions to handle those case: input string is empty, no matched results, only one result matched, more than one results

```
3  const status = () => {
4    if (search === "") {
5      if (notes.length === 0) {
6        return <Text style={styles.statusText}>Please add a new note</Text>;
7      }
8      return <Text style={styles.statusText}>Total {notes.length} notes</Text>;
9    } else {
10     if (filteredNotes.length === 0) {
11       return <Text style={styles.statusText}>Not found!</Text>;
12     } else {
13       return (
14         <Text style={styles.statusText}>
15           Found {filteredNotes.length}{ " "}
16           {filteredNotes.length ≥ 1 ? "note" : "notes"} match with {search}
17         </Text>
18       );
19     }
20   }
21 }
```

Achieved Results



**KEEP
GOING**

- In summary, our team has successfully implemented the core functionalities required for this project. We have completed the essential features and delivered a working application.
- However, regarding the additional requirement of the Folder Screen, we have only managed to implement around 10% of it. Unfortunately, we were unable to complete the bonus of the Folder Screen within the given time frame.



STOP

Thank you!

Have
a good
weekend!