Subject Name: **Front End Engineering**

Subject Code: CS186

Cluster: iGamma   Group: **G19**

Department: **DCSE**

CHITKARA UNIVERSITY

Project Name: Keeper (A note keeping app)

Submitted By:
Udit
2110992026
G19

Submitted To:
Ms. Pritpal Kaur

React Note-Keeping App: Keeper

Introduction

Welcome to the documentation of our React-based Note-Keeping App. In this documentation, we will explore the key components that make up our application and their respective functionalities.

Our Note-Keeping App is designed to help users capture and organize their thoughts, ideas, and important notes in a simple and intuitive manner. It offers a user-friendly interface and a range of features for managing notes efficiently.

Technologies Used:

- HTML
- CSS
- JavaScript
- React

Key Components

1. App Component

- The central component of our application that manages the state of notes.

- Allows users to add and delete notes.

- Coordinates the interactions between other components.

2. CreateArea Component

- Enables users to create new notes.

- Offers input fields for note title and content.

- Provides a button to submit new notes.

3. Header Component

- Represents the application's header, displaying the app title with an icon for visual appeal.

4. Search Component

- Displays the notes only containing the searched text.

5. Note Component

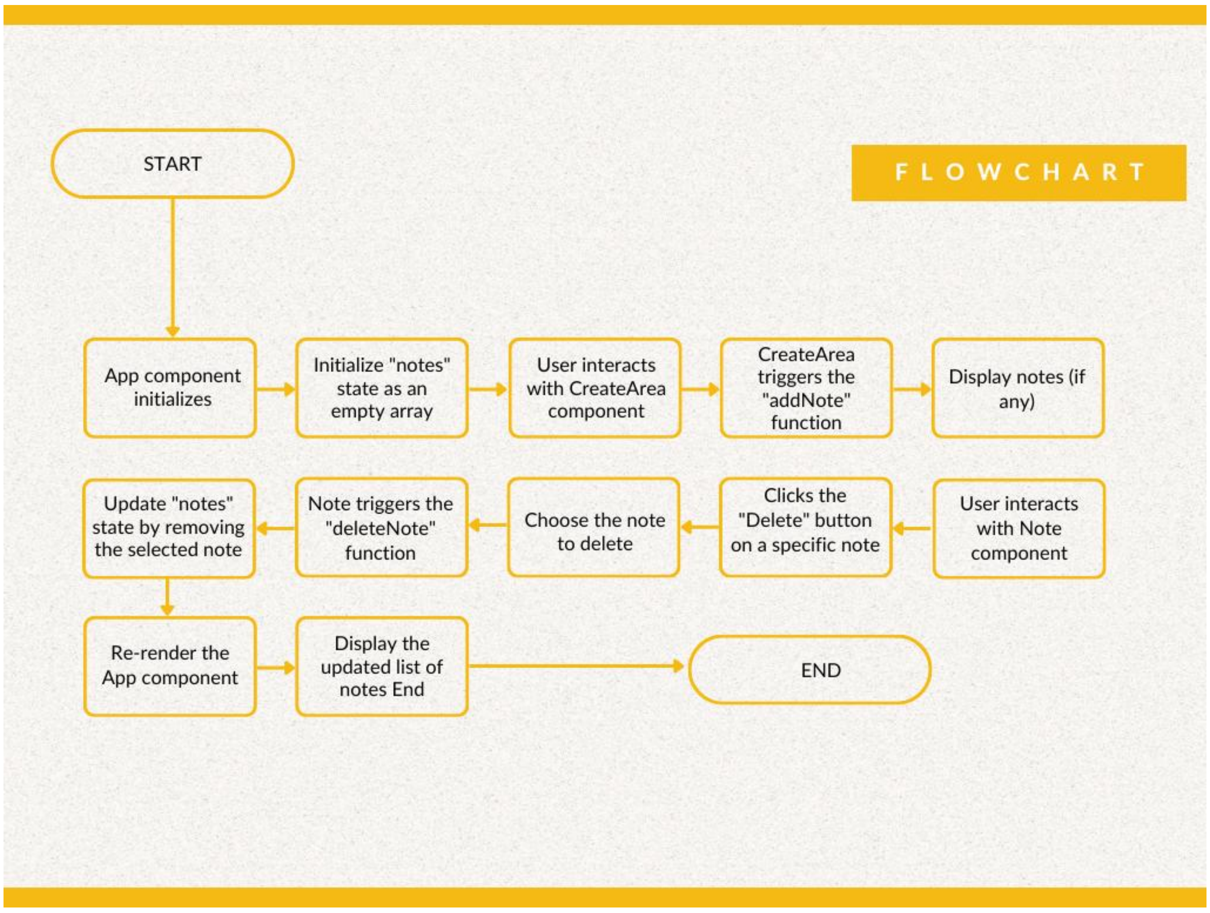
- Represents an individual note within the application.
- Displays the note's title and content.
- Provides a delete button for removing specific notes.

Usage

These components work together to create a seamless and user-friendly note-keeping

"Header," "Search," and "Note" components contribute to the overall functionality and visual appeal of the application.

Flow Chart:


START
FLOWCHART
App component initializes
Initialize "notes" state as an empty array
User interacts with CreateArea component
CreateArea triggers the "addNote" function
Display notes (if any)
User interacts with Note component
Clicks the "Delete" button on a specific note
Choose the note to delete
Note triggers the "deleteNote" function
Update "notes" state by removing the selected note
Re-render the App component
Display the updated list of notes End
END

-----------------------------**Components Documentation**-------------------------------

Index.js

The "index.js" file is the entry point for your React application. It initializes the application by rendering the root component, "App," into the root HTML element on the web page.

Import Statements:

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
```

Root Element:

The `createRoot` method is used to create a root element for rendering the React application. It selects the HTML element with the id "root" as the container for the application.

```
const root = ReactDOM.createRoot(document.getElementById("root"));
```

Rendering the Application:

The `render` method is used to render the "App" component within a `<StrictMode>`. `<StrictMode>` is a wrapper component provided by React that performs additional runtime checks and helps identify potential issues in your application.

```
root.render(
```

```
ReactDOM.render(
  <App />,
 document.getElementById('root')
);
```

1. App.js

The "App" component is the core component of the note-keeping application. It manages the state of notes, allowing users to add and delete notes. It also serves as the container for other components like "Header," "Footer," "Create Area," and a list of "Note" components.

Import Statements:

```
import { useState, useEffect } from "react";
import NotesList from "./Components/NotesList";
import { nanoid } from 'nanoid';
import Search from "./Components/Seach";
import Header from "./Components/Header";
```

State Management:

The component uses the `useState` hook to manage the state of notes.

- `notes` (Array): An array that stores note objects.

Functions:

1. `addNote(note)`: This function adds a new note to the `notes` state array. It uses the `setNotes` function to update the state.

2. `deleteNote(id)`: This function deletes a note based on its index (id) in the `notes` array. It uses the `setNotes` function to update the state by filtering out the note with the specified id.

Render Method:

The component renders the following elements:

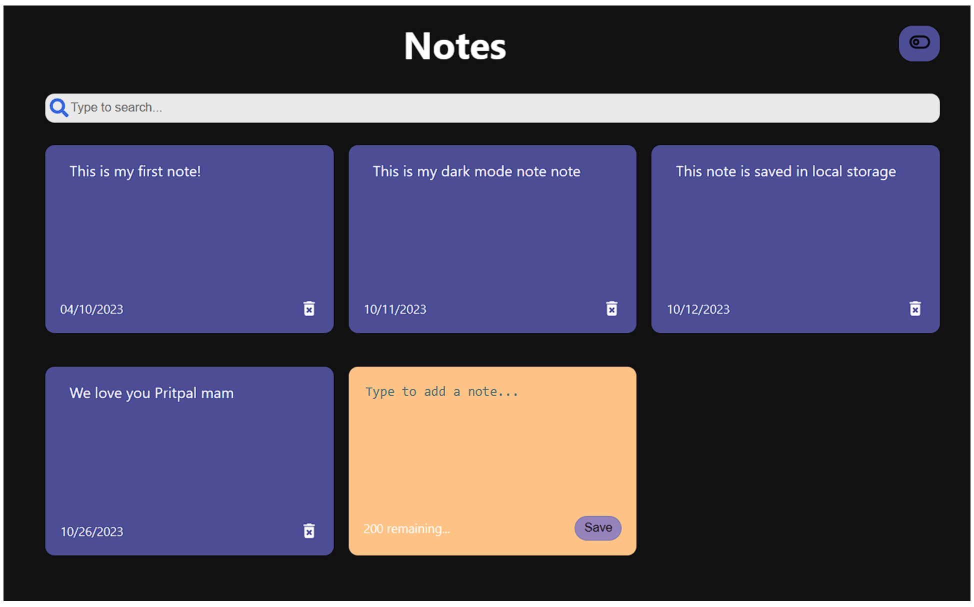
- `<Header />`: A header component.

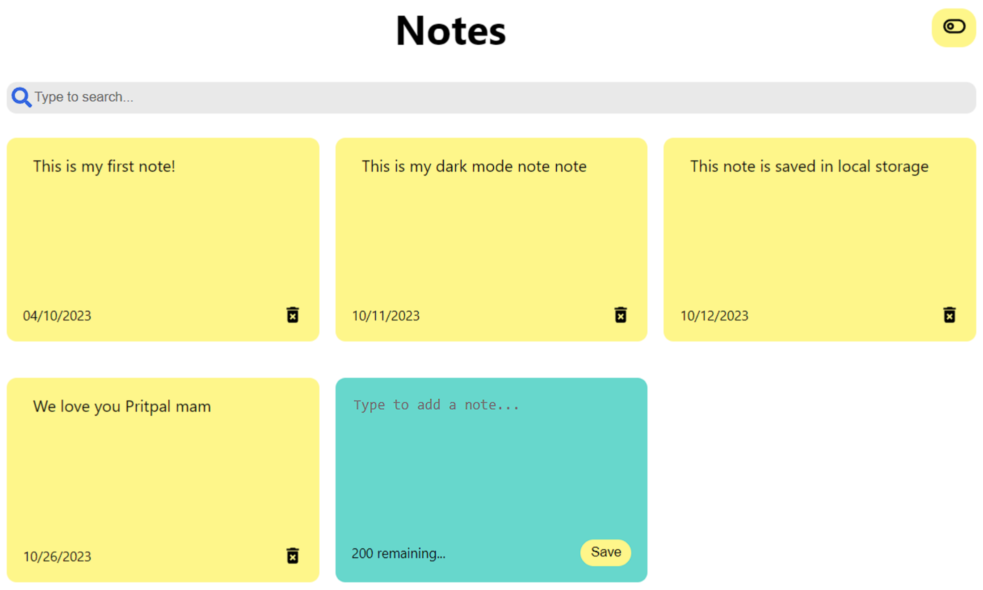
- `<CreateArea onAdd={addNote} />`: A component for creating and adding new notes. It passes the `addNote` function as a callback to this component.

- A list of `<Note />` components: Each represents an individual note, with properties like `key`, `id`, `title`, and `content`. Users can delete notes by clicking the delete button on each note. The `deleteNote` function is used as a callback to handle note deletion.

- `<Footer />`: A footer component.

Dark Mode:
I have also added a Dark Mode toggle button to switch my Note Taking App in Dark mode.





2. AddNote.js

Description:

The "CreateArea" component is responsible for enabling users to create new notes. It offers a form for input, including a title and content field, and a button to submit the note. The component manages its own state and triggers a callback function when a note is added.

Import Statements:

import { useState } from "react";

State Management:

The component uses the `useState` hook to manage its state:

- `isExpanded` (Boolean): Indicates whether the input field is expanded.

- `note` (Object): Contains the properties `title` and `content` for the note being created.

Functions:

1. `handleChange(event)`: This function is called when an input field's value changes. It updates the `note` state object accordingly.

2. `submitNote(event)`: When a user submits a note, this function is called. It invokes the `onAdd` callback passed as a prop with the `note` object, effectively adding the note to the parent component's state. It then resets the input fields.

3. `expand()`: This function is responsible for expanding the input field when the user clicks inside it.

Render Method:

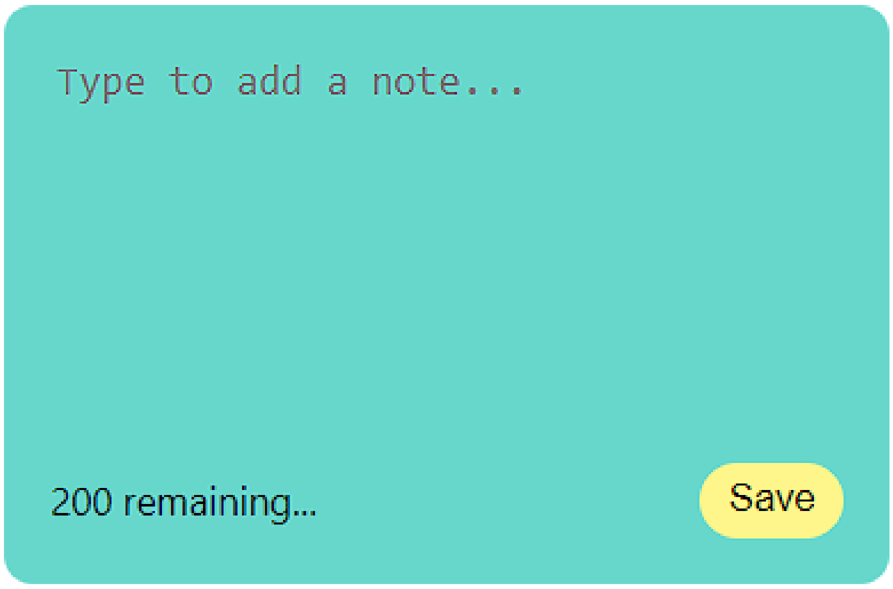
The component renders a form with the following elements:

- An input field for the note's title, which is displayed when `isExpanded` is `true`.

- A textarea for the note's content.

- A "Zoom" effect that is applied to the submit button when `isExpanded` is `true`.

- A submit button represented by a Material-UI `Fab` component with an "Add" icon.



3. Header.js

Description:

The "Header" component serves as the application's header, displaying the application's title along with a stylized icon. It provides a visual identifier for the application.

Import Statements:

```
import React from "react";
import { UilToggleOff } from '@iconscout/react-unicons'
```

Render Method:

The component renders a `<header>` element containing the application title and an icon:

- `<header>`: A container for the header content.

- `<h1>`: Displays the application title "Keeper" alongside an icon.

- `<HighlightIcon />`: A stylized icon used for visual enhancement.

4. Footer.jss

Description:

The "Search" component is responsible for displaying all the notes that contains the Searched text.

State Management:

The component does not use any state.

Render Method:

The component renders a `<Search>` element containing a copyright notice.

- `<Search>`: A container for the displaying search content.

5. Note.js

Description:

The "Note" component represents an individual note within the note-keeping application. It displays the note's title, content, and a delete button, allowing users to remove a specific note.

Import Statements:

import {MdDeleteForever} from 'react-icons/md';

Functions:

- `handleDeleteNote()`: This function is called when the user clicks the delete button. It invokes the `onDelete` callback, passing the `id` of the note to be deleted.

Render Method:

The component renders a `<div>` element representing a note, which includes the following elements:

- `<h1>`: Displays the note's title.

- `<p>`: Displays the note's content.

- A delete button represented by a Material-UI `DeleteIcon`. Clicking this button triggers the `handleClick` function to delete the note.

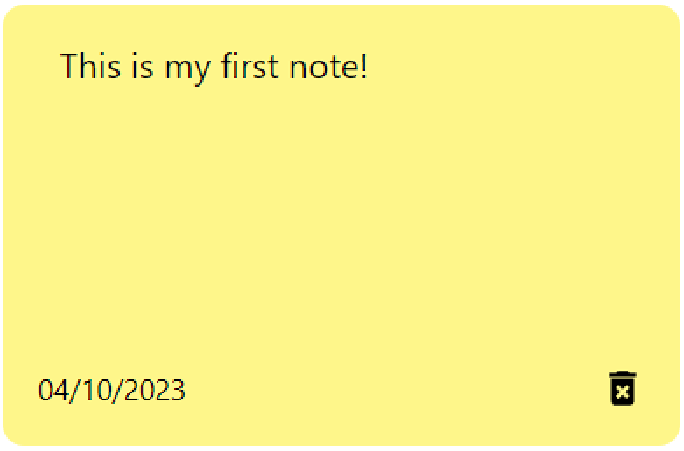
Props:

- `title` (String): The title of the note.

- `content` (String): The content of the note.

- `id` (Number): An identifier for the note.

- `onDelete` (Function): A callback function passed as a prop from the parent component. It is used to delete the note when the delete button is clicked.


This is my first note!

04/10/2023