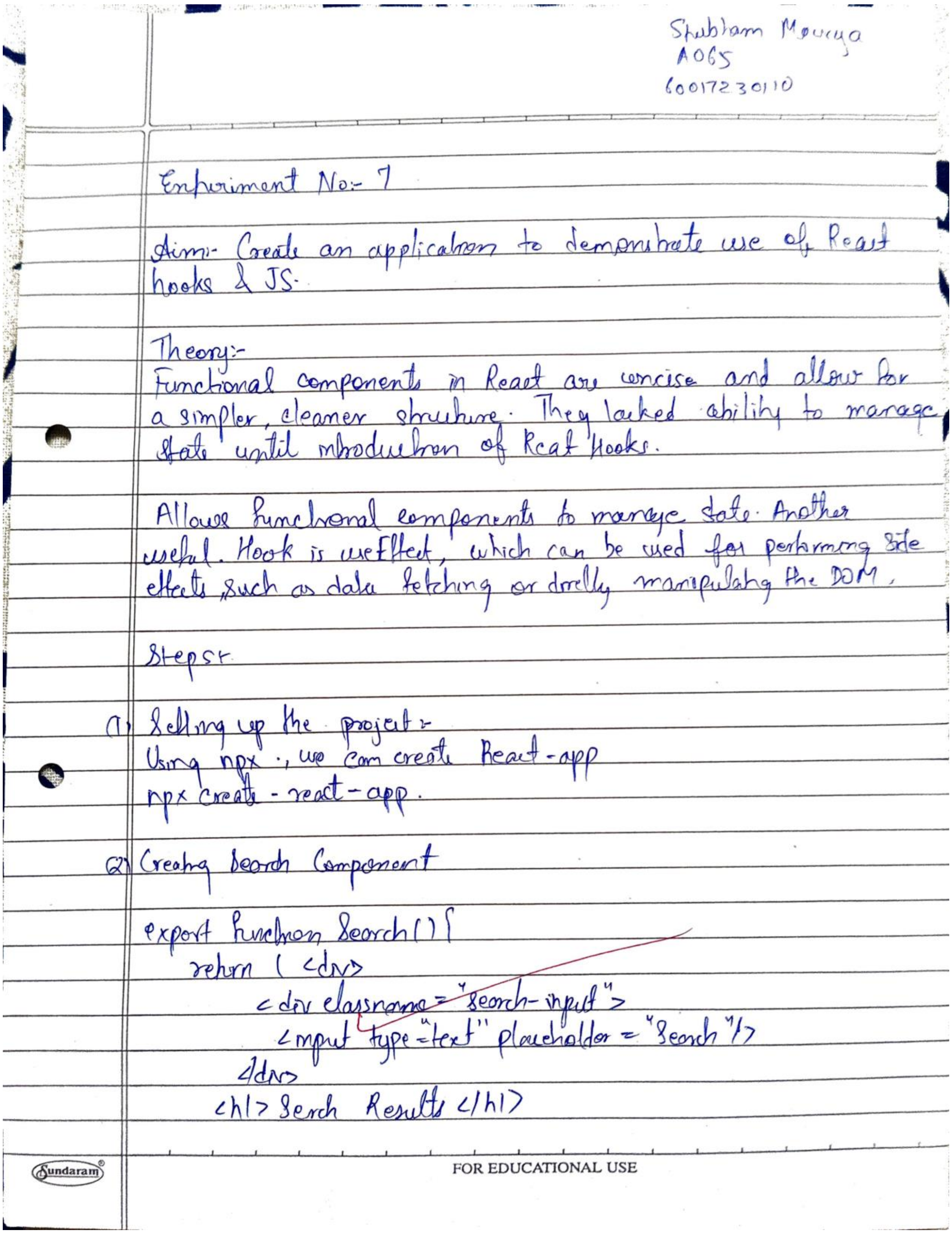
Department of Artificial Intelligence and Machine Learning
B.Tech. Sem: V Subject: Full Stack Development Laboratory (DJS22AML504)

Experiment 7

Name: Shubham Mourya **SAP ID: 60017230110**



Shubham Mourya
A065
60017230110

Experiment No:- 7

Aim:- Create an application to demonstrate use of React hooks & JS.

Theory:-
Functional components in React are concise and allow for a simpler, cleaner structure. They lacked ability to manage state until introduction of React Hooks.

Allows functional components to manage state. Another useful. Hook is useEffect, which can be used for performing side effects such as data fetching or directly manipulating the DOM.

Steps:-

(1) Setting up the project:-
Using npx :, we can create React-app
npx create-react-app.

(2) Creating Search Component

```
export function Search() {
   return ( <div>
         <div classname = "search-input">
          <input type="text" placeholder = "Search"/>
       </div>
     <h1> Serch Results </h1>
```

FOR EDUCATIONAL USE

(3) Adding State using hooks:- With useState hook, we add state to search component, to dynamically render book data. By modifying the input data to capture user queries, the application will call an external API to fetch relevant books.

```
const [results, setResults] = useState([]);
const handleSearch = (e) => {
        searchBooks(e.target.value).then(response =>
            { setResults(response.docs);
        });
    };
```

(4) Fetching Data with Fetch():-
We utilize browser's fetch() function to send the HTTP requests to Open Library API. Data fetched is then mapped into list of books details that will be rendered in UI.

(5) Adding authentication using Clerk:-
Integrating user authentication using Clerk. With Clerk's simple setup, we restrict certain section of app to logged-in users.

Conclusion:- We successfully created React Application using Hooks to manage state & fetch book data from an API.

[illegible]

Department of Artificial Intelligence and Machine Learning
B.Tech. Sem: V Subject: Full Stack Development Laboratory (DJS22AML504)

Date:	**03/09/2024 and 05/09/2024**
Aim	**Create an application to demonstrate use of React hooks and JS.**
Software	
Pre-requisite	Active internet connection
Theory	Functional components are much shorter, they contain less boilerplate code, and everything is contained in one function. Until recently, there was another big difference between class components and functional components. Functional components could not contain any state. These stateless components are lightweight, and they encourage separating the presentation from the application logic. Then React introduced Hooks. Hooks allow you to obtain data and a callback function that can modify the data. This allows you to add state to functional components, making them much more powerful. In this way, you can create complete stateful React components using the terse functional style. **Creating a React Application with Hooks** A typical Node installation comes together with two command-line tools, npm and npx. npm is used for installing packages into a project, and npx is used to run Node commands from the command line. The beauty of npx is that the commands don't necessarily need to be installed on your system. npx will first look in your current project folder to see if a command is installed there. When it can't find it on your computer, it will look in the npmjs.com repository, load the latest version of the command script and run it, without installing it locally. This feature can be used to create a skeleton React application in just a few key presses. Open a terminal in a folder of your choice and run the following command. `npx create-react-app react-books-with-hooks` This will create a new folder react-books-with-hooks and initialize it with a basic React application. You can now open up the project in your favorite IDE. Inside the project, you will see a src folder with the main application component, App.js. When you look inside this file, you can see that it contains a single function App(). This function returns an element, and it uses an extended JavaScript syntax, known as JSX, to define the component. JSX allows you to write HTML-style template syntax straight into your JavaScript file. The React

toolchain is set up to convert this mix of JavaScript and HTML into pure JavaScript that renders the HTML element.

You can define your own React components simply by writing a function that returns a JSX element. Try it out. Create a new file, src/Search.js, and paste the following code into it.

```
import React from 'react';



export function Search() {

  return (

    <div>

      <div className="search-input">

        <input type="text" placeholder="Search"/>

      </div>

      <h1 className="h1">Search Results</h1>

      <div className="books">

        <table>

          <thead>

            <tr>

              <th className="title-col">Title</th>

              <th className="author-col">Author</th>

              <th className="year-col">Pub Year</th>

            </tr>

          </thead>

          <tbody></tbody>
      </table>

    </div>

  </div>

 );

}
```

This is all you need to create a component. Of course, it doesn't yet do anything and only displays an empty table. But you can already use the Search component in the application. Open up src/App.js again and add the following import to the top of the file.

```
import { Search } from './Search';
```

Now remove the import of the logo.svg and then replace the contents of returned value in the App() function with the code below.

```
<div className="App">

  <header>

    Books with Hooks

  </header>

  <Search/>

</div>
```

You will notice the <Search/> element has been used as if it was an HTML element. The JSX syntax allows you to include components in this way directly in your JavaScript code. You can already test your application by running the following command in your terminal.

```
npm start
```

This will compile the application and open your default browser to http://localhost:3000. You can leave this command running while you're developing your code, and it will keep on updating the application and reloading the browser page every time you modify and save the code.

So far, the application works, but it doesn't look nice, and it doesn't react to any user input.

Adding State with React Hooks

In this section, will see how to use Hooks to create state and update the search results depending on that state. First, create a function that loads content from the Open Library web service. Open src/Search.js and paste the following code after the import statements.

```
const baseUrl = 'http://openlibrary.org';



export function searchBooks(query) {

    const url = new URL(baseUrl + '/search.json');

    url.searchParams.append('title', query);



    return fetch(url).then(response => response.json());

}
```

This uses the browser's fetch() API to get data from a server and return a JavaScript Promise that resolves with the server's response.

Now, inside the Search() function, before the return statement, add the following code.

```
const [results, setResults] = React.useState(0);



const handleSearch = (event) => {

  searchBooks(event.target.value).then(response => {

    setResults(response.docs);
  });

};



const resultList = (results || []).map((book) =>

  <tr key={book.key}>

    <td>{book.title}</td>

    <td>{book.author_name && book.author_name.join(', ')}</td>

    <td>{book.first_publish_year}</td>

  </tr>

);
```

The first line calls React.useState() to obtain a stateful variable. useState() returns an array with two entries. The first entry is the current value of the state variable. This will be undefined until you update the state. The second entry is a function that you can call to update the state. In the example above, I have called the state variable results and the callback setResults().

After obtaining the state, the code above defines an event handler. This simply calls the searchBooks() function and, once a response from the server is received, calls the setResults() callback to update the state. You do not have to worry about telling React to re-render the component. When you update the state, React will automatically check which parts of the application have changed and re-render them. Finally, a resultList is created that represents the search results in an array of HTML table rows.

You can now add the handleSearch() event handler to the input element. Modify the <input> element that is part of the returned JSX code to match the code below.

```
<input onChange={handleSearch} type="text" placeholder="Search"/>
```

To render the results inside the table, modify the <tbody> element to render the resultList.

```
<tbody>{resultList}</tbody>
```

In both cases, the curly braces are used to insert the value of variables into the rendered HTML.

Add Authentication to Your React App

Real-life web applications require access control. Some parts of the application should be restricted to a limited number of users. Creating your own user management and securing your application is difficult and requires a lot of expertise. Okta allows you to set up authentication with just a few lines of code.

Before you begin, you'll need a free Okta developer account. Install the Okta CLI and run `okta register` to sign up for a new account. If you already have an account, run `okta login`. Then, run `okta apps create`. Select the default app name, or change it as you see fit. Choose **Single-Page App** and press **Enter**.

Use `http://localhost:3000/callback` for the Redirect URI and accept the default Logout Redirect URI of `http://localhost:3000`.

What does the Okta CLI do?

Take note of the **Client ID**. This needs to be pasted into your JavaScript code.

To make use of Okta in your React app, open the terminal in your project directory, and install the Okta React SDK with the React router by running the following commands.

```
npm install -E @okta/okta-react@3.0.4 react-router-dom@5.2.0
```

In `src/App.js`, add the imports for these two packages to the top of the file.

```
import { BrowserRouter as Router, Route, Link } from 'react-router-dom';

import { LoginCallback, SecureRoute, Security } from '@okta/okta-react';

import { Home } from './Home';
```

The router is responsible for looking at the route part of the URL and selecting the right React component to render. To add the router to your application, replace the component returned in the `render()` function with the code below.

```
<div className="App">
```

```
  <Router>

    <header>

      <div>Books with Hooks</div>

      <ul className="menu"><li><Link
to="/">Home</Link></li><li><Link
to="/search">Search</Link></li></ul>

    </header>

    <Security issuer='https://{YourOktaDomain}/oauth2/default'

              clientId='{ClientId}'

              redirectUri={window.location.origin + '/callback'}

              pkce={true}>

      <Route path='/' exact={true} component={Home}/>

      <SecureRoute path='/search' exact={true} component={Search}/>

      <Route path='/callback' component={LoginCallback}/>

    </Security>

  </Router>

</div>
```

Here {YourOktaDomain} is your Okta developer domain. You can find this on the Okta dashboard tab. {ClientId} is the client ID that you obtained earlier when you registered the application. I have added a reference to a Home component. Implement this by creating a new file src/Home.js and pasting the following code into it.

```
import React from 'react';

import { useOktaAuth } from '@okta/okta-react';



export function Home() {
```

	`const { authState, authService } = useOktaAuth();`
	
	`const login = () => { authService.login('/'); }`
	`const logout = () => { authService.logout('/'); }`
	
	`const userText = authState.isAuthenticated`
	`? <div><p>You are signed in!</p><button onClick={ logout }>Logout</button></div>`
	`: <div><p>You need to sign in to use the application!</p><button onClick={ login }>Sign In</button></div>;`
	
	`return <div className="page-home"><h1>Welcome to Books with Hooks</h1>{ userText }</div>;`
	`}`
	
	Add Some Finishing Touches
	
	Styling web applications is done using Cascading Style Sheets (CSS). You might have noticed the import of App.css at the top of the App.js file. React configures your application so that CSS files can be directly imported into the component JavaScript files. The styles will then automatically be applied to the component. You can add some styling by opening the src/App.css file and replacing its contents with the following code.
	
	`.App header {`
	`background-color: #282c34;`
	`display: flex;`
	`flex-direction: row;`
	`align-items: center;`

	```
	  justify-content: space-between;

	  color: white;

	  padding: 0.5rem 1rem;

	}


	ul.menu {

	  list-style: none;

	}


	ul.menu li {

	  display: inline;

	  padding: 12px;

	}


	ul.menu a {

	  color: #ffffff;

	}


	.page-home {

	  text-align: center;

	}


	.content {
	```

```
  text-align: left;

  display: inline-block;

  background-color: #ffffff;

  width: 100%;

  max-width: 1232px;

  padding: 16px;

  box-sizing: border-box;

}


h1 {

  text-align: center;

}


.books table {

  width: 100%;

}


.title-col {

  max-width: 60%;

}


.search-input {

  padding: 4px;
```

<table>
<tr><td></td><td>

```
  text-align: center;
}


.search-input input {
  display: inline-block;
  width: 50%;
}
```

You can run the following command again if it isn't still running.

```
npm start
```

In your browser at http://localhost:3000, you will be redirected to the Okta sign-in page. After successfully entering your credentials, you should see something like the following.
</td></tr>
<tr><td>Code</td><td>
AddNote.jsx

```
import React, { useState } from 'react';

const AddNote = ({ addNote }) => {
    const [note, setNote] = useState({ title: "", description: "", tag: "" });

    const handleClick = (e) => {
        e.preventDefault();
        addNote(note.title, note.description, note.tag);
        setNote({ title: "", description: "", tag: "" });
    };

    const onChange = (e) => {
        setNote({ ...note, [e.target.name]: e.target.value });
    };

    return (
```
</td></tr>
</table>

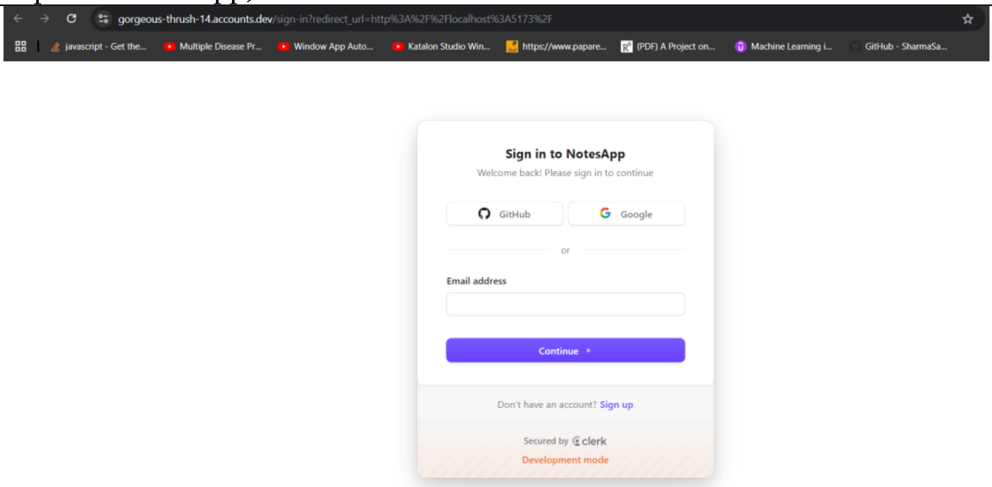
```
        <div className="max-w-md mx-auto my-5 p-5 border rounded-lg shadow-lg bg-white">
          <h2 className="text-2xl font-bold mb-4">Add a Note</h2>
          <form className="space-y-4">
            <div>
              <label htmlFor="title" className="block text-sm font-medium text-gray-700">Title</label>
              <input
                type="text"
                className="mt-1 block w-full border-gray-300 rounded-md shadow-sm focus:border-blue-500 focus:ring focus:ring-blue-500 focus:ring-opacity-50"
                id="title"
                name="title"
                value={note.title}
                onChange={onChange}
                minLength={5}
                required
              />
            </div>
            <div>
              <label htmlFor="description" className="block text-sm font-medium text-gray-700">Description</label>
              <textarea
                className="mt-1 block w-full border-gray-300 rounded-md shadow-sm focus:border-blue-500 focus:ring focus:ring-blue-500 focus:ring-opacity-50"
                id="description"
                name="description"
                value={note.description}
                onChange={onChange}
                minLength={5}
                required
              ></textarea>
            </div>
            <div>
              <label htmlFor="tag" className="block text-sm font-medium text-gray-700">Tag</label>
              <input
                type="text"
                className="mt-1 block w-full border-gray-300 rounded-md shadow-sm focus:border-blue-500 focus:ring focus:ring-blue-500 focus:ring-opacity-50"
                id="tag"
                name="tag"
                value={note.tag}
                onChange={onChange}
              />
          </div>
          <button
            type="submit"
              className="w-full bg-blue-500 text-white py-2 rounded-md
hover:bg-blue-600 transition duration-200"
            onClick={handleClick}
          >
            Add Note
          </button>
        </form>
      </div>
  );
};

export default AddNote;

Navbar.jsx

import React from 'react';
import { Link } from 'react-router-dom';
import { SignedIn, SignedOut, UserButton, SignInButton } from
"@clerk/clerk-react";

const Navbar = () => {
  return (
    <nav className="bg-white shadow">
      <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
        <div className="flex items-center justify-between h-16">
          <div className="flex-shrink-0">
                   <Link className="text-xl font-bold text-gray-900"
to="/">NotesApp</Link>
          </div>
          <div className="flex space-x-4">
            <SignedIn>
                 <Link className="text-gray-900 hover:text-blue-500"
to="/">Home</Link>
                 <Link className="text-gray-900 hover:text-blue-500"
to="/">All Notes</Link>
              <UserButton />
            </SignedIn>
            <SignedOut>
                <SignInButton className="bg-blue-500 text-white px-4
py-2 rounded hover:bg-blue-600 transition duration-200" />
            </SignedOut>
          </div>
        </div>
      </div>
    </nav>
  );
};

export default Navbar;

NoteItem.jsx

import React, { useState } from 'react';

const NoteItem = ({ note, deleteNote, editNote }) => {
  const [isEditing, setIsEditing] = useState(false);
  const [editedNote, setEditedNote] = useState({ ...note });

  const handleEdit = () => {
    setIsEditing(true);
  };

  const handleSave = () => {
            editNote(note.id,    editedNote.title,    editedNote.description,
editedNote.tag);
    setIsEditing(false);
  };

  const handleChange = (e) => {
    setEditedNote({ ...editedNote, [e.target.name]: e.target.value });
  };

  if (isEditing) {
    return (
       <div className="max-w-sm mx-auto my-4 p-5 border rounded-lg
shadow-md bg-white">
        <div className="flex flex-col">
          <input
            type="text"
            name="title"
            value={editedNote.title}
            onChange={handleChange}
            className="text-xl font-semibold mb-2 p-1 border rounded"
          />
          <textarea
            name="description"
            value={editedNote.description}
            onChange={handleChange}
            className="text-gray-700 mb-2 p-1 border rounded"
          />
          <input
            type="text"
            name="tag"
            value={editedNote.tag}
            onChange={handleChange}
            className="text-gray-500 text-sm mb-4 p-1 border rounded"
          />
          <div className="flex justify-between">
            <button
               className="bg-green-500 text-white px-4 py-2 rounded-
md hover:bg-green-600 transition duration-200"
              onClick={handleSave}
            >
              Save
            </button>
            <button
              className="bg-gray-500 text-white px-4 py-2 rounded-md
hover:bg-gray-600 transition duration-200"
              onClick={() => setIsEditing(false)}
            >
              Cancel
            </button>
          </div>
        </div>
      </div>
    );
  }

  return (
      <div className="max-w-sm mx-auto my-4 p-5 border rounded-lg
shadow-md bg-white">
      <div className="flex flex-col">
        <h5 className="text-xl font-semibold mb-2">{note.title}</h5>
        <p className="text-gray-700 mb-2">{note.description}</p>
                    <p className="text-gray-500 text-sm mb-
4"><small>{note.tag}</small></p>
        <div className="flex justify-between">
          <button
              className="bg-red-500 text-white px-4 py-2 rounded-md
hover:bg-red-600 transition duration-200"
            onClick={() => deleteNote(note.id)}
          >
            Delete
          </button>
          <button
             className="bg-blue-500 text-white px-4 py-2 rounded-md
hover:bg-blue-600 transition duration-200"
            onClick={handleEdit}
          >
            Edit
          </button>
          </div>
        </div>
      </div>
    );
};

export default NoteItem;

NoteList.jsx

import React, { useState } from 'react';
import NoteItem from './NoteItem';

const NotesList = ({ notes, deleteNote, editNote, addNote }) => {
    const [title, setTitle] = useState('');
    const [description, setDescription] = useState('');
    const [tag, setTag] = useState('');

    const handleAddNote = () => {
        if (title.trim() !== '' && description.trim() !== '') {
            addNote(title, description, tag);
            setTitle('');
            setDescription('');
            setTag('');
        } else {
            alert('Title and Description are required!');
        }
    };

    return (
        <div className="max-w-4xl mx-auto p-5">
            <h2 className="text-2xl font-bold my-3">All Notes</h2>
            {notes.length === 0 ? (
                <p className="text-gray-500">No notes to display</p>
            ) : (
                <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3
gap-4">
                    {notes.map(note => (
                        <NoteItem
                            key={note.id}
                            note={note}
                            deleteNote={deleteNote}
                            editNote={editNote}
                        />
                    ))}
                </div>
            )}
```

	```
	            <h3 className="text-xl font-semibold mt-8 mb-4">Add a New Note</h3>
	          <div className="mb-3">
	            <input
	              type="text"
	                className="w-full p-2 border border-gray-300 rounded-md focus:outline-none focus:ring focus:ring-blue-500"
	              placeholder="Title"
	              value={title}
	              onChange={(e) => setTitle(e.target.value)}
	            />
	          </div>
	          <div className="mb-3">
	            <textarea
	                className="w-full p-2 border border-gray-300 rounded-md focus:outline-none focus:ring focus:ring-blue-500"
	              placeholder="Description"
	              value={description}
	              onChange={(e) => setDescription(e.target.value)}
	            ></textarea>
	          </div>
	          <div className="mb-3">
	            <input
	              type="text"
	                className="w-full p-2 border border-gray-300 rounded-md focus:outline-none focus:ring focus:ring-blue-500"
	              placeholder="Tag"
	              value={tag}
	              onChange={(e) => setTag(e.target.value)}
	            />
	          </div>
	          <button
	               className="w-full bg-blue-500 text-white py-2 rounded-md hover:bg-blue-600 transition duration-200"
	            onClick={handleAddNote}
	          >
	            Add Note
	          </button>
	        </div>
	    );
	};

	export default NotesList;
	```
	 App.jsx
	```
	import React, { useState } from 'react';
	import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
	```

```
import { SignedIn, SignedOut, RedirectToSignIn, ClerkProvider } from
"@clerk/clerk-react";
import AddNote from './components/AddNote';
import NotesList from './components/NotesList';
import Navbar from './components/Navbar';

const App = () => {
  const [notes, setNotes] = useState([]);

  const addNote = (title, description, tag) => {
    const newNote = { id: Date.now(), title, description, tag };
    setNotes([...notes, newNote]);
  };

  const deleteNote = (id) => {
    setNotes(notes.filter(note => note.id !== id));
  };

  const editNote = (id, title, description, tag) => {
    const updatedNotes = notes.map(note =>
      note.id === id ? { ...note, title, description, tag } : note
    );
    setNotes(updatedNotes);
  };

  return (
    <ClerkProvider
publishableKey={import.meta.env.VITE_CLERK_PUBLISHABLE_KEY}
>
      <Router>
        <Navbar />
        <div style={{ padding: "20px" }}>
          <Routes>
            <Route
              path="/"
              element={
                <>
                  <SignedIn>
                    <NotesList notes={notes} deleteNote={deleteNote}
editNote={editNote} addNote={addNote} />
                  </SignedIn>
                  <SignedOut>
                    <RedirectToSignIn />
                  </SignedOut>
                </>
              }
            />
            <Route
```

```
                    path="/add-note"
                    element={
                      <>
                        <SignedIn>
                          <AddNote addNote={addNote} />
                        </SignedIn>
                        <SignedOut>
                          <RedirectToSignIn />
                        </SignedOut>
                      </>
                    }
                  />
                </Routes>
              </div>
            </Router>
          </ClerkProvider>
  );
};

export default App;
```

Result



gorgeous-thrush-14.accounts.dev/sign-in?redirect_url=http%3A%2F%2Flocalhost%3A5173%2F

javascript - Get the... | Multiple Disease Pr... | Window App Auto... | Katalon Studio Win... | https://www.papare... | (PDF) A Project on... | Machine Learning i... | GitHub - SharmaSa...

Sign in to NotesApp
Welcome back! Please sign in to continue

GitHub | Google

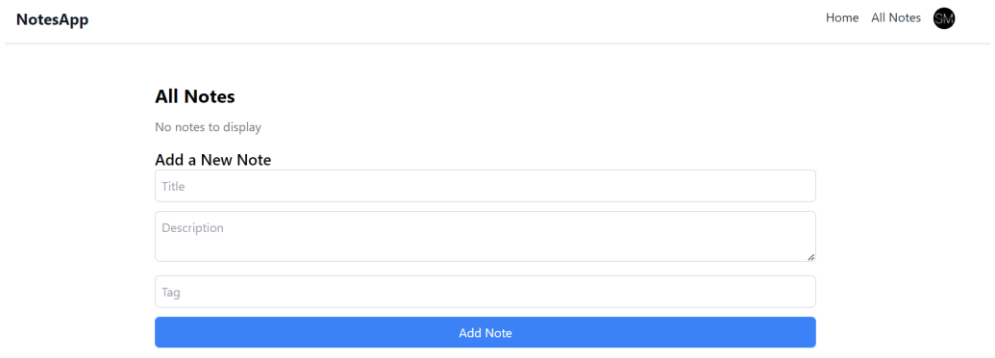
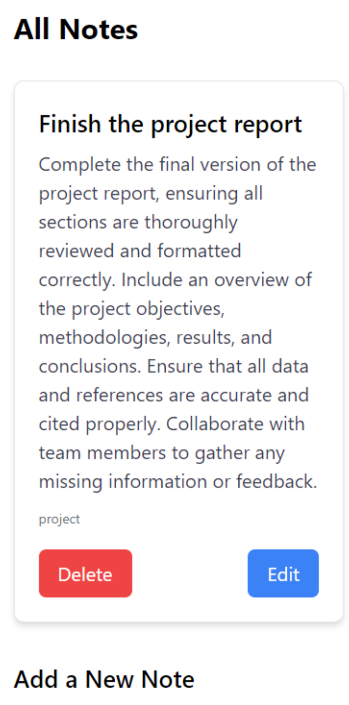
or

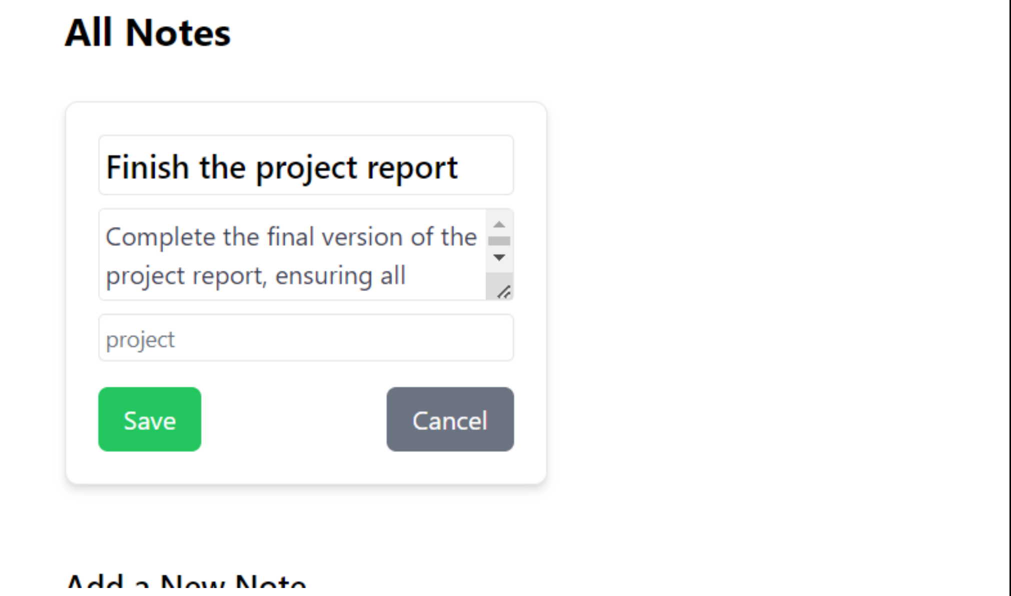
Email address

Continue

Don't have an account? Sign up

Secured by clerk
Development mode

	 NotesApp Home All Notes GM All Notes No notes to display Add a New Note Title Description Tag Add Note  All Notes Finish the project report Complete the final version of the project report, ensuring all sections are thoroughly reviewed and formatted correctly. Include an overview of the project objectives, methodologies, results, and conclusions. Ensure that all data and references are accurate and cited properly. Collaborate with team members to gather any missing information or feedback. project Delete Edit Add a New Note

	 All Notes Finish the project report Complete the final version of the project report, ensuring all project
Conclusion	**Thus we create an application to demonstrate use of React hooks and JS.**