Who am I?

Humera Tariq

PhD, MS, MCS (Computer Science), B.E (Electrical)

Postdoc (Medical Image Processing, Deep Neural Networks)

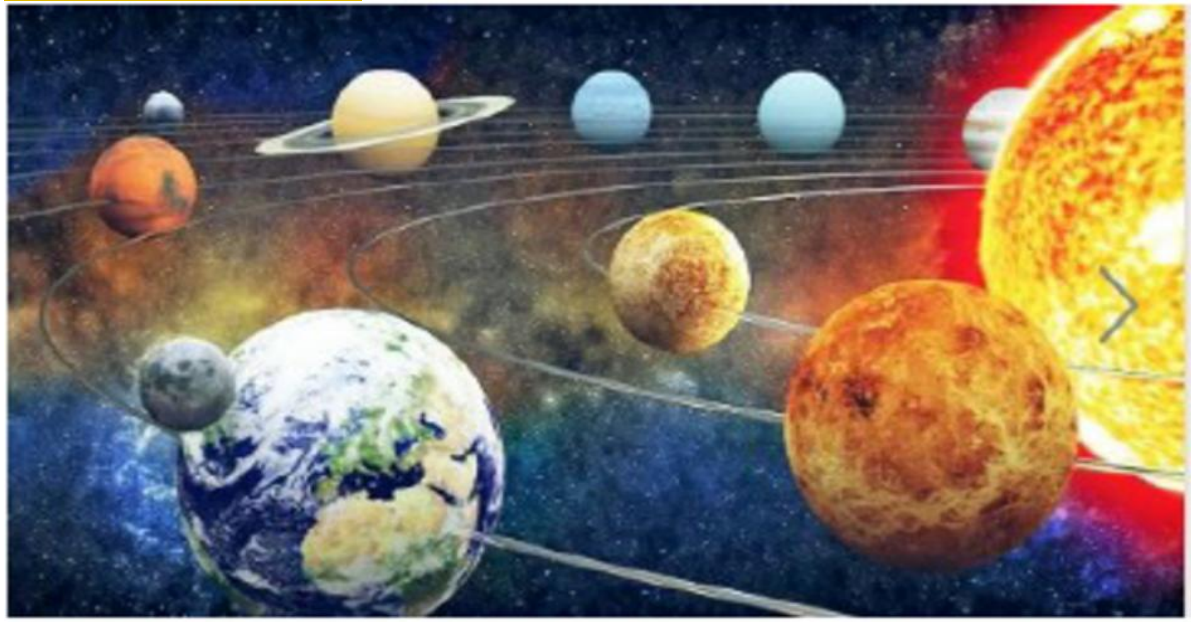
Email: humera@uok.edu.pk

Web: https://humera.pk/

Discord: https://discord.gg/xeJ68vh9

Starting in the name of Allah,

the most beneficial,
the most merciful.



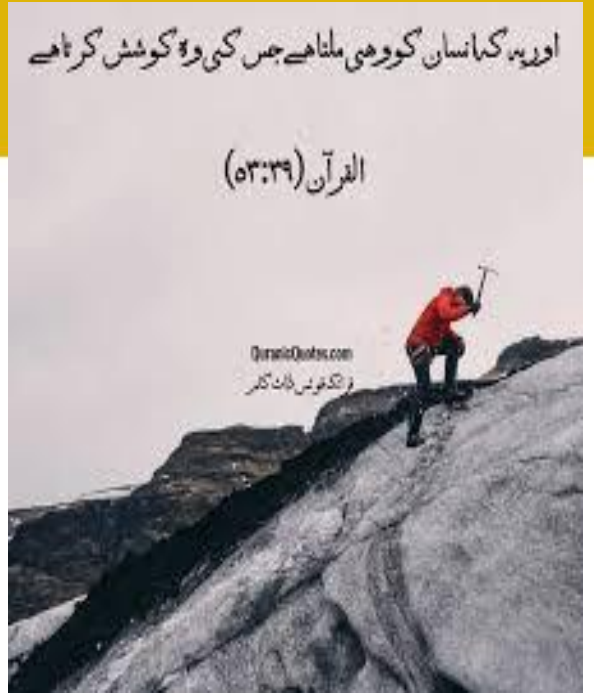
أَمْ لِلْإِنسَانِ مَا تَمَنَّىٰ ﴿٢٤﴾

کیا انسان کو ہر وہ چیز حاصل ہے جس کی اس نے تمنا کی؟

UNIVERSITY OF KARACHI

Surah An-Najm Chapter 53 Verse 39

And there is not for man except that [good] for which he strives.



UNIVERSITY OF KARACHI

Week 09

Internet Application Development

Department of Computer Science (DCS/UBIT)
University of Karachi
January 2025

Week 09 Agenda/ Flow

Firebase

Top React Design Patterns

- ✓ **Example 1 :** Review Mini List Project (addProject, deleteProject, Array of Objects, importance of Backticks)

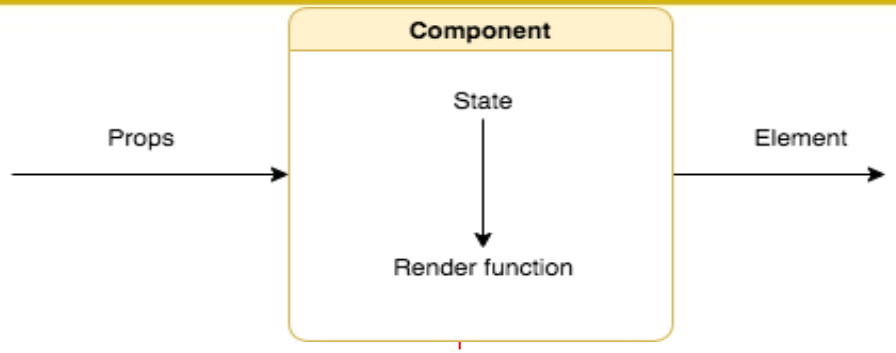
- ✓ **Example 2 :** Building and hosting a small React app with Fire base authentication. Firestore portion will be covered in week 10 inshallah.

- ✓ **Example 3 :** Container-Presenter Pattern with clean and optimized fetch logic discussion. (fetch .then vs async await fetch debate).

UNIVERSITY OF KARACHI

Class vs functional components

Class Component



Functional Component

render() method

```
class MyClassComponent extends Component {
  render() {
    return <h1>Hello from Class Component!</h1>;
  }
}
```

```
export default MyClassComponent;
```

Directly returns JSX

```
const MyFunctionalComponent = () => {
  return <h1>Hello from Functional Component!</h1>;
};
```

```
export default MyFunctionalComponent;
```

UNIVERSITY OF KARACHI

Have any one practiced Multi-component List?

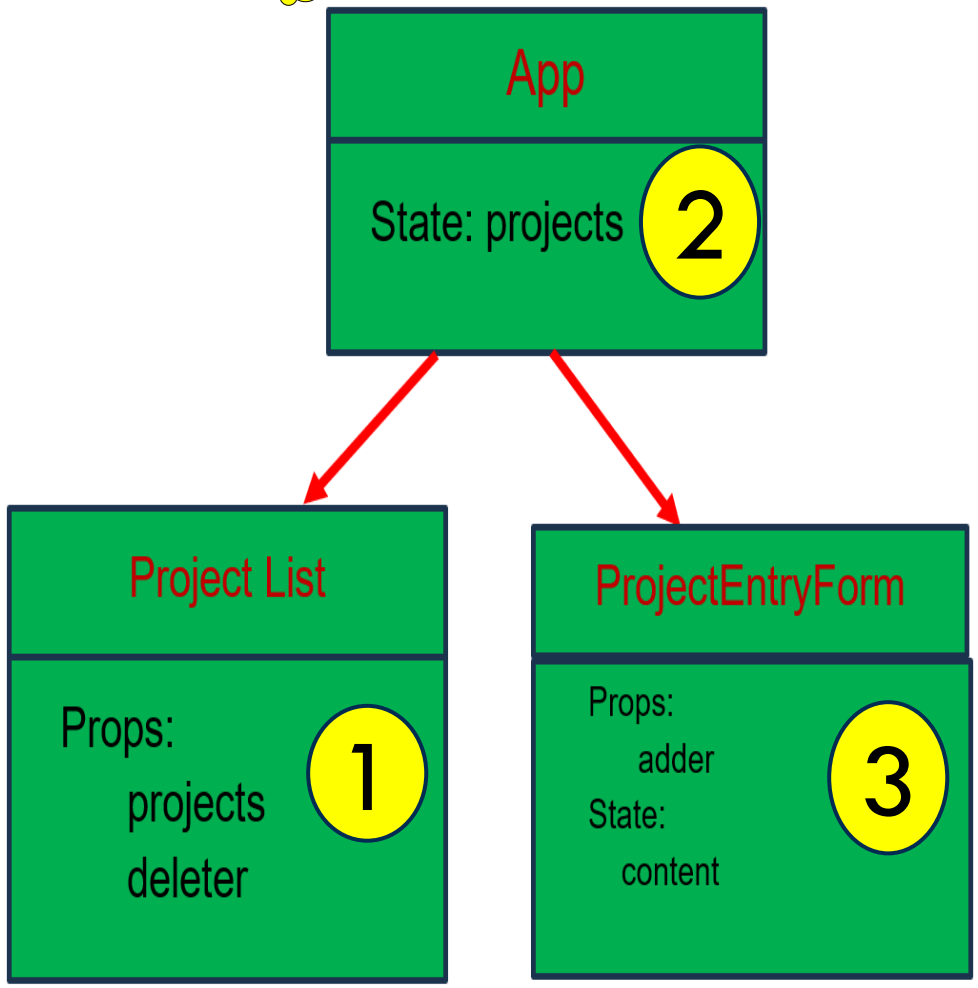
What is take away message?

Git repositories

Discord group

??

Follow split design and establish communication by passing down functionality from parent to child and in future receiving notification from child

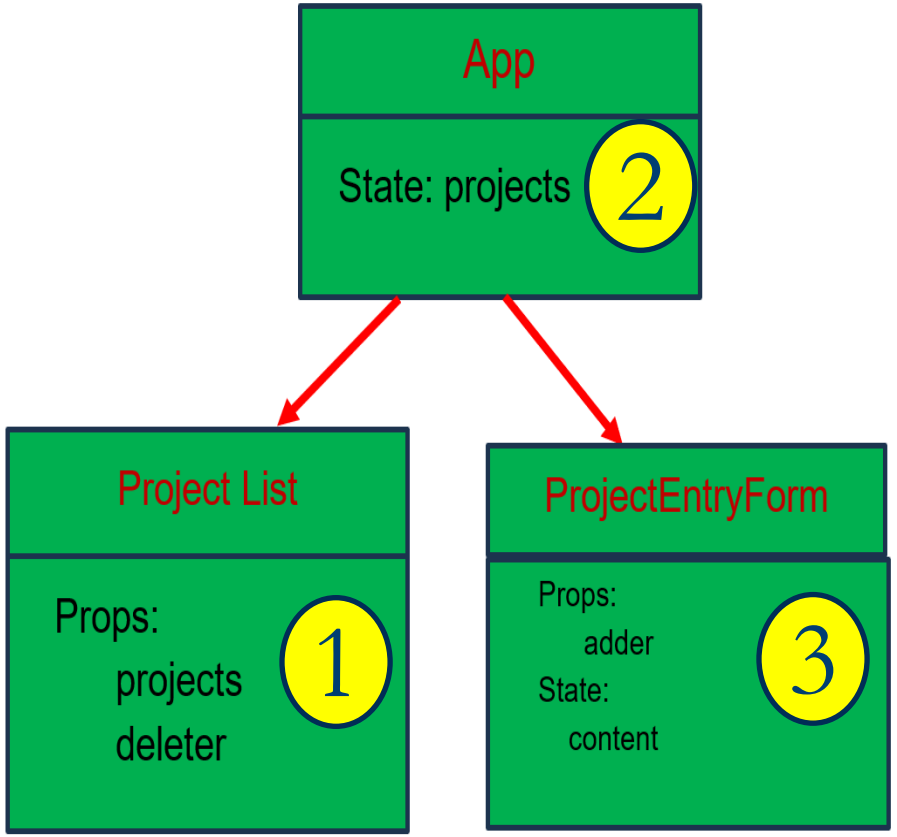


Use given word bank and explain the architectural plan in 3-4 lines!

- ✓Components
- ✓.jsx
- ✓App.jsx
- ✓Parent ,child
- ✓List of projects
- ✓State,Prop
- ✓Passing ?
- ✓Render
- ✓Respond to click
- ✓Lift state

Who is sending and who is receiving what?

https://github.com/humeraaa/react-component-list



App.jsx (Parent) - Holds state (projects)

- ~~ProjectList.jsx (Child) - Receives: projects~~
- DeleteFromList.jsx (Child) - Receives: projects, deleter
- ProjectEntryForm.jsx (Child) - Receives: adder

Redundant code

UNIVERSITY OF KARACHI

@@@@@@@@@@@@@@@@ Version 3: src/ProjectList.jsx @@@@@@@@@@@@@@@@@@@@

@@@@@@@@@ Inline JSX with Ternay Operator @@@@@@@@

Uses **inline JSX , no pList variable**

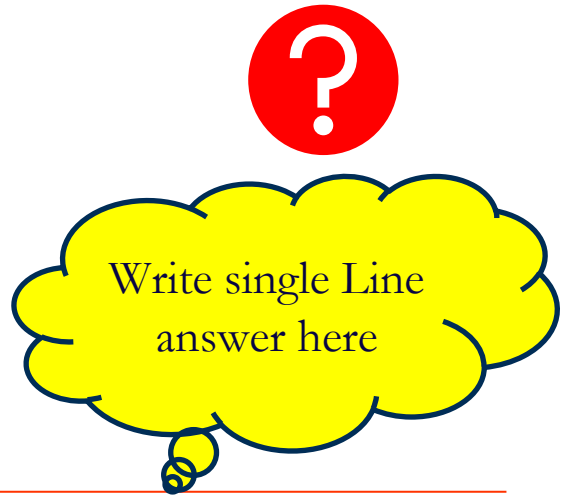
- projects.length ? … : … → This is the **ternary operator**.
- **Condition:** projects.length
- **If true:** map() over the projects and display them.
- **If false:** Show a No Projects left .

Redundant code that displays projects ProjectList.jsx ??

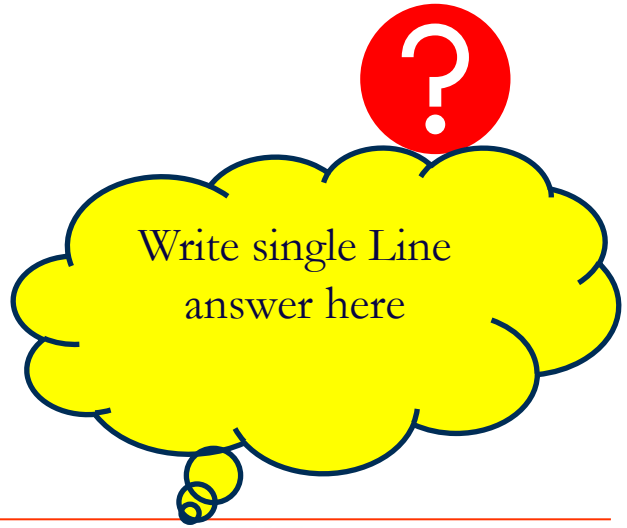
```
export default function _____ ({ projects }) {
    const pList = _____ ? (
      .map() to iterate over an array of objects,
        transform it and return .jsx elements
    ) : (
      <p>No Projects left</p>
    );
    return <div>{pList}</div>;  }
```

Take Away Message from List Exercise

Usually what React developer doing? is



and



UNIVERSITY OF KARACHI

pass down functionality to children in props,

and

pass up notifications from children in events
(or better yet: dispatch).

D: > ___BSCS_IAD > week08 > www > src > App.jsx > ...

```jsx
import './App.css'
import { useState } from 'react'; // Import useState from React to use state in a func
import ProjectList from './ProjectList';   //{/*Think of <ProjectList /> like a reusab
```

```jsx
{/*Create a state variable called "projects" and a setter function called "setProjects"
 This stores array of objects to manage the list of projects, each with an id and content*/}
const [projects, setProjects] = useState([
  {id: 1, content: 'DIP with Python'},
  {id: 2, content: 'FYP with undergrads'},
]);
```

Missing functions ??

Project Structure (Simplified)

bash

```
/my-react-app
  ├── /src
  │     ├── App.jsx          # Main component
  │     ├── main.jsx         # React entry point
  │     ├── index.css        # Global styles
  │     ├── assets/          # Images, icons, etc.
  ├── package.json
  ├── vite.config.js         # Vite configuration
  ├── index.html             # Main HTML file
```

```jsx
return (
  <>
    <h1>Capstone Projects List</h1>
    {/* / Only Display Projects → Pass only projects.        Display + Delete Projects
    <ProjectList projects={projects} />
    <DeleteFromList projects={projects} deleter={deleteProject} />
    <ProjectEntryForm adder={addProject} />
  </>
);
}
```

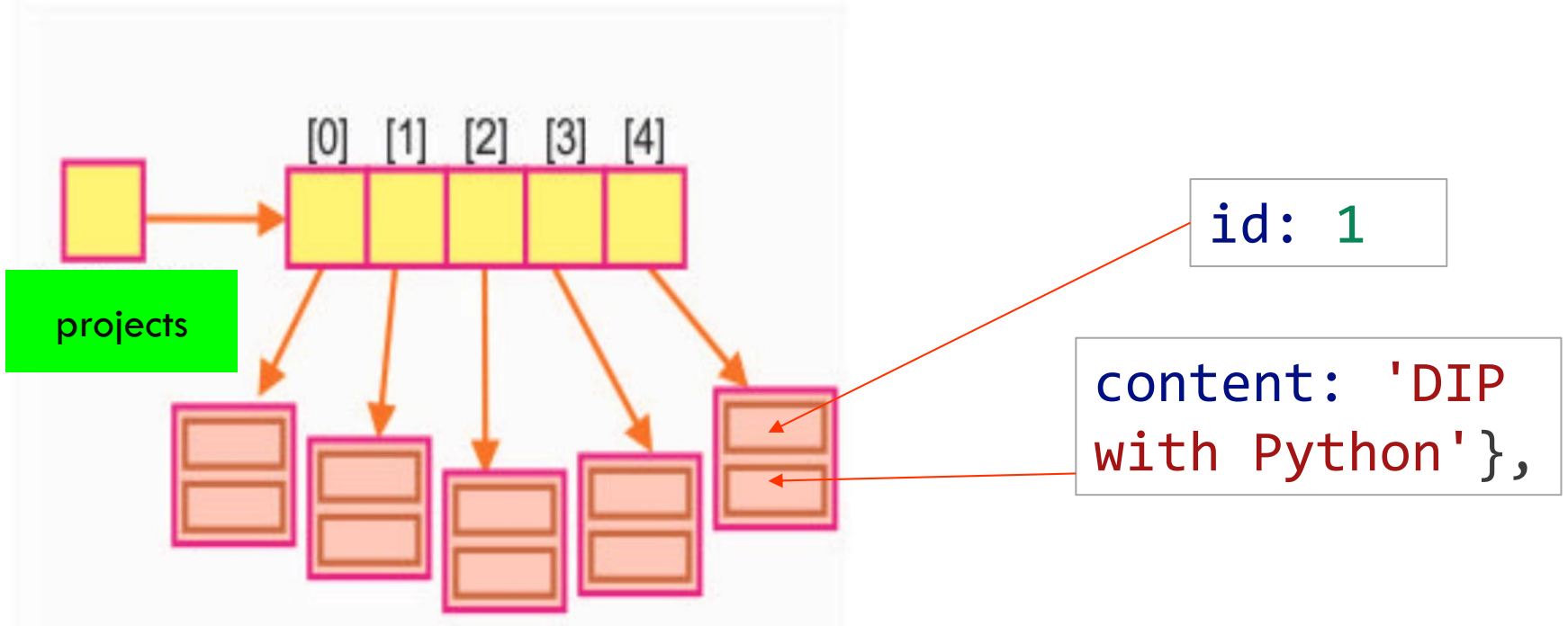
Recall the take-away message??

```
const [projects, setProjects] = useState([
    {id: 1, content: 'DIP with Python'},
    {id: 2, content: 'FYP with undergrads'},
    { ____________________________________ },
  ]);
```

useState update the list dynamically (e.g., adding, removing, editing, items)

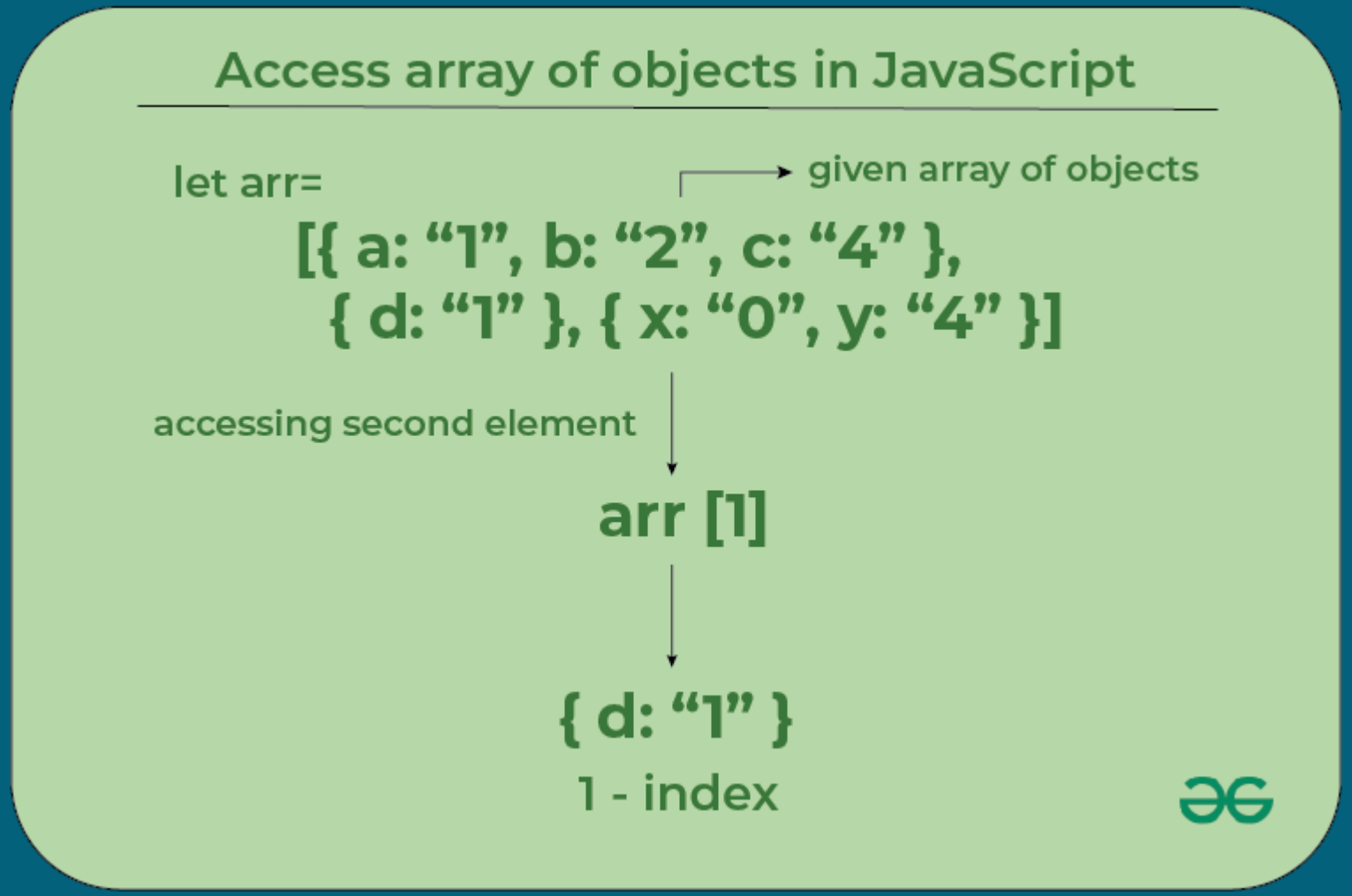
Here's the shortest yet clear interpretation:

projects holds an array of project objects, and setProjects updates it.



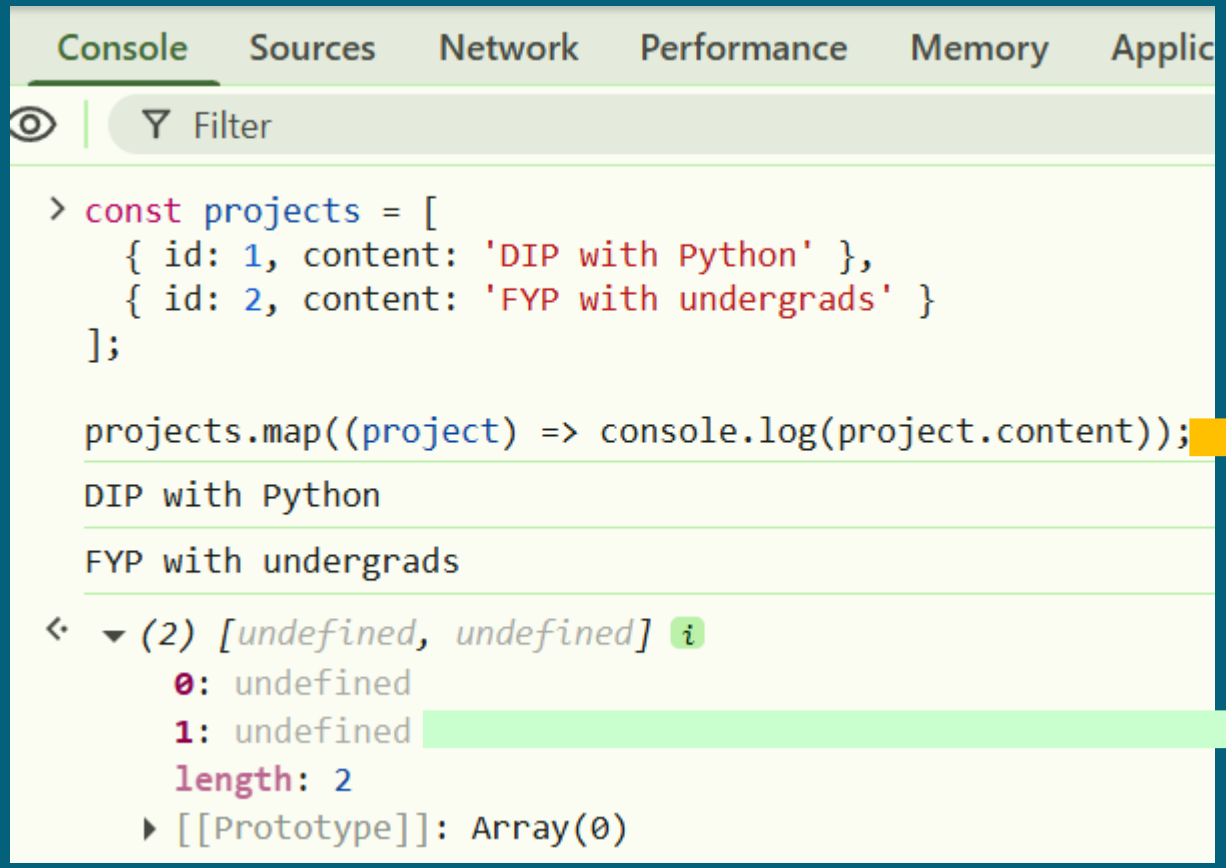
Array of Objects stored in useState()

Replacing useState with a let/const Variable (Static Data)



Java script Array of Objects

- **Press F12** or **Ctrl + Shift + J**
- **create the same array of objects on developers' console**



Inside .map(), console.log(project.content) executes synchronously for each element in the array as .map() iterates.

How to avoid undefined ??

projects.map() is an array method that iterates over each element in the projects array, executing the provided callback function. The callback function receives an individual project object and logs its content property to the console.

To avoid undefined either use .forEach() OR return newArray!

```
> projects.forEach((project) => console.log(project.content));
DIP with Python
FYP with undergrads
< undefined
```

```
> const newArray = projects.map((project) => project.content);
console.log(newArray);
▸ (2) ['DIP with Python', 'FYP with undergrads']
< undefined
```

```
const projects = [
  { id: 1, content: "DIP with Python" },
  { id: 2, content: "FYP with undergrads" }
];

projects.forEach(p => console.log(p.id + " > " + p.content));
```

Java script

React

```
const pList = projects.length ? (
  projects.map(p => <div key={p.id}>{p.content}</div>)
) : (
  <p>No Projects left</p>
);
```

```
// Check if projects exist, then map over them; otherwise, show a message
const pList = projects.length
  ? projects.map(p => p.content) // Extract content from each project
  : ["No Projects left"]; // Default message if no projects

console.log(pList); // Output the result
```

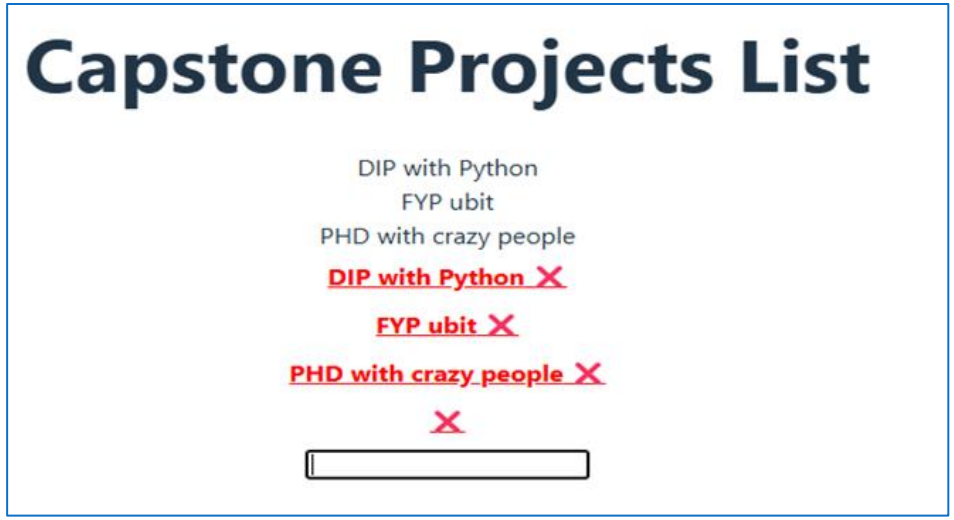
projects.map() goes through each item in the projects array, takes one project at a time, and runs the given call back function on it. In this case, it extracts and logs the content property of each project.

Example 1

Let’s recall User Interface (UI) of Mini-List React Project

student performances > mine

Recall the User Interface for display and delete




```
projects.map(p => <div key={p.id}>{p.content}</div>)
```

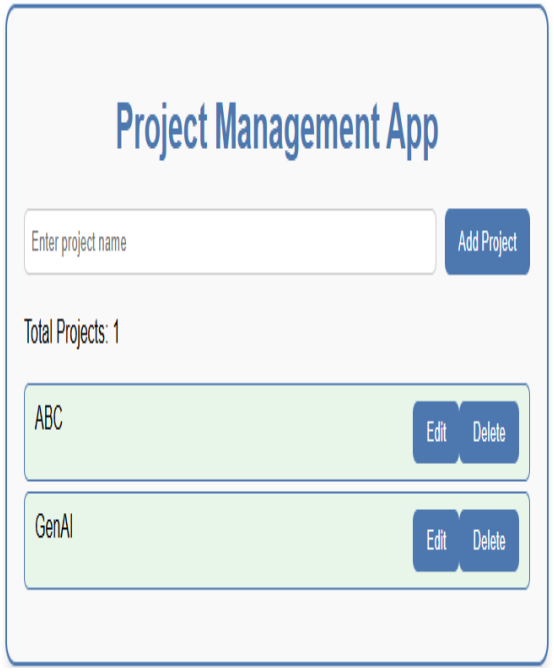
Rendering project items by mapping over projects, using id as the key."

```
onClick={() => deleter(project.id)}
```

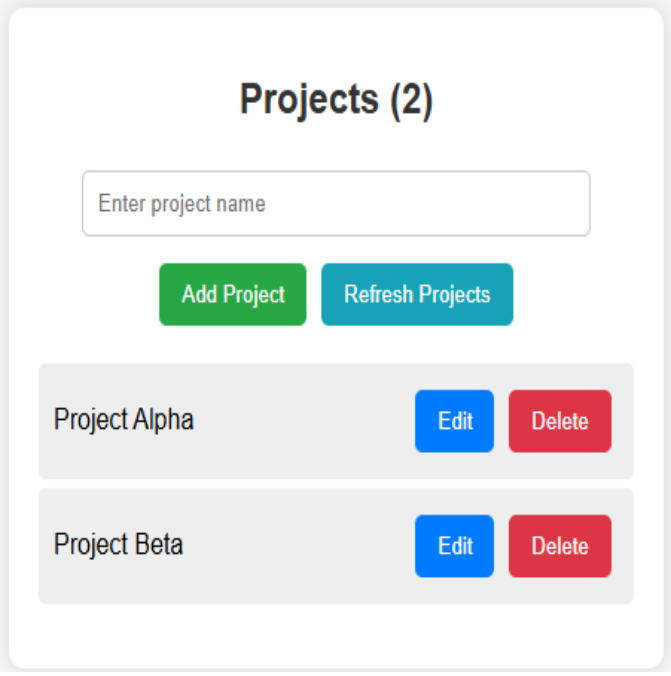
Clicking invokes deleter(project.id), passing the project's ID to trigger deletion.

Sample UIs from Group 4 (Section A)

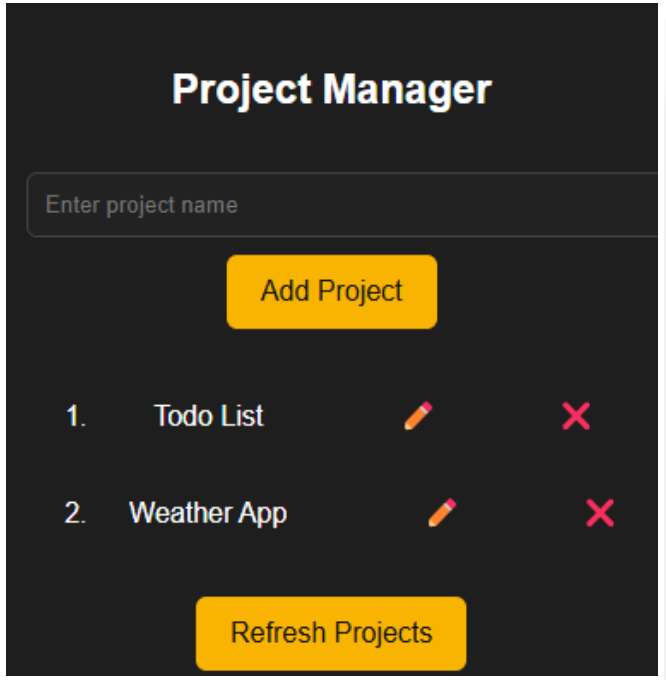
Mehak Fatima B21110006057



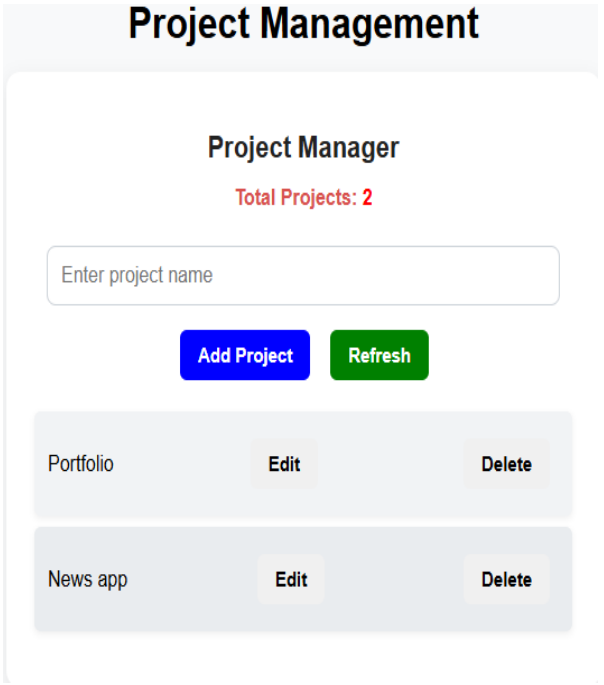
Tahrim Bilal B21110006153



Yumna Mubeen B21110006165

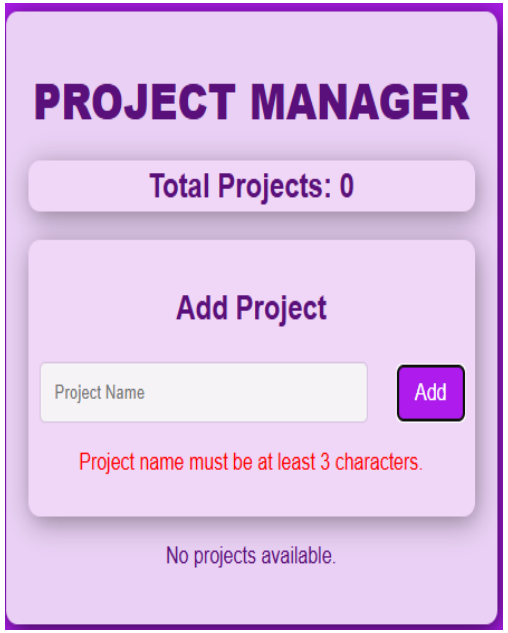


Rimsha Laraib B21110006107

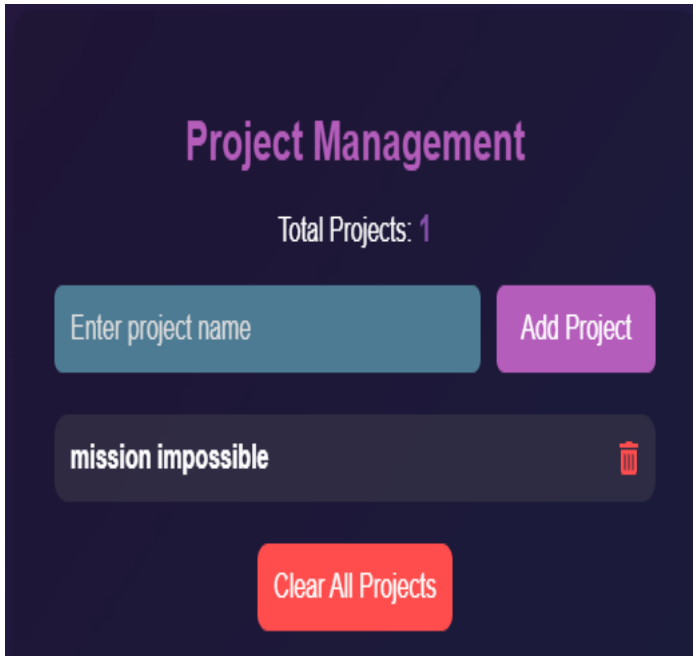


Sample UIs from Group 7 (Section B)

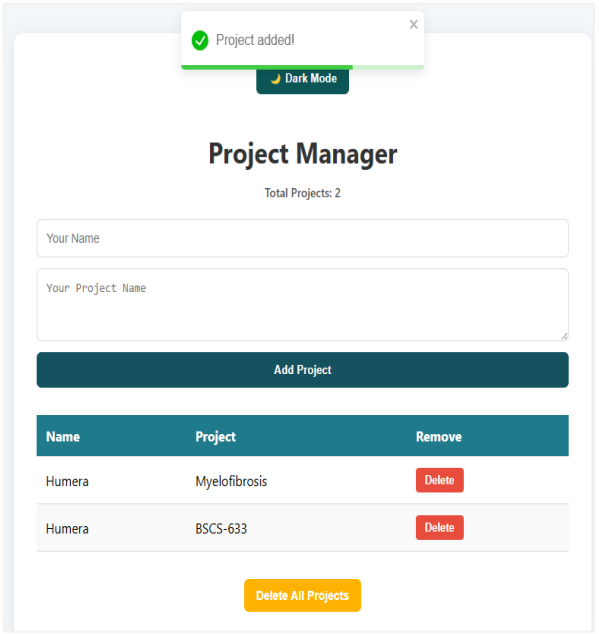
Areesha Abdullah B21110006019



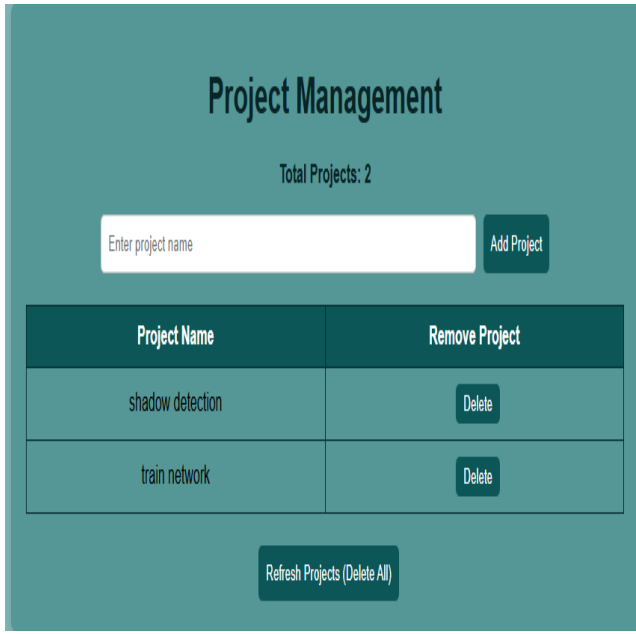
Manal Amin B21110006054



Aiman Ali B21110006011



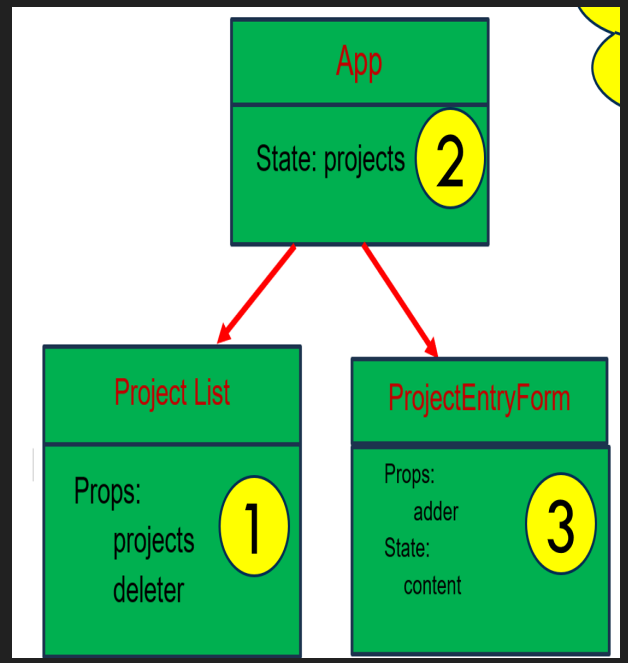
Haniya Hussain B21110006037



The App component passes the deleteProject function as a prop (deleter) to the DeleteFromList component. When a project is clicked for deletion, DeleteFromList calls deleter(project.id), triggering a state update in App via setProjects, which removes the project from the list.

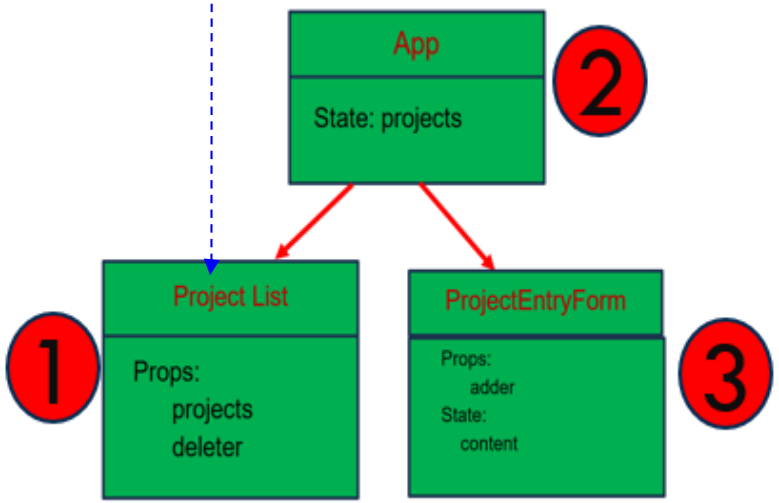
Let's Revise DeleteFromList component in ProjectList.jsx

```
// Delete-Enabled Project List (Named Export)
export function DeleteFromList({ projects, deleter }) {  ??     }
```



DeleteFromList ()

ProjectList.jsx
1st child component




```
//=============================================================
  // delete on click
 //=============================================================

// Delete-Enabled Project List (Named Export)
export function DeleteFromList({ _____ , _____ }) {
    if (!projects.length) return <p>No Projects left</p>;

    return (
      <div>
        {projects.map((project) => (
          <div key={project.id}>
            <span
              onClick={() => [          ]}
              style={{
                cursor: "pointer",
                color: "red",
                textDecoration: "underline",
                fontWeight: "bold",
                display: "  [          ]  ,
                padding: "5px",
              }}
            >
              [          ]
            </span>
          </div>
        ))}
      </div>
    );
```

Always concentrate on received data and function

The DeleteFromList component receives _____as a prop from ____and uses it inside _____event handler."

What are other possible alternatives of ?

Capstone Projects List

DIP with Python
FYP ubit
PHD with crazy people
DIP with Python ✕
FYP ubit ✕
PHD with crazy people ✕
✕
[   ]

```
export default function App() {  // parent component

  const [projects, setProjects] = useState([
    {id: 1, content: 'DIP with Python'},
    {id: 2, content: 'FYP with undergrads'},
  ]);
```

```
// Delete function: Removes a project by filtering out the matching ID

 function deleteProject(id) {

   _____(_____ => existingProjects.filter(project => _____ ));

 }
```

The deleteProject function _____ the state by filtering out the project with the given id. It calls _____ , passing a new array that excludes the matching project, ensuring _____(Hint: immutablility/mutability) and triggering a _____in React."

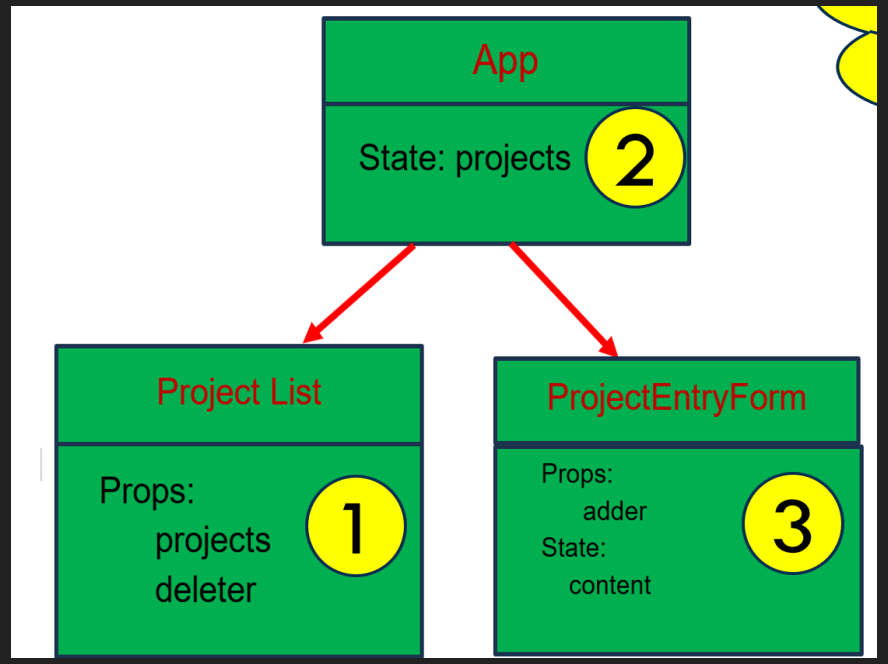
Recall App.jsx

```jsx
// File App.jsx
import './App.css'
import { useState } from 'react'; // Import useState
import ProjectList from './ProjectList';
import { DeleteFromList } from "./ProjectList"; // Named export (Correct!)

export default function App() {  // parent component
   {/*Create a state variable called "projects" and a setter function called
   "setProjects". This stores array of objects, each with an id and content*/}

// Delete function: Removes a project by filtering out the matching ID
  function deleteProject(_____ ) {

      setProjects( _____ => projects.filter( _____ );
  }
  return (
    <>
      <h1>Capstone Projects List</h1>
      {/* / Only Display Projects → Pass only projects.
       Display + Delete Projects → Pass both projects and deleter */}
      <ProjectList projects={projects} />
      <DeleteFromList projects={projects} deleter={ _____ } />
    </>
  );
}
```

Passing projects and deleteProject() as props to child component DeleteFromList

App will pass a function `addProject(newProject)` to the form, as a prop, to update App's state.

Let's code ProjectEntryForm




```
export default function ProjectEntryForm( { adder }) {
  const [content, setContent] = useState('');
```

```
function submit(e) {
        //??
}
```

```
return (
  <form onSubmit={submit}>
    <input value={content} onChange={e => setContent(e.target.value)} />
  </form>
)
```

Event Handling while adding projects:

React supports various event handlers, such as **onClick** and **onChange**, which allow you to execute functions when specific events occur. In this exercise, we'll use the **onSubmit** event handler to handle the submission and deletion of project items, and the **onChange** event handler to update the input field value.

```
11  export default function ProjectEntryForm({ adder }) {
12    const [content, setContent] = useState('');
13    function submit(e) {
14      e.preventDefault();
15      if (content.trim()) {
16        adder({ id: Math.random(), content });
17      }
18      setContent('');
19    }
20
21
22    return (
23      <form onSubmit={submit}>
24        <input value={content} onChange={e => setContent(e.target.value)} />
25      </form>
26    )
27  }
```

`handleSubmit` adds a new todo item to the `projects` array when the "Add project" button is clicked. Before adding the item, it checks if the trimmed input value is not empty to prevent adding empty or whitespace-only items.

We need to prevent empty string project

Do some user testing. You might find this bug: If you hit Enter with an empty project in your form, you'll actually have an empty string project in your list, but it renders as nothing, and you can't get your cursor over it to delete it and you'll never see the "No Project left" ever again.

The form behaves dynamically!

It is controlled by state (content):

- ✓ The <input> field is bound to content, meaning its value changes as the user types.
- ✓ The onChange handler (setContent(e.target.value)) updates the content state in real time.

It triggers updates and re-renders:

- ✓ React automatically re-renders the component whenever the content state changes.
- ✓ This is different from a static HTML form where input fields do not automatically update any underlying state.

It handles form submission dynamically:

- ✓ submit(e) prevents the default form submission behavior (e.preventDefault()).
- ✓ If content.trim() is not empty, the adder function is called to add an item with a unique id.
- ✓ The setContent('') function resets the input field after submission.

UNIVERSITY OF KARACHI

existingProjects → current state (array of projects).

...existingProjects → Spread Operator creates a **new array copy** of the existing projects.

newProject → This is the new project that we want to add.

setProjects([...existingProjects, newProject]) → This updates the state by adding newProject **to the end** of the copied array.

```
import './App.css'
import { useState } from 'react'; // Import useState
import ProjectList from './ProjectList';
import { DeleteFromList } from "./ProjectList"; // Named export (Correct!)
import ProjectEntryForm from './ProjectEntryForm';

export default function App() {  // parent component

  const [projects, setProjects] = useState([
    {id: 1, content: 'DIP with Python'},
    {id: 2, content: 'FYP with undergrads'},
  ]);

  // Delete function: Removes a project by filtering out the matching ID
  function deleteProject(id) {
    //setProjects(projects => projects.filter(project => project.id !== id));
    setProjects(existingProjects => existingProjects.filter(project => project.id !== id));
  }

  function addProject(newProject) {

    setProjects(existingProjects => [...existingProjects, newProject]);
  }
```

```
35  function addProject(newProject) {
36
37    setProjects(existingProjects => [...existingProjects, newProject]);
38  }
```

Have any one practiced A-B-C-D practice ?

Git

Discord group

??

```jsx
import React, { useState } from "react";

function ParentChild() {
  const [x, setX] = useState(0);

  const handleXClick = () => {
    setX(x + 1);
  };
```

https://github.com/humeraaa/react-state-lifting

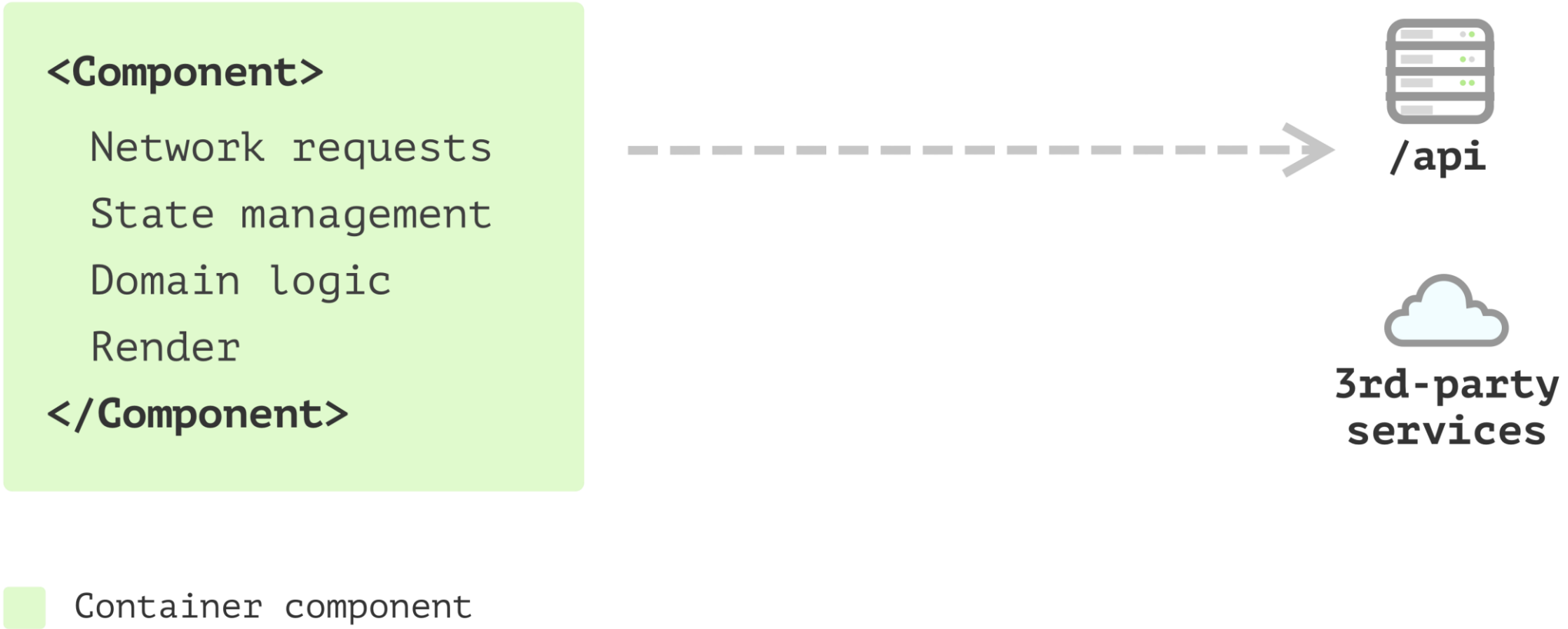
Share Your Experience

```jsx
  return (
    <div className="component">
      <h1>Component A</h1>
      <p>this.state.x = {x}</p>
      {/* Not working becoz JSX treats => as part of the HTML, not JavaScript.  */}
      {/*this.handleXClick => {"this.setState({x: this.state.x + 1})"} <br />*/}
      <p>this.handleXClick =&gt; {"setX(x + 1)"}</p>  {/* ✅ FIXED: &gt; for ">" */}

      {/* ✅ FIXED: backticks */} {/*Solution 2: Wrap Entire Content in {} and inside backticks*/}
      <p>{`this.handleXClick => setX(x + 1)`}</p>
      <B x={x} onXClick={handleXClick} />
      <D x={x} onXClick={handleXClick} />
    </div>
  );
}
```

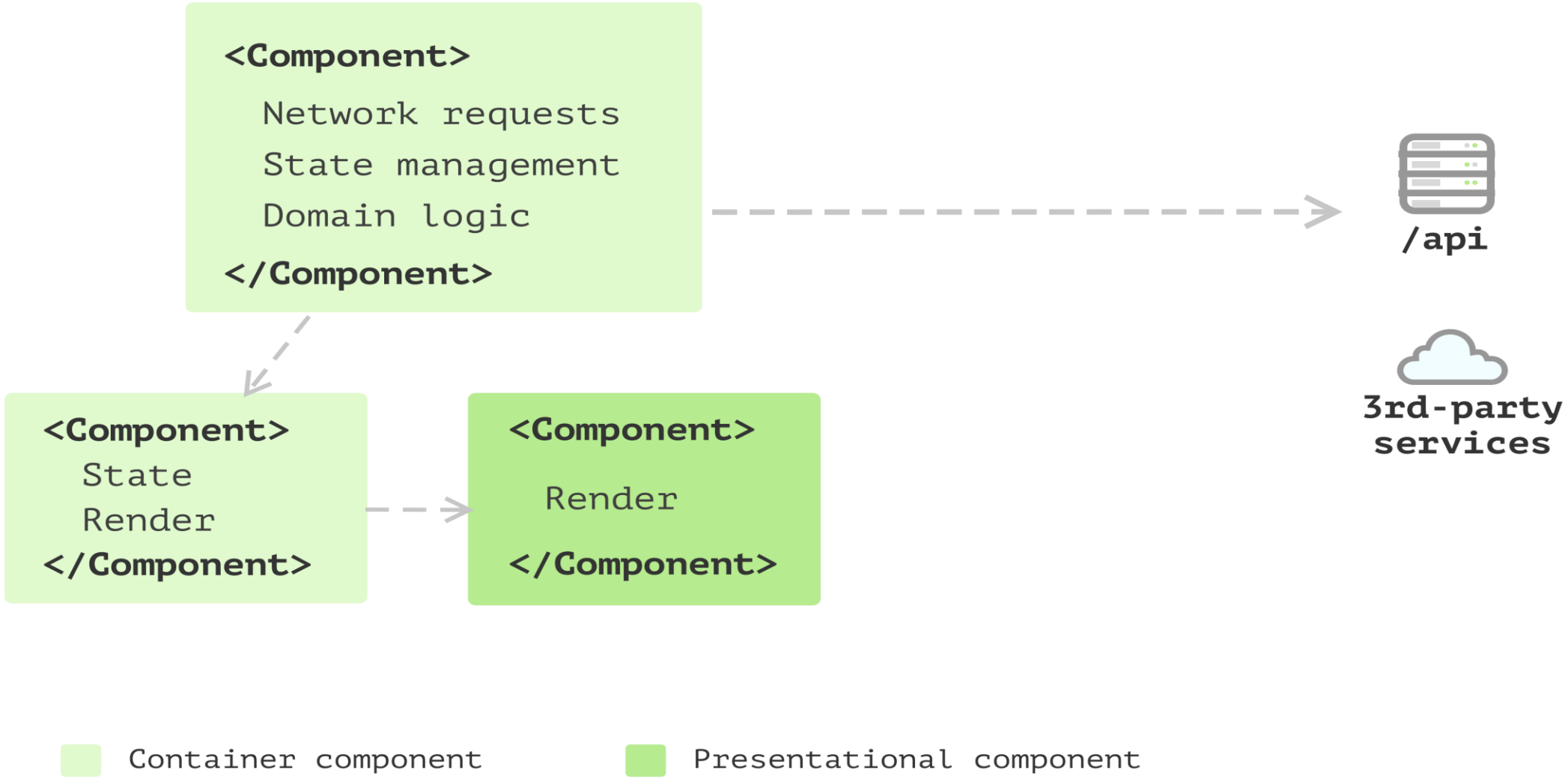
JSX is not html

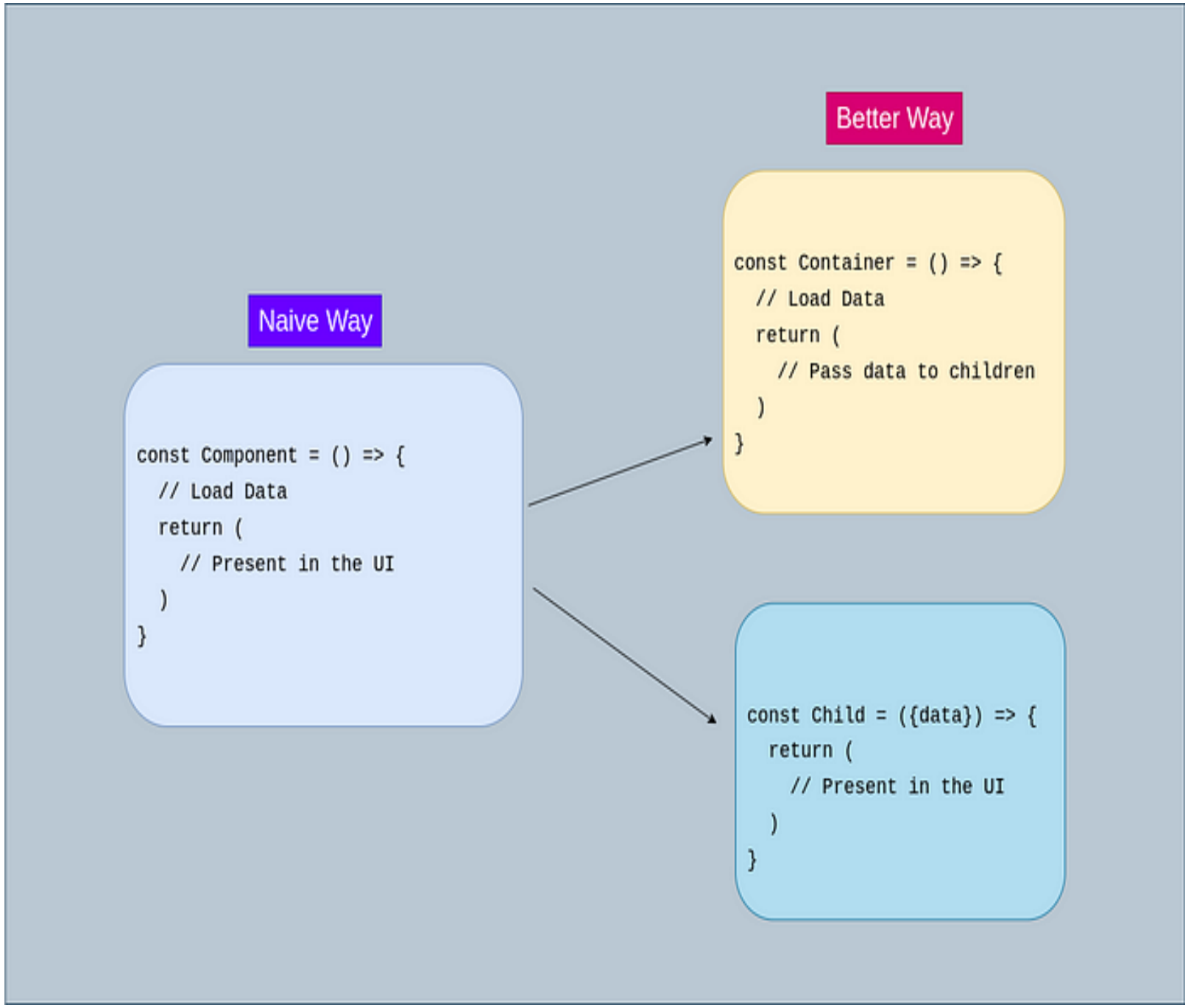
What we learned so far ?

Packing all the code into components may work in small applications like a Todo List / Project List or one-form application.



Multiple Component Application with predefined roles is good design approach





In this pattern, Container Components are responsible for managing data and state logic.

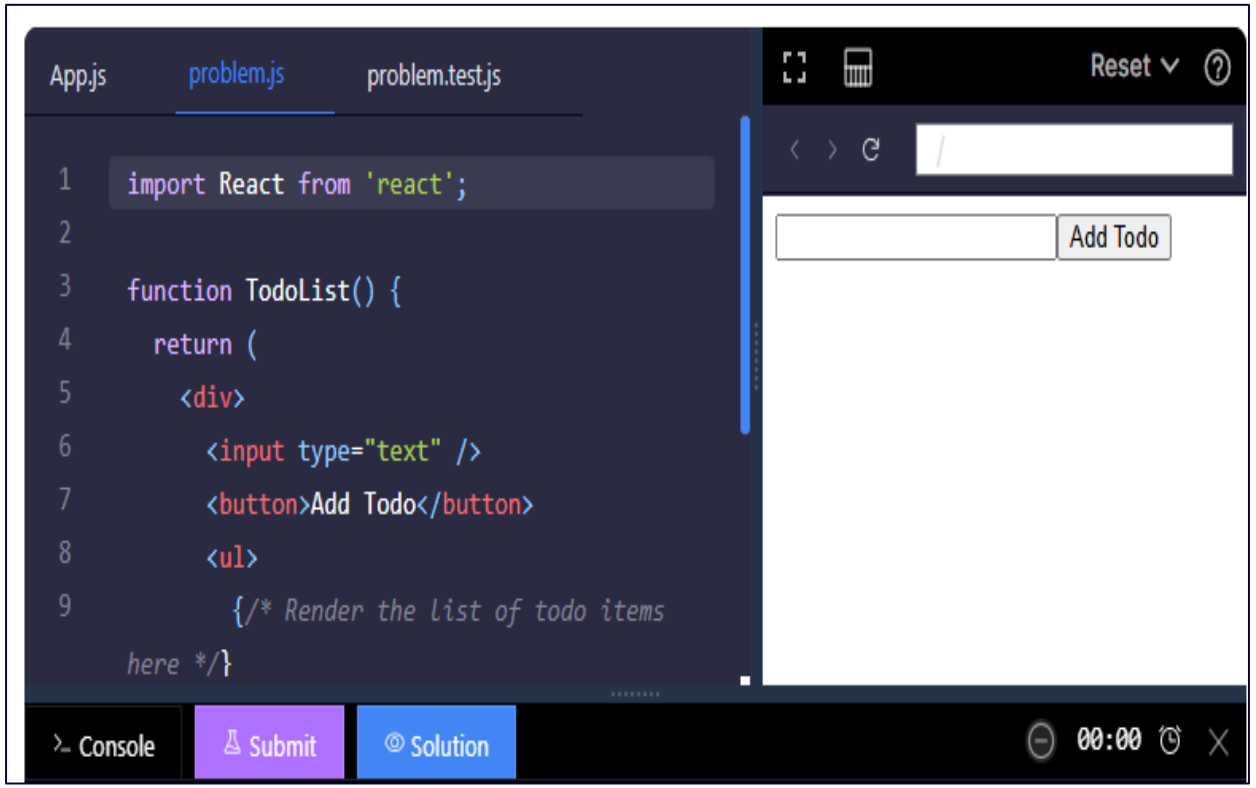
They fetch data from external sources, manipulate it if necessary, and pass it down to Presentational Components as props.

They are often connected to external services, Redux stores, or context providers.

Event Handling in General!

[Practice React's useState hook with 8 Interactive Exercises | Clientside Goodies]

React supports various event handlers, such as **onClick** and **onChange**, which allow you to execute functions when specific events occur. In this exercise, we'll use the **onSubmit** event handler to handle the submission and deletion of project items, and the **onChange** event handler to update the input field value.




```
import React from 'react';

function TodoList() {
  return (
    <div>
      <input type="text" />
      <button>Add Todo</button>
      <ul>
        {/* Render the list of todo items here */}
```

Rendering Draggable List Items:

We render the list items with the `draggable` attribute and attach the appropriate event handlers for the drag events:

Copy code

```
<ul>
  {items.map((item, index) => (
    <li
      key={index}
      draggable
      onDragStart={() => handleDragStart(index)}
      onDragOver={() => handleDragOver(index)}
      onDragEnd={handleDragEnd}
    >
      {item}
    </li>
  ))}
</ul>
```

Firebase

Let's build a blog web application using React with Firebase Authentication, Firestore, and Hosting.

Firebase lets React interact with Firestore **directly** (without writing API endpoints).

Firebase Firestore is a **NoSQL database**, similar to MongoDB.

Firebase simplifies backend logic for small apps

Firestore CRUD operations

Firebase Products

As of October, 2024, Firebase provides these 21 products:

Authenti-cation
About • Docs

Cloud Firestore
About • Docs

Realtime Database
About • Docs

Data Connect
About • Docs

Cloud Storage
About • Docs

Cloud Functions
About • Docs

In-App Messaging
About • Docs

Cloud Messaging
About • Docs

Machine Learning
About • Docs

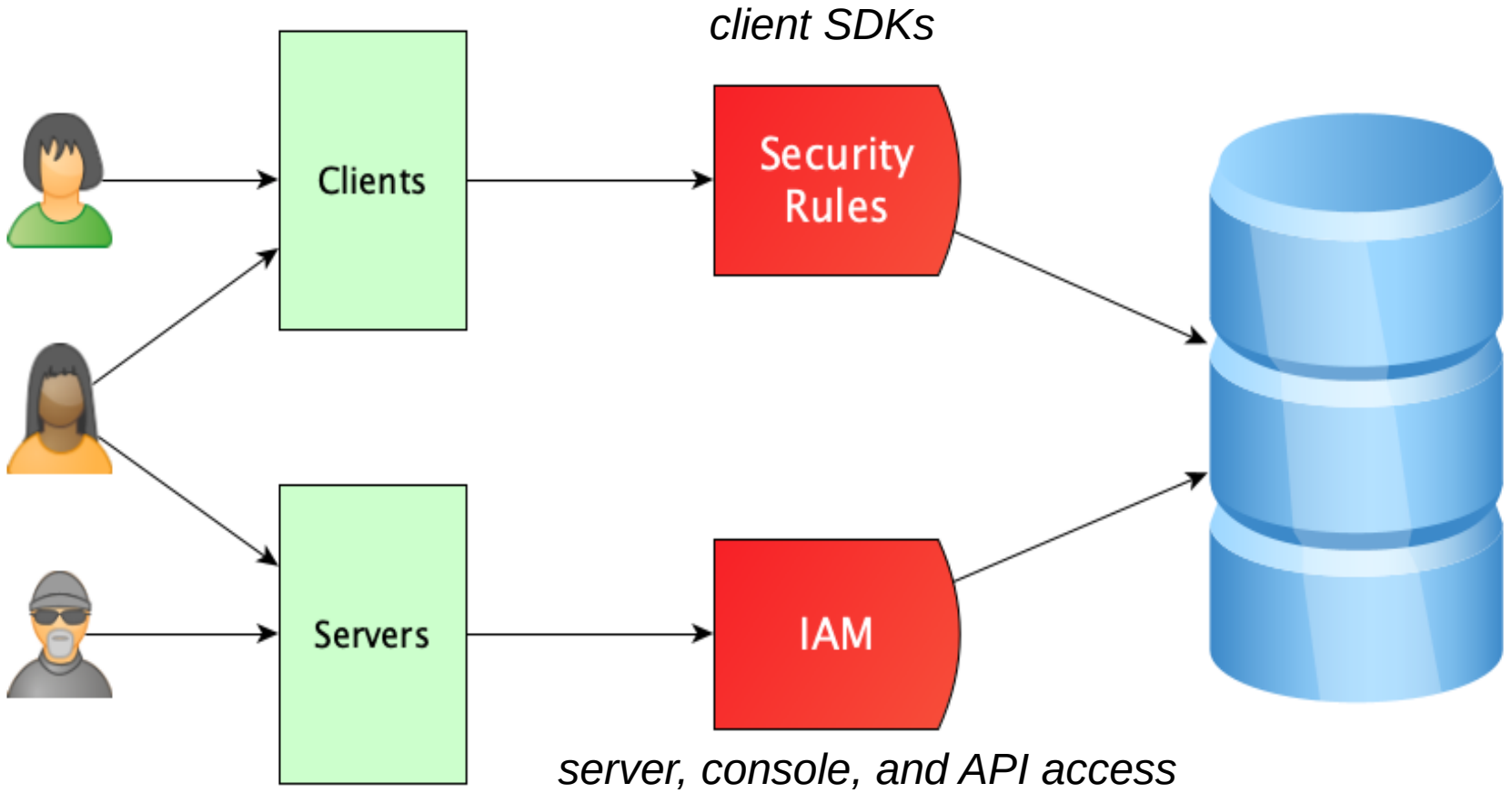
if you are using Firebase form a server, check the docs for how to get set up with the Admin SDK.

Firebase setup

There is complete documentation for Firebase setup on:

- ✓ The web (JavaScript)
- ✓ iOS, macOS, tvOS, watchOS, etc (Swift and Objective C)
- ✓ Android (Java and Kotlin)
- ✓ Unity
- ✓ Flutter

Firebase is part of the Google Cloud, so if you are connecting to Firebase services from a server, or the console, your authentication and authorization will be configured through Google Cloud's **Identity and Access Management** IAM.



But what about that SDK for the web client?, How can a web client talk to...a database 😬 ? The answer is: **security rules**. You configure rules that will examine all requests from a client—the same kinds of rules you would put in a server that your write yourself.

The configured security rules apply only to client SDKs; server, console, and API access use IAM instead.

Security & DB Controls Hints !

Chat GPT: Trust but verify!

Aspect	Client (Vite - React Frontend)	Server (Express - Backend)
File Name	.env / .env.local	.env
Access Method	import.meta.env.VITE_VAR_NAME	process.env.VAR_NAME
Variable Naming	Must start with VITE_ (e.g., VITE_FIREBASE_API_KEY)	No prefix needed (e.g., MONGO_URI, JWT_SECRET)
Security Level	**Low** – Exposed in browser (visible via DevTools)	**High** – Hidden from clients (server-side only)
Sensitive Data?	Never store secrets (use only public API keys)	**Store secrets** (DB credentials, JWTs, API keys)
Database Control	Uses **Firebase Security Rules** for access control	Uses **Backend authentication & middleware** (e.g., JWT, OAuth)
Best Practices	- Restrict Firebase API key in Firebase Console - Use Firestore rules to limit access	- Use .gitignore to prevent committing .env - Use authentication & authorization (JWT, OAuth)

It's a good idea to learn everything you can about security rules, both in terms on the general approach and the specifics for different products:

- ✓ Start here for Firebase Security Rules
- ✓ Firestore-specific security
- ✓ Cloud Storage-specific security

So, if a web client is talking directly for Firebase, there is no way to hide your database credentials at all, and you must rely on the security rules to prevent all the bad things.

You can "do a little more" beyond the security rules, for example, you can:

- ✓ Place your firebase config in a file that is not committed to GitHub. This does not actually prevent people from finding your config, but it's nice to do.
- ✓ Arrange that Firestore will only respond to requests coming from certain places, by setting the allowed HTTP referrers. This makes development a little clunky, and referrers can be spoofed, but again, it's something.
- ✓ Use App Check.
- ✓ There are some articles with security tips floating around the web, including How to Keep Your Firebase Project Safe and Secure from everyone.

Step 0: Prerequisites

Make sure you have:

- ✓ a Google account,
- ✓ VS Code,
- ✓ Node.js

Also, you should be pretty good with

- ✓ JavaScript and
- ✓ have previously written React applications with at-least useState and useEffect.

Step 1: install the Firebase CLI tools

> npm install -g firebase-tools

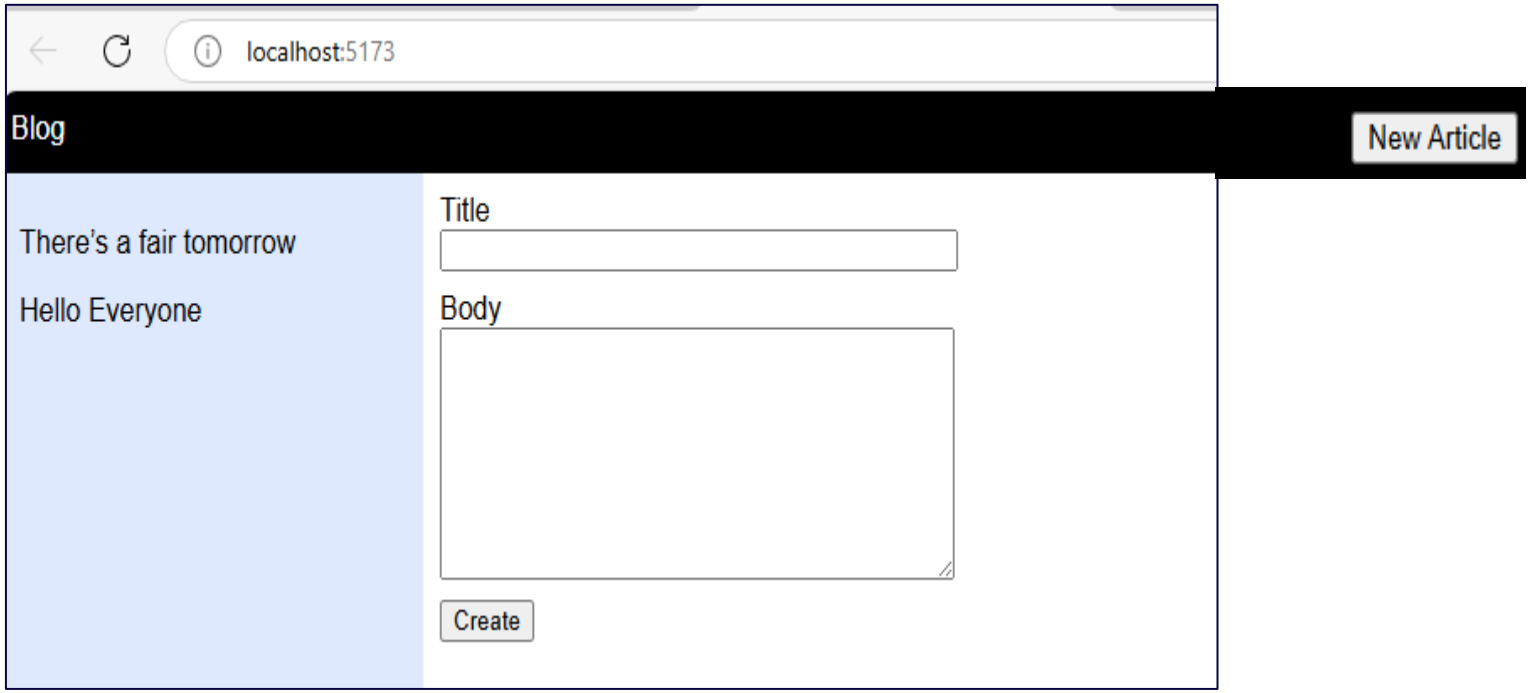
> firebase --version

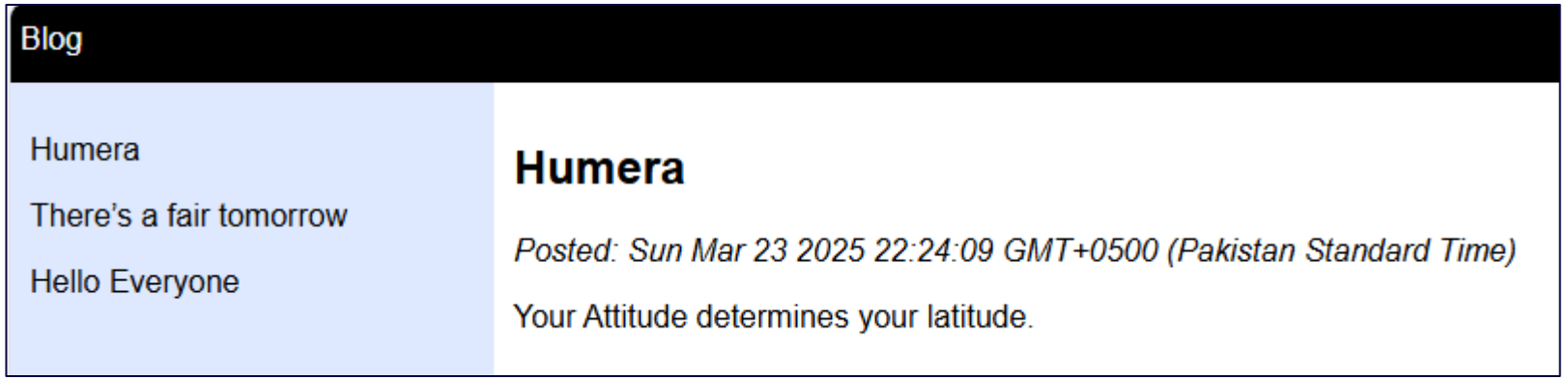
> firebase login

C:\Users\humer>firebase login:list
Logged in as humera@uok.edu.pk

Step 2: Get the Starter code

```
├── README.md
├── eslint.config.js
├── vite.config.js
├── index.html
├── package-lock.json
├── package.json
└── src
    ├── main.jsx
    ├── components
    │   ├── App.css
    │   ├── App.jsx
    │   ├── Article.jsx
    │   ├── ArticleEntry.jsx
    │   └── Nav.jsx
    └── services
        └── articleService.js
```

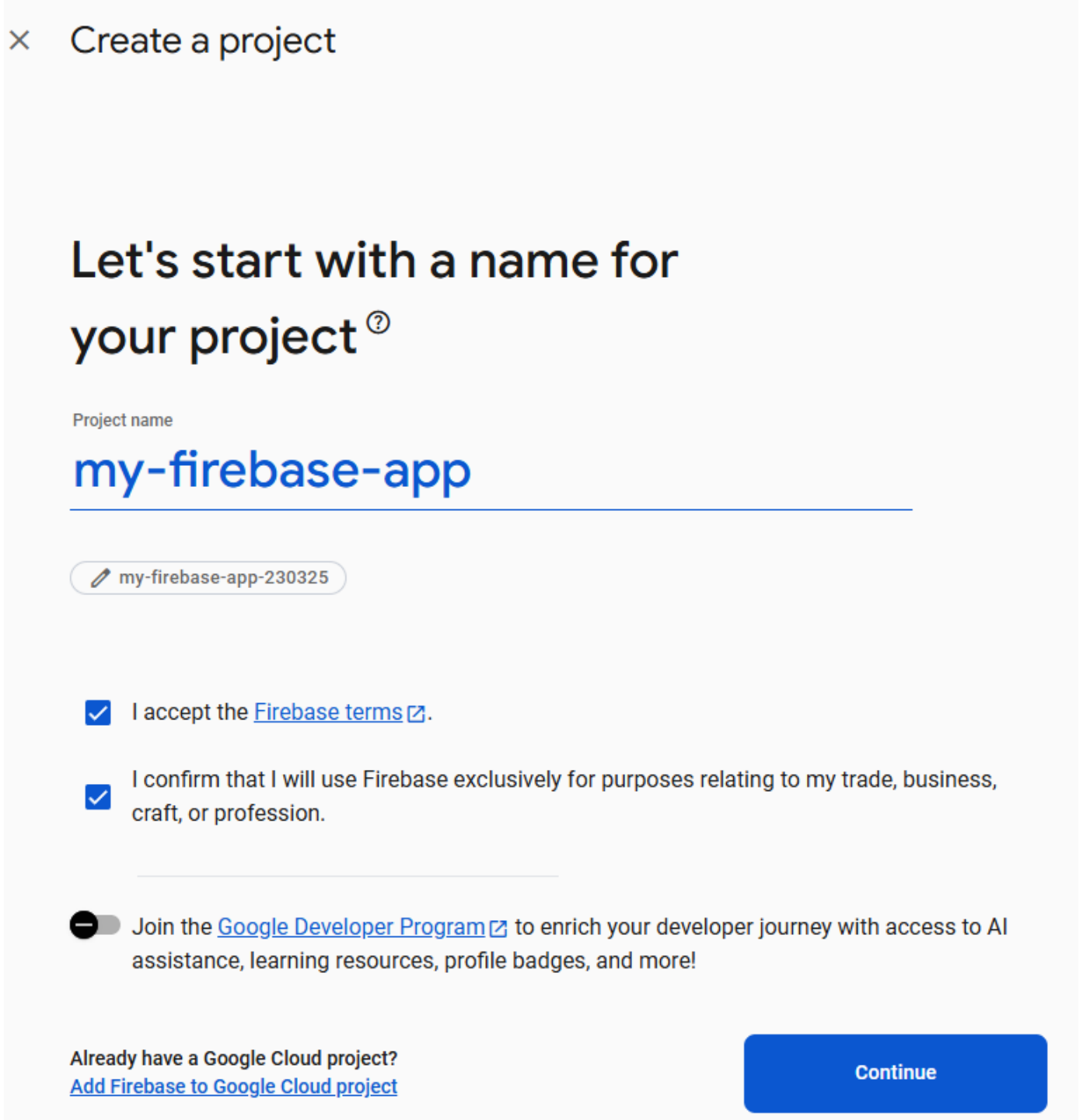

localhost:5173
Blog
New Article
There's a fair tomorrow
Hello Everyone
Title
Body
Create


Blog
Humera
There's a fair tomorrow
Hello Everyone
Humera
Posted: Sun Mar 23 2025 22:24:09 GMT+0500 (Pakistan Standard Time)
Your Attitude determines your latitude.

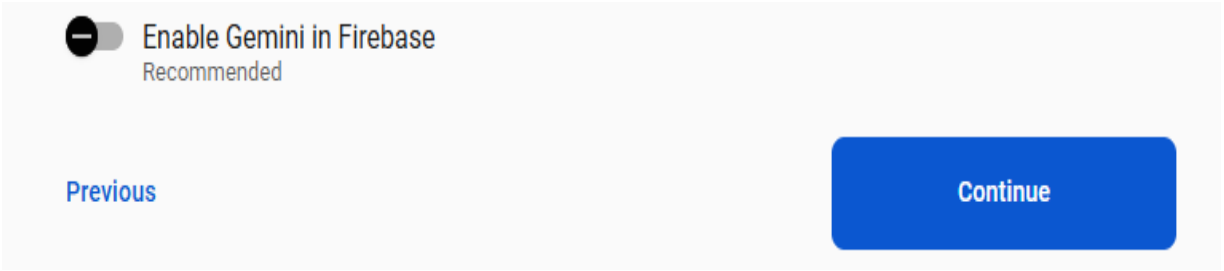
Step 3: Create the Project in the Firebase Console

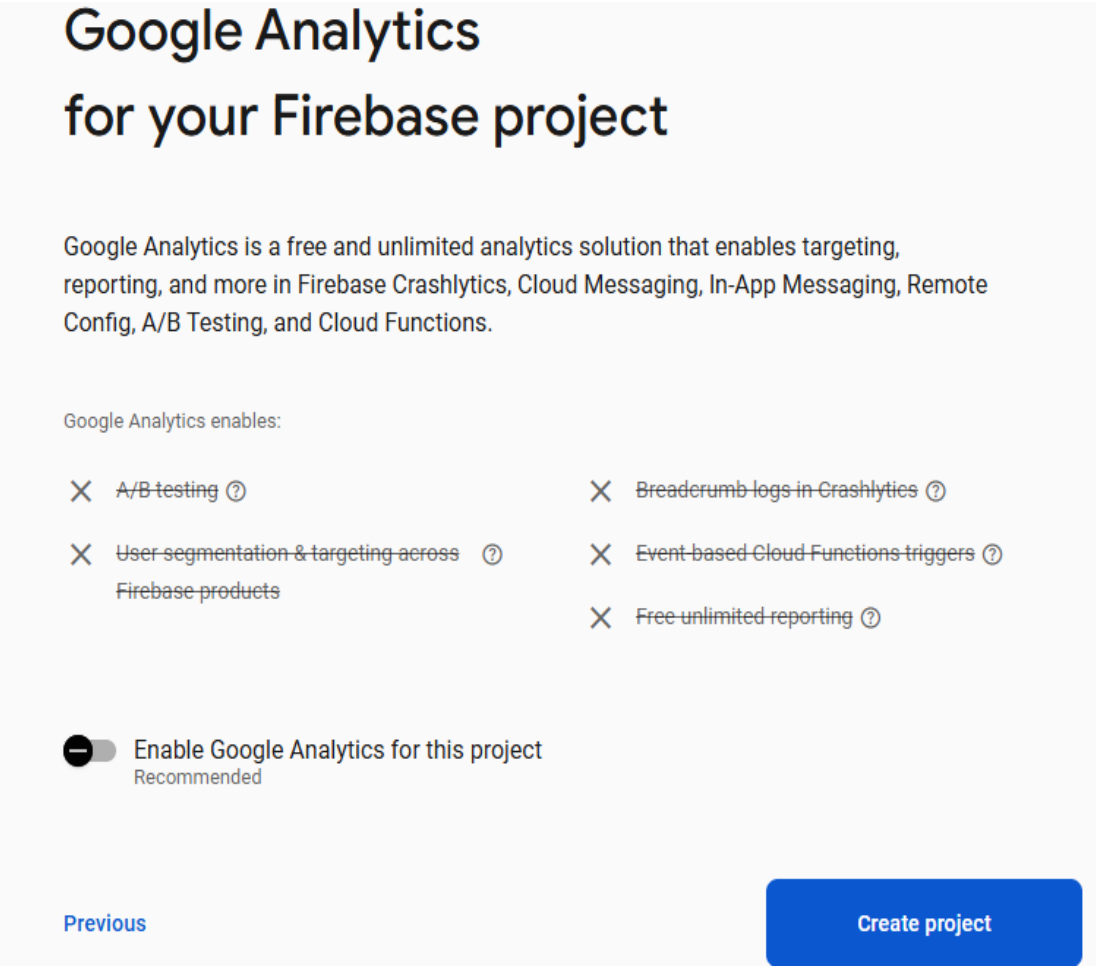
Step 3: Create the Project in the Firebase Console

Open your browser and visit:
👉 https://console.firebase.google.com/

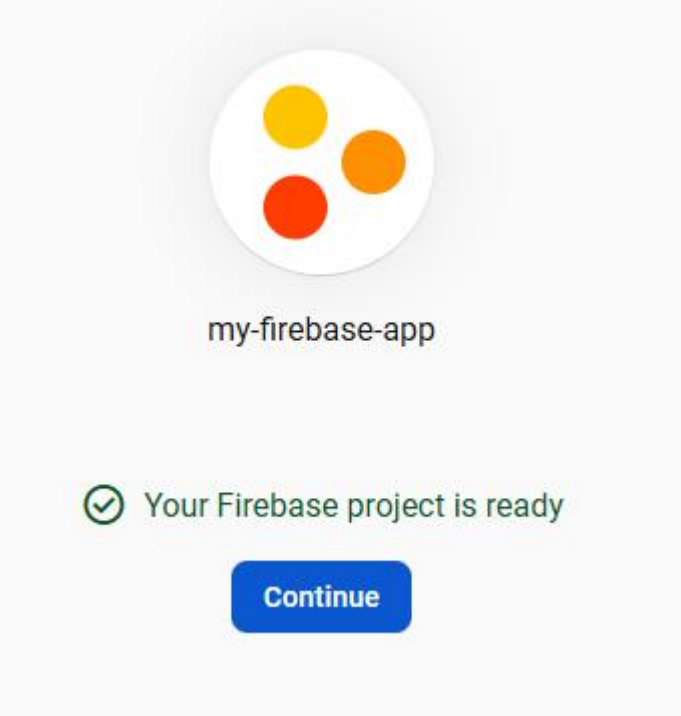


Disable

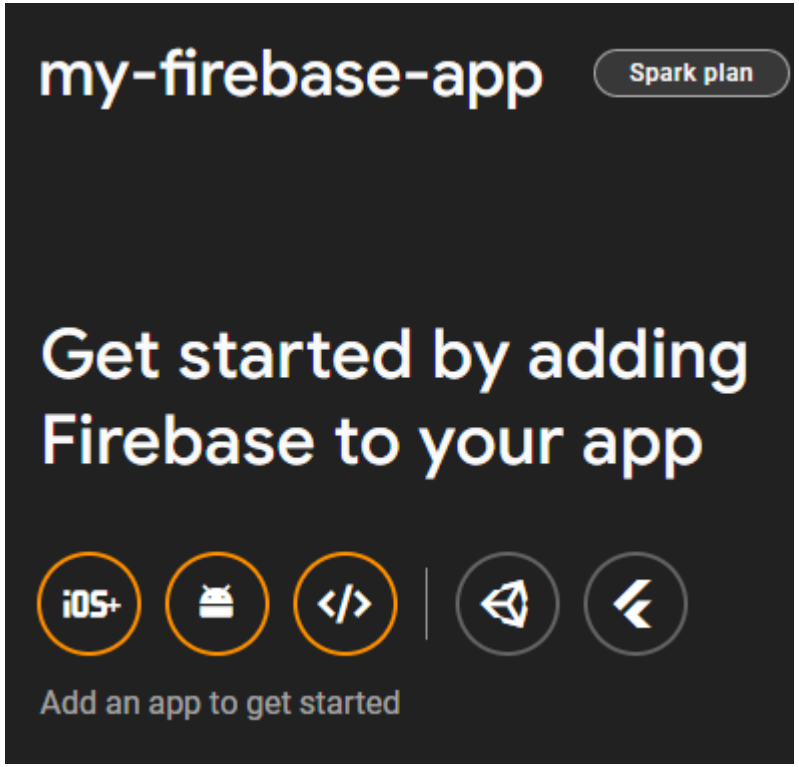

Enable Gemini in Firebase
Recommended
Previous
Continue

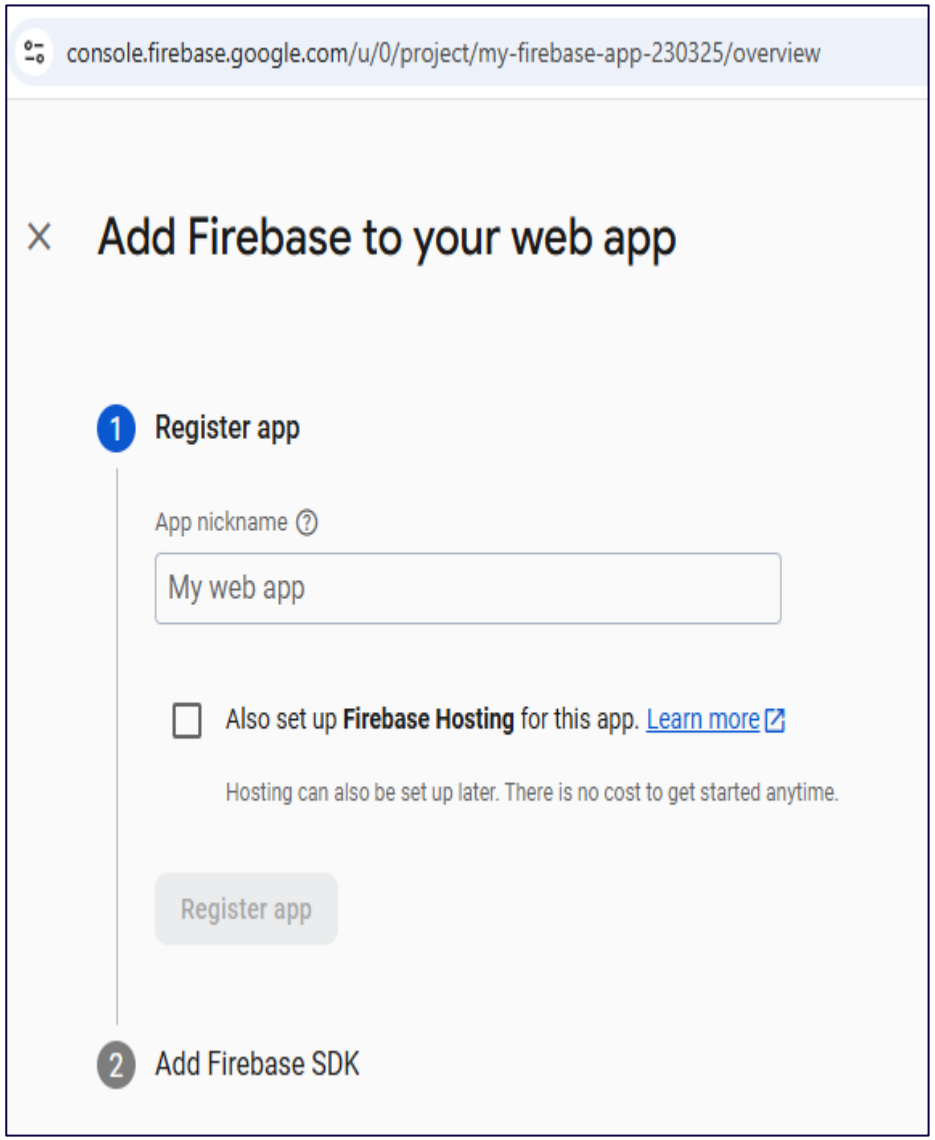

Google Analytics
for your Firebase project
Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, and Cloud Functions.
Google Analytics enables:
A/B testing
User segmentation & targeting across Firebase products
Breadcrumb logs in Crashlytics
Event-based Cloud Functions triggers
Free unlimited reporting
Enable Google Analytics for this project
Recommended
Previous
Create project

Go to your Firebase console and do Create Project. Enter a project name, but do *not* add Google Analytics for this, as it is just a code-along.



In **Add Firebase to your App**, select Web (it look like </>)





Register app

2 Add Firebase SDK

Use npm Use a <script> tag

If you're already using npm and a module bundler such as webpack or Rollup, you can run the following command to install the latest SDK (Learn more):

```
$ npm install firebase
```

Then, initialize Firebase and begin using the SDKs for the products you'd like to use.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyATqeFXugQm70B9deePWtCrouxjlJo9SPk",
  authDomain: "my-firebase-app-230325.firebaseapp.com",
  projectId: "my-firebase-app-230325",
  storageBucket: "my-firebase-app-230325.firebasestorage.app",
  messagingSenderId: "434225677666",
  appId: "1:434225677666:web:c581702d83feb51348d83b"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
```

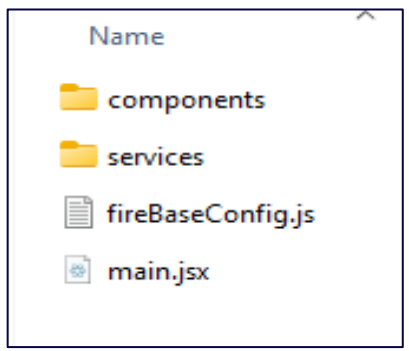
Note: This option uses the modular JavaScript SDK, which provides reduced SDK size.

Learn more about Firebase for web: Get Started, Web SDK API Reference, Samples

Continue to console

```
├── README.md
├── eslint.config.js
├── vite.config.js
├── index.html
├── package-lock.json
├── package.json
└── src
    ├── main.jsx
    ├── components
    │   ├── App.css
    │   ├── App.jsx
    │   ├── Article.jsx
    │   ├── ArticleEntry.jsx
    │   └── Nav.jsx
    └── services
        └── articleService.js
```

https://console.firebase.google.com/u/0/project/my-firebase-app-230325/overview



JS fireBaseConfig.js ● JS articleService.js

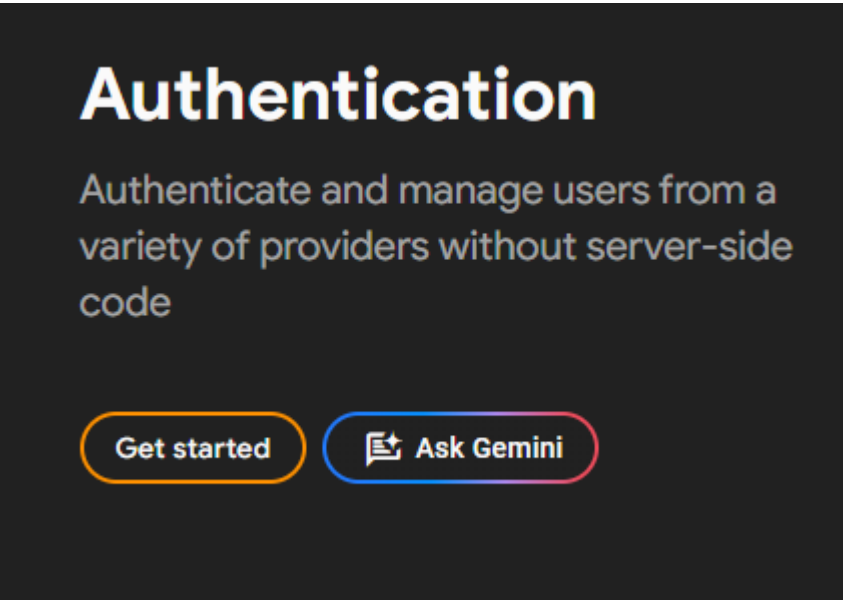
src > JS fireBaseConfig.js > ...

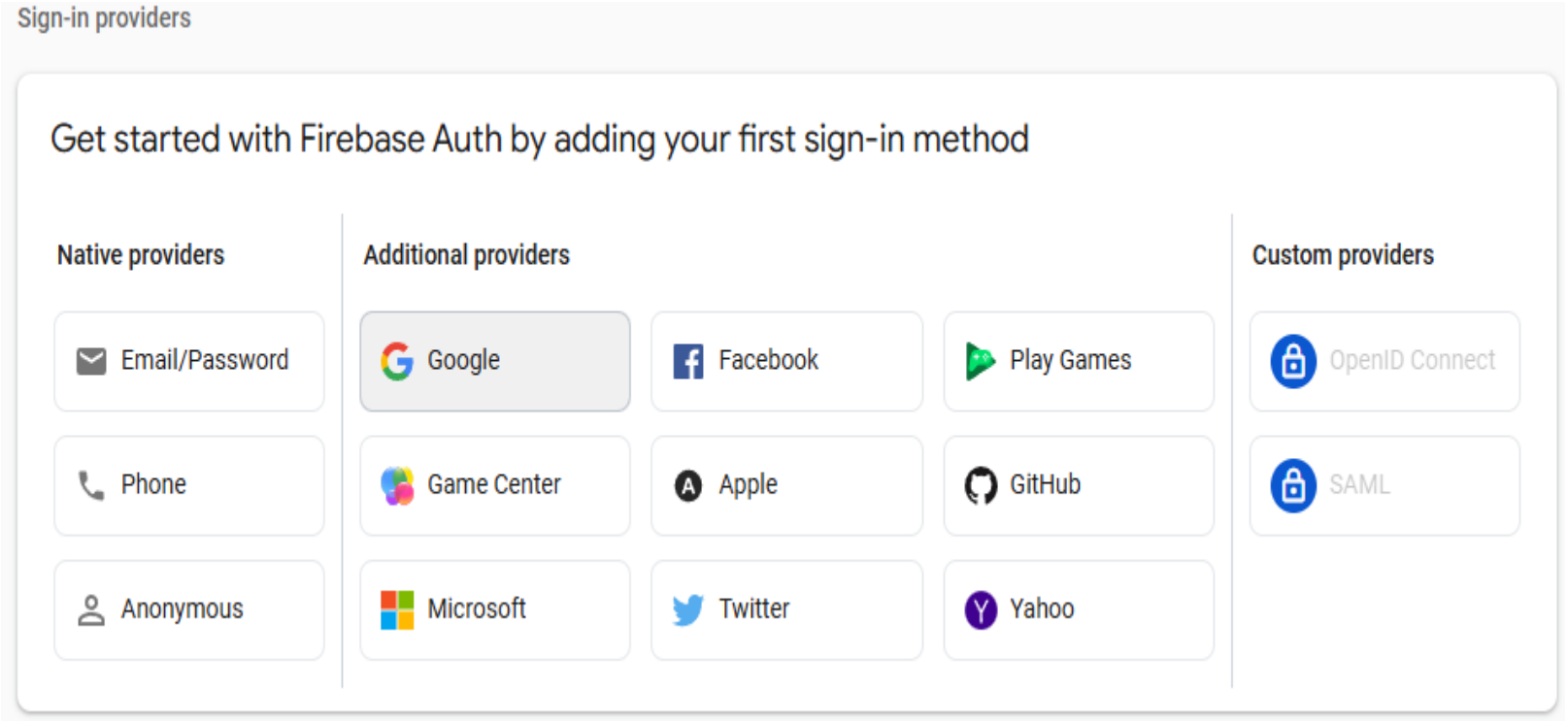
```
1  // Import the functions you need from the SDKs you need
2  import { initializeApp } from "firebase/app";
3  // TODO: Add SDKs for Firebase products that you want to use
4  // https://firebase.google.com/docs/web/setup#available-libraries
5
6  // Your web app's Firebase configuration
7  const firebaseConfig = {
8    apiKey: "AIzaSyATqeFXugQm70B9deePWtCrouxjlJo9SPk",
9    authDomain: "my-firebase-app-230325.firebaseapp.com",
10   projectId: "my-firebase-app-230325",
11   storageBucket: "my-firebase-app-230325.firebasestorage.app",
12   messagingSenderId: "434225677666",
13   appId: "1:434225677666:web:c581702d83feb51348d83b"
14 };
15
16 // Initialize Firebase
17 const app = initializeApp(firebaseConfig);
```

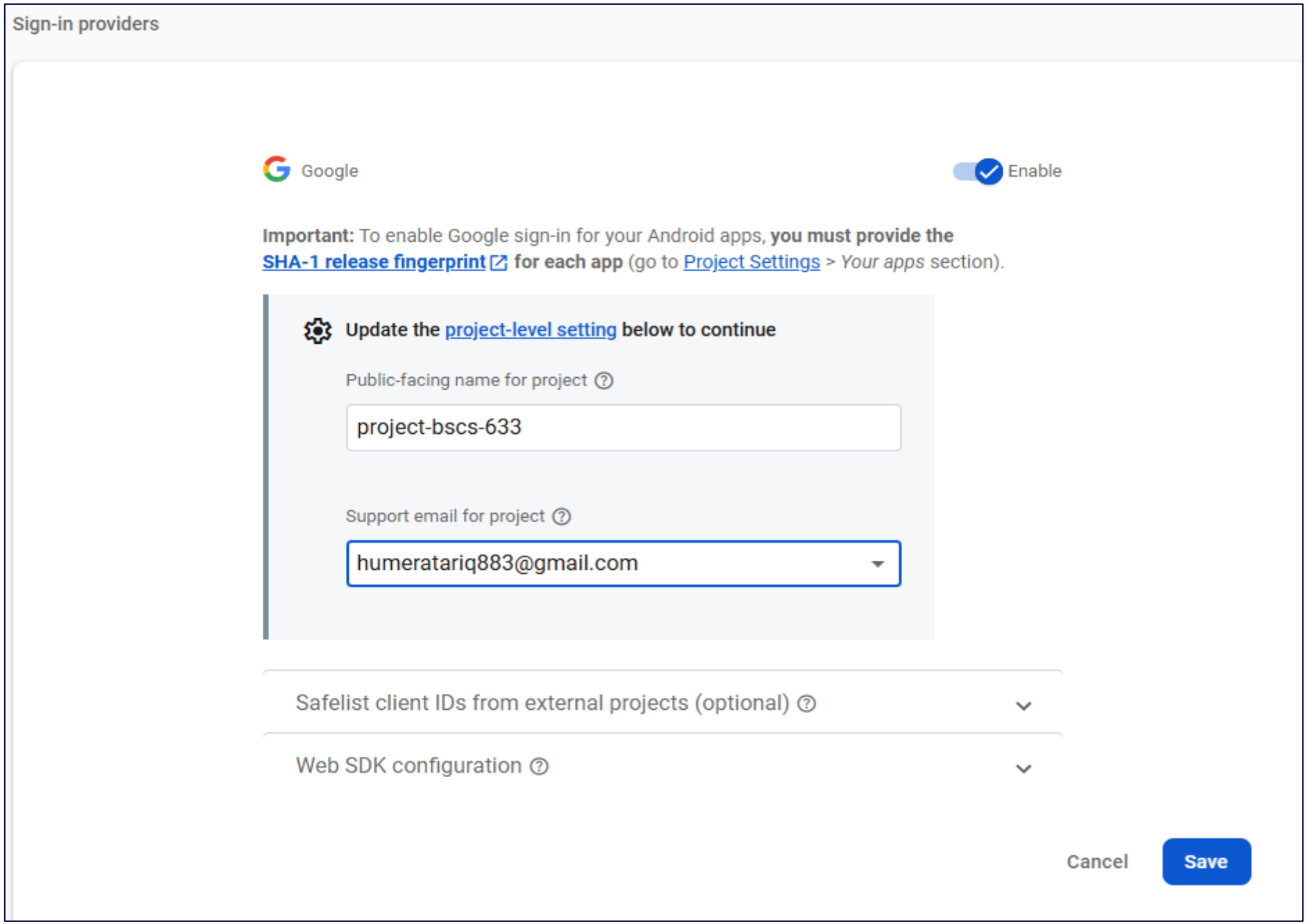
Continue to console

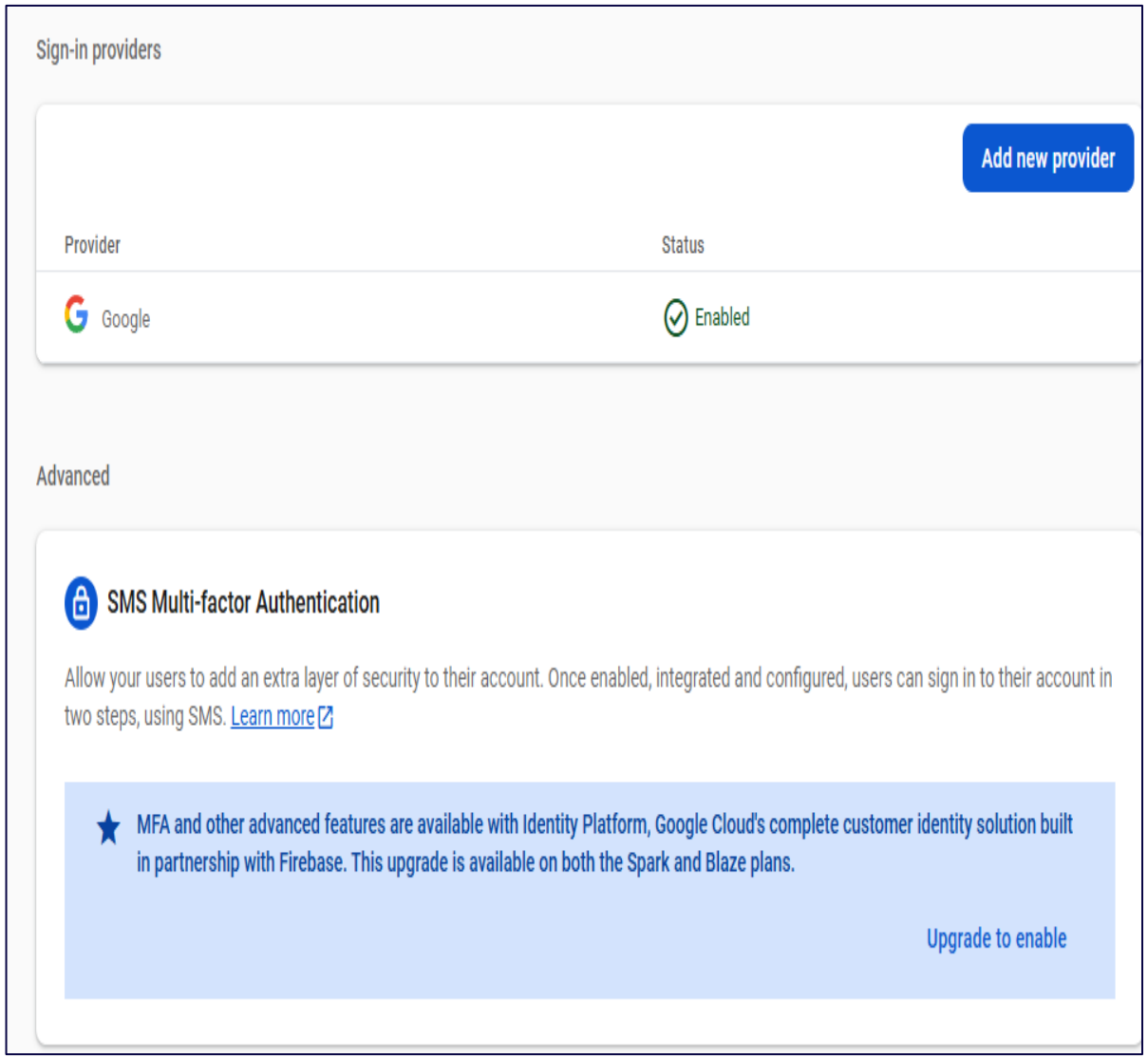
Back in the console, let's add two products: Authentication and Firestore.

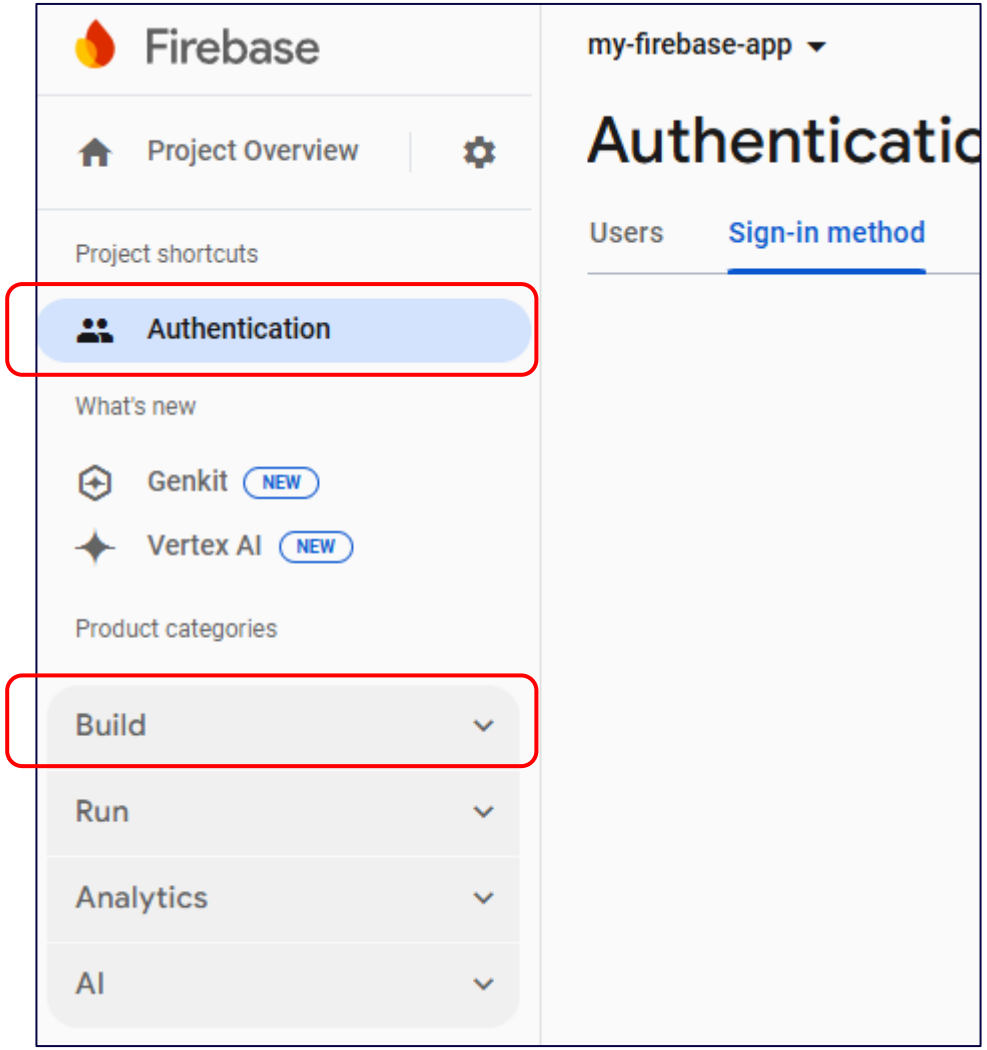
- When adding and setting up Authentication, enable Google only, adding yourself as the main user.
- When adding and setting up Cloud Firestor, create a database and put the initial security settings in full-on production mode, so no one can read or write. (we can fix this later but *always start with the most locked-down settings possible*).



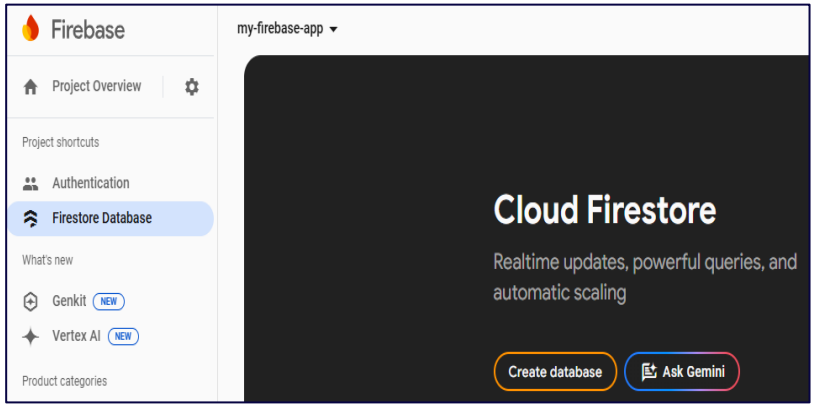


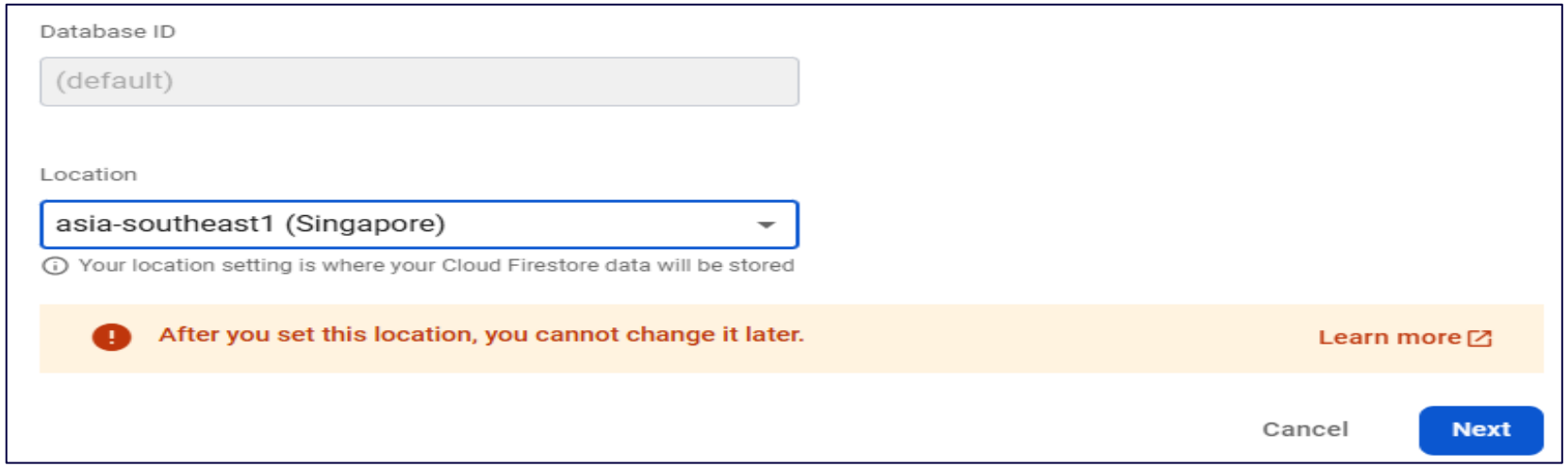

Sign-in providers
Google
Enable
Important: To enable Google sign-in for your Android apps, you must provide the SHA-1 release fingerprint for each app (go to Project Settings > Your apps section).
Update the project-level setting below to continue
Public-facing name for project
project-bscs-633
Support email for project
humeratariq883@gmail.com
Safelist client IDs from external projects (optional)
Web SDK configuration
Cancel
Save


Sign-in providers
Add new provider
Provider
Status
Google
Enabled
Advanced
SMS Multi-factor Authentication
Allow your users to add an extra layer of security to their account. Once enabled, integrated and configured, users can sign in to their account in two steps, using SMS. Learn more
MFA and other advanced features are available with Identity Platform, Google Cloud's complete customer identity solution built in partnership with Firebase. This upgrade is available on both the Spark and Blaze plans.
Upgrade to enable

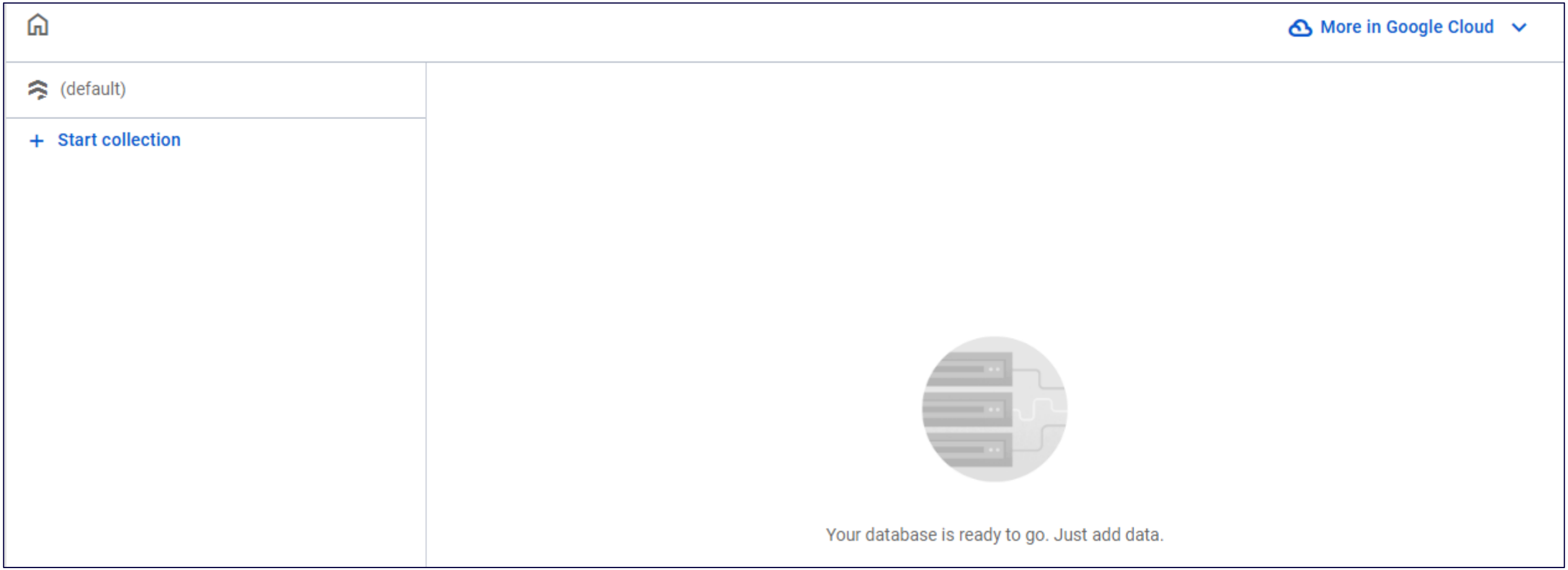

Firebase
my-firebase-app
Project Overview
Authenticatic
Users
Sign-in method
Project shortcuts
Authentication
What's new
Genkit
NEW
Vertex AI
NEW
Product categories
Build
Run
Analytics
AI

Reminder: When adding and setting up Cloud Firestor, create a database and put the initial security settings in full-on production mode, so no one can read or write. (Yes, we'll fix this later but *always start with the most locked-down settings possible*).





Your DB is ready to Go



Now back in the code, let's make sure our app can see Firebase. Expand *src/firebaseConfig.js*, as follows:

JS fireBaseConfig.js × JS articleService.js

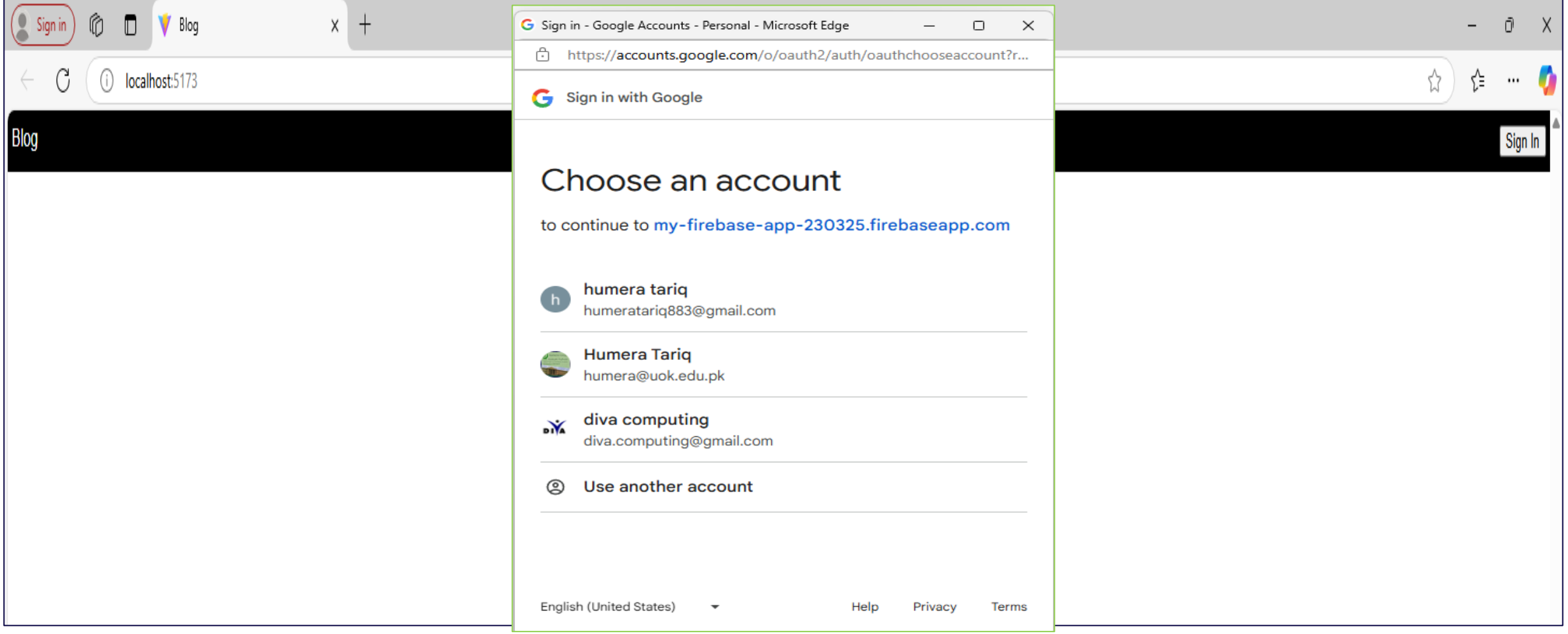
src > JS fireBaseConfig.js > db

```
1   // Import the functions you need from the SDKs you need
2   import { initializeApp } from "firebase/app";
3   // TODO: Add SDKs for Firebase products that you want to use
4   // https://firebase.google.com/docs/web/setup#available-libraries
5
6   import { getAuth } from "firebase/auth"
7   import { getFirestore } from "firebase/firestore"
8
9
10  // Your web app's Firebase configuration
11  const firebaseConfig = {
12    apiKey: "AIzaSyATqeFXugQm70B9deePWtCrouxjlJo9SPk",
13    authDomain: "my-firebase-app-230325.firebaseapp.com",
14    projectId: "my-firebase-app-230325",
15    storageBucket: "my-firebase-app-230325.firebasestorage.app",
16    messagingSenderId: "434225677666",
17    appId: "1:434225677666:web:c581702d83feb51348d83b"
18  };
19
20  // Initialize Firebase
21  const app = initializeApp(firebaseConfig);
22  export const auth = getAuth(app)
23  export const db = getFirestore(app)
```

Our goal is a layered, modular architecture.

Step 4: Authentication

we are not going to take any shortcuts, but rather do things in an organized, disciplined manner, with, um, lots of little files. It will look like more work than necessary, but this code-along strives to teach proper *separation of concerns*.


Sign in
Blog
localhost:5173
Blog
Sign In
Sign in - Google Accounts - Personal - Microsoft Edge
https://accounts.google.com/o/oauth2/auth/oauthchooseaccount?r...
Sign in with Google
Choose an account
to continue to my-firebase-app-230325.firebaseapp.com
humera tariq
humeratariq883@gmail.com
Humera Tariq
humera@uok.edu.pk
diva computing
diva.computing@gmail.com
Use another account
English (United States)
Help
Privacy
Terms


localhost:5173
Blog
New Article
Hello, humera tariq
Sign Out
There's a fair tomorrow
Hello Everyone
No article selected

The big idea!

Components never talk to Firebase directly.

Only the services do.

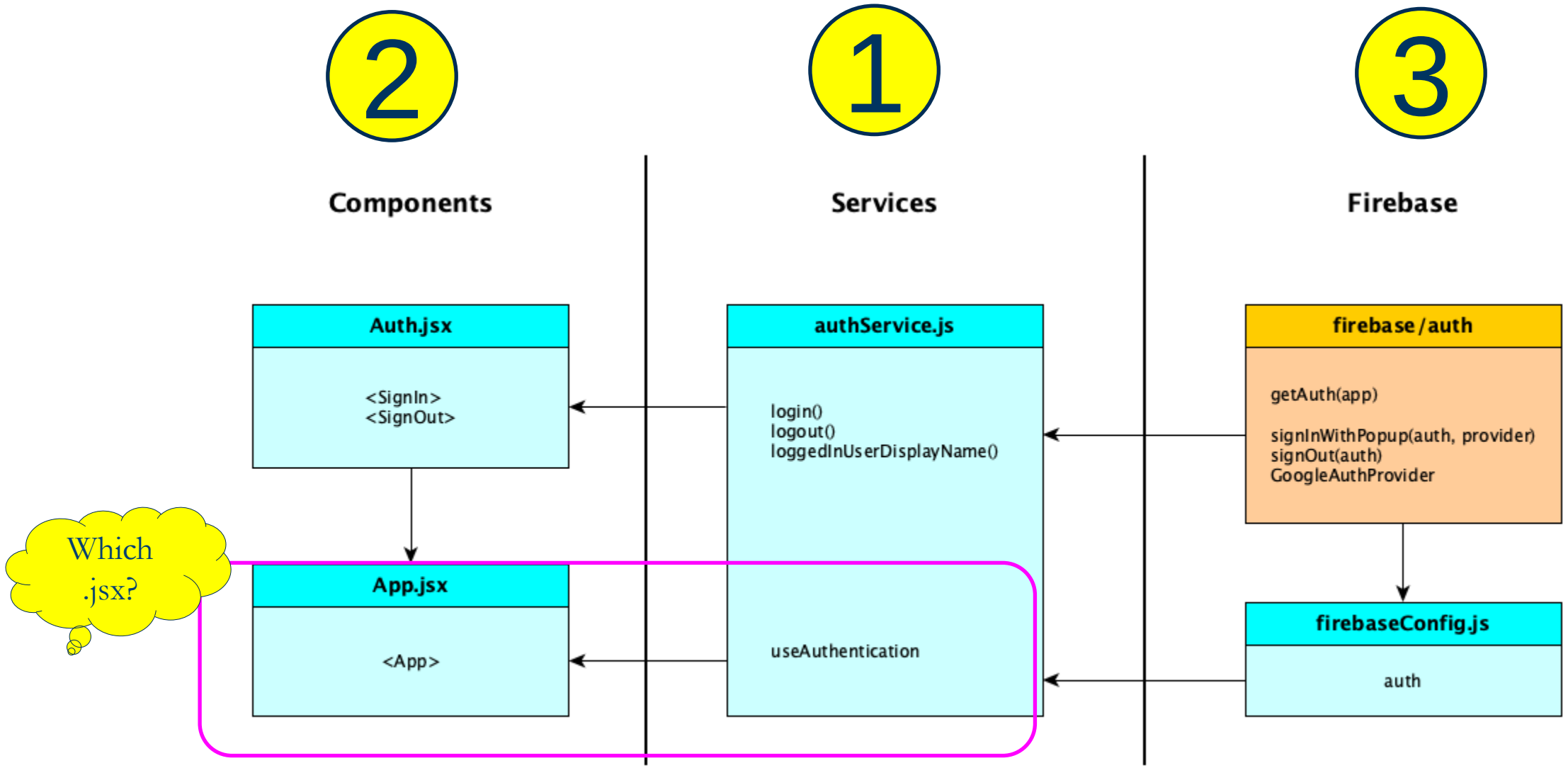
You will notice that the services hide the complexities of Firebase from the components.

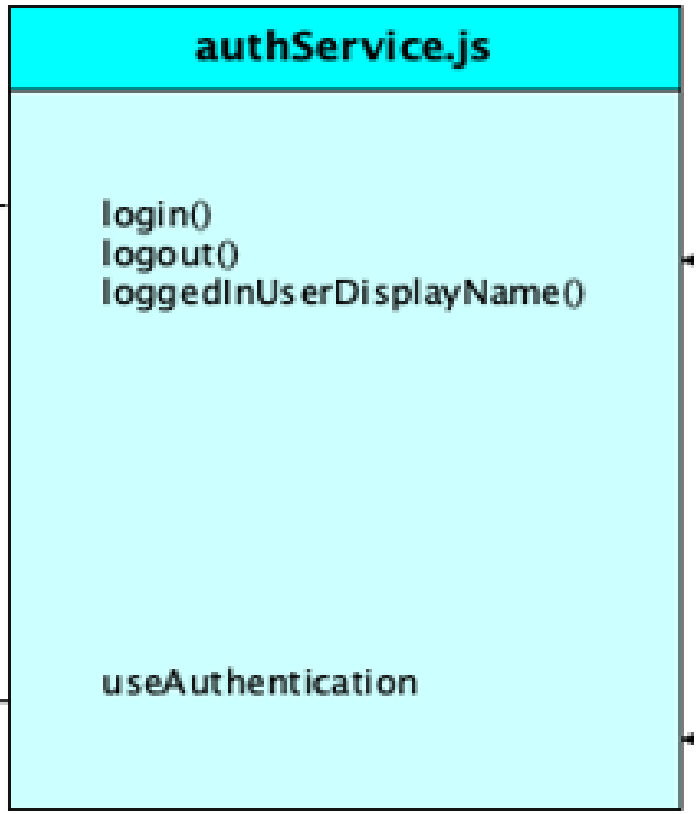
Why Services are awesome ?

Components will only call login and logout and don't even need or care that an auth object is needed, nor that authentication providers are needed. Heck, they don't even know that Firebase is there!

In principle, we can swap out Firebase with a different backend without changing our components.

Layered, modular architecture





- ✓Create a new file called *src/services/authService.js*.
- ✓ This file will provide very simple services to the components, called login and logout, hiding the complexities of Firebase auth.
- ✓ It will also contain a custom hook to notify components when the authentication state changes.

src/services/authService.js

```
import { useState, useEffect } from "react"
import { signInWithPopup, GoogleAuthProvider, signOut } from "firebase/auth"
import { auth } from "../firebaseConfig"
```

signInWithPopup(auth, provider) → Opens a pop-up for Google sign-in

GoogleAuthProvider → Represents Google as an authentication provider.

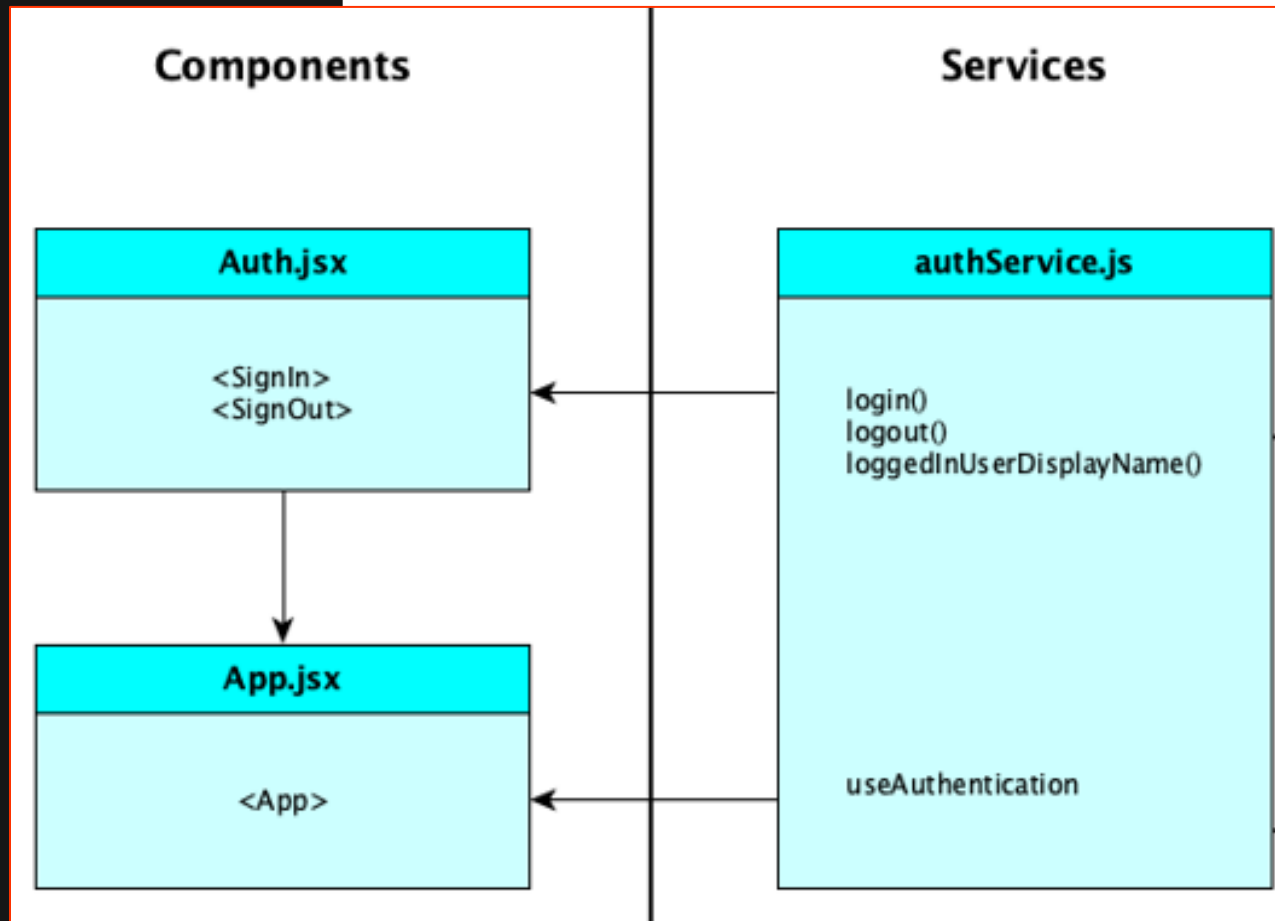
signOut(auth) → Logs out the user.

auth → This is the authentication instance that we initialized in firebaseConfig.js.
It contains Firebase authentication settings for our app.

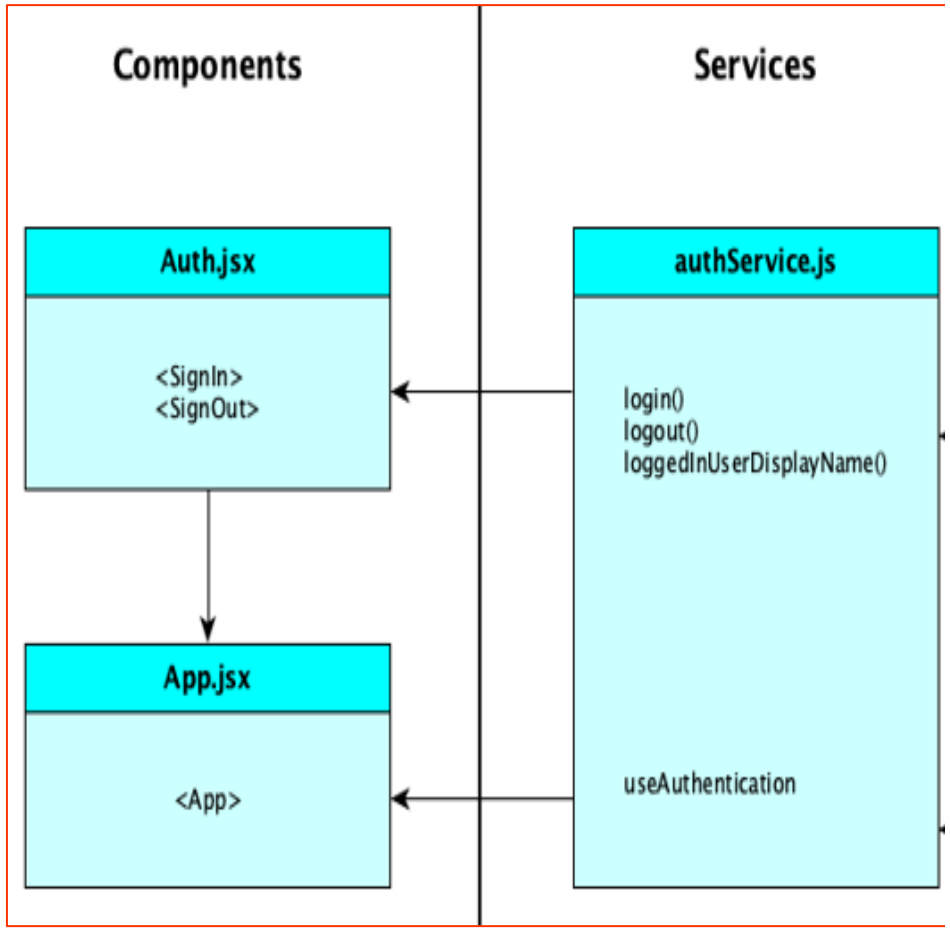
✅ firebaseConfig.js exists in the correct directory.
✅ The import path matches the actual file structure.

```
import { useState, useEffect } from "react"
import { signInWithPopup, GoogleAuthProvider, signOut } from "firebase/auth"
import { auth } from "../firebaseConfig"

export function login() {
  return signInWithPopup(auth, new GoogleAuthProvider())
}

export function logout() {
  return signOut(auth)
}

export function loggedInUserDisplayName() {
  return auth.currentUser.displayName
}

export function useAuthentication() {
  const [user, setUser] = useState(null)
  useEffect(() => {
    return auth.onAuthStateChanged((user) => {
      user ? setUser(user) : setUser(null)
    })
  }, [])
  return user
}
```

src/services/authService.js



Build components for managing login and logout. We'll put both components in the same file, just because we can call the file *src/components/Auth.jsx*:




```jsx
import { login, logout, loggedInUserDisplayName } from "../services/authService"

export function SignIn() {
  return <button onClick={login}>Sign In</button>
}

export function SignOut() {
  return (
    <div>
      Hello, {loggedInUserDisplayName()}
      <button onClick={logout}>Sign Out</button>
    </div>
  )
}
```

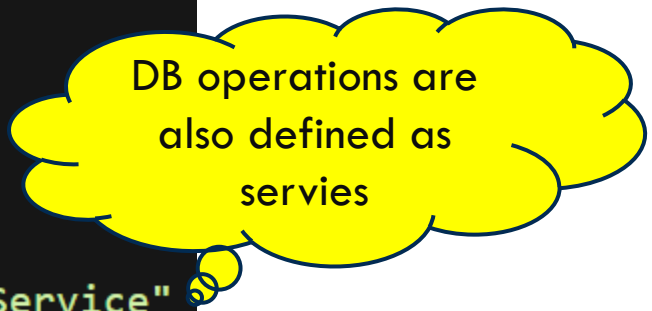
src/components/App.jsx

Step 5:

Now let's get this working in our app. In *src/components/App.jsx* we want that if someone is logged in, you show the app as in the starter code *(with the logged in user's name and New Article button)* but now, in addition, the Sign Out component on the header bar. If no one is logged in, we show the Sign In button on the header bar.

src/components/App.jsx

```jsx
import { useEffect, useState } from "react"
import Nav from "./Nav"
import Article from "./Article"
import ArticleEntry from "./ArticleEntry"
import { SignIn, SignOut } from "./Auth"
import { useAuthentication } from "../services/authService"
import { fetchArticles, createArticle } from "../services/articleService"
import "./App.css"
```

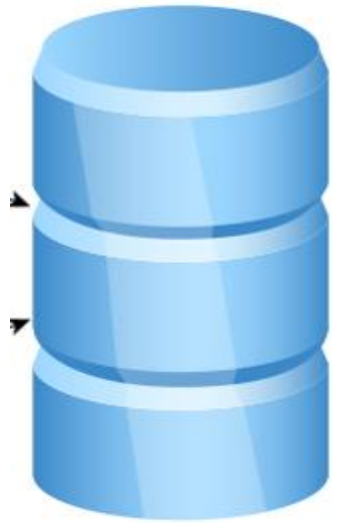


src/components/App.jsx

```jsx
export default function App() {
  const [articles, setArticles] = useState([])
  const [article, setArticle] = useState(null)
  const [writing, setWriting] = useState(false)
  const user = useAuthentication()

  // This is a trivial app, so just fetch all the articles only when
  // a user logs in. A real app would do pagination. Note that
  // "fetchArticles" is what gets the articles from the service and
  // then "setArticles" writes them into the React state.
  useEffect(() => {
    if (user) {
      fetchArticles().then(setArticles)
    }
  }, [user])
```

src/components/App.jsx

```jsx
// Update the "database" *then* update the internal React state. These
// two steps are definitely necessary.
function addArticle({ title, body }) {
  createArticle({ title, body }).then((article) => {
    setArticle(article)
    setArticles([article, ...articles])
    setWriting(false)
  })
}
```



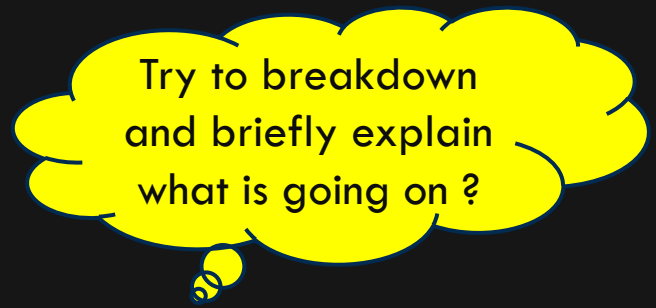
createArticle() sends the article data to the database (e.g., via an API).

setArticle(article) → Updates **which article** is currently displayed.

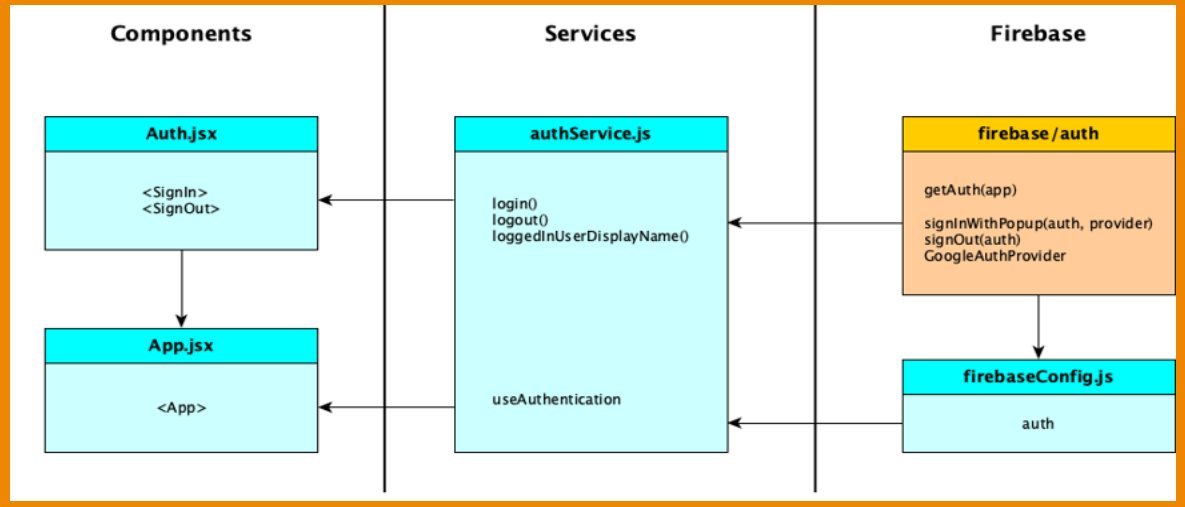
setArticles([article, ...articles]) → Updates the **React state array** with the new article.

setWriting(false) → Switches back to reading mode after submission.

```
  return (
    <div className="App">
      <header>
        Blog
        {user && <button onClick={() => setWriting(true)}>New Article</button>}
        {!user ? <SignIn /> : <SignOut />}
      </header>

      {!user ? "" : <Nav articles={articles} setArticle={setArticle} />}

      {!user ? (
        ""
      ) : writing ? (
        <ArticleEntry addArticle={addArticle} />
      ) : (
        <Article article={article} />
      )}
    </div>
  )
}
```



Step 6: Firestore



Suppose we're building a social media dashboard where users can view their friends' posts and interact with them.

(Container and presentation pattern)
(A clean and optimized fetch discussion)

Right now, your Friend Feed app is **static** in terms of user interaction because it only **fetches and displays posts** but does not allow any **user-driven actions** like:

- ✓ Adding new posts
- ✓ Liking or commenting on posts
- ✓ Deleting or editing posts
- ✓ Refreshing data manually

```
/src
  ├── /components
  │     ├── FriendFeed.jsx
  │     ├── FriendFeedContainer.jsx
  ├── /utils
  │     ├── fetchData.js ✅ (New Utility Function)
  ├── App.jsx
  ├── index.jsx
```

Welcome to Friend Feed

Friend Feed

sunt aut facere repellat provident occaecati excepturi optio reprehenderit
Posted by: User 1

qui est esse
Posted by: User 1

ea molestias quasi exercitationem repellat qui ipsa sit aut
Posted by: User 1

eum et est occaecati
Posted by: User 1

nesciunt quas odio
Posted by: User 1

App.jsx > ...

```jsx
import React from "react";
import "./App.css";
import FriendFeedContainer from "./components/FriendFeedContainer";

const App = () => {
  return (
    <main>
      <h1>Welcome to Friend Feed</h1>
      <FriendFeedContainer />
    </main>
  );
};

export default App;
```

Why <main> Instead of ?

Welcome to Friend Feed

Friend Feed

sunt aut facere repellat provident occaecati excepturi optio reprehenderit
Posted by: User 1

qui est esse
Posted by: User 1

ea molestias quasi exercitationem repellat qui ipsa sit aut
Posted by: User 1

eum et est occaecati
Posted by: User 1

nesciunt quas odio
Posted by: User 1

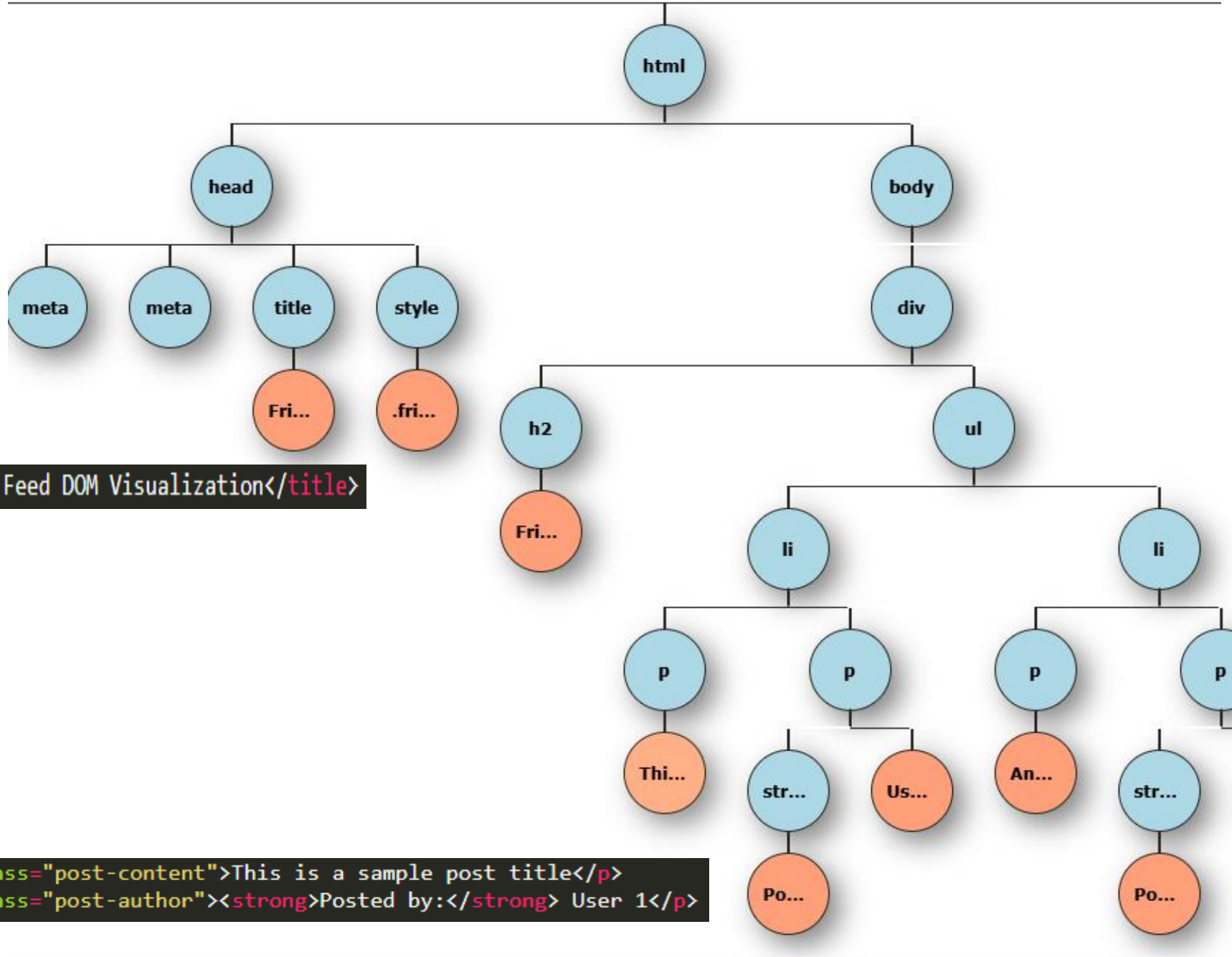
Think of index.css
And
FriendFeed.css

```css
.friend-post {
  background: #f9f9f9;
  margin: 10px 0;
  padding: 15px;
  border-left: 5px solid #007bff;
  border-radius: 5px;
  transition: transform 0.2s ease-in-out;
}
```

Child/presenter FriendFeed.jsx

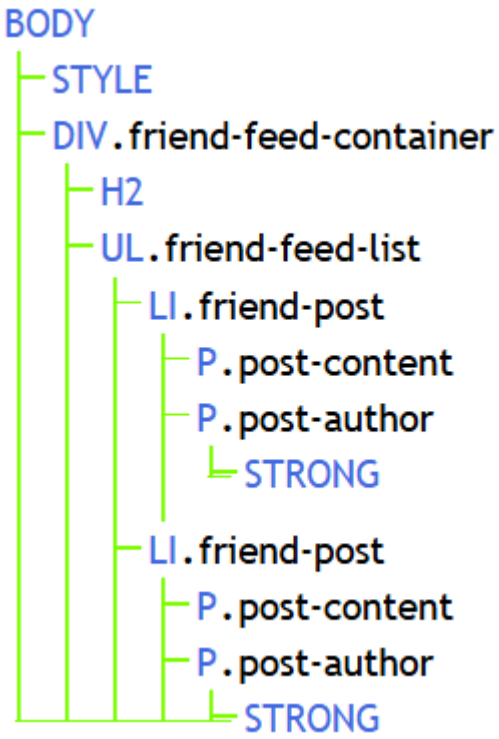
```jsx
<li key={post.id} className="friend-post">
  <p className="post-content">{post.content}</p>
  <p className="post-author"><strong>Posted by:</strong> {post.author}</p>
</li>
```

Interactive CSS Based DOM Visualization




```
<title>Friend Feed DOM Visualization</title>
```

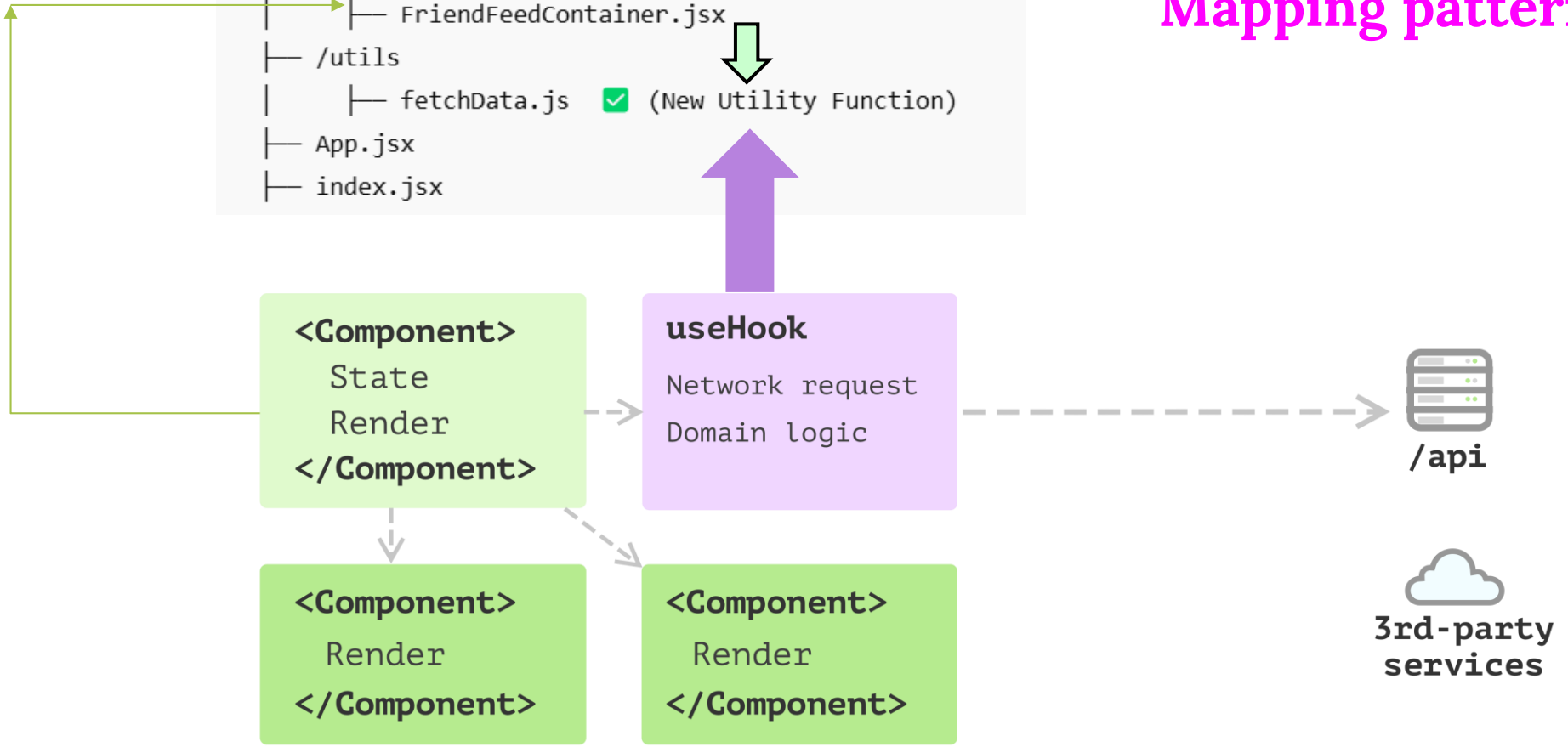
```
<p class="post-content">This is a sample post title</p>
<p class="post-author"><strong>Posted by:</strong> User 1</p>
```



State management with hooks

Mapping pattern to App()

```
/src
  ├── /components
  │     ├── FriendFeed.jsx
  │     ├── FriendFeedContainer.jsx
  ├── /utils
  │     ├── fetchData.js  ✅ (New Utility Function)
  ├── App.jsx
  ├── index.jsx
```



Container component Presentational component React hook

/ Container Component (Handles Data Fetching)

// Presenter Component (Only Displays UI)

```jsx
import React, { useState, useEffect } from "react";

// Container Component (Handles Data Fetching)
import { fetchFriendPosts } from "../utils/fetchData";

// Container pass data to Presenter component
import FriendFeed from "./FriendFeed";
```

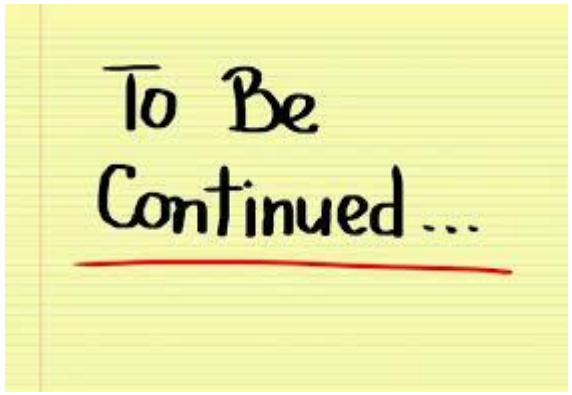
```jsx
// Container Component (Handles Data Fetching)
const FriendFeedContainer = () => {
  const [friendPosts, setFriendPosts] = useState([]);

  //Data Fetching ??

  return <FriendFeed posts={friendPosts} />;
};
```

```jsx
import React from "react";
```

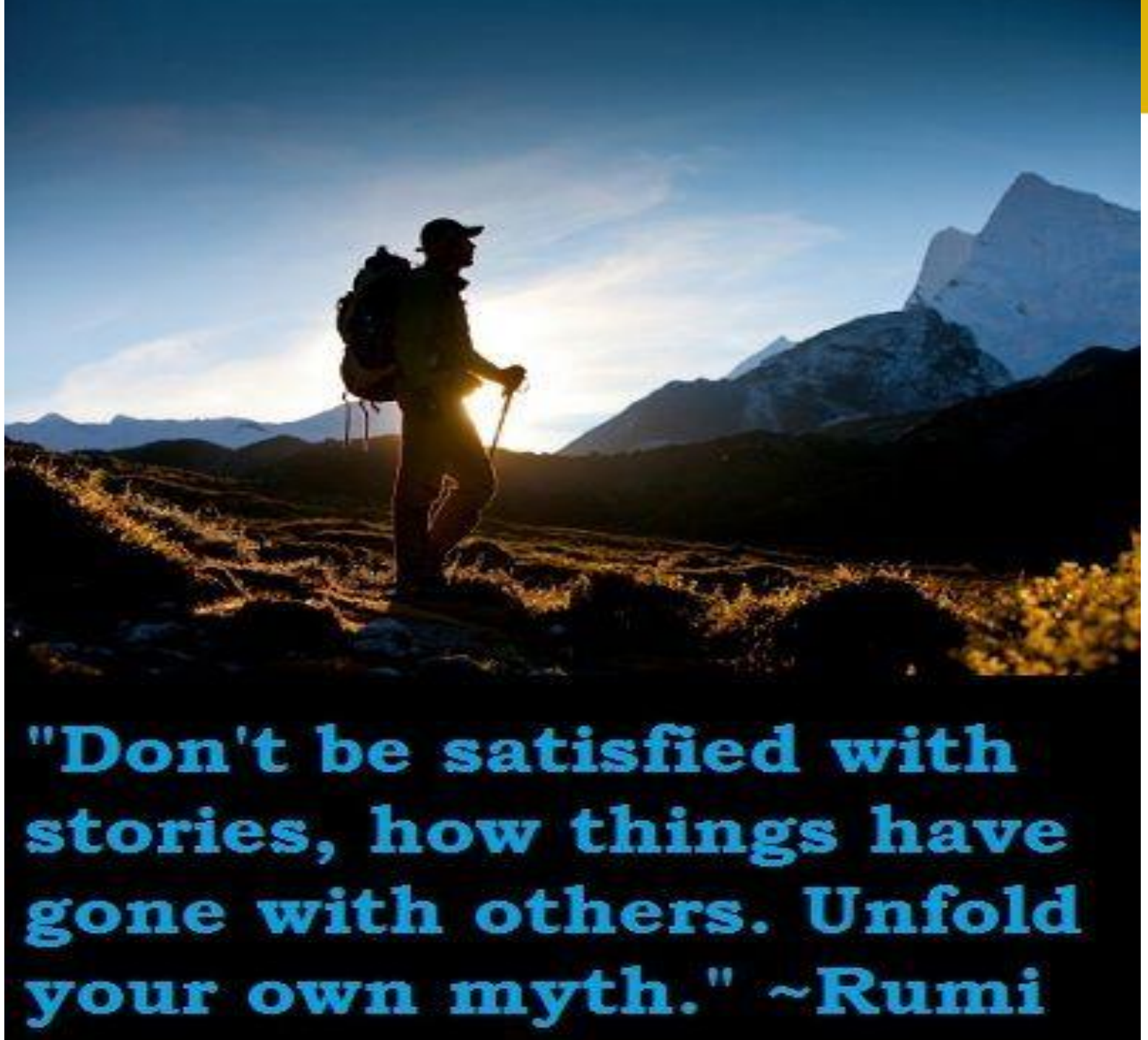
```jsx
// Presenter Component (Only Displays UI)
const FriendFeed = ({ posts }) => {
  return (
    <ul>

      Rendering a list of posts dynamically using .map().

      Each post is displayed inside an <li> with its content and author

    </ul>
  );
};
```

Async () => { Await }

```
useEffect(() => {
  fetchFriendPosts().then(  ??  );
}, []);
```



UNIVERSITY OF
KARACHI



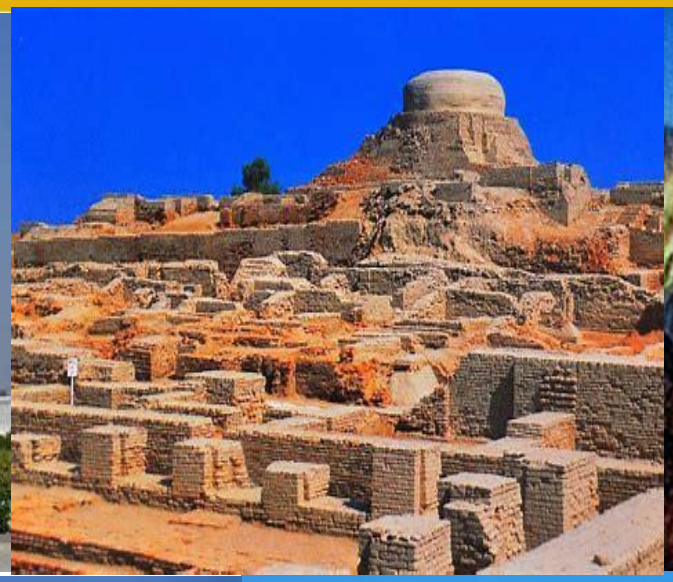
UNIVERSITY OF KARACHI

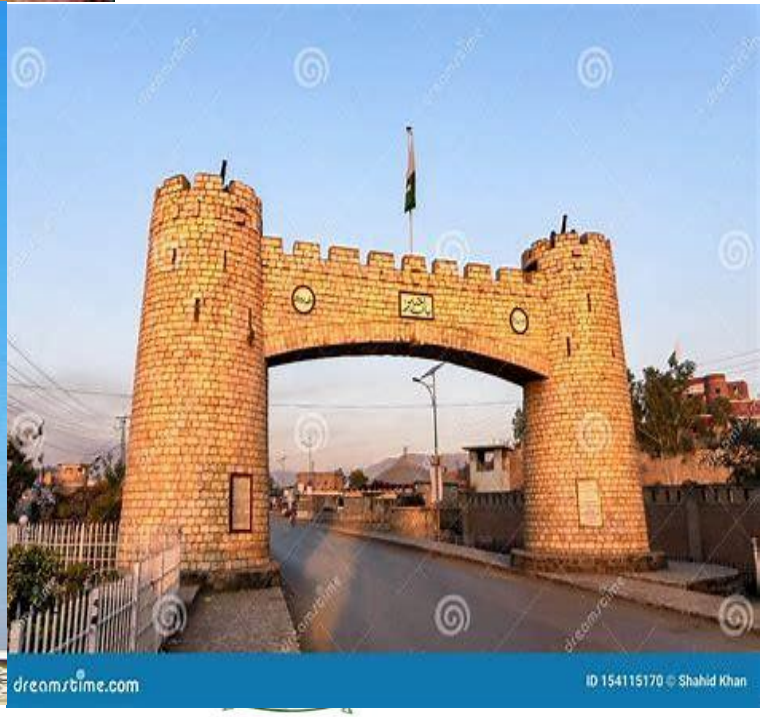
Department of Compute Science (UBIT Building), Karachi, Pakistan.

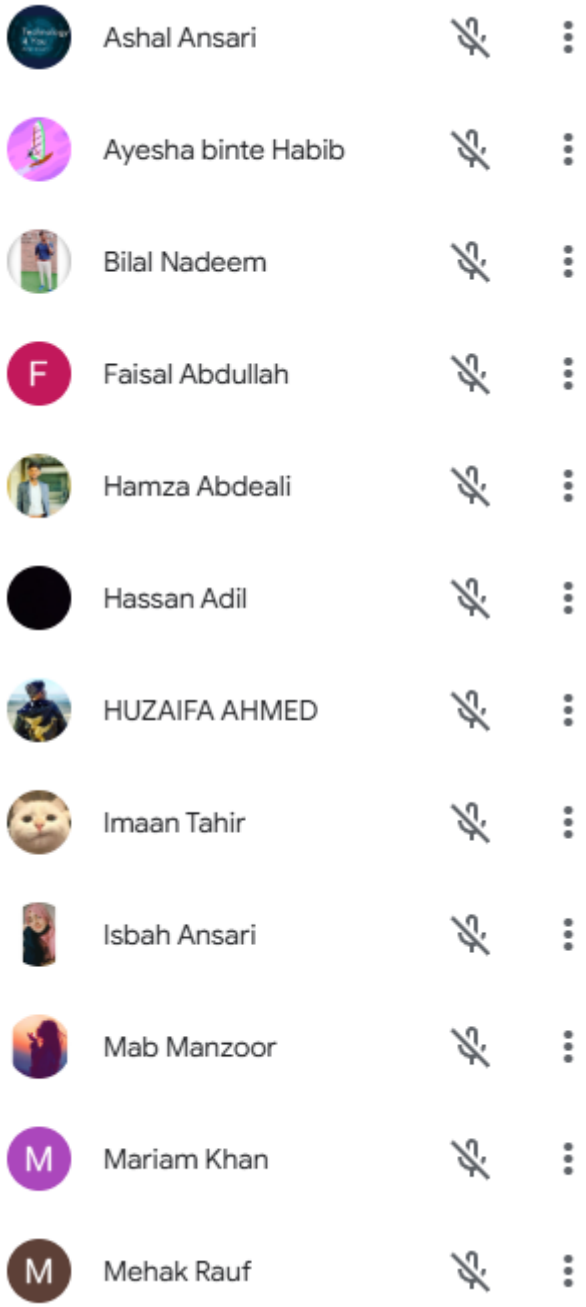
1200 Acres (5.2 Km sq.)
53 Departments
19 Institutes
25000 Students

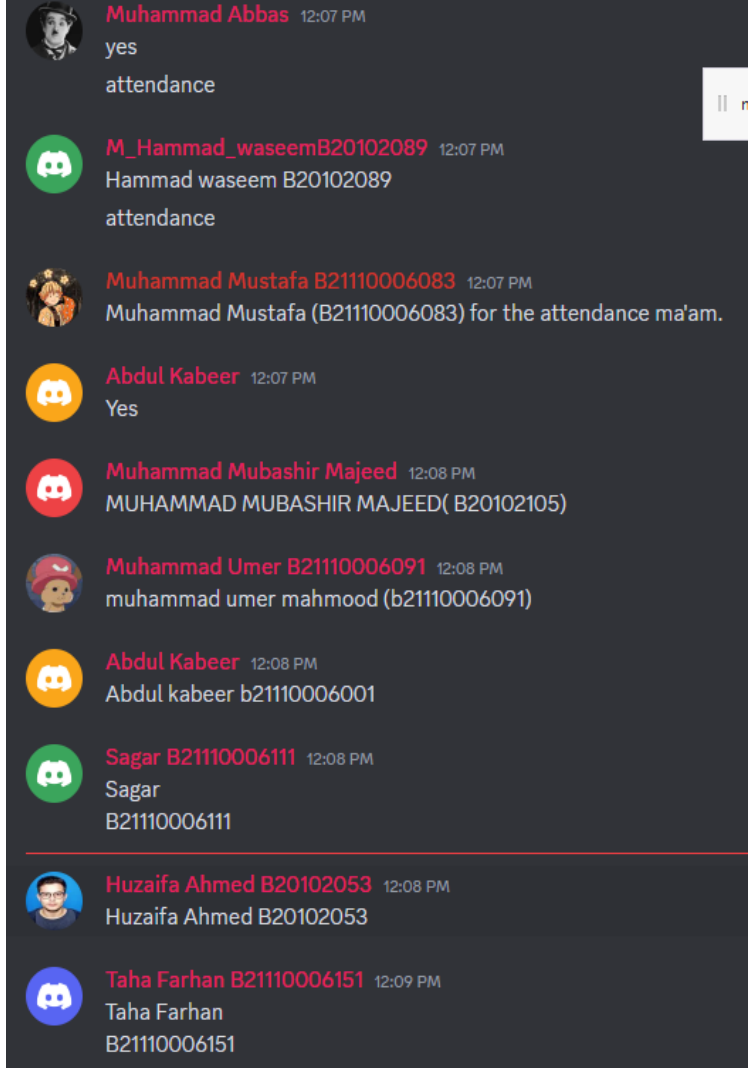
UNIVERSITY OF KARACHI

My Homeland Pakistan

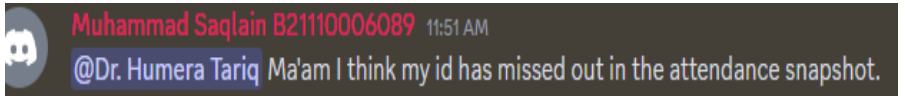





Ashal Ansari
Ayesha binte Habib
Bilal Nadeem
Faisal Abdullah
Hamza Abdeali
Hassan Adil
HUZAIFA AHMED
Imaan Tahir
Isbah Ansari
Mab Manzoor
Mariam Khan
Mehak Rauf
Qambar Ali
Rafia Asim
Rimsha Larib
Rimsha Masood
Sabeeh hassan farooqui
Saman Aslam
syed wahaj
Tarim Bilal
Wahab Tariq
Yumna Mubeen
Zehra Meethawala
Zoya Ali


Muhammad Abbas 12:07 PM
yes
attendance
M_Hammad_waseemB20102089 12:07 PM
Hammad waseem B20102089
attendance
Muhammad Mustafa B21110006083 12:07 PM
Muhammad Mustafa (B21110006083) for the attendance ma'am.
Abdul Kabeer 12:07 PM
Yes
Muhammad Mubashir Majeed 12:08 PM
MUHAMMAD MUBASHIR MAJEED(B20102105)
Muhammad Umer B21110006091 12:08 PM
muhammad umer mahmood (b21110006091)
Abdul Kabeer 12:08 PM
Abdul kabeer b21110006001
Sagar B21110006111 12:08 PM
Sagar
B21110006111
Huzaifa Ahmed B20102053 12:08 PM
Huzaifa Ahmed B20102053
Taha Farhan B21110006151 12:09 PM
Taha Farhan
B21110006151


Muhammad Yaseen B21110006097 12:09 PM
Muhammad Yaseen 6097


Muhammad Saqlain B21110006089 11:51 AM
@Dr. Humera Tariq Ma'am I think my id has missed out in the attendance snapshot.

Abdul Moiz

A Abeera Taj

Ahmed Azhar

Ahmed Ibrahim

Aiman Ali

Anas Hussain

Aun hassan

Bilal Lodhi

Bismah Nasir

F Faraz Ali

F Fatima Rizvi

Hammad Kamal

Haniya Hussain

H Huzaifa Shamsi

Ibad Hussain

Inshal Mansoor

K Khizra Faisal

M. Naqeebullah Shah

M Maha Meherrr

MARIA QURESHI

mohammad abdullah

Moin UD-din

M Muhammad Fakhir

Muhammad Jawwad Kh...

M Muhammad Yousuf

M Muhammd Hamza Siddi...

Muneeb Raza

You can't unmute some

Muntazir hassan

N Nighat Raza

Qaim Hassan

Rida

Saad Khan

Sadaqat Zia

saher hussain

S Sarah Sajid

Sheraz Asghar

S Shumail Arshad

S Syed Yaseen Arham

Sheraz Asghar

S Shumail Arshad

S Syed Yaseen Arham

S Syeda Laiba

S Syeda Tooba

S Syeda Wajiha Ali

T Tehreem Javed

Tehreem Khurram

umema batool

urooj shahab

Uzaifa Siddique

You can't unmute someon

Yahya Salman

marium zehra 10:03AM

Miss count me in for the attendance please I was late