Google Developer Groups

REACT FIREBASE

Google Developer Groups

AUGISTA EKSOFON PRADANA

SOFTWARE ENGINEERING,DEPARTMENT OF INFORMATICS ITS

Experience

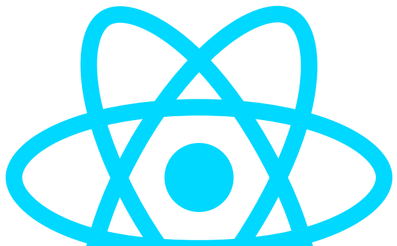
- Expert Staff IT Dev Mabacup 2024
- Expert Staff IT Dev Schematics ITS 2025
- Chief Technology Officer of PT. CoNation
- Frontend Engineer at JagoTeknik (Education Startup)
- Backend Engineer at IMAC 2024

Organization

- Head Bureau C of Internal Affairs HMTC ITS 2025
- Project & Journal Staff IEEE ITS Student Branch 2024
- Project Analyst 180DC Consulting ITS

Google Developer Groups

WHAT IS REACT?



React

- React is a JavaScript library for building user interfaces.
- Developed by Facebook (now Meta).
- Purpose: Create fast, scalable, and simple UI for web applications.
- React is Component-based architecture.

Google Developer Groups

WHY REACT MOST USED FOR WEB DEV?

- **Reusable Components**: Build once, use everywhere.
- **Virtual DOM**: Faster rendering by updating only changed elements.
- **Unidirectional Data Flow**: Makes data handling easier to manage.
- **Strong Ecosystem**: Rich community and tools.

Google Developer Groups

REACT COMPONENT

```jsx
import React from 'react';

function WelcomeMessage() {
  return <h1>Welcome to React!</h1>;
}

export default WelcomeMessage;
```

Components: The building blocks of a React application.

Google Developer Groups

JSX

What is JSX?: A syntax extension that allows mixing HTML and JavaScript.

Benefit JSX :
Readable and declarative UI code.

```
const element = <h1>Hello, World!</h1>;
```

```
ildren: [
con(icon, color: color
ontainer(
 margin: const EdgeIns
 child: Text(
   label,
   style: TextS
```

STATE MANAGEMENT

- State: Used to store data that can change over time.
- Function : State updates trigger a re-render of the component.

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

const Counter: React.FC = () => {
  const [count, setCount] = useState<number>(0);

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
};

export default Counter;
```

useState manages the count state.
Clicking the button increments the count.

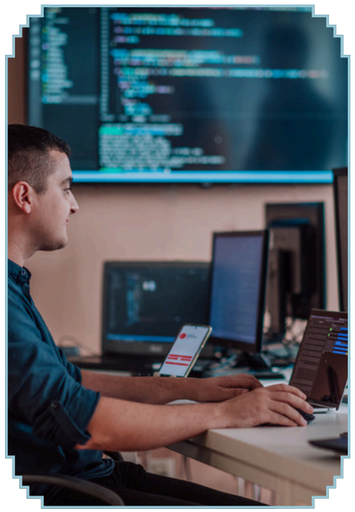
PROP & EVENT HANDLING

PROP

Props (Properties)
Props: Read-only data passed from parent to child components.

Passing data to child components using props:

```
function Greeting(props: { name: string }) {
  return <h1>Hello, {props.name}!</h1>;
}

export default function App() {
  return <Greeting name="Steve" />;
}
```



EVENT HANDLING

```
function ClickButton() {
  const handleClick = () => {
    alert('Button clicked!');
  };

  return <button onClick={handleClick}>Click Me</button>;
}

export default ClickButton;
```

onClick is used to attach the handleClick function to the button.

Google Developer Groups

Google Developer Groups

HOOK

- Hooks: Functions that allow functional components to manage state and lifecycle.
- Common hooks:
- useState: Manage state.
- useEffect: Handle side effects like data fetching.
- useContext: Manage global state.

Google Developer Groups

useEffect Hook

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

function Timer() {
  const [seconds, setSeconds] = useState(0);

  useEffect(() => {
    const interval = setInterval(() => {
      setSeconds((prev) => prev + 1);
    }, 1000);
    return () => clearInterval(interval);
  }, []);

  return <p>Timer: {seconds} seconds</p>;
}

export default Timer;
```

useEffect sets up an interval and cleans it up when the component unmounts

Google Developer Groups

useContext Hook

```jsx
import { createContext, useContext, useState } from "react";


const ThemeContext = createContext();

const ThemeProvider = ({ children }) => {
  const [theme, setTheme] = useState("light");

  return (
    <ThemeContext.Provider value={{ theme, setTheme }}>
      {children}
    </ThemeContext.Provider>
  );
};

export { ThemeProvider, ThemeContext };
```

- createContext() creates a global context.
- useContext(ThemeContext) is used in components to get the value of the context.
- Allows us to change themes without having to manually pass props to each component.

Google Developer Groups

useRef Hook

useRef is a React Hook used to:

- **Save a reference to a DOM element without causing a re-render.**
- **Store a value between re-renders without causing a state change.**

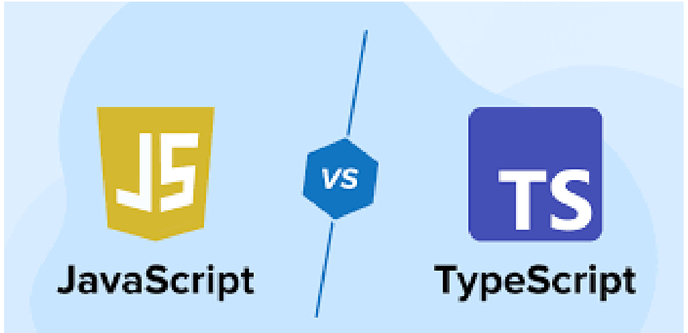
```
const scroll = useRef<HTMLDivElement | null>(null);
const chatContainerRef = useRef<HTMLDivElement | null>(null);
```

🔹 **scroll → Used to handle automatic scroll down when there is a new message.**
🔹 **chatContainerRef → Used to store the chat container reference so that it can be accessed directly.**

Google Developer Groups

WHAT DIFFERENT BETWEEN .JS VS .TS?

Perbedaan	JSX (JavaScript XML)	TSX (TypeScript XML)
Tipe Data	Tidak memiliki tipe data bawaan	Memiliki static typing
Error Handling	Error hanya muncul saat runtime	Error terdeteksi saat compile
Keamanan	Kurang aman karena tidak ada validasi tipe	Lebih aman karena memiliki type checking



Google Developer Groups

WHAT DIFFERENT BETWEEN .JS VS .TS?

```
const Greeting = ({ name }) => {
  return <h1>Hello, {name}!</h1>;
};
```

There's No type Checking for ***name***

```
interface Props {
  name: string;
}

const Greeting: React.FC<Props> = ({ name }) => {
  return <h1>Hello, {name}!</h1>;
};
```

Props prohibits ***name*** must ***string*** type

Google Developer Groups

WHAT IS TYPESCRIPT?

Google Developer Groups

- A superset of JavaScript with static typing.
- Developed by Microsoft to improve scalability and maintainability.
- Compiled to JavaScript to run in a browser.

TypeScript Advantages:

- ✅ Static Typing - Prevent errors from the start with strict data types.
- ✅ Interface & Generics - The code structure is neater and reusable.
- ✅ Support for OOP - Can use classes, inheritance, encapsulation.

Google Developer Groups

REACT + TYPESCRIPT

- React is still declarative & component-based, but more strict and safe.
- Uses interfaces to define props and state.
- Can use React Hooks with clear typing.

```tsx
import React from "react";

interface Props {
  name: string;
  age: number;
}

const UserCard: React.FC<Props> = ({ name, age }) => {
  return (
    <div className="p-4 border rounded-lg">
      <h2>{name}</h2>
      <p>Age: {age}</p>
    </div>
  );
};

export default UserCard;
```

Google Developer Groups

TYPESCRIPT: FUNCTION, INTERFACE

1.FUNCTIONS WITH PARAMETER AND RETURN TYPES

IN TYPESCRIPT, WE CAN SPECIFY DATA TYPES FOR PARAMETERS AND RETURN VALUES TO MAKE THE CODE SAFER.

```typescript
interface User {
  name: string;
  age: number;
}

function getUserInfo(user: User): string {
  return `${user.name} is ${user.age} years old.`;
}

const user: User = { name: "Augista", age: 20 };
console.log(getUserInfo(user)); // Outputnya: Augista is 20 years old.
```

Google Developer Groups

TYPESCRIPT: CLASS ACCESS IDENTIFIER & GENERIC

TYPESCRIPT SUPPORTS THE CONCEPT OF CLASS-BASED PROGRAMMING WITH ACCESS MODIFIERS SUCH AS PUBLIC, PRIVATE, AND PROTECTED.

```typescript
class Person {
  public name: string;
  private age: number;
  protected job: string;

  constructor(name: string, age: number, job: string) {
    this.name = name;
    this.age = age;
    this.job = job;
  }

  public introduce(): string {
    return `Hi, I'm ${this.name}, and I work as a ${this.job}.`;
  }
}

const person = new Person("Augista", 20, "Software Engineer");
console.log(person.introduce());
```

Google Developer Groups

TYPESCRIPT: CLASS ACCESS IDENTIFIER & GENERIC

```typescript
function identity<T>(value: T): T {
  return value;
}

console.log(identity<string>("Hello"));
console.log(identity<number>(123));
```

GENERICS ALLOW US TO CREATE REUSABLE CODE FOR VARIOUS DATA TYPES.
<T> IS A VARIABLE DATA TYPE PARAMETER

- IDENTITY<STRING>("HELLO") → CALLING A FUNCTION WITH A STRING

Google Developer Groups

NEXT.JS

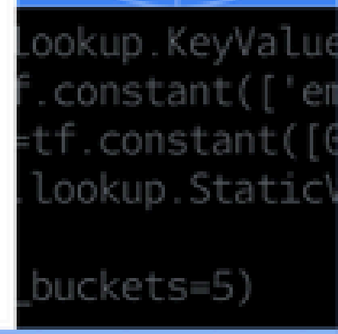
Google Developer Groups

WHAT IS NEXT JS?

Next JS is React framework for building server-side rendering (SSR) and static site generation (SSG) applications.

Supports automatic routing, API Routes, and performance optimization.

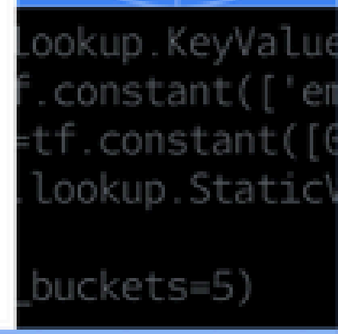
Built-in support for TypeScript!



Google Developer Groups

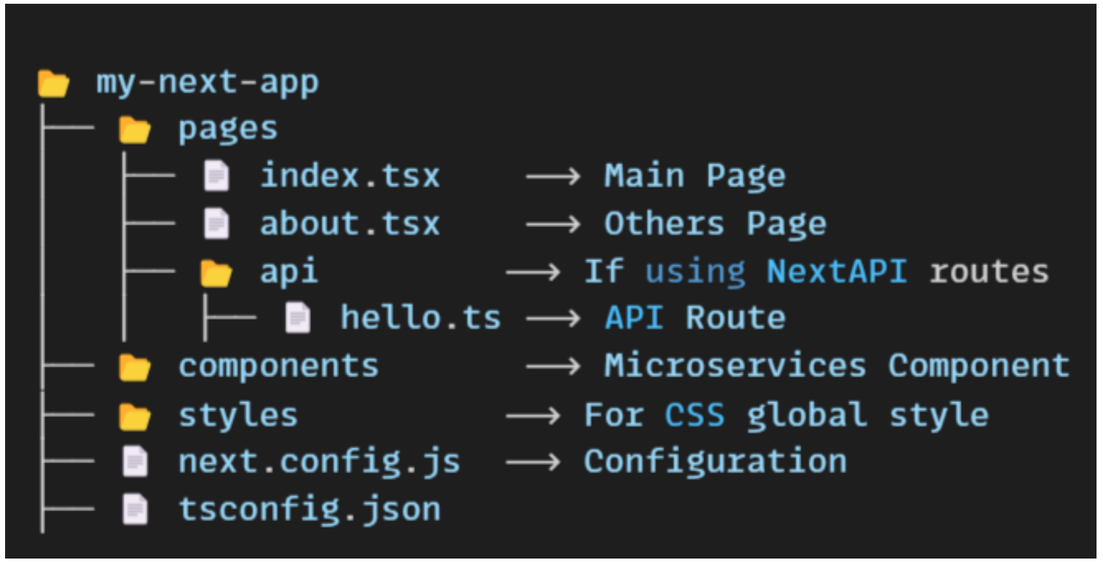
WHY WE MUST TRY NEXT JS?

- **Server-Side Rendering (SSR) → Render pages on the server before sending them to the client.**
- **Static Site Generation (SSG) → Generate HTML at build time for faster performance.**
- **API Routes → Create backend APIs directly within your Next.js project.**
- **Image Optimization → Use next/image to load more optimized images.**
- **Internationalization (i18n) → Built-in multi-language support.**



Google Developer Groups

FOLDER STRUCTURE IN NEXT JS



Google Developer Groups

AUTOMATIC ROUTING

- **Next.js uses a file-based routing system, meaning every file in the pages/ folder automatically becomes a route.**

- **No need for React Router → No additional configuration required.**
- **Supports Nested Routes → Can create folders in pages/ for nested routes.**

```
import { useRouter } from "next/router";
```

Google Developer Groups

PRE-RENDERING

Next.js has two types of pre-rendering to improve performance and SEO:

Rendering When Does Rendering Happen?

- **Static Site Generation (SSG) When build time is Fast, ideal for static content**
- **Server-Side Rendering (SSR) When requesting Data is always up-to-date**

```tsx
import { GetStaticProps } from "next";

export default function Home({ message }: { message: string }) {
  return <h1>{message}</h1>;
}

export const getStaticProps: GetStaticProps = async () => {
  return {
    props: { message: "Hello from SSG!" },
  };
};
```

```tsx
import { GetServerSideProps } from "next";

export default function Home({ time }: { time: string }) {
  return <h1>Current Time: {time}</h1>;
}

export const getServerSideProps: GetServerSideProps = async () => {
  return {
    props: { time: new Date().toISOString() },
  };
};
```

Google Developer Groups

IMAGE OPTIMIZATION

- **Automatic lazy loading**
- **Using modern formats (WebP, AVIF) for better performance**
- **Optimizing images based on screen size**

```
import Image from "next/image";
```

Google Developer Groups

MIDDLEWARE NEXTAUTH.JS (OTENTIKASI)

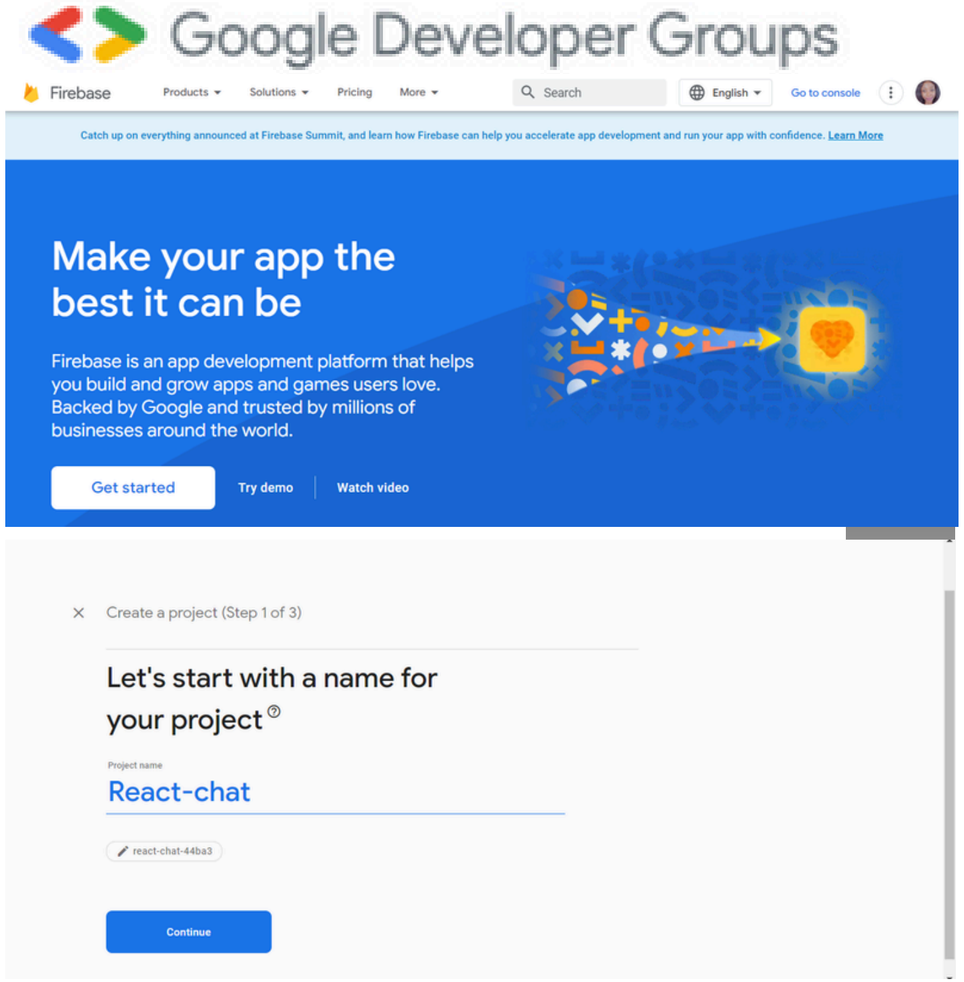
```
import NextAuth from "next-auth";
import GoogleProvider from "next-auth/providers/google";

export default NextAuth({
  providers: [
    GoogleProvider({
      clientId: process.env.GOOGLE_CLIENT_ID!,
      clientSecret: process.env.GOOGLE_CLIENT_SECRET!,
    }),
  ],
});
```

Firebase

Using React with Firebase config to connect realtime database and server

Google Developer Groups

FIREBASE SETUP

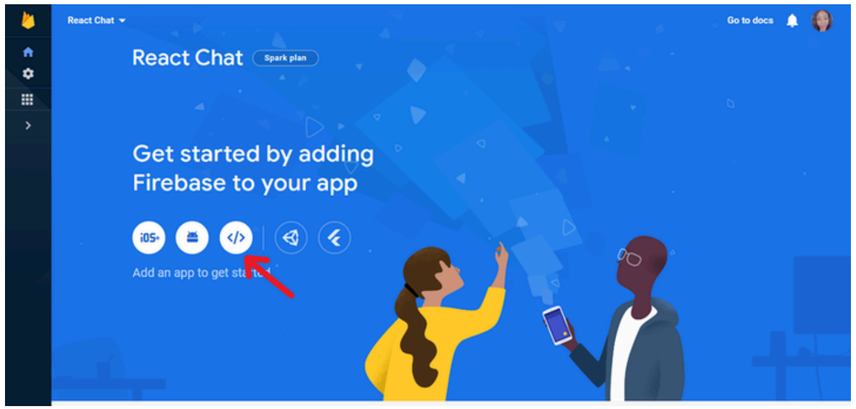


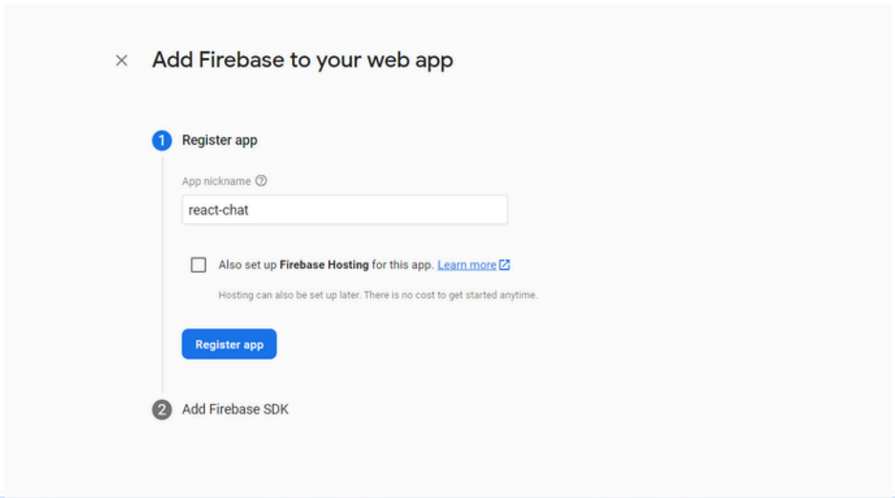
isi **Create a project** form by providing a project name.

created, click on Continue.

Google Developer Groups

FIREBASE SETUP

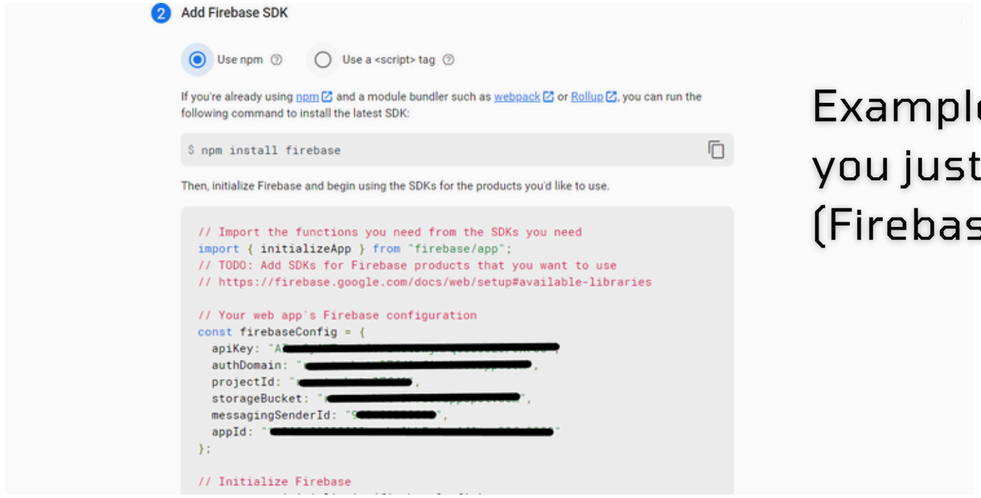




Fill in the title of the application you want to add to your Firestore

Google Developer Groups

FIREBASE SETUP



Example after creating you will be showed Firebase SDK, you just need copy and paste to your Firebase Config (Firebase.ts)

Google Developer Groups

FIREBASE CONFIG

```ts
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAuth } from "firebase/auth";
import { getFirestore } from "firebase/firestore";
import { getAnalytics } from "firebase/analytics";
// 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
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "API",
  authDomain: "API",
  projectId: "API",
  storageBucket: "react-chat-4c0a6.firebasestorage.app",
  messagingSenderId: "API",
  appId: "tesd",
  measurementId: "G-ECGGY3Y4NQ"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
export const auth = getAuth(app);
export const db = getFirestore(app);
```

Code Block: firebase.ts

Initializes Firebase using initializeApp.

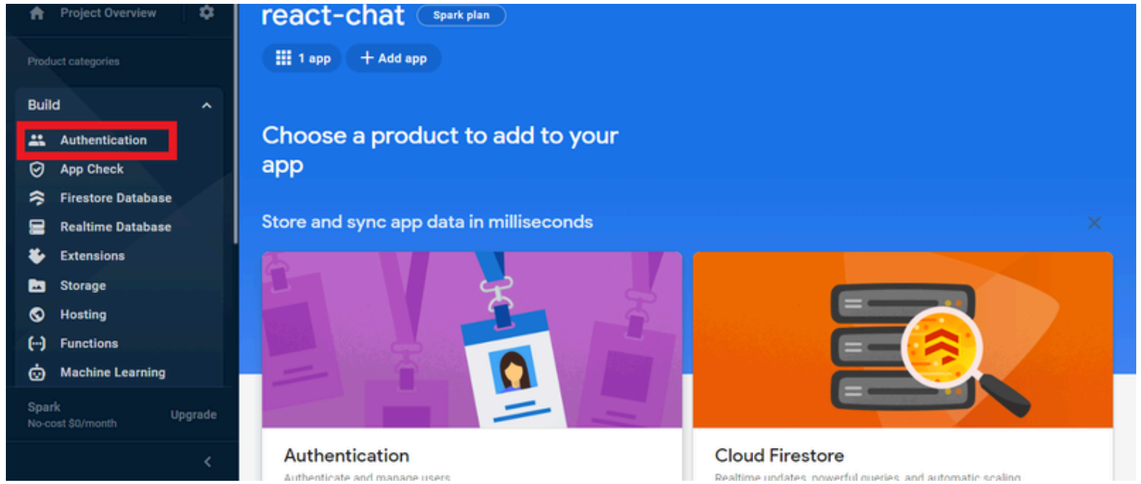
Connect the app to Firebase services for authentication and data storage.

Sets up **getAuth** for user authentication.

Uses **getFirestore** for real-time database.

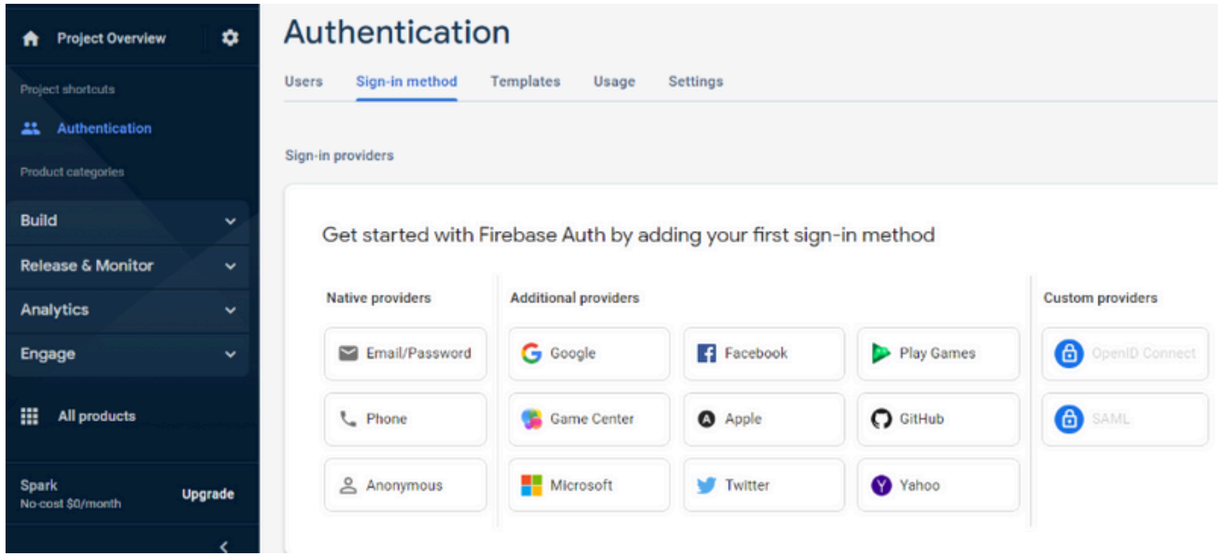
Google Developer Groups

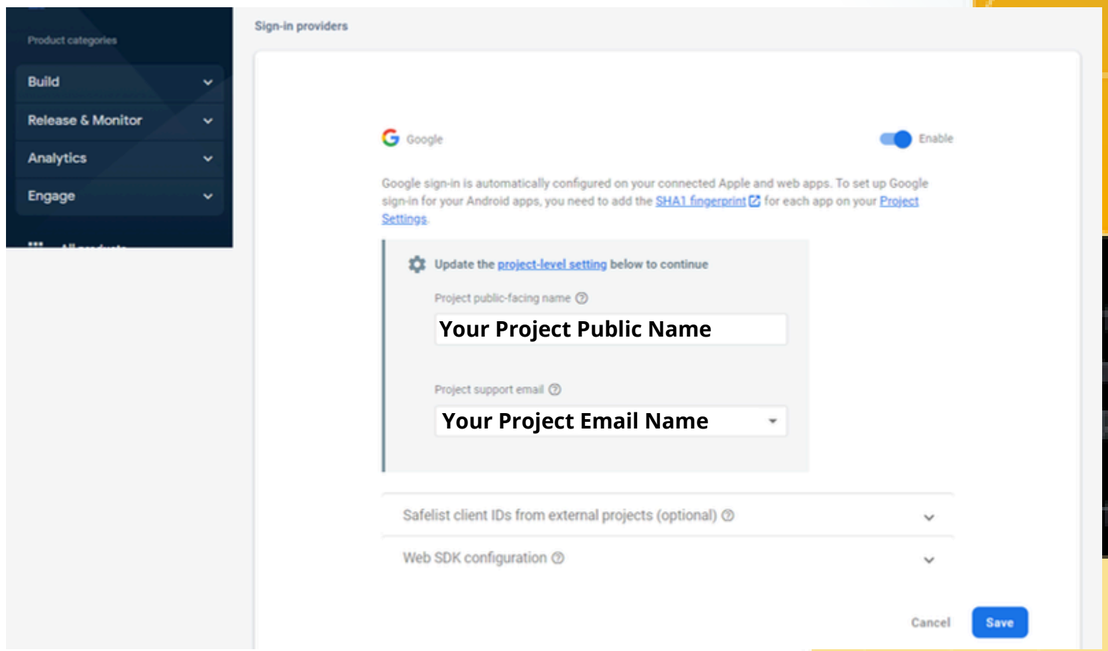
FIREBASE SETUP



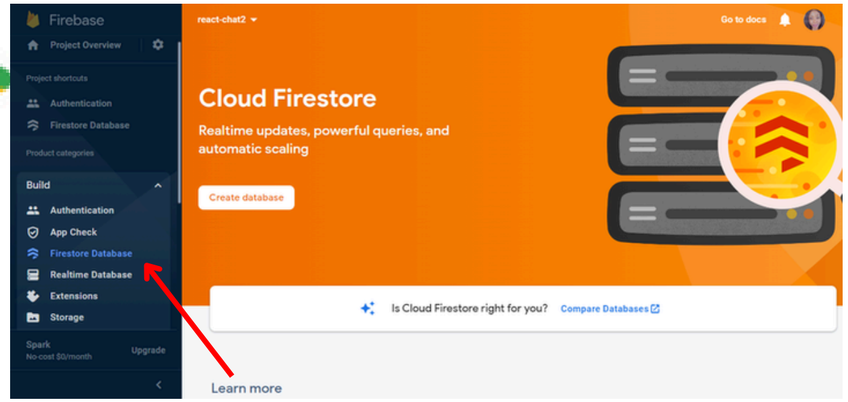
Google Developer Groups

FIREBASE SETUP

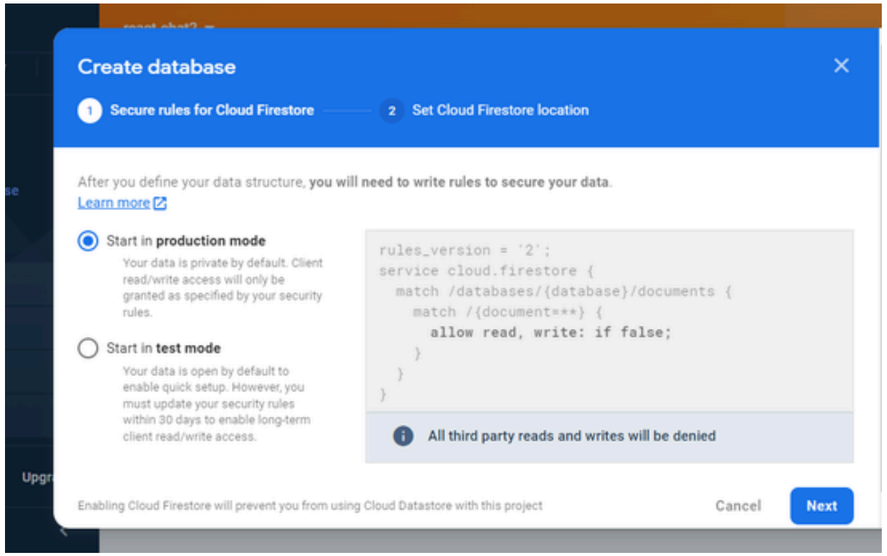




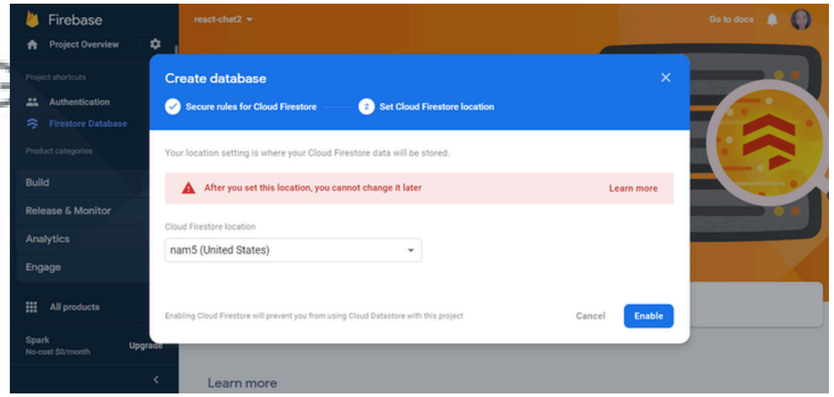
Choose your authentication sign in method
Enable **Google**, choose your Project support email, and click Save



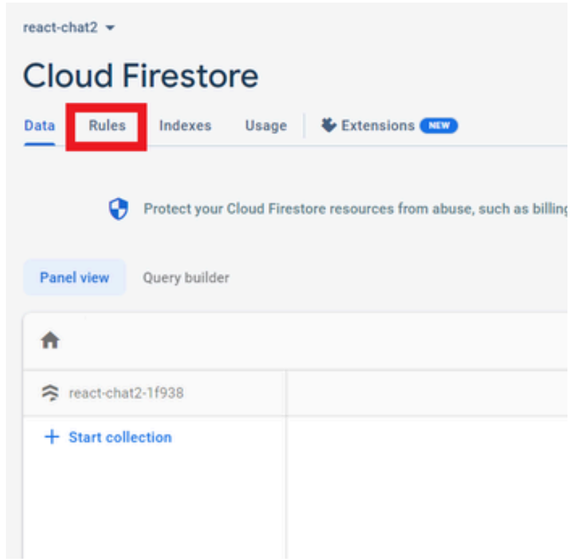
Select **Firestore** in Build Menu



Select **Production Mode** and Next.



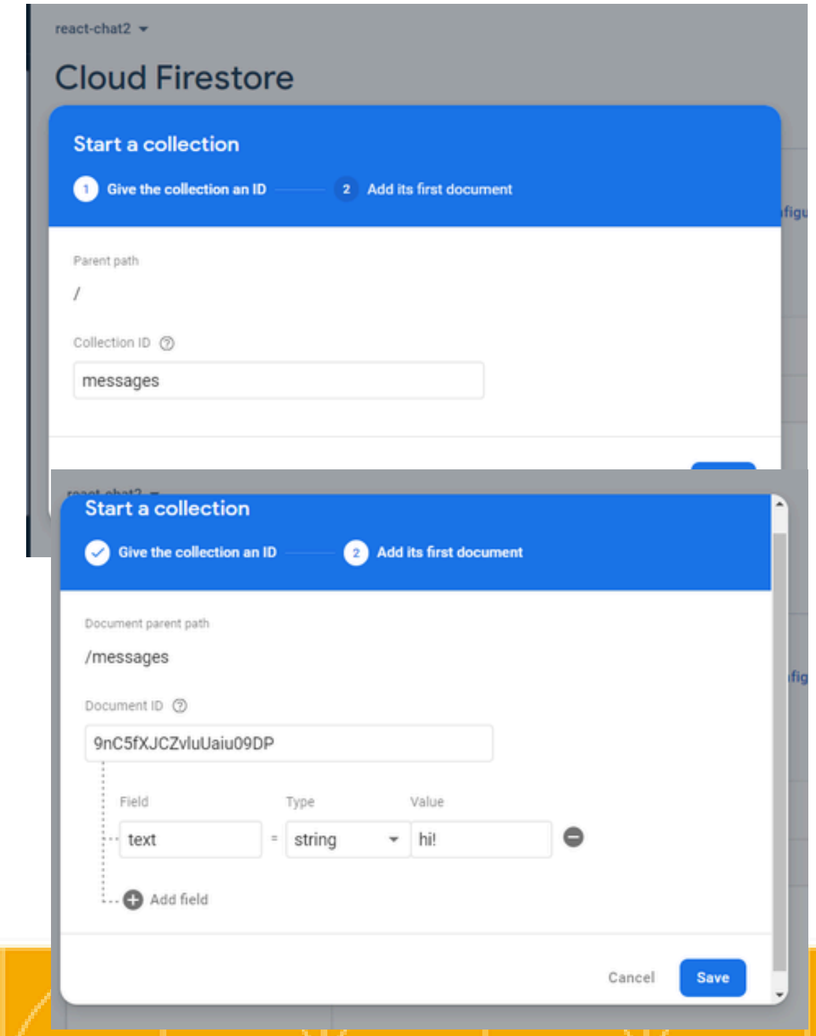
Choose for the Region where you use this




```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read: if true;
      allow create, update, delete, write: if request.auth != null;
    }
  }
}
```

Before next step, we must Edit our **Firestore rules**

Google Developer Groups



we can **Start a collection** in Firestore

Last we can create **Collection** in our database to save our messages request

THANK YOU

GET IN TOUCH WITH:

AUGISTA_EKSOFON

AUGISTA EKSOFON PRADANA

Google Developer Groups

END.