1. Google’s BERT Model (AI & Text Analysis)

**Integration Steps:**

1. Use a pre-trained BERT model via **Google’s TensorFlow Hub** or **Hugging Face Transformers**:
   * Install dependencies:

bash

pip install transformers tensorflow

* + Load the BERT model:

python

from transformers import BertTokenizer, TFBertForSequenceClassification

tokenizer = BertTokenizer.from\_pretrained("bert-base-uncased")

model = TFBertForSequenceClassification.from\_pretrained("bert-base-uncased")

1. Deploy a fine-tuned BERT model via **Google Cloud AI Platform** for API-based inference.

**2. GeoJSON (Mapping & Spatial Data)**

Used for:

* Representing geospatial data (affected areas, disaster zones)

**Integration Steps:**

1. Store location data as GeoJSON format:

Json

{

"type": "FeatureCollection",

"features": [

{

"type": "Feature",

"geometry": {

"type": "Point",

"coordinates": [-122.4194, 37.7749]

},

"properties": {

"disaster": "Wildfire",

"severity": "High"

}

}

]

}

1. Use **Leaflet.js (React)** or **Mapbox** to visualize GeoJSON on maps.
   * Example in **React**:

jsx

import { MapContainer, TileLayer, GeoJSON } from 'react-leaflet';

function MapComponent({ data }) {

return (

<MapContainer center={[37.7749, -122.4194]} zoom={10}>

<TileLayer url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png" />

<GeoJSON data={data} />

</MapContainer>

);

}

**3. GDACS API (Global Disaster Alerts)**

Used for:

* Fetching real-time disaster alerts (earthquakes, tsunamis, etc.)

**Integration Steps:**

1. Make an API request to fetch GDACS alerts:

python

import requests

url = "https://www.gdacs.org/xml/rss.xml" # GDACS RSS feed for real-time updates

response = requests.get(url)

if response.status\_code == 200:

print(response.text) # Parse XML response to extract alerts

1. Parse the XML response and integrate alerts into your dashboard.

**4. Google Maps API / OpenStreetMap (Mapping & Location Services)**

Used for:

* Visualizing disaster locations
* Routing & navigation

**Google Maps API Integration (React Native)**

1. Install dependencies:

bash

npm install react-native-maps

1. Add the component:

jsx

import MapView, { Marker } from 'react-native-maps';

function MapScreen() {

return (

<MapView

style={{ flex: 1 }}

initialRegion={{

latitude: 37.7749,

longitude: -122.4194,

latitudeDelta: 0.1,

longitudeDelta: 0.1,

}}

>

<Marker coordinate={{ latitude: 37.7749, longitude: -122.4194 }} title="Disaster Zone" />

</MapView>

);

}

**OpenStreetMap Integration (React Web)**

1. Install Leaflet:

bash

npm install leaflet react-leaflet

1. Use OpenStreetMap in React:

jsx

import { MapContainer, TileLayer } from 'react-leaflet';

function OpenStreetMap() {

return (

<MapContainer center={[37.7749, -122.4194]} zoom={10}>

<TileLayer url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png" />

</MapContainer>

);

}

**5. Twilio API / Firebase Cloud Messaging (Notifications & SMS Alerts)**

Used for:

* Sending real-time notifications & alerts

**Twilio API (SMS Notifications)**

1. Install Twilio SDK:

bash

pip install twilio

1. Send an SMS via Twilio:

python

from twilio.rest import Client

account\_sid = "your\_account\_sid"

auth\_token = "your\_auth\_token"

client = Client(account\_sid, auth\_token)

message = client.messages.create(

body="Emergency Alert: Wildfire detected in your area!",

from\_="+1234567890",

to="+9876543210"

)

print(message.sid)

**Firebase Cloud Messaging (Push Notifications)**

1. Install Firebase SDK:

bash

npm install firebase

1. Initialize Firebase in React Native:

jsx

import messaging from '@react-native-firebase/messaging';

async function requestUserPermission() {

const authStatus = await messaging().requestPermission();

console.log('Permission status:', authStatus);

}

**6. Firebase / PostgreSQL (Database & Authentication)**

Used for:

* Storing disaster reports
* Managing user authentication

**Firebase Integration (React Native Authentication)**

1. Install Firebase Auth:

bash

npm install @react-native-firebase/auth

1. Authenticate user:

jsx

import auth from '@react-native-firebase/auth';

function signIn(email, password) {

auth().signInWithEmailAndPassword(email, password)

.then(user => console.log('User logged in:', user))

.catch(error => console.error('Login error:', error));

}

**PostgreSQL Integration (Node.js + FastAPI)**

1. Install PostgreSQL driver:

bash

npm install pg

1. Connect to PostgreSQL in Node.js (Express backend):

javascript

const { Pool } = require("pg");

const pool = new Pool({

user: "your\_user",

host: "your\_host",

database: "your\_db",

password: "your\_password",

port: 5432,

});

app.get("/disasters", async (req, res) => {

const result = await pool.query("SELECT \* FROM disaster\_reports");

res.json(result.rows);

});

**1. Steps to Integrate APIs in Any App**

1. **Choose the Right API** – Determine if the API fits your use case (e.g., mapping, notifications, AI).
2. **Get API Keys / Credentials** – Sign up for the service and generate authentication keys.
3. **Make API Requests** – Use HTTP requests (REST API) or SDKs provided by the service.
4. **Process the Response** – Parse JSON/XML responses and use them in your app.
5. **Optimize for Real-Time** – If needed, use WebSockets, Firebase, or polling for real-time updates.

**2. API Integration Methods**

Each API service has a different integration method:

**(A) REST API (Standard HTTP Requests)**

* **Use when:** Fetching disaster alerts, geo-coordinates, AI processing.
* **How:** Use fetch() (JavaScript) or requests (Python).

**Example (Fetching GDACS Disaster Alerts via REST API)**

javascript

fetch("https://www.gdacs.org/xml/rss.xml")

.then(response => response.text()) // Parsing XML response

.then(data => console.log(data))

.catch(error => console.error("Error:", error));

**(B) SDK Integration (Easier & Optimized)**

* **Use when:** Integrating Firebase, Twilio, Google Maps, etc.
* **How:** Install SDK via npm (for JavaScript) or pip (for Python).

**Example (Firebase Push Notifications via SDK)**

1. **Install Firebase SDK:**

bash

npm install firebase

1. **Request notification permission & receive messages:**

javascript

import messaging from "@react-native-firebase/messaging";

async function requestPermission() {

const authStatus = await messaging().requestPermission();

console.log("Notification permission status:", authStatus);

}

**(C) WebSockets (For Real-Time Updates)**

* **Use when:** Need real-time disaster alerts (e.g., GDACS, Firebase).
* **How:** Use WebSocket API in JavaScript or Python.

**Example (Real-time updates via WebSocket in Node.js)**

javascript

const WebSocket = require("ws");

const ws = new WebSocket("wss://example.com/disaster-updates");

ws.on("message", (data) => {

console.log("New alert:", data);

});

**(D) AI Model Integration (Google BERT)**

* **Use when:** Need text classification for emergency messages.
* **How:** Use Hugging Face Transformers in Python.

**Example (Classify Emergency Messages with BERT)**

python

from transformers import pipeline

classifier = pipeline("text-classification", model="bert-base-uncased")

result = classifier("Wildfire detected near California")

print(result)

**(E) Mapping APIs (Google Maps / OpenStreetMap)**

* **Use when:** Visualizing disasters on a map.
* **How:** Install react-leaflet for OpenStreetMap or react-native-maps for Google Maps.

**Example (Google Maps in React Native)**

1. **Install Google Maps SDK:**

bash

npm install react-native-maps

1. **Display Map Component:**

jsx

import MapView, { Marker } from "react-native-maps";

function DisasterMap() {

return (

<MapView style={{ flex: 1 }}

initialRegion={{

latitude: 37.7749,

longitude: -122.4194,

latitudeDelta: 0.1,

longitudeDelta: 0.1,

}}

>

<Marker coordinate={{ latitude: 37.7749, longitude: -122.4194 }} title="Wildfire Alert" />

</MapView>

);

}

**3. How to Secure API Integrations**

1. **Store API Keys Securely** – Use **environment variables** (e.g., .env file).
2. **Rate Limit Requests** – Avoid excessive API calls to prevent bans.
3. **Use Authentication** – APIs like Firebase require **OAuth or JWT tokens**.

**Example (Hiding API Keys in .env file)**

**1. Store API Key in .env**

ini

MAPS\_API\_KEY=your\_google\_maps\_key

**2. Load API Key in JavaScript**

javascript

const apiKey = process.env.MAPS\_API\_KEY;

**4. Best Practices for API Integration**

✅ Use **caching** (e.g., Redis) to reduce repeated API calls.  
✅ Implement **error handling** (try...catch) to handle API failures.  
✅ Use **pagination** for large datasets (e.g., disaster reports).  
✅ Regularly **update API versions** to avoid deprecation issues.