/\*\*

\*

\* main() will be run when you invoke this action

\*

\* @param Cloud Functions actions accept a single parameter, which must be a JSON object.

\*

\* @return The output of this action, which must be a JSON object.

\*

\*/

Var request = require(“request-promise”);

Const DiscoveryV1 = require(“watson-developer-cloud/discovery/v1”);

Function getRandomInt(max) {

Return Math.floor(Math.random() \* Math.floor(max));

}

Const statesMap = {

Alaska: “99501:US”,

Alabama: “35801:US”,

Arkansas: “72201:US”,

American\_samoa: “96799:US”,

Arizona: “85001:US”,

California: “90001:US”,

Colorado: “80201:US”,

Connecticut: “06101:US”,

District\_of\_columbia: “20001:US”,

Delaware: “19901:US”,

Florida: “33124:US”,

Georgia: “30301:US”,

Guam: “GU:US”,

Hawaii: “96801:US”,

Iowa: “50301:US”,

Idaho: “83254:US”,

Illinois: “60601:US”,

Indiana: “46201:US”,

Kansas: “67201:US”,

Kentucky: “41701:US”,

Louisiana: “70112:US”,

Massachusetts: “02101:US”,

Maryland: “21201:US”,

Maine: “04032:US”,

Michigan: “49036:US”,

Minnesota: “55801:US”,

Missouri: “63101:US”,

Mississippi: “39530:US”,

Montana: “59044:US”,

North\_carolina: “27565:US”,

North\_dakota: “58282:US”,

Nebraska: “68901:US”,

New\_hampshire: “03217:US”,

New\_jersey: “07450:US”,

New\_mexico: “87500:US”,

Nevada: “89501:US”,

New\_york: “10001:US”,

Ohio: “44101:US”,

Oklahoma: “74101:US”,

Oregon: “74101:US”,

Pennsylvania: “15201:US”,

Puerto\_rico: “00600:US”,

Rhode\_island: “02840:US”,

South\_carolina: “29020:US”,

South\_dakota: “57401:US”,

Tennessee: “37201:US”,

Texas: “78701:US”,

Utah: “84321:US”,

Virginia: “24517:US”,

Virgin\_islands: “00801:US”,

Vermont: “05751:US”,

Washington: “98004:US”,

Wisconsin: “53201:US”,

West\_virginia: “25813:US”,

Wyoming: “82941:US”,

};

Function formatStates(state) {

State = state.toLowerCase();

State = state.replace(“ “, “\_”);

Return state;

}

Async function main(params) {

If (params.type === “api”) {

Try {

Const summary = await request({

Method: “GET”,

Uri: <https://api.covid19api.com/summary>,

Json: true,

});

If (params.location) {

// country was the old param, could be states in us.

State = formatStates(params.location);

If (state in statesMap) {

Const uri = `https://api.weather.com/v3/wx/disease/tracker/state/60day?postalKey=${statesMap[state]}&format=json&apiKey=${params.twcApiKey}`;

Const data = await request({

Method: “GET”,

Uri: uri,

Json: true,

});

Return {

Result: `Total Cases: ${data.covid19.confirmed[0]}\nTotal Deaths: ${data.covid19.deaths[0]}\n\nSource: ${data.covid19.source[0]}`,

};

}

For (var I = 0; I < summary.Countries.length; i++) {

If (

Summary.Countries[i].Country.toLowerCase() ===

Params.location.toLowerCase() ||

Summary.Countries[i].CountryCode.toLowerCase() ===

Params.location.toLowerCase()

) {

Return {

Result: `Total Cases: ${summary.Countries[i].TotalConfirmed}\nTotal Deaths: ${summary.Countries[i].TotalDeaths}\nTotal Recovered: ${summary.Countries[i].TotalRecovered}\n\nSource: Johns Hopkins CSSE`,

};

}

}

Return { error: “did not find location” };

}

Let totalCases = summary.Global.TotalConfirmed;

Let totalDeaths = summary.Global.TotalDeaths;

Let totalRecovered = summary.Global.TotalRecovered;

Return {

Result: `Total Cases: ${totalCases}\nTotal Deaths: ${totalDeaths}\nTotal Recovered: ${totalRecovered}\n\nSource: Johns Hopkins CSSE`,

};

} catch (err) {

Return { error: “it failed : “ + err };

}

} else {

Const discovery = new DiscoveryV1({

Version: “2019-03-25”,

Iam\_apikey: params.api\_key,

url: params.url,

});

Const offset = getRandomInt(50);

Const queryParams = {

Environment\_id: params.env\_id,

Collection\_id: params.collection\_id,

Natural\_language\_query:

“corona virus “ + params.input || “corona virus news”,

Count: 3,

Offset: offset,

};

Try {

Data = await discovery.query(queryParams);

Let response = data.results.map((v, i) => {

Return `${v.title}

${v.text}

${v.url}`;

});

Return {

Result:

“Here are some news articles we found. We can’t verify the accuracy of all of these sources.\n\n” +

Response.join(“\n\n”),

};

} catch (err) {

Return { error: “it failed : “ + err };

}

}

}