Dependencies

```
const {WebhookClient} = require('dialogflow-fulfillment');
const {Card, Suggestion} = require('dialogflow-fulfillment');
```

Import the 2 libraries above to interact with Dialogflow.

The following function defines the actions to be taken for each intent that requires fulfilment.

exports.dialogflowFirebaseFulfillment = functions.https.onRequest((request, response) => {});

```
let intentMap = new Map();
intentMap.set('Default Welcome Intent', welcome);
intentMap.set('End Conversation', endConversation);
intentMap.set('employment', employment);
intentMap.set('partner', partner);
intentMap.set('expenses-spendings', expensesSpendings);
agent.handleRequest(intentMap);
```

intentMap.set(*intentName*, *functionName*) is used to associate a function with an intent. The first parameter is the name of the intent, and the second parameter is the function name. If the fulfilment of an intent is enabled, the associated function is called every time the intent is matched. Every associated function has agent as the parameter.

```
function welcome(agent) {
    const session = {'name': 'session', 'lifespan': 10000, 'parameters': {
      'session_id': agent.session,
      'client_name': agent.originalRequest.payload.name,
      'ew_id': agent.originalRequest.payload.ew_id
    }};
    agent.setContext(session);
}
```

The code above is function that's associated with the 'Default Welcome Intent', "agent" is passed into the function when it's called. "agent" object contains functions to retrieve the information of the agent, manage contexts, and add a response to the end user. Some commonly used functions are listed below.

setContext(session)
It's used to add an active context to the conversation. The flow of the conversation can be changed by setting contexts. A context has the structure as shown below

```
function welcome(agent) {
  const context = {'name': 'session', 'lifespan': 10000, 'parameters': {
    'session_id': agent.session,
    'client_name': agent.originalRequest.payload.name,
    'ew_id': agent.originalRequest.payload.ew_id
  }};
  agent.setContext(context) ;
```

The first field is the name of the session. Lifespan is the number of conversational turns of this context (a conversational turn consists of an agent's question and a user's response). When the number of conversational turns is reached, the context is removed from the conversation. "parameters" field contains key-value pairs that are stored in the context.

agent.getContext("session")
It's used to retrieve an active context from the conversation. "Context" is used to store parameters in Dialogflow. To get all the parameters stored in a context, use **context.parameters**.

Deploy fulfilment

Inline editor

To use the inline fulfilment editor in Dialogflow console, simply edit the fulfilment code there and add dependencies in the package.json file, then click deploy. Dependencies specified in the package.json file will be automatically installed upon deployment.

Cloud functions in firebase

The other option is to use firebase cloud function to manage the fulfilment code. To use firebase cloud function, install firebase tools from the command line:

npm install -g firebase-tools

Then initialize the function:

firebase init functions
cd functions

Once the fulfilment code is complete, deploy the function to firebase by entering

firebase deploy --only functions

Inside the functions directory from the command line.

To add this function to the Dialogflow agent. Find the URL of this function from firebase console, then copy this URL to the webhook of the Dialogflow agent. Note that, if you use firebase cloud functions for fulfilment, the inline editor for fulfilment will be disabled. If you

change your mind and want to use inline editor, the associated cloud function must be deleted.