First Holiday – Chat Agent

Matthew McCormack

March 29, 2023

White Stuff

Table of Contents

[Introduction 2](#_Toc131152597)

[Design 2](#_Toc131152598)

[UML 2](#_Toc131152599)

[Wireframe 2](#_Toc131152600)

[Data model - Sequence Diagram 3](#_Toc131152601)

[Project structure 3](#_Toc131152602)

[Limitations and additions 4](#_Toc131152603)

[Testing 4](#_Toc131152604)

[Returns a reply from the answersArray when a matching utterance is found 4](#_Toc131152605)

[User Guide 4](#_Toc131152606)

[Step 1 5](#_Toc131152607)

[Step 3 5](#_Toc131152608)

[Step 4 5](#_Toc131152609)

[Step 5 5](#_Toc131152610)

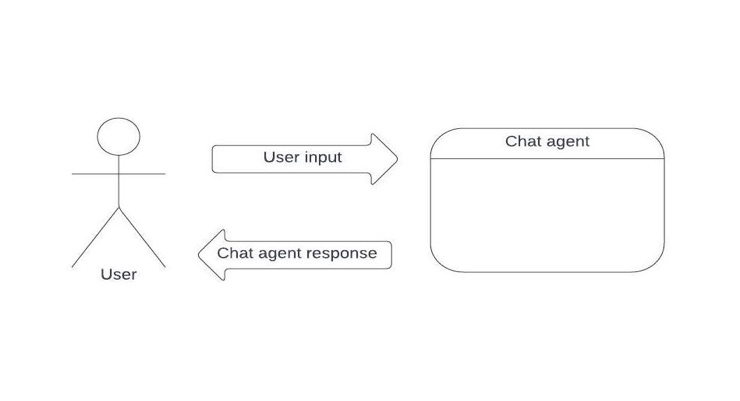
[Step 6 6](#_Toc131152611)

## Introduction

This chat agent is a web application to be used on the end user’s browser. It is designed to take in user input (text) and respond via a text output based on its processing of the user input. In short, the response is to make sense relative to the user input. It will pick up on key words ‘flight’ and ‘holiday’ in order to respond with a question for the user. This is to guide the user into finding a holiday which suits their preference by using the features provided by my app. Along with this response a button will appear giving access to a filter form allowing the user to enter information to filter through the options. The user can leave the form blank in order to see all options within the table. Once the filter form button is then clicked the filtered table will display bellow. The holiday packages data will be pulled in through an API from a MySQL database hosted locally.

## Design

## UML



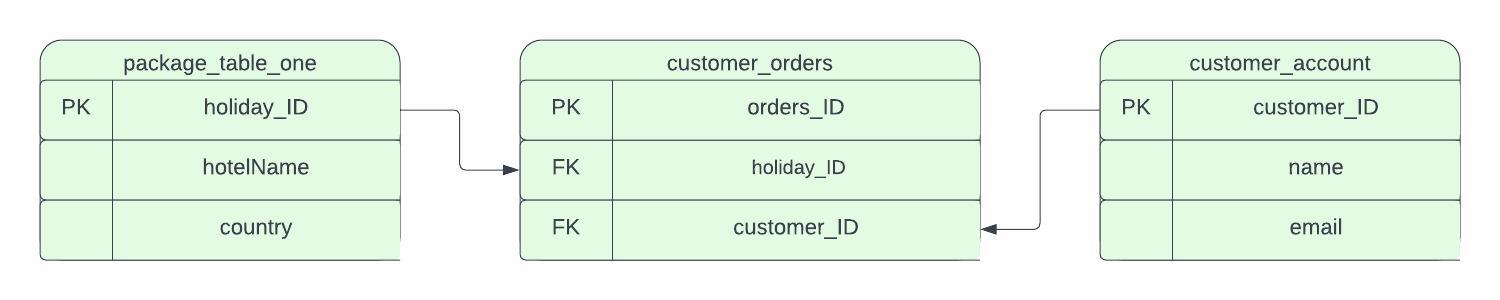
Display   
- filter form   
- relevant data table

Wireframe

Table

Description automatically generated with medium confidence

Data model - Sequence Diagram



Above is a data model based on relational database architecture. In this project I have created a single table in MySQL DB represented in a basic form on the far left of the image above.

Project structure

Diagram

Description automatically generated

## Limitations and additions

I have taken this approach after conducting research on other chat bots of similar capacities, I noticed they tend to have a small amount of dialogue between the user and the bot before pointing them in the direction of their product in this case the holiday packages. However, it is possible to expand on the possible interactions between the bot and the user by simply adding to the “utterances” and “answers” arrays accordingly. This may cause issues further down the line as in order to cover all eventualities that may occur within a conversation would mean to have an extensive set or utterances and answers. This would be a great way to improve its ability to interact with the end user in a more convincing manner.

Further limitations include the two catch words “flight” and “holiday” embedded within the logic. As this will trigger the response…

"So you're looking to go on holiday. Would you like to see our holiday packages? Just click Yes bellow!"

…from the answers array when the user inputs anything with these words included. This response will include the displaying of a button which when clicked on will produce a filter form allowing the user to enter key filter information to narrow down their search. This is useful if the data provided is very extensive. It also saves the user from having to refresh the page every time they wish to make a different decision or go through the entire chat process again.

My API only has a GET function as opposed to utilising the entire CRUD repertoire. As my web app only displays data from a data base to the user, I initially have only given the app the ability to “read” the data. The rest of the CRUD functionality could be added as a future improvement to allow users to save their account information and update it accordingly.

Another way to expand this project for wider use is to make use of the relational database. This could hold customer accounts and customer orders allowing for the use of foreign keys to build larger more complex data sets based on the relationship between the tables. The relational quality of the database is identifiable as the relationships between the data in the table can be inferred based on the data itself. For example, we can tell which hotels are located in which cities or countries by looking at the values in the corresponding columns. This allows for expansion based on the introducing more tables such as customer\_orders and customer\_account etc. (Please see above: Data Model – Sequence Diagram for graphical representation).

## Testing

To run unit test: Copy index.html file, found in the Tests directory, and paste into the browser. The test will run on page load automatically.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test No. | Description | Tools | Expected outcome | Results | Pass/Fail |
| 1 | Unit testing the compare function in script.js | Mocha/chai, vscode, chrome | Returns a reply from the answersArray when a matching utterance is found | Correct array index response was generated | Pass |
| 2 | Unit testing the compare function in script.js undefined response | Mocha/chai, vscode, chrome | Returns a reply from the alternatives array when no matching or keyword is found in input | Correct array index response was generated | Pass |

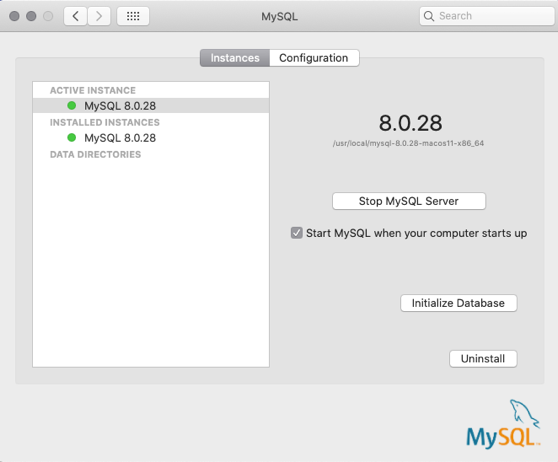
## User Guide

For reference

* Github repo: https://github.com/mcc-ctrl/first-holiday-chat-agent
* MySQL Server local host
* Node server local host
* Local development server VS Marketplace Link: <https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer>

### Step 1

Start the MySQL server



For this web app I am using a MySQL server hosted locally on my machine. This is to enable me to create a database using the data provided in the brief. The API then uses a get function to

### Step 2

Start the node server by entering the command node index.js into the command line when in the root folder “first-holiday-chat-agent” of the application. This will allow the API to parse the data from the database to the local host URL <http://localhost:3000/first-holiday-chat-agent> in JSON format.

You should see this confirmation message in the command line interface – “Example app listening at http://localhost:3000”

### Step 3

If you haven’t done so already then download the Local development server extension below in VSCode:

Name: Live Server Id: ritwickdey.LiveServer

Description: Launch a development local Server with live reload feature for static & dynamic pages Version: 5.7.9

Publisher: Ritwick Dey

VS Marketplace Link: <https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer>

Once this extension has been downloaded to VSCode you can then click on “Go live” in the bottom right-hand corner of the VSCode window

Graphical user interface, text, application

Description automatically generated

This will launch the web app in your default browser.

### Step 4

From here you can use the chatbot in your browser as you please. There will be a “write something…” prompt for you to start the interaction. The premise is built on the user saying “Hi” or “Hello” in which case the bot will respond with a question. This then prompts the user to click the “yes” button to view the filter form enabling the user to choose their destination more accurately.

##### Chat agent

Graphical user interface, text, application, chat or text message

Description automatically generated

Once this button is clicked then the user can see the table below based on the results of the filter form submission. If left empty the filter form will produce the entire table of holiday packages. You can edit and re-submit the filter form as many times as you like to retrieve different results.

##### Filter form

Graphical user interface, text, application, chat or text message

Description automatically generated

### Step 5

View the data produced in table format.