20/09/2022

Isaac Naylor

Jet2

Synoptic Project

Holiday Chat Bot

Contents

[Design 2](#_Toc114821042)

[Requirements. 2](#_Toc114821043)

[Data. 2](#_Toc114821044)

[Wireframes. 2](#_Toc114821045)

[Use Cases. 3](#_Toc114821046)

[Models 4](#_Toc114821047)

[Development 5](#_Toc114821048)

[Implementation 5](#_Toc114821049)

[Data 5](#_Toc114821050)

[Project Setup. 6](#_Toc114821051)

[Unit Testing 7](#_Toc114821052)

[Manual Testing 7](#_Toc114821053)

[Conclusion 8](#_Toc114821054)

[Design Limitations 8](#_Toc114821055)

[Improvements 8](#_Toc114821056)

[User Guide 8](#_Toc114821057)

# Design

Before deciding how I would like the user interface to look I would like to get an abstracted list of requirements.

## Requirements.

* The chat bot must ask minimum 2 questions
* The chat bot must feedback to invalid input from the user
* The chat bot must provide a set of recommended holidays

One assumption I have made about the feedback to the user is that not allowing an empty input to be sent counts as feedback.

## Data.

I have decided that I would like to load my dataset from a JSON format. This is because I find the JSON format easier to read and write, both myself and from code.

## Wireframes.

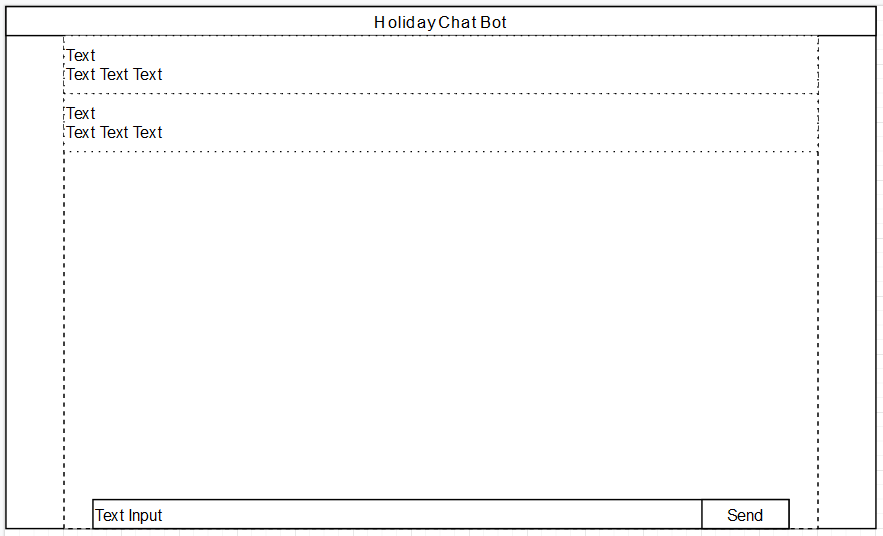
Figure 1 Shows the basic Wireframe that I would like to create my application to look like.

Figure 1: Chat Bot Wireframe

I have decided that the most logical way to implement the Minimal Viable Product is by having it be a full-page application, however I think if this were to be incorporated into a larger website, it could also be shrunk down to be a small component in the corner of the screen.

The wireframe in figure 1 shows the trail of conversation the user has had with the chat bot, with the top row of text being used to show who has sent a message (Chat bot or User), with the bottom row of text representing the message sent by either party.

## Use Cases.

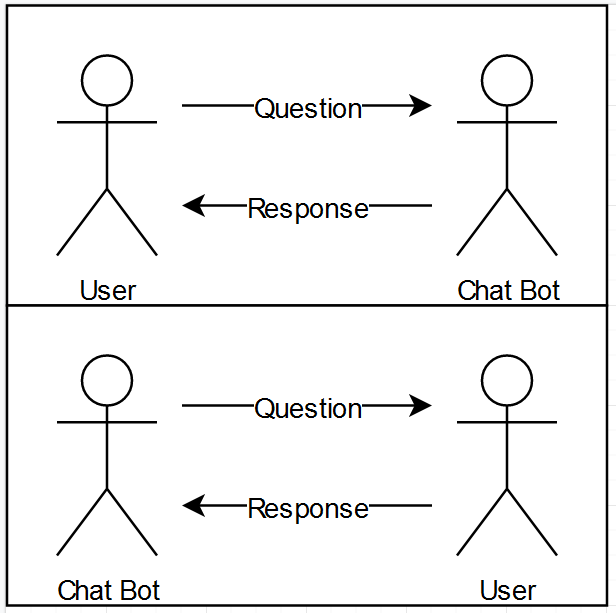


Figure 2: Use Case Diagram

The main use cases for this application are for the user to ask the bot questions to get a holiday recommendation, or the user to be asked questions by the bot and replying to be given a holiday.

## Models

Figure : Class Diagram

One of the most important components of this application will be the data. Figure 3 shows the classes that will be used for data storage and manipulation throughout the project. The Holiday class will be a class purely for containing data, as it will hold all of the information that is known about the holiday and used to filter it. The Question class is also going to be for data, as it will

# Development

## Implementation

I have decided to implement this project using the Python programming language. Within Python I am using a web development library called Flask. This is because I am familiar with Python and Flask and I think that they suit this sort of quick turnaround style of development.

I am also going to be using the Bootstrap CSS framework to provide me with some common utility classes in CSS to improve the speed at which I can deliver a nice user interface.

## Data

As stated earlier, I decided to use JSON to store the holiday data. I chose to do this because it is a standard way of sending and receiving data on the web, and it is easy to read both from a human and code perspective.

Below is a screenshot of one of the data sets converted to JSON.



I wrote and ran a small python script which read in the data csv file and converted it’s contents to JSON. This script is included in my project (“csv\_to\_json.py”)

## Project Setup.

Before I can get started on my project, I must first decide what tools I want to use. The primary tool involved in the creation of this project will be Visual Studio Code, a code editor that is designed for working with modern web development. Alongside my text editor, I will also be using a Git client called “Fork” for my version control management, I also used the command line interface for Git within my terminal application, Windows Terminal. For the documentation and testing of my app, I will also be using some of the Microsoft Office tools, Word and Excel.

Once I have all the tools I need, I need to get my project environment set up. Because the language I have chosen to use is Python, I need to set up a Virtual Environment as that is the recommended convention for python projects. I named this environment “.env” as it means that it will stay at the top of my folder structure making it slightly easier to ignore than if it were in the middle. Once I have created the virtual environment, I need to activate it and install the packages I need. These packages are: Flask, Flake8, Pytest, Coverage, and Python-Dotenv. Once they have all been installed, I run the the command “python -m pip freeze > requirements.txt”. This creates a text file which be used to install all necessary packages to run the project.

Once I have my virtual environment set up, it’s time to configure everything to make it easy to run. Firstly, I created a .flaskenv file, which would set up environment variables, to allow running my flask app to be easy. These variables simply state where to run my application, and what mode to run it in. Then I need to create a .flake8 file, which will configure the linter for my python code, telling me when I have written code that does not follow the recommended conventions. This configuration file simply tells Flake8 (A python linter) what folders to ignore, and some other settings such as when to use colour in the output. The final configuration file I have written is my .coveragerc file. This file will tell the python library “Coverage.py” what settings to run when I try to run the coverage command. In this case I told Coverage to run Pytest, and generate a HTML report for my code coverage statistics.

The final part of my setup is creating the folders and initial files I will need to create and test my application. The first folder I made, was my “app” folder. This is where all the source code for my application is going to go. Within the app folder I need 2 subfolders, “templates” and “static”. The templates folder is where all the HTML for my application will go, and the static folder is where I will put CSS and JavaScript files for improving the user interface and experience of my application. The next folder I created was the data folder. This folder will hold any data files I need, initially this will just be the Holidays.json file that I created from the provided dataset. The next and final folder that I will need to create is my tests folder. This folder is where all of the unit tests that I write for my application will go, as well as the setup and configuration of these tests.

Testing

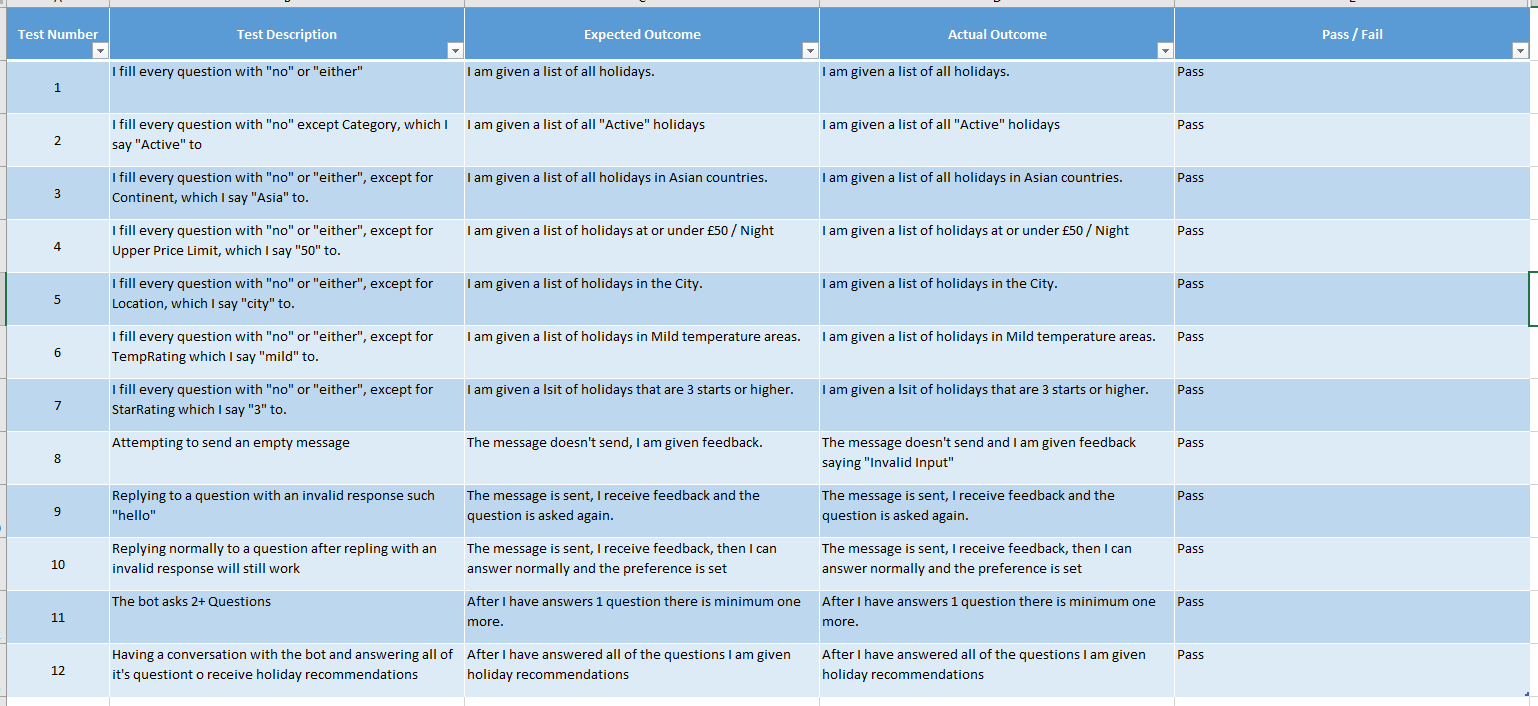
## Unit Testing

For this project I used 2 python libraries, Pytest and Coverage. Pytest is a library used to write and run tests, whereas coverage is used to generate code coverage reports. Using these 2 libraries together allowed me to write unit tests for my code and receive a code coverage report, which allowed me to reach 100% code coverage.

## Manual Testing

Manual Testing is a big part of testing as it allows me to experience what the end user will experience and find any issues I find with the functionality, as well as the usability.

My Manual tests have been documented in the Excel file named “Manual Testing”



# Conclusion

## Design Limitations

I think that the way I have designed and implemented my app means that it would be relatively difficult to scale up to be part of a large website, due to the coupling of the front and back ends.

I also think that the application is limited by the use of a global instance of the bot for tracking the current users’ preferences. I think that this could be resolved using an account system, and permanent storage of each users’ preferences.

## Improvements

I think one way that I could improve this project further would be adding an account system that can track a user’s preferences. Currently, all interactions with this bot would be stored into one set of preferences, I think that a major improvement to this project would be allowing concurrent users, however that is outside of the scope of the requirements.

Another significant improvement I think could be made to this project would be to improve the recognition of a user’s answers. The current implementation requires that a user not make any spelling mistakes in answering the questions, which could cause frustration for users if they answer a question wrong and are required to start the entire holiday recommendation process from the beginning. Some sort of autocorrect feature in the text input box could be a potential solution to this problem, allowing users to be recommended the potential answers to a question, or perhaps in the code for the Chat Bot itself, allowing it to recognize and correct spelling mistakes to it’s best approximation.

I think another small improvement could be to ensure that upon refreshing the page, the bot is set back to it’s default state, as it currently will continue to build up the preferences of the previous user when you refresh, and you have to reset the bot through the button I added to the screen.

## User Guide

Upon loading the Holiday Chat Bot page, you will be greeted by a message from the bot asking for your name. This can be any name you like, and will be what you are referred to as for the duration of your conversation with the bot.

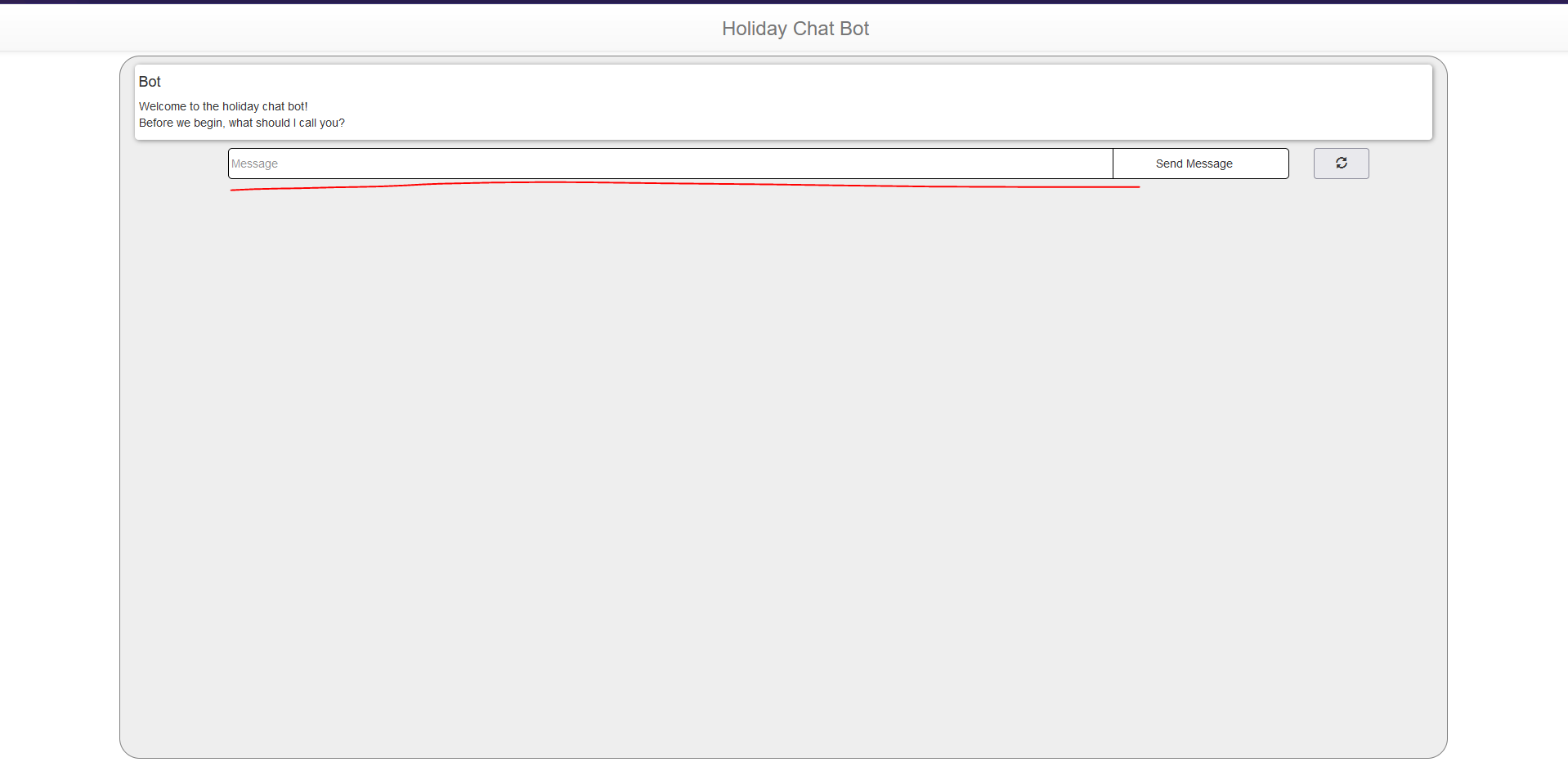
To speak to the bot, you simply need to type into the message box in the central window. 

Figure 4: The Chat Window

To send a message to the bot, you can either press the “Send Message” button, on the right hand side of the text input, or you can press enter.

The bot will ask you questions about your preferences, for example it will ask you if you have a preferred continent that you would like to look at holidays in. When it asks if you have a preferred Continent, or Temperature or another holiday factor, you simply need to reply with something positive, such as “Yes”, “Of course” or something like those words and phrases, on the other hand, if you have no preference, you can simply reply with something negative, such as “No”, or “Nah”. If you input something that the bot cannot determine is positive or negative, it will ask you to repeat your answer and repeat the question to you. In the case that you have a preference, it will then ask a follow-up question, asking you what your preference is. Sticking to the example of Continents, you could reply with “Asia” and the bot will then filter the holidays down to only those within Asian countries.

Once you have answered all the questions the bot has to offer, it will provide you with a list of holidays it believes are suitable based on the answers you have given. The information provided to you is: The hotel name, city, and country. The Continent the hotel is in, as well as the location (sea, city or mountain). It also will tell you the rating of the hotel, the average temperature range of the area, and how expensive it is per night.

On the right side of the message box, is a button with a circular arrow on it. Clicking this button will reset the state of the bot, setting it back to asking for your name, and setting the preferences back to default.