MonkeyWeather Milestone 2 Report.

Contents

[Table of contents i](file:///C:\Users\baraa\Downloads\toc.docx#_Toc5630352)

[Introduction ii](file:///C:\Users\baraa\Downloads\toc.docx#_Toc5630355)

[Plan ii](file:///C:\Users\baraa\Downloads\toc.docx#_Toc5630356)

[Research ii](file:///C:\Users\baraa\Downloads\toc.docx#_Toc5630357)i

[Modifications ii](file:///C:\Users\baraa\Downloads\toc.docx#_Toc5630357)i

[Design](file:///C:\Users\baraa\Downloads\toc.docx#_Toc5630360) v

[Scripts](file:///C:\Users\baraa\Downloads\toc.docx#_Toc5630361) v

[Classes](file:///C:\Users\baraa\Downloads\toc.docx#_Toc5630362) iv

[Future Additions](file:///C:\Users\baraa\Downloads\toc.docx#_Toc5630363) xv

Contributors viii

Reference………………………………………………………………………………….

Introduction.

MonkeyWeather is a chatbot that communicates weather forecasts to the user based on the users desired locations. It aims to provide information on the weather conditions and advise the user on what clothing they should bring with them on their journey to the chosen location.

Monkey weather was developed using HTML and CSS for GUI and JavaScript for functionality. We also insured that all our code is tested and is running as expected using QUnit testing. We decided to use JavaScript as our programming language as it is easy to implement a GUI with JavaScript.

Plan.

For this review we have built more functionality to the bot so that it could behave more responsively to the user’s input. We have also improved our GUI and developed more onto it to make it easy on the user’s eye and for it have a more sophisticated appearance.

The bot will take in and process user input as needed. The user will be able to select their desired destination and confirm it with the bot. The bot will analyse this information and prompt the user to select the next location they wish to visit. After the user has selected all five destinations for their three-day vacation, the robot will advise the user on what clothes to bring for each destination based on the weather conditions.

Given that this plan implements the majority of the functionality we anticipate the chatbot to give, we hope to be able to focus on the quality of the bot's answers and provide the user with as much information as possible to provide a pleasant trip with clarity on what to expect at each destination. We'd also like to provide the bot with more advanced details and behaviours.

Research.

After completing the first milestone, we decided to conduct more API research in order to obtain the most accurate weather data available. Previously we were using the 5 day/ 3 Hour forecast API from the OpenWeatherMap website. We were conducting weather data for every three hours by passing the user inputted city name to the API. Although it was working as we wished it to however, we noticed that it was quite inefficient. The 5 day/3 Hour forecast was providing us with a forecast on three-hour intervals of the day, which we saw as unreasonable and unnecessary information therefore we decided to use One Call API from the OpenWeatherMap site.

One call provided a forecast for the weather at the current time, a minute forecast for an 1hour, an hourly forecast for 48 hours, a daily forecast for 7 days, with that information in mind we decided that use a daily forecast for 7 days would be most reasonable and efficient to our chatbot.

We noticed when using the city name inputted by the user directly to into the API we were going to run into so inconvenience considering that there are several cities around the world with the same name. We did some research on using API to get accurate geographical locations. We found the Free Geocoding API from <https://geocode.maps.co/> . This API was able to get us explicit data on the locations using city names, which was more convenient to avoid confliction between cities with the same name.

Modifications

We have updated the CSS so that the chat bot appears more presentable to the user.

Screenshot of before:

Graphical user interface, application

Description automatically generated

Screenshot of after:

Graphical user interface, application, website

Description automatically generated

We have added more functionality to the chatbot. It is more responsive to the user input.

The user can now select five different locations that they want to travel to during the trip.

The bot will process these locations and will get the weather forecast for the upcoming three days of the requested location using the OpenWeatherAPI.

The chat bot will then suggest recommendations of what clothes the user will bringing depending on the weather conditions and temperatures level of the location, printing out the suggest clothes to the user.

We have created a state machine for our bot to maximise the efficiency of the code processing when the bot is communicating with the user.

We also decided to use OOP to increase efficiency and readability of our code. The code design is described in more detail in the design section.

Design

**Scripts**

~ main.js:

* This is a JavaScript file that connects directly to our index.html. It implements the functionality of our webpage. This script stores the chatBox element and userInput as variables which is further used for the functions of our bot.

Text

Description automatically generated

* It also allows user input to be passed to the chatBox under two events: Send button is clicked or when the enter button is pressed. The bellow screenshot shows how these functions were implemented.
* We also included a handleInput function that passes the user input to the chatBox object, so that the input could be updated and stored inside that chatBox class to be further used for other functionalities of our bot.

Text

Description automatically generated

**Classes**

~ ChatBox:

This class creates a chatBox object that will take in user input and then display it inside the chatbot in the HTML.

Functions

* getInput function gets the value inputted by the user.

Text

Description automatically generated

* The retrieveInput function stored the value that is inputted by the user and returns the stored value.

Text

Description automatically generated

* UpdateChatBox function updates the chatBox by creating a HTML document and then appends it as a child to the chat box.

Text

Description automatically generated

~ ChatBot:

This class uses a state machine that goes through all of the states the bot will go through from the first interaction with the user to the last response of clothes recommended to the user.

It uses a clothesRecomenation object to output the recommended clothes to the user in regard to weather conditions of each location the user is looking to travel to.

**States:**

* Greeting State, at this state the bot is waiting for the user to greet it for it start with asking for information from the user.
* Starting the state; after the user has greeted the bot, the bot will ask the user if they want to choose the locations they are looking to go to.
* Location State; after the user have inputted their location the bot will proceed to the Location State. During this state the bot will get the precise location information using the LocationFetcher class.
* Confirming Location State: after the bot has processed the location inputted by the user. The bot will ask the user to confirm the retrieved location. If the user confirms the bot will move the Forecast state otherwise it will move to the Alternative Location State.
* Alternative Location State: during this state the bot will display a list of other locations with the same city name as the one the user inputted and will ask the user to choose from those options. After the user chooses it will move to the forecast state.
* Forecast State: at this state the bot will get the weather forecast using the ForecastFetcher class.
* Date State: After getting the forecast of the location the bot will ask the user to choose a date of when they will be travelling to the location.
* Destination State: during this state the bot will store the user destination and then add the destination to a list of places the user will go on their three-day journey and will update the recommended clothes accordingly.
* Confirm Destination State: The user is asked to confirm if they are satisfied with all their chosen locations.
* Recommendations state: The bot will get all the clothes recommendations for each chosen destination will list them to the user.

The screenshot bellow is a State machine Diagram showing all states of the bot.

Diagram

Description automatically generated

~ ForecastFetcher Class:

This class uses OpenWeatherAPI to get the weather forecast of the upcoming seven days for the desired location of the user.

Functions

* GetWeather function, passes the location to the API to retrieve the weather data and then returns the weather data to the bot.

Text

Description automatically generated

~LocationFetcher Class:

This class gets the precise location using the location name the user input. The location name is passed to an API that returns accurate location data of the requested location.

Text

Description automatically generated

~Location Class

This class creates a location object with a name, longitude, and latitude.

* getLocationFromData function generates an array of Location Objects from the data received from Geocoding API. Then returns an array of locations corresponding to each element in the input array.

Text

Description automatically generated

Destination Class

This class creates a destination object using the data of the confirmed location inputted by the user.

Text

Description automatically generated

Functions

* getDates function returns an array of all the dates stored in the forecast array.

Text

Description automatically generated

* selectDates function updates the forecast for the chosen destination to only contain weather data for the chosen dates the user will visit the location.

Text

Description automatically generated

* getAllForecastTemperatures function returns the temperature levels at the destination for all the forecasts.

Text

Description automatically generated

* getAllForecastWeather function returns an array containing all the weatherMain for the destination.

Text

Description automatically generated

* updateClothesRecommendation updates the clothes recomendations based on the temperature levels and the weather conditions.

Text

Description automatically generated

~ ClothesRecommendations Class

This class stores recommended clothes that the user should consider bringing with them. The clothesRecomendation object gets a response message that is delivered by the bot to the user informing the user what clothes to bring depending on the temperature levels and weather conditions.

Functions

* updateByTemperature function updates the clothes variables depending on temperature levels. The clothes type increments in consideration of how many forecasts are found within the upper and lower bound temperatures of each clothes type.

A screenshot of a computer

Description automatically generated with medium confidence

* updateByWeather function increments the clothes based on weather conditions such as rainy weather.

Text

Description automatically generated

* temperatureBetween function returns true if the temperature is between the set maximum and minimum temperatures.

A picture containing text

Description automatically generated

* getMessage function returns a list element of all the clothes the user will need for their trip. It checks whether the clothe type has been incremented and returns a message accordingly

Text

Description automatically generated ……..

Text

Description automatically generated

* combineRecomendations functions combines the clothes recommendations of two clothesRecomendation objects. The purpose of this function is that if the is travelling to countries of similar weather forecasts they’re told to bring the necessary clothes just once rather than bot sending the same information several times.

Text

Description automatically generated

Future Additions

We are looking to make the side bar functional so that the location, weather forecast, date and clothes recommendation appear inside the side bar when the user confirms their location choices for their trip.

We also want to include an input suggestion bar above the user input box so that the user can choose stored keywords.

A picture containing text, black, electronics, white

Description automatically generated

Testing.

We used qUnit for our first milestone, but after conducting some further study on testing in JavaScript, we realized that JEST would be the best option for us.

JEST has more up-to-date information and resources available online, therefore we've made this decision. JEST is also a lot more legible and simpler to create code with.

JEST also makes mocking HTML elements for testing reasons easier.

Contributors

Bara’ah Nidal Afana -3054933

Arthur Martins – 3028568

Mert Bekar – 3050376

Referencing

Jestjs.io. 2022. *ES6 Class Mocks · Jest*. [online] Available at: <https://jestjs.io/docs/es6-class-mocks> [Accessed 20 April 2022].

Jestjs.io. 2022. *Expect · Jest*. [online] Available at: <https://jestjs.io/docs/expect> [Accessed 20 April 2022].

<https://stackoverflow.com/questions/41098009/mocking-document-in-jest>

<https://jest-bot.github.io/jest/docs/configuration.html>

<https://github.com/jsdom/jsdom>