Frederick Laroche

NMIT 2024

SDV602

Assessment 2 ,part 1

Contents

[1.Overview 2](#_Toc178946787)

[Application purpose: 2](#_Toc178946788)

[Motivation: 2](#_Toc178946789)

[2.App Design and Documentation 3](#_Toc178946790)

[Login 3](#_Toc178946791)

[Top command interface 4](#_Toc178946792)

[DES one - Current condition 5](#_Toc178946793)

[DES two – Historical data 7](#_Toc178946794)

[DES three – Yearly comparison 9](#_Toc178946795)

[References: 11](#_Toc178946796)

# 1.Overview

## Application purpose:

This scientific weather app is designed for users that wants to get accurate weather data from all over the world. With this data , they can get the current weather from anywhere or build some analysis on the climate change and write theory about the subject by using the comparison feature.  
  
The app has 3 main features:

* view the current weather conditions,
* view historical weather data ,
* compare weather data from different years.

## Why build this app?

My motivation to build this specific app comes a need of accurate weather data to do research and analysis and help other people with this tool. Climate change is “trending” at the moment but let’s make our own observation first . Not only this app will provide data to do research on climate change, but it can also help to build forecast predictions tool by having such a large dataset .

# 2.App Design and Documentation

## Login

A screenshot of a computer screen

Description automatically generated

|  |  |  |
| --- | --- | --- |
| No. | Element | User action/interaction |
| 1 | Title | This is only a title. The user can’t interact with it. |
| 2 | Username label and input box | The user enters their username in the input box. It must be at least 7 at least 7 characters, have 1 uppercase letter and 1 special character. |
| 3 | Password label and input box | The user enters their password in the input box. It must be at least 7 at least 7 characters, have 1 uppercase letter and 1 special character. |
| 4 | Submit button and logic | After entering the username and password, the user clicks on the “Submit*”* button.  The system will verify the credentials using a JSON service, and if valid, this window closes and then the top command interface is displayed. If incorrect, a message is shown depending of the error type . |
| 5 | Quit button | This will close the app. |

## Top command interface

A screenshot of a computer

Description automatically generated

|  |  |  |
| --- | --- | --- |
| No. | Elements | User action/interaction |
| 1 | Title | This is only a title. The user can’t interact with it. |
| 2 | Current condition button | The user clicks this button to navigate to DES 1 (Current Condition(Today)). |
| 3 | Historical data button | The user clicks this button to navigate to DES 2 (Historical Data). |
| 4 | Yearly comparison button | The user clicks this button to navigate to DES 3 (Yearly Comparison), comparing weather data from two selected years for a chosen location(city). |
| 5 | Quit button | This will close the app. |

## DES one - Current condition

A screenshot of a chat

Description automatically generated

|  |  |  |
| --- | --- | --- |
| No. | Elements | User action/interaction |
| 1 | Title | This static title tells the user which DES they are on. The user cannot interact with it. |
| 2 | Navigation buttons (previous/next) | The user can navigate between the different DES by clicking the left or right arrows (Current condition, historical data, yearly comparison) |
| 3  &  9 | Data type label and  Drop-down menu | “Data type:”(#3) is only an indicator to tell the user to choose a data type in the drop-down menu (#9).  The system will then set the data type argument to that specific one and add it to the API call function and payload .  A drop-down menu is easier for user to know which kind of data type is available to choose from as they don’t have to guess by typing multiple test input. |
| 4 & 8 | City label and City name | “City:”(#4) is only an indicator to tell the user to enter a valid city name in the input box(#8).  The system will check if the city name exists, if it does, the system will then set the city name argument to that specific one and add it to the API call function and payload . |
| 5 | Today | This element indicates the current date of the weather data (ex: 2024-09-12). The user can’t interact with it.  When the DES current condition is chosen from the top command interface , it gets the current data to display to the user.  I don’t know if I will need a specific library for it yet or if there is a built-in function in python to track data. |
| 6, 7, 12 | Chat , send button and chat input box | The user can type questions or anything in the chat box input.  After typing a message chat box in the input the chat box, the user clicks the “Send” button to submit their message. The message should be stored either on the json online service or in dictionary.  The message will be displayed in the chat box with their name as argument and maybe time if a built-in function or library is available in python for it. |
| 10 | Chart | The chart is populated by data based on the city and data type. The user can see the chart but can’t interact with it.  (As I don’t know if it is possible with math.plot lib to zoom, download the plot as PNG as I know it is possible with plotly). |
| 11 | Quit button | This will close the app. |
| 13 | Fetch weather button | This is a button that triggers the API call to populate a chart with data related to the city name and data type.  If there is no city name , an error message will be displayed to the user stating : “Need a city name” .  For the data type, by default it will be temperature so there is no error with this element. |

## DES two – Historical data

A screenshot of a chat

Description automatically generated

|  |  |  |
| --- | --- | --- |
| No. | Elements | User action/interaction |
| 1 | Title | This static title tells the user which DES they are on. The user cannot interact with it. |
| 2 | Navigation buttons (left/right arrows) | The user can navigate between the different DES by clicking the left or right arrows (Current condition, historical data, data comparison) |
| 3 & 5 | Data type label and Drop-down menu | “Data type:”(#3) is only an indicator to tell the user to choose a data type in the drop-down menu (#5).  The system will then set the data type argument to that specific one and add it to the API call function and payload .  A drop-down menu is easier for user to know which kind of data type is available to choose from as they don’t have to guess by trying multiple test input. |
| 4 & 6 | City label and City name | “City:”(#4) is only an indicator to tell the user to enter a valid city name in the input box(#6).  The system will check if the city name exists, if it does, the system will then set the city name argument to that specific one and add it to the API call function and payload . |
| 7 | Year label and year’s input box | This is an input box where the user needs to enter the year he wants to get data from and get it displayed in the chart. The user can’t interact with the label. |
| 9, 10, 12 | Chat box, send button, and chat input box | The user can type questions or anything in the chat box input.  After typing a message chat box in the input the chat box, the user clicks the “Send” button to submit their message. The message should be stored either on the json online service or in dictionary.  The message will be displayed in the chat box with their name as argument and maybe time if a built-in function or library is available in python for it. |
| 8 | Chart | The chart is populated by data based on the city and data type. The user can see the chart but can’t interact with it.  (As I don’t know if it is possible with math.plot lib to zoom, download the plot as PNG as I know it is possible with plotly). |
| 11 | Quit button | The user clicks this button to close the app. |
| 13 | Fetch weather button | This is a button that triggers the API call to populate a chart with data related to the city name and data type.  If there is no city name , an error message will be displayed to the user stating : “Need a city name” .  For the data type, by default it will be temperature so there is no error with this element. |

## DES three – Yearly comparison

A screenshot of a chat

Description automatically generated

|  |  |  |
| --- | --- | --- |
| No. | Elements | User Action/Interaction |
| 1 | Title | This static title tells the user which DES they are on. The user cannot interact with it. |
| 2 | Navigation buttons (left/right arrows) | The user can navigate between the different DES by clicking the left or right arrows (Current condition, historical data, data comparison) |
| 3 & 5 | Data type label and Drop-down menu | “Data type:”(#3) is only an indicator to tell the user to choose a data type in the drop-down menu (#5).  The system will then set the data type argument to that specific one and add it to the API call function and payload .  A drop-down menu is easier for user to know which kind of data type is available to choose from as they don’t have to guess by typing multiple test input. |
| 4 & 6 | City label and City name | “City:”(#4) is only an indicator to tell the user to enter a valid city name in the input box(#6).  The system will check if the city name exists, if it does, the system will then set the city name argument to that specific one and add it to the API call function and payload . |
| 7, 9, 12 | Chat box, send button, and chat input box | The user can type questions or anything in the chat box input.  After typing a message chat box in the input the chat box, the user clicks the “Send” button to submit their message. The message should be stored either on the json online service or in dictionary.  The message will be displayed in the chat box with their name as argument and maybe time if a built-in function or library is available in python for it. |
| 8 | Chart | The chart will display a comparison of weather data between two selected years for the chosen city and data type. The user cannot directly interact with the chart. This chart will have two line to be able to compare the data between the two different years. |
| 10 | Years input box. | This element is for the user to enter their chosen years to compare data with .In this example on the prototype, I used 2002 and 2021. |
| 11 | Quit button | This button will exit the app. |
| 13 | Fetch weather button | This is a button that triggers the API call to populate a chart with data related to the city name, data type and year selected  If there is no city name , an error message will be displayed to the user stating : “Need a city name” .  If there is no time(year) was selected or only 1 , an error message will be displayed to the user stating : “Need to select 2 different years to compare each other” .  For the data type, by default it will be temperature so there is no error with this element. |

References:

Matplotlib Developers. (2024). matplotlib.pyplot.subplots. Matplotlib Documentation.

<https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.subplots.html>

Python Software Foundation. (2024). datetime. Python Documentation.

<https://docs.python.org/3/library/datetime.html#module-datetime>

PySimpleGUI. (2024). Creating the Window. PySimpleGUI Documentation.

<https://pysimplegui.readthedocs.io/en/latest/#creating-the-window>

PySimpleGUI. (2024). expand\_x and expand\_y Parameters. PySimpleGUI Documentation.

<https://pysimplegui.readthedocs.io/en/latest/#expand_x-expand_y>

PySimpleGUI. (2024). Multiline Element. PySimpleGUI Documentation.

<https://pysimplegui.readthedocs.io/en/latest/#multiline-element>

PySimpleGUI. (2024). Push Element. PySimpleGUI Documentation.

<https://pysimplegui.readthedocs.io/en/latest/#push-element>

PySimpleGUI. (2024). Window. PySimpleGUI Documentation.

<https://pysimplegui.readthedocs.io/en/latest/#window>

PySimpleGUI. (2024). Keypad Entry. PySimpleGUI Documentation.

<https://pysimplegui.readthedocs.io/en/latest/cookbook/#keypad-entry>