Weather App User Manual

Kyle Chestnut, Caleb Newman, Trey Coleman, Alan Huebschen

November 27, 2022

Installation

Simply download the executable file and place the file in your desired location, the only requirement is that you be running the x64 version of Windows 10. The executable can be run as-is. 300MB of storage required.

Usage

To start the forecast app, you will need to double click WeatherApp.exe. Something to note: there is a significant startup time when starting the application. The app will allow you to search the 5-day forecast of a location. The program will detect the user's current location based on their external IP address. When the program is first ran, the program will display the weather based on the estimated location of the user's IP. If the user would like to view the weather at a different location, they may enter the name of a city or a zip code in the text input box and click the "Get Weather" button at the bottom. Using the zip code of the desired location is recommended to ensure finding the correct location.

1 Bugs and test cases

Test Case #	Input	Expected Output	Rationale	Pass/ Fail
1	Put in Very long text into prompt. Ex. Copy and paste hamlet	Catches the long output without crashing. Returns no town found.	We should limit the user to only input what is necessary for cities.	Pass
2	Check for cursor blink throughout tests.	No output for this.	The user interface should not appear buggy.	Pass
3	Drag and drop long text	Catches the long output without crashing. Returns no town found.	While copy and paste is allowed for longer city names. We should make sure that the input is valid.	Pass
4	Check with different language options	Works normally, returning nothing if not recognized, or returning weather of location if accounted for by API	If users do not primarily use the english language we should account for whatever language they speak and write in.	Pass

5	Hold down a single key for an extended period	Returns no town found.	We should only allow valid cities.	Pass
6	Try to input an image in multiple fashions (different file types, drag and drop, copy and paste.)	Returns no town. Might just reject the input.	The text box should only allow for text input.	Pass
7	Test for null/ blank	Does not search, or no town found.	Users may accidentally delete their city entry before getting their forecast	Pass
8	Max character test	Catches that there are more characters than the box allows, or caps the amount of characters when put into the box.	There should not be a way for the user to "overflow" our program or the API we are using so a max length on the text box will be added	Pass
9	Symbols of varying origin (Alt key symbols, characters from different alphabets, numbers, symbols, numbers, etc.)	Works normally, or returns default value.	Some cities may contain letters not a part of the English alphabet. As long as the city name is valid the city should be returned.	Pass
10	Ascii characters.	Ascii is not supported and therefore should return an error.	The user should only be able to enter valid text.	Pass
11	Control characters	Control characters are not supported and therefore should return an error	The user should only be able to enter valid text.	Pass
12	Boundary values test	If the city is valid and is within the boundaries that have been set then we return the city. If not we return an error	The boundary values that were set for the text box should not be violated.	Pass
13	Are the characters properly visible in text input?	Normal output. Should be able to see text input.	The user should clearly see what city they are attempting to input.	Pass

14	Basic city name test	Returns the town name and weather	The user should be able to enter a town name and get the weather for that town.	Pass
15	Basic zip code test	Returns the town name and weather based on the zip code provided.	The user should be able to enter a zip code and get the weather for that location.	Pass
16	Check for pasting unusual characters in the text field.	Should not crash or glitch.	With unusual input the program itself should not crash and just return an error that a city cannot be found.	Pass
17	Enter spaces in the prefix and suffix of what you type.	The program handles spaces before and after the town name, and return proper weather results	As long as the name of the town is valid, it should return the results.	Pass
18	Does any elements of the program change position through repeated use of the program?	No output. Just watch.	The user should be able to input as many locations as they want in a singular run of the program. Although it will be done one at a time, the location of each component should be static and the information should not deviate from its original position.	Pass
19	Check if repeatedly pressing the forecast button many times crashes the program or glitches it out. (do this with both an empty field and non empty field)	The program will function every time without crashing.	If the user is clicking the "Get Forecast" button repeatedly the program should function no differently.	Pass
20	Check for only spaces as input.	Returns no city found because spaces are not a valid input for a city.	We want to account for most if not all cases of user input.	Pass
21	Does copying from the text box work?	No crash or error.	A user should be able to copy and paste into the text box.	Pass
22	Enter a mix of numbers and characters into the text box.	Return no city found	A user should be allowed to enter a city or zip code but not a combination of the two.	Pass

23	Enter a city name with a space in the middle of the name. Example: Spring field	Return no city found	A user may accidentally space in the middle of a city name without noticing. We want to be able to catch that error	Pass
24	Enter a city name with a space in the middle of a name. Example: San Jose	Return the proper city and weather.	A user may enter a city that specifically has a space in between two words.	Pass
25	VPN and proxy	Should return the weather based on the default I.P.	Users may have an active VPN or proxy server on their machine.	Fail