NYU INFO 2335 "Freestyle" Project
Planning and Requirements Document
Jessica (JiaJia) Zhang (jjz266@stern.nyu.edu)
2019-07-03

Tomorrow's Forecast Reminder

Background

Weather forecast is like fortune telling for me—I seesaw between completely relying on it to drive my decisions, or highly skeptical of what I am being told and often betting against the forecast and wishing for the best. Although I have an all-powerful iPhone with multiple weather applications and weather forecasting capabilities, I lack the second nature to check the weather app on a daily basis. Therefore, I have many times arrived at the office in the morning, only to find out it's raining in the afternoon and have to buy a new umbrella. Or worst, New York City weather deceives me into thinking it's just a cold Winter day, only to find out it's snowing by 5PM and I have to find my way home in heels; hence, adding an element of surprise to my day.

But in reality, I don't appreciate being caught in these kinds of circumstances. And so, I thought how great would it be if I had someone to remind me of the next day weather so I don't have to later hate myself for not doing this simple thing as checking the weather app?

Problem Statement

User Needs

1. As a busy Langone student, I would like to be reminded of next day's weather, so that I can always be prepared for the weather condition.
2. As a smartphone user, I would like to be actively reminded of tomorrow's weather, so that I can prepare and pack for the next day.
3. As someone who has way too many umbrellas at home, I would like to be reminded of the weather, so that I can stop spending money on more umbrellas.

Current State Processes

Collecting weather forecast is a conscious decision that I have to make every day. To do that, I utilize a mixture of my smartphone, Amazon Alexa and my smart watch, where I have to intentionally navigate to the weather app/widget or ask Alexa for the forecast.

When using my smartphone, more often than not, I either get distracted by a text message or I navigates straight to my Instagram, simply out of habit. And through these distractions, I forget to check the weather all together and hence brings me back to the initial dilemma of not knowing tomorrow's weather forecast.

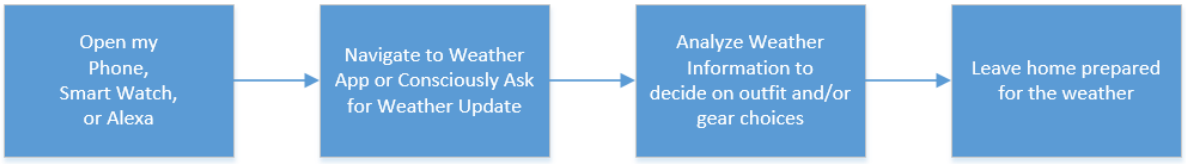


Figure 1 - Current Process Diagram

Proposed Solution

The goal of my proposed solution is to remove the intentional aspect of this weather checking “chore”. Rather than actively pulling/seeking this information from a weather app/widget or Amazon Alexa, the end user of my program can more intuitively consume this information in as a SMS text message.

System Objectives

My program is called “Tomorrow’s Forecast Reminder” (a.k.a. TFR). Its job is to check next day’s weather forecast, provide an outfit and/or weather gear recommendation, and finally send this recommendation to the end user via SMS text message. And even when there is no outfit or gear recommendation, my program should still send a text message notifying the end user of the temperature and normal weather conditions.

Future State Processes

My program should remove the need to actively navigate to a weather application to collect forecast information, at the same time, my program offers an additional layer of improved user experience by feeding the end user outfit recommendations.

With this service, the future state process is simplified into a single step: Receive a notification in a SMS text message form with tomorrow’s weather forecast and outfit/gear recommendation. Basically, relieving the end user of the need to proactively seek out the next day’s weather forecast.

Figure 2 - Future State Process Diagram

Functionality Requirements / Features

Collecting End User Zip Code

REQUIRED: The system should prompt the end user to input a zip code as the search criteria. It should only accept a numerical value, in standard zip code format. If in the case that the inputted zip code does not correspond to a valid location, then the end user is prompted to enter another input. The system should continue to prompt for a zip code until a valid one is inputted.

FUTURE SCOPE: The system should allow the end user to enter another search criterion other than zip code, for example, the end user can input a city name.

Store Sensitive Information in a Secure File

REQUIRED: The system should store credentials, such as API Keys and Token(s), in a ".env" file that protects privacy.

REQUIRED: The system should store private information, such as the Sender's and the Recipient's (End User) phone numbers in a secure and private location (".env" file).

FUTURE SCOPE: The system should allow the end user to input new and additional Recipient's phone numbers, so that more than one user can use this service at a time.

Read and Digest Weather Forecast Information

REQUIRED: The system should request and receive the weather forecast information from the OpenWeatherMap API.

REQUIRED: The system should parse the weather information to arrive at a recommendation based on the weather condition(s) and temperatures of the next day.

Send Forecast Notification

REQUIRED: If all of the above conditions are met, the system should send the end user a succinct SMS message of tomorrow's weather forecast.

REQUIRED: The SMS message should include city name, weather condition and temperature range for the next day.

FUTURE SCOPE: The system should allow the end user to input a Recipient phone number so that it allows for flexibility and transferability between different users. For example, the end user can input a different phone number to receive weather forecast for a different zip code.

(Note: This capability is limited by the fact that Twilio has a verification process for each new recipient number.)

FUTURE SCOPE: The system should allow the end user to communicate with the service via SMS messages, so end user can input a different zip code or ask for additional information on the weather condition.

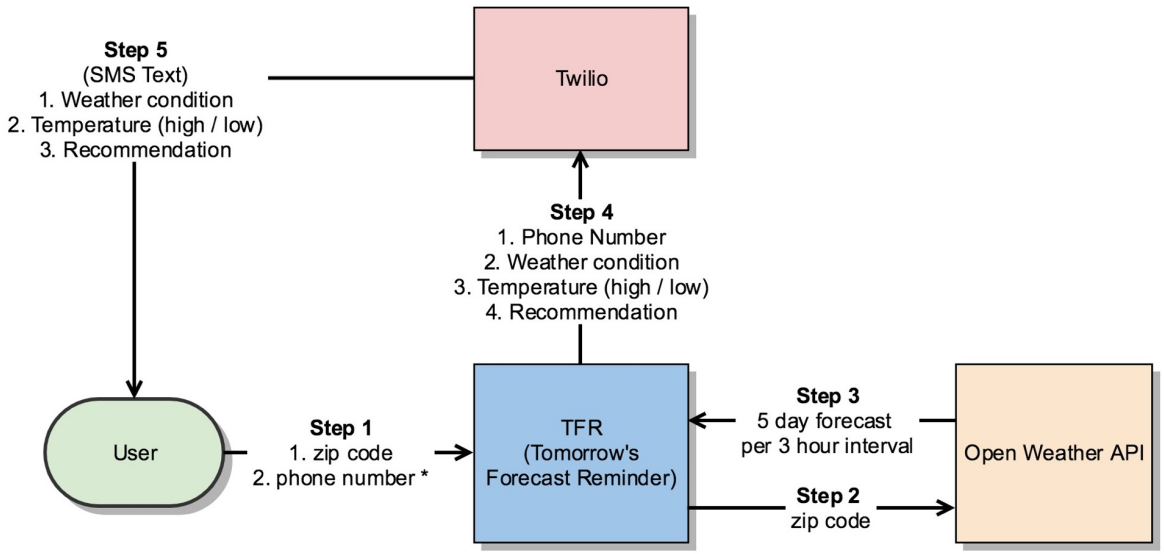
Information Requirements

Information Inputs

1. End user's **zip code**, which currently only covers U.S. zip codes.
2. **OpenWeatherMap API's 5 day / 3 hour forecast data**, which includes the weather forecast information for every 3 hours of the next 5 days in JSON format. Fields of interest include UTC values or range for tomorrow's date, main weather condition, and temperature.
3. **OpenWeatherMap's API key** in order to access OpenWeatherMap data.
4. **Twilio API Key and Token** in order to access Twilio Programmable SMS capabilities.
5. **Sender and Recipient numbers** in order to know who to send and who to receive SMS.

Information Outputs

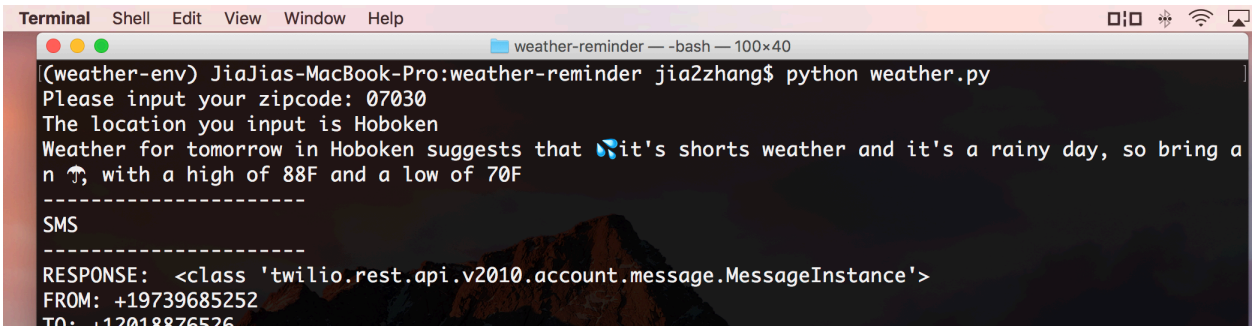
1. **Next day's weather forecast and condition**, along with outfit & gear recommendations, sent by SMS that contains emoji(s).

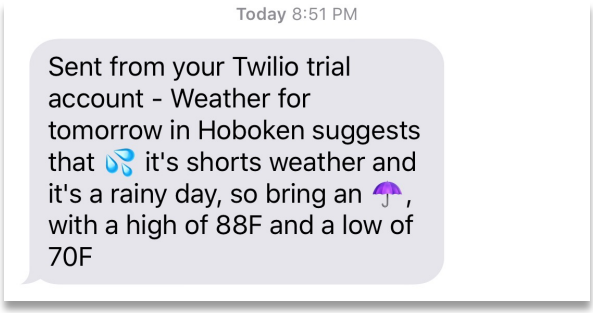


Interface Requirements

The system currently requires the end user to run the program via the command line, and then the end user will receive the SMS on his or her phone.

In the future, I hope to improve this system so that it stores a dictionary of zip code and recipient number in a one-to-one format, so that I can schedule this program to run on a daily basis and reference this dictionary to see who wants the weather forecast for which location.





Technology Requirements

The system requires the OpenWeatherMap API, where the input data is then downloaded and processed to derive a weather forecast and recommendation.

The system also requires the Twilio API in order to interact with the Python application to send SMS messages to the end user.

In order to leverage these two APIs, the following modules and packages were used:

- (Necessary) “python-dotenv” package to store secret credentials in an environment variable
- (Necessary) “requests” package to help download weather information from an HTTPS
- (Necessary) “zipcodes” module to help ensure inputted zip code abides by zip code format
- (Necessary) “twilio” package to send SMS notifications
- (Necessary) “pytest” package to implement automated tests for quality control

- (Necessary) “os” module for directory operations
- (Necessary) “json” module to handle JSON format data
- (Necessary) “datetime”, “calendar” and “time” modules to handle UTC to standard date time format and conversions
- (Helpful) “emoji” package in order to insert emoji(s) in SMS message
- (Helpful) “pprint” module to customize the format of print statements

Development Plan

For the development of this service, I plan to split up the work in the following 3 increments:

Collect Weather Forecast Data
First, I need to ensure I can connect to the weather forecast database using the OpenWeatherMap API. And immediate after connecting, I need to make sure the information I need are in the database and I can import and digest this data to provide a recommendation.

Form Recommendation Logic
Subsequently, I plan on creating a recommendation logic based on the weather forecast data, at which I will write conditional logic to then return different outputs.

Send Weather Notification
Finally, if all the above goes well, I plan on utilizing the Twilio Programmable SMS capability to send the recommendation directly to the end user's phone.

(If time permits)

1. I plan to allow the end user to input their recipient phone number so that it is less hardcoded and more interactive via the command line.
2. I also plan to deploy this service to a server using Heroku and have it scheduled to run every evening, so that I don't have to manually execute this Python program on the command line.
3. And finally, I plan to scale this service, so that there is a dictionary of zip codes and recipient phone numbers that my program can loop through to send SMS messages to all the people contained in the dictionary.