Daily Debrief Information Service Requirements

Table of Contents

Problem Statement

Keeping updated on each news update, sports score, stock price, or even weather prediction often seems like a full-time job. For someone who has a busy schedule, this routine task becomes time consuming or even completely unmanageable. In a world full abundant information everyone should be able to know exactly what is going on wherever they wish. For this reason, it is clear that there is a need for a quicker process for news information acquisition.

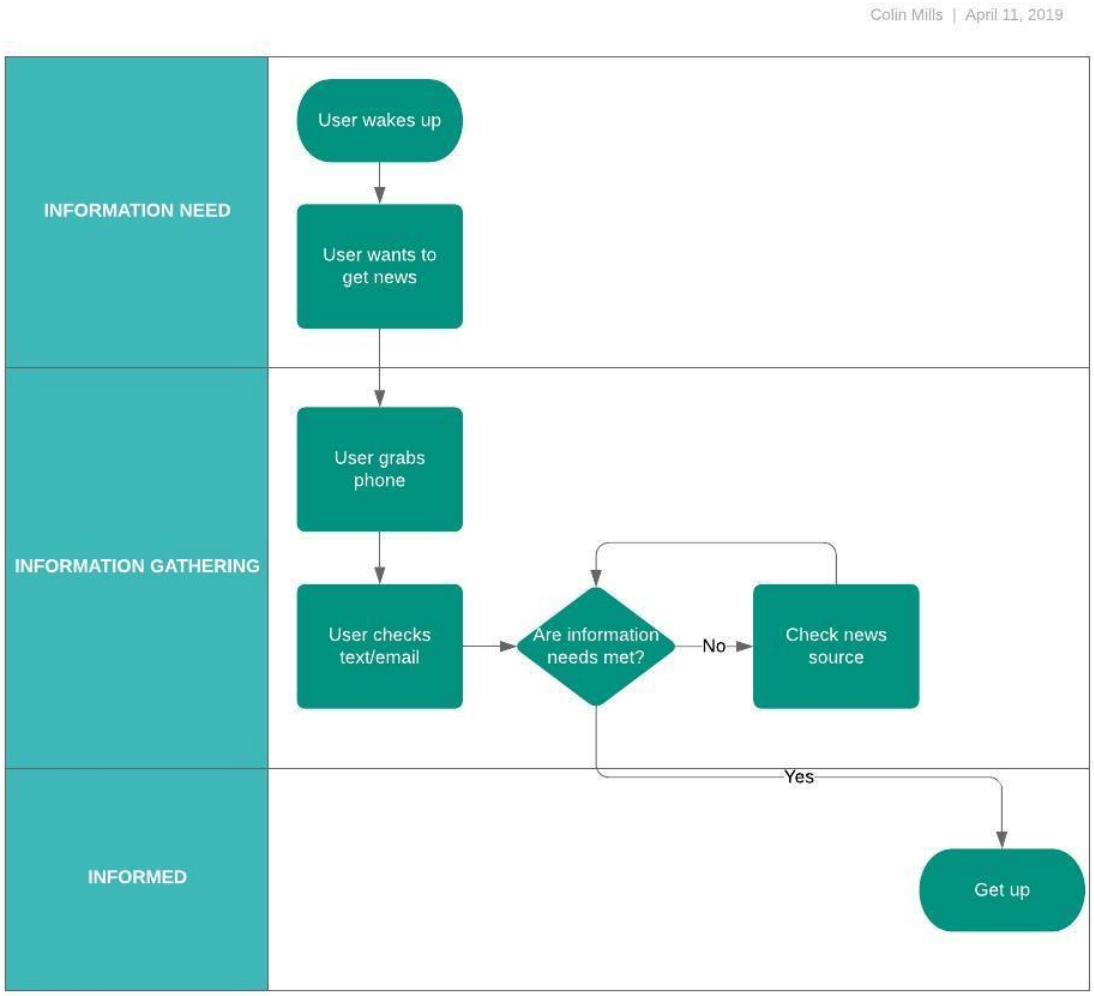
User Needs

This is a problem for those who wish to stay educated on each of their interests but do not have time to spare. As person who likes to stay educated on various subjects, I spend a lot of my time switching through different apps and websites to gather the updated news. In retrospect this process of "keeping up" is nothing more than repeatedly and manually checking each individual site with information that is pertinent to the user.

The user needs a service that provides this information to them in a centralized location. Through this method they do not have to spend their valuable time visiting numerous sites, nor is there the user error of failing to remember to check certain sites.

Current State Processes

This diagram displays the current state process that a user goes through in the status quo.



Pain Point

In the current process the user is faced with a pain point of repeatedly checking news sources one at a time until their information needs are met. This process loops indefinitely, with each step adding to the amount of time it takes the user to get up.

Potential Solutions

Status Quo:

The status quo is not an undesirable aspect to many users, especially if they have low information needs or an abundance of time to spend. In this case the user continues to check each news sight themselves in a routine that is merely habitual. This method allows for variability in information needs, depending on what is going on at the time. On the other hand, this method is time consuming and tedious. Additionally, through this manner the burden of checking every sight the user wishes to see falls ripe to human error. In this instance the user might forget to check certain sights at times, causing them to be less educated on a subject.

Push notifications:

Through this method the user could receive notifications on their phone from each different source, eliminating the possibility of forgetting to check one. This allows the user to place a higher degree of burden on a technological solution and provides them with snippets of information throughout the day. The downside to this solution is that these notifications come inconsistently, causing the user to have to spend time throughout the day checking these. Also, the information is still scattered amongst sources, which are messy and scattered to the end user.

Watch daily news:

This is a method that many people have elected to choose for decades. It is easy as the user simply has to turn on a TV or other video stream and listen as they get ready. Furthermore, the user does not have to spend time gathering information from various sources or researching as this has been done for them by a team for that news channel. However, many of these news channels cover their information over the course of an hour airtime, which is much more time than many busy users have to spend in the morning. Additionally, although the channel might cover a slew of topics, this does not let the user elect their own scope of information.

Ask educated friends:

Another solution is to avoid news sources altogether and to leave the burden of educating oneself on friends who tend to be educated in those subjects (i.e. sports guy, politics friend, finance friend). This method is less than desirable as it leaves the burden to humans to inform you as a secondary source. As a secondary source it is possible that they are biased, outdated, or even incorrect in their information as well as the fact that this requires timely interactions with each one.

Daily Debrief Information Service:

This system would allow the user to select their news information preferences and compile it into a single environment. Through this manner the user both has volition over their information output as well as a centralized location to view this in. This system will be limited by the developer's ability to add news sources for selection as well as the ability to provide graphics that are available on these news sources.

Proposed Solution

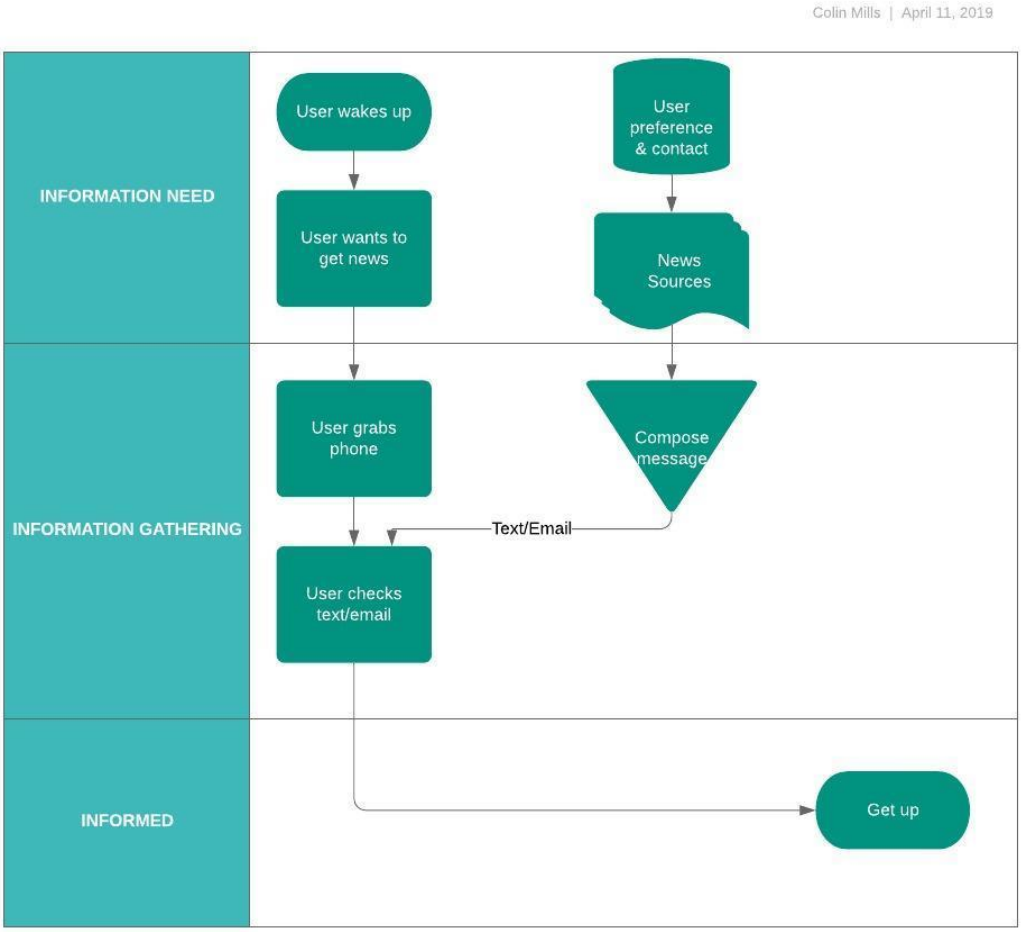
The proposed solution is called “The Daily Debrief” which will gather a user’s contact information and news preferences on a user available website and provide the service of parsing the most important news stories for each user which will be sent to them each morning. For certain news sources such as stocks and weather the information will be provided directly, whereas in other situations the user will be provided with the most important links for their information needs.

Hypothesis

This process will simplify and expedite a user’s daily routine through automating a repeated manual routine allowing the user to be more educated on current topics in less time.

Future State Processes

This diagram displays the future state process that a user goes through with this service.



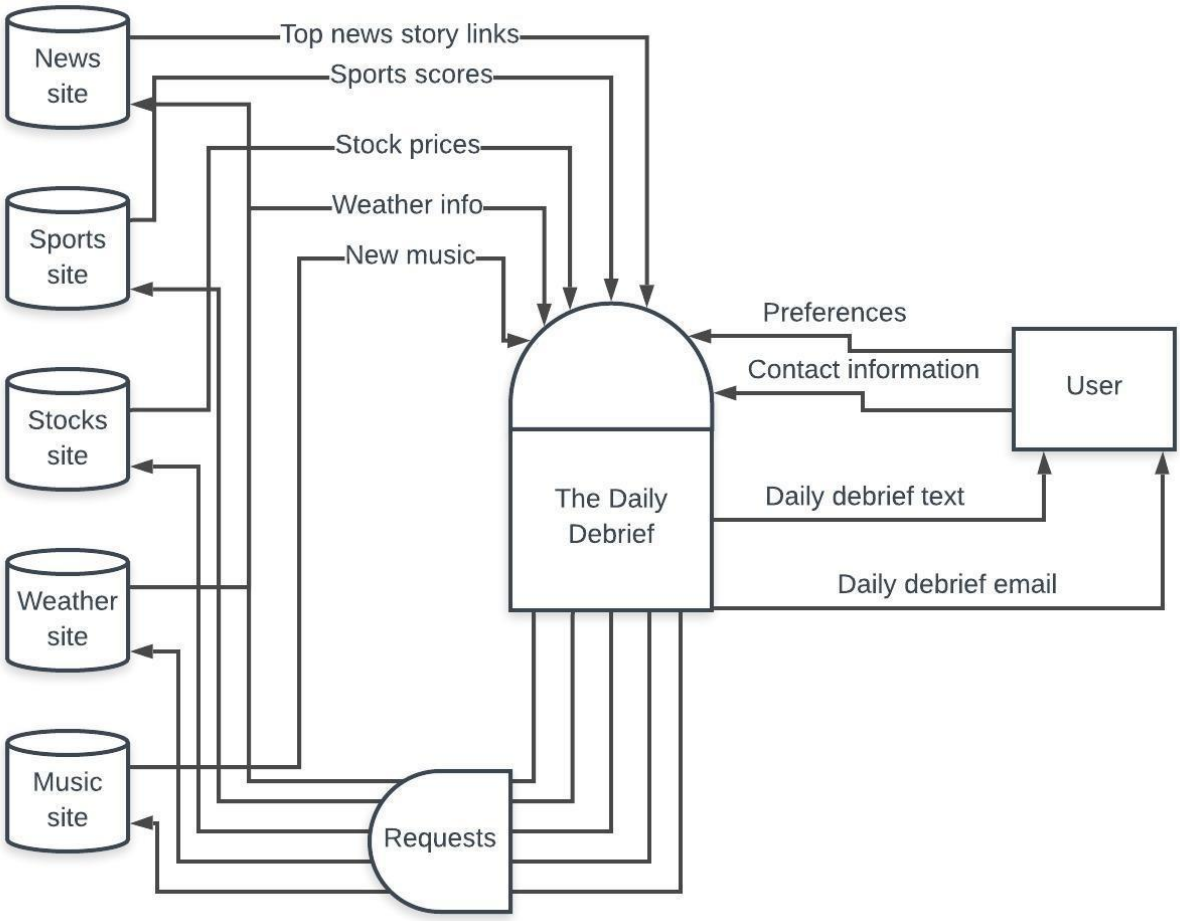
Simplification

In this future process the compilation of news and information is done automatically and sent to the user through a route that they already would have elected. This simplifies the process on the user end as well as saving them the time of looping through multiple sources.

System Objectives

This system's primary objective is to provide a service that integrates all users' news sources into a single environment through a daily text or email. This will include but not limited to top daily news stories, desired stock information, weather reports, clothing recommendations, and sports scores. The information output will depend on the user input on an external website through which they provide which news sources they would like information from and their preferred contact information. Through this the user will benefit from the simplicity and time savings of interacting with a single environment.

Information Requirements



Information inputs: User contact and preferences, news from various API's
Information Outputs: A text or an email

Functionality Requirements

The Daily Debrief will be a service that the user will have access to through a user facing website. This website will prompt the user to fill out and submit a form of contact information and preferences. Next this form will update a data store within a google sheet which holds all users' information. This information will be protected through two factor authentications to access the developer's google account. Next, a remote server will run a python script, initiated by a scheduler at 8:00 AM in the morning. This script will initiate a chain of functions to call upon the data store, access each news source accordingly, and compile an individual message for each user. This information will then be sent to its respective users.

Sign-Up Process:

This part of the process requires the user to navigate to the Daily Debrief's website. Once there they will select if they prefer to receive a text message or an email and the respective contact information. Next, they will have a list of news options that they are able to opt for; they may choose one, many, or all. Once submitted, this information will then be submitted as a new entry in a google sheet using the python script, submit data. Each row will begin with the name, followed by their contact preferences and a column for each news source with a binary system (0,1) marking which news sources they which to receive.

Data Fetching:

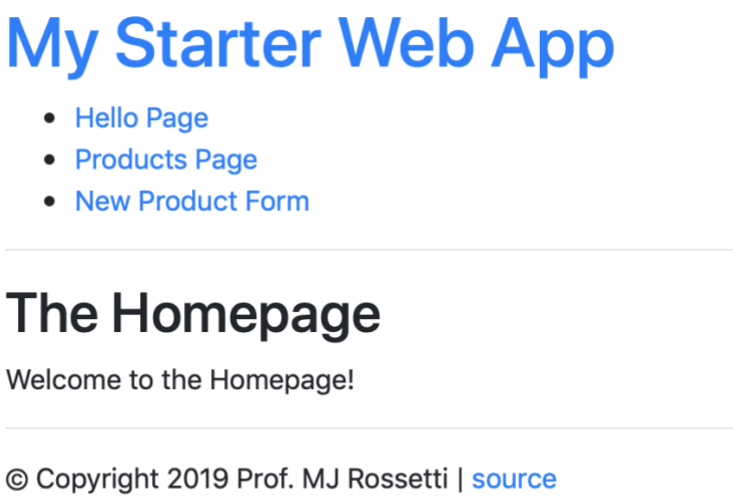
Heroku will call the python script, daily debrief, which will be triggered by Heroku Scheduler at 8:00 AM every day. This script will read each user row of information into a list of dictionaries which it will use to choose which contact method to pursue and which information to collect. This information will determine which python functions need to be called to collect and send each debrief. These functions will be in separate files consisting of send_text, send_email, as well as a function for each news source that is named after its respective site.

Message Sending:

The script will finally call either Twilio or SendGrid to send this information, stored in a formatted string of data. Each call will send an individualized message to the user. The user will at whatever point, wake up and check their phone. Seeing that they received this message they will read and inform themselves.

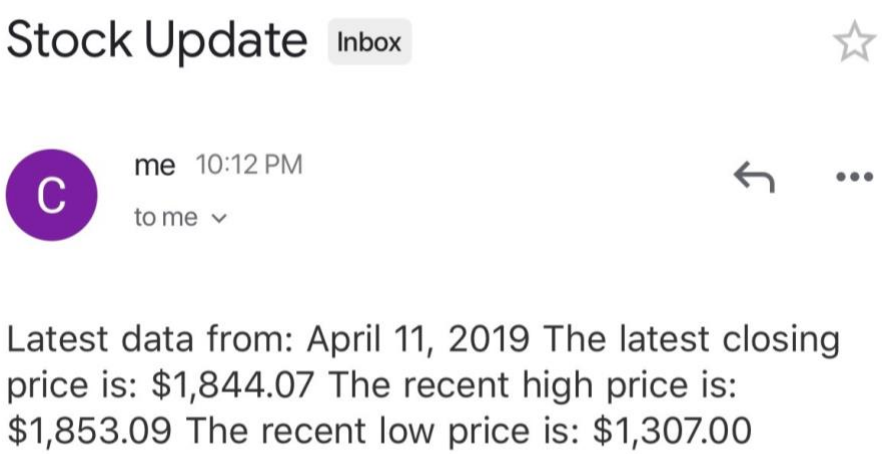
Interface Requirements

The user will interact with the system at two points. The first will be through the website in which the user submits their information. Below is an example website that will be similar to the website that will be created for the user. There will be a home page with a link to another page in which the user can submit a form.



**Credit to Professor MJ Rossetti for the website example

The second point will be in the user's own text messages or emails. They will receive an email according to their preferences in their preferred form of contact as seen below. Below is an example of a user who is receiving stock updates through email.



Technology Requirements

APIs: The system will use a variety of APIs to access a datastore and to collect information.

- The API needed for the datastore will be google drive's API. This will allow the python script to edit the google sheet.

- Next, the information APIs: NY Times, CBS, NPR, MSNBC, Fox News, CBS sports, Alpha Vantage, Open Weather Map, and Spotify. These will be used to gather and process news information.

Packages:

- Flask: Helps build the framework for a website with many features including buttons and drop-down lists.

- Dotenv: Keeps local secret variables secret by allowing the developer to store these variables in a separate .env file

- SendGrid: facilitates the sending of emails

- Twilio: facilitates the sending of texts

- Requests: Allows scripts to pull, post, and delete information from APIs

- Json: Allows for the interpretation of data formatted in CSV files

- Pytest: Allows for continuous integration testing during development

Languages:

- Python
- HTML