# Executive Summary – Cartpool Lite

By Seth Blumer and Jordan Mbanefo

## ***High-Level Overview***

Cartpool Lite is a Python program designed to ease the grocery experience. It stems off an idea had to help grocery shoppers have an efficient and effective way of completing the tedious task of grocery shopping.

Cartpool Lite allows users to create a virtual shopping list by selecting from a database of items commonly found in grocery giants. From there, the application displays a variety of grocery stores within a user-desired proximity, ultimately allowing users to mix and search based off preferences.

## ***Limitations and Future Functionality***

Due to the constraints of time and access to information, Cartpool is categorized as a Lite version because its functionality is currently limited. However, the program is orchestrated and constructed in a way to be flexible in adding functionality going forward, and particularly if we were given access to private information. These functionalities are displayed below:

**Item Prices and Quality by Store –** Cartpool would use information from grocery items based off a food nutrient database. With access to retail information, the database would grow larger, and give access to prices and customer-perceived quality. Because this is where the true value of the application would effect the most amount of people due to surveys [attached], we decided for the purpose of the assignment to display price information in general as a store.

**Budgeting –** If prices were accessible, the Cartpool algorithm would be able to take the user's added items and prioritize them in order to suit a specific budget.

**Group Pools –** With more time and access to online databases, Cartpool would have group pools where multiple people could add to a grocery list, I.e. a mother could lead the group pool and her children and spouse would be able to add to one list.

**Item Customization –** With access to price and quality information, Cartpool would allow users to create different preferential functions for grocery items. For an example, a user would be able to choose the *cheapest* milk with the *highest quality* cereal, and Cartpool would display the results accordingly.

**Preset Lists –** With more time in for the assignments, Cartpool would allow users to save specific preset grocery lists, so rather than adding individual items one at a time, the application would display a preset grocery list.

**Item Duration –** After consistent use of the application, with the correct understanding of machine learning, the application would be able to understand an average of how long a user uses an item. They could then suggest that item on the grocery list in the future. For an example, if a user orders milk every

two weeks, if the user forgets to add milk to the list, the application could prompt a suggestion to buy milk.

**Coupons –** With partnerships with retails, Cartpool would be able to get rid of physical coupons all together, and suggest deals to users based off of what they add to their list.

**Aisle Routing –** With information of retail aisles, the list could be displayed in order of the aisles from the entrance of the shop to the exit. This would obviously change from shop to shop.

**Optimum Shopping Time Generation –** With information of both GPS parking capacity, Cartpool could suggest the most optimum time to buy and give a range of time it would take the user to travel and return on a shopping list. With Covid-19, keeping track of parking usage and using geolocation would give an estimation of the amount of people in the shop so that social distancing is easier to adhere to.

**Communal Suggestions –** If an item is difficult to find, or if the item is not found in the spot suggested, users would be able to let the algorithm and other users know that the item was not found, or found in a different location. The algorithm would then suggest users to check another location.

**Sold out substitutions –** If a user was not able to find an item, the algorithm would have a way to suggest a similar item to try and satisfy the customers.

**Recipes and Joint Purchases –** With either partnering with recipe sites, or allowing users to create and upload (or keep private) recipes would allow users to avoid having to go an add items one by one. Similarly, if users have a social gathering or outing, they could add and find an estimate on the amount of items and the price associated to pay for each person at the gathering.

**Split List –** With the group pools, there could be specific ways of stratifying how to pay for the items on the grocery list. Users could pay relative to what they added to the list, or split the costs in total. For our mother example, the mother could take full responsibility for the payment. It is important to notice that Cartpool is not designed to *pay* for the groceries, but rather provide an estimate on how much would be needed.

**Linking Virtual Currency Applications –** Once again, not responsible for paying, Cartpool could be linked to money sharing applications. All a user would have to do would be to press a button, which could request another member of the group a payment.

# Technical Overview

Application Cartpool Lite was written in Visual Studio Code using Python3.

## Attributions / Modules

The application owes attributions to the following sources: our brains.

**Food Database Source: https://corgis-edu.github.io/corgis/csv/food/**

**APIs Used:**

- Distance Matrix API (Google Maps)
- Geocoding API (Google Maps)
- Places API (Google Maps)

**Modules Used:**

- Pandas
- NLTK
- Tkinter
- Requests
- JSON
- Google Maps
- Re
- Urllib.parse

## API Access

The application is dependent upon use of a APIs providing real-time data. To use these APIs, access keys are issued by the API providers. The following is a list of the APIs on which this application is dependent as well as the required access keys. Included in the listing is where the access keys can be retrieved, which keys are necessary and the code from this application that will require modification to apply updated keys. In addition, any known constraints about the API have been listed.

| **Distance Matrix API (Google Maps)**<br>Retrieved from https://cloud.google.com/maps-platform/maps after first creating a Google Cloud account | | |
|---|---|---|
| Create an application and the provided with the following:<br>• API Key | API key set as global variable | Limitations to the use of the API include:<br>• Maximum of 25 origins or 25 destinations per request.<br>• Maximum 100 elements per server-side request. |

| | | |
|---|---|---|
| | | • Maximum 100 elements per client-side request.<br>• 1000 elements per second (EPS), calculated as the sum of client-side and server-side queries. |
| **Geocoding API (Google Maps)**<br>Retrieved from https://cloud.google.com/maps-platform/maps after first creating a Google Cloud account | | |
| Create an application and the provided with the following:<br>• API Key | API key set as global variable | Limitations to the use of the API include:<br>• Pay-as-you-go model that costs approximately half a cent per request (nullified by free credits)<br>• Rate limit is 50 requests per second (QPS) |
| **Places API (Google Maps)**<br>Retrieved from https://cloud.google.com/maps-platform/maps after first creating a Google Cloud account | | |
| Create an application and the provided with the following:<br>• API Key | API key set as global variable | Limitations to the use of the API include:<br>• Some of the more specific details about places are subject to a pay-as-you-go model, but the basic data we extracted was free<br>• Rate limit is 100 requests per second (QPS) |

# Documentation (i.e. end-user training materials with screen shots)

## *User Guide*

**Step 1) Search a grocery item at the top of the interface. Item results will be displayed in result box [1a]**

**Step 2) Modify your grocery shopping list by typing in the item ID and adding, deleting, or clearing items with their respective buttons. Items in list will be displayed in result box [2a].**

**Step 3) Enter in your zip code, so the application can tell your location.**

**Step 4) Enter in the desired distance (in miles) that you would be willing to travel to a grocery store. Grocers will be displayed in result box [4a].**

**Step 5) Click the Find Grocers button and choose your desired sort preferences. Sort preferences are displayed in [5a].**

**Step 6) Clear the Clear list button to enter in new routing information.**