Ryan Collins Andrew Sidorchuk

In this project, we aim to create a program that creates routes and delivery schedules for a restaurant with a delivery service. We will create a system that implements java graphics and framing to have a simple and appealing user interface. The program will implement the Google Maps API to create efficient routes. It will then implement the Java Mail API to send data through an email.

To begin, it should be understood what this project is not. This project is not a system a restaurant can use to keep track of orders, purchases or item prices. Rather, it can be considered supplementary software that restaurants can incorporate to their already installed systems to include delivery services. Sample data is passed in at times that would otherwise be provided by the already existing system. Employee information, such as the courier’s email addresses has been generated for the sake of demonstrating the performance of the software.

The software was designed in consideration of the courier. What kind of information does the courier need to be most efficient and successful and how can they receive that information as quickly as possible? We believe that sorting the information into three distinct containers is the best way to easily keep track of what goes where. We begin at the smallest unit, the item. This is the basically the food that the customer orders. This can be a pizza, burger or anything that’s food really. Each item contains information about the food such as the estimated time required to prepare it. The next container is the order. Every time a customer calls, they initiate an order which can contain multiple food items. At the end, a customer can have 3 burgers, 4 pizzas, and 2 fries and the order with this information will be dedicated to that specific customer. The order also contains information relevant to the customer such as their name, address, and telephone number. The last container is the load which is a collection of orders. The load is what the courier comes to the restaurant and picks up to go deliver. The load will contain orders for different customers with different food items. We believe this would be the best way to keep track and organize the purchases for the delivery driver.

To prevent a user from inputting an order with ascribing to its vital details, the item buttons have been deactivated until the address, name and phone number have been submitted. Once submitted, the program calls a method to enable the buttons for further processing. Once the load is complete, the addresses are taken and run through an algorithm using the Google Maps API to find the most efficient route the courier should take when delivering. We’ve defined the most efficient route to mean the first stop for delivery is the closest address to the restaurant. The next stop would be the next address closest to the previously mentioned address.

Finally, the program sends one email per load containing information about each order including the most efficient delivery route. This receipt displays the address, the phone number of the customer, the customer’s name and all the food items associated with that order.

The software can be modified to meet the need of the client. For example, the algorithm implemented in the code organizes the addresses in order from the address closest to the restaurant, then the next address closest to the previous address mentioned and so on. However, other algorithms could base their organization off the time it takes to travel to each destination rather than the distance between destinations. Another modification that can be included is limiting the number of orders assigned to each load.

System Requirements

|  |  |  |
| --- | --- | --- |
| Identifier | Priority | Description |
| req1 | 3 | The system will calculate the total distance between each destination point and home base – internal |
| req2 | 2 | The system will arrange the addresses based off an algorithm (closest, closest to closest, repeat) |
| req4 | 2 | The system will create a load to hold orders with specific items |
| req7 | 3 | The system will present a simple GUI for creating loads and orders |
| req8 | 1 | The system will utilize Google Maps APIs to create a delivery schedule |
| req9 | 5 | The system will utilize Java Mail API to email a receipt to a courier |

A list of terms for reference:

Home Base The restaurant the client will appoint to be the delivery pickup location

Courier: The employee that will deliver orders from home base to the customer

Route: The path the courier will take from home base to the customer

Order: The individual purchase by each customer

Load: All the orders the courier takes at the same time, a package of each individual

order

Item: Food choices that customers can add to their order

USER STORIES

|  |  |  |
| --- | --- | --- |
| Identifier | Points | User Stories |
| ST-2 | 8 | As a user, I can use the software to create a delivery schedule |
| ST-6 | 8 | As a courier, I can see which orders are in my load |
| ST-7 | 9 | As a courier, I will receive the delivery schedule information through an email |

REPOSITORY: https://github.com/collinsr729/csc380part2

UML class diagram: