Divvy Van Navigator Final Report

A Document for Divvy Van Navigation Functionalities and Information

***Prepared by Andy Hansana, Kevin Elliott, Vincent Weaver, Ryan Crowley for use in* CS 440**
at the
University of Illinois Chicago

November 2021

Table of Contents

List of Figures

List of Tables

I Project Description

1 Project Overview

The Divvy Van Navigator is an application that assists the Divvy van driver by automatically navigating them throughout the city of Chicago to each Divvy station in the easiest and most convenient pathway possible, as well as notifying the driver of which stops need bicycles and scooters repaired and picked up to be placed into the truck. This application shall be user friendly for a driver and not too distracting as this person will be focused on driving and it must be as simplistic and efficient as possible.

2 Project Domain

The domain for the divvy van navigator would be targeted towards the company Divvy as this application will be used for their drivers and the bikes labeled under their brand name. In the near future this application could possibly be optimized so that other companies would be able to implement this within their systems as the time would come.

3 Relationship to Other Documents

The relationship between the description and the requirements are in relation to this document as it finalizes all functionalities from the prototype of the Divvy Van Navigator as this document summarizes all of the given requirements such as the description, requirements, and also design.

4 Naming Conventions and Definitions

4a Definitions of Key Terms

Van: Refers to the vehicle issued to a Divvy maintenance employee, most commonly a storage haul type of van with two front seats and large storage behind them and in back.

Station/Stop/Platform: All refer to a location where divvy bikes are available to be rented by customers, change in names depends on how the location is situated in the surrounding area. For example a location in the middle of the city is sometimes called a station because it has a large capacity and more features.

Bike: refers to any divvy bicycle whether electric or standard

Scooter: refers to the electric scooters that are available through the divvy locations

4b UML and Other Notation Used in This Document

This document generally follows the standard UML diagram format, with any exceptions noted when used.

4c Data Dictionary for Any Included Models

Maintenance procedure: get next location + route to location + perform maintenance on available bikes + replace bikes and scooters as needed + update vehicle count of bikes

Restock Procedure: Pause list of stops + go to nearest restock location + update stock + return to listed location

Greedy Algorithm used for navigating drivers to each closest stop in the shortest path possible.

Traveling knapsack algorithm used for keeping track of van inventory such as bikes and scooters and prioritizing stops that fit the current amount best.

II Project Deliverables

Overall, the application that our group developed was a prototype similar to the original documentation of what the divvy van navigator would be. The application is able to show the driver directions of what divvy stop to go to and it sorts each divvy station in the chicagoland area based on the greedy algorithm where the driver puts their location marker of what stop they exactly want to go, and then it sorts it based on distance. The GUI is a simple basic visual for the driver to maintain driving while navigating his way around the area in a timely manner.

1 First Release

The functionality of the first release was getting started on how the application would look as well as designing the GUI and how it would portray towards a drivers eye. A lot of elements were thought about in terms of what needed to be displayed on the application, as well as layouts of what each task could be and where certain things could be laid out such as the map, and the stop markers. After the design process, 3 prototypes of the GUI were created eventually leading to which GUI we would be continuing on with. Also, the divvy API was created by pulling using unirest so that the information from each divvy station could be displayed towards the GUI so that the information of each stop could be shown.

2 Second Release

The second release needed to incorporate the map and how it would display onto the GUI as that is one of the most important features that this application would need to navigate the driver of where to go. Different approaches were used on how to display the map as well as experimenting with different API's such as Bing, Google, Yahoo, and Apple. Eventually, we decided that Google Maps would be most efficient and effective as it is one of the more updated API's and is friendly towards navigating with Divvy. Displaying that also required using webview and then we implemented using the greedy algorithm in order to sort the divvy stop by shortest distance possible from where the user currently was which was the inputted divvy station.

3 Comparison with Original Project Design Document

This prototype compares to the full project described by the previous group in certain ways such as implementing the application with all of its uses such as marking what bikes and scooters were needed and also being friendly towards the driver and their directions. Many key features were implemented towards this application with the maps, traveling knapsack algorithm, and overall just convenience toward the drivers ease, however there were certain features that weren't implemented such as in the full project it was stated to "keep left turns to a minimum", in order for the driver to be in the safest lane possible. After consideration of this feature, we thought that there wouldn't really be a way to implement this as the city of Chicago has construction going on 247/ as well as certain road closures, we thought that this feature wouldn't be necessary to implement as left turns don't necessarily make the driving more dangerous but keeping the driver less focused on the road and more elsewhere would be more dangerous.

III Testing

1 Items to be Tested

Items to be tested: List of stops, complete stop button, initial stop

2 Test Specifications

66# - Nearest Stop test

Description: When the user inputs an initial stop, the stops in the table should be filled with the closest stops to it, in order of distance.

Items covered by this test: Initial stop, List of stops, complete stop button

Requirements addressed by this test: Valid initial stop

Environmental needs: Valid login for driver, start location

Intercase Dependencies:

Valid initial stop is needed before the test can continue, otherwise distance cannot be determined

Test Procedures: Ensure that the driver software has valid access to all stops, then run automation script

Input Specification: Provide the automated script with the first stop that is being tested, along with the amount of stops to be checked for correctness. A custom list of stops and locations can be provided if necessary, otherwise test will run the current applications API list.

Output Specifications: Test will output stops in a list with fist on top, with first column being name and location, and second column being distance from start

Pass/Fail Criteria: Test passes if each rows distance from start is greater than the previous one, and there is not a closer one that is not included in the list

3 Test Results

66# - Nearest Stop Test

Date(s) of Execution: Tests were run multiple times with different data, locations on the north, west, and south sides of the city.

Staff conducting tests: QA Team conducted the tests

Expected Results: Expected results of test was an output of ascending distances from initial location

Actual Results: Actual results was a list of divvy locations sorted by closest to farthest from start location

Test Status: Pass: for each input a different result was output with each having correct data following initial entry.

4 Regression Testing

Regression testing on the bike and scooter counters after the newest update maintained passing status, all buttons and counters were covered during these tests. Therefore the GUI has maintained functional status over the recent update to the map and divvy stop sort implementation.

IV Inspection

1 Items to be Inspected

API data parsing to useable data, Integration of google maps to GUI, Population of API data in Table, sorting of API data via coordinates

2 Inspection Procedures

Checklist: Is the code commented? Does the code have warnings? Does the code have any unused/useless code fragments? Does the code operate in a clearly inefficient manner?

Meetings were held online when all team members were present, which started with an overview of how the code section fit into the overall function of the program and the components of said section. After the code's role in the program was clear the team went through the checklist and asked each question. Each team member would then respond with if they thought the code met the criteria, and if it did not then the team discussed what makes it fail and what is needed to fix it. After every criteria was evaluated then the notes on the block of code were added as a comment to the top of the function for ease of access during the next development phase.

3 Inspection Results

Each Item that was inspected was done by the entire dev team. In regards to the API data parsing to usable data, a small inefficiency was noted in the process of parsing strings that could have been reduced to a single loop. The Integration of google maps to GUI met all criteria as listed. The Population of API data in Table was found to have multiple sections of unused code, which was removed from the loop and verified unused after functionality was the same after. The sorting of API data via coordinates did not have any logical errors or inefficiencies, but it did have warnings due to type conversion that were not guaranteed, this was resolved by implementing a safety mechanism in case the data could not be converted. This change was small enough to not warrant a bug fix case later and instead was resolve

V Recommendations and Conclusions

VI Project Issues

1 Open Issues

In the current state of the program, it has not been adapted to any mobile applications yet. This brings a problem with having to reorganize the GUI to be compatible with IOS, Android, and possibly other mobile devices. This is a limiting factor in the number of DIVVY bike drivers who will have access to our program.

2 Waiting Room

Some of the things that are being considered for future releases later down the line are more features involving user data/profile, a communication system between other drivers who are online, real time data involving current drivers to other drivers, and more.

3 Ideas for Solutions

One idea for establishing a system for keeping track of solutions is to utilize a UML design to keep track of design updates. Another solution could be utilizing a virtual discussion board strictly for posting ideas/answers for certain questions that come up. These could all be kept track using a shared GitHub repository that all members of the design team would have access to and could make edits/adjustments at any time.

4 Project Retrospective

At the conclusion of this Divvy Van Project, many things went well and some mistakes did occur throughout the process of this project. What worked well for this group in particular was having assigned tasks for each individual to accomplish based on their strengths, and communication amongst each other was very important. We ran into some errors when merging all of the code together as well as troubleshooting bugs such as figuring out what specific packages weren't imported on each user's machine. In the near future to improve from the mistakes that were made, we would have looked more into getting everyone to use the same compiler such as Eclipse, rather than some using x compiler and some using y compiler and when code would be merged, it wouldn't compile on one compiler and it would on the other. Learning from this was a great experience to one another as this would benefit and give great communication and teamwork skills that are often used by many companies that implement scrumology and sprints in their work life.

VII Glossary

UML Diagram: Unified Modeling Language is a general-purpose, developmental, modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.

DIVVY Station: The location where the DIVVY bikes and scooters are placed when they are not in use.

DIVVY Van: The type of vehicle that the DIVVY company uses to transport the DIVVY bikes and scooters across Chicago.

VIII

IX References / Bibliography

[1] Bernd Bruegge and Allen H. Dutoit, "Object-Oriented Software Engineering - Using UML, Patterns, and Java", Third Edition, Prentice Hall

[2] Thapliyal, Sengupta, Broachwala, Traveling Knapsack Software for Divvy Biking System

X Index