DIVVY VAN ASSISTANT SOFTWARE

Coding Project Report

Prepared by

Prajwal Athreya Jagadish

Sachin Srinivas

Aliasgar Zakir Merchant

Arjjun Prasaad

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. PROJECT DESCRIPTION

Divvy Van Assistant is a GUI application for Divvy Van drivers. The application aims to help transport divvy bikes and scooters from one station to another seamlessly while ensuring optimal travel routes.. The goal of the project is to increase the efficiency of the entire system without affecting the day to day activities in and around the city.

a. Project Overview

The system is a desktop application where Divvy delivery drivers will be able to receive real time instructions on where to transfer bikes and the most efficient routes to take when making transfers. The system, which keeps track of the number of bikes at each station in real time, will use an algorithm to determine which stations require bikes as well as the quantity of bikes to be delivered. The system will also show the most efficient route a driver should take in order to reach a destination in the shortest amount of time.

b. Project Domain

The domain of our project work is an Assistant software (GUI Application) for the Divvy Maintenance van. At the core, our project is very closely related to the knapsack application problem as we are designing software for efficient transportation of divvy bikes and scooters from one station to another.Also we are considering a few different aspects for ranking the tasks that needs to be completed and prioritizing the same. For obtaining the optimal route we are using the google Maps API.

c. Relationship to Other Documents

The coding project report is created with reference to the original development project report that was developed by a group in the previous semester. This dev project was chosen by our group as a part of the coding project we have to develop during the course of the semester. All contents in this document contain the execution steps and documents the steps and the processes that were initiated to code and develop the project to obtain the desired results as mentioned in detail in the original development project report. Additionally this document contains key definition terms of various technologies and frameworks used to develop the project.

d. Naming Conventions and Definitions

Some of the naming conventions in this document are efficiency, mapping, strategy, task-ranking, MongoDB, Pandas.

i. Definitions of Key Terms

1. **Efficiency:** The tool of measurement to determine whether the system has produced a significant reduction in time.
2. **Mapping**: The system that will allow delivery workers to locate from one station to another station.
3. **Strategy**: This is defined as a series of steps to take by executive management based on the success of the system. This term is mentioned to explain the use cases.
4. **Task Ranking**: Based on the excess and deficit bikes in a certain dock, the task is ranked and provided to the user based upon priority.
5. **Plus Codes**: We are basically using the Plus Codes to identify a specific location to get the optimal route from Google Maps. Since the codes are simple, we can easily use them for computation. Plus codes are based on latitude and longitude

values of any given location.

6. **MongoDB**: MongoDB is a document database with the scalability and flexibility that you want with the querying and indexing that you need. We use MongoDB for storing the user/login information, location of the divvy docks, the number of bikes each dock holds and for logging the task completion.
7. **Pandas**: Pandas is basically a python package that can be imported and used for data manipulation and data analysis.
8. **PySimpleGUI** : This is the python library which we used to implement the GUI for our product. This Python library that wraps tkinter, Qt (pyside2), wxPython and Remi (for browser support), allowing very fast and simple-to-learn GUI programming.

ii. Data Dictionary for Included Models

Column	Data Type	Description
Email	String	Takes users email
Username	String	Takes users Username
Password	String	Takes users password
logins_for_the_day	Int 32	Counts the number of logins
user_created_on	String	Mentions date of user creation
login_history	String	Mentions login history of the user
Distance from Base	Double	Distance value from the base is stored
Address	String	Address of the station is stored
Bikes Available	Int 32	Mentions the number of bikes in a station
Latitude	Double	Stores the latitude value of station
Longitude	Double	Stores the longitude value of station
Station ID	Int 32	Stores the station ID
Station Name	String	Stores the station name
Station Plus-Codes	String	Stores the station plus code values

Station Status	String	Mentions the status of the station
Total Docks	Int 32	Stores the total number of docks in a station
Zip Code	Double	Stores the zip code of the station
Task Number	Int 32	Stores the task number
Start Station Name	String	Stores the start station name
End Station Name	String	Stores the end station name
Bikes Transported	Int 32	Stores the number of bikes transported
Distance	Int 32	Stores the distance from base to the station
logins_done	Int 32	Stores the number of logins done
Tasks_Completed	Int 32	Stores the number of tasks completed
Distance_Driven	Double	Stores the distance driven by Divvy driver

Table 1 : Data Dictionary used in the project

Content

The above data dictionary table contains all type of data stored in the mongo Database, the values mentioned in the table are stored under different collections within the mongo Database

Motivation

The table provides an accurate description of all data that is being collected and stored in the database, all of this information is used to store data which is accessed by the program at run time. Additionally the data that is stored in the database will be used to generate various reports which would enable the system administrator to measure the effectiveness of the application.

2. PROJECT DELIVERABLES

After working on the coding project for about 3 months we as a group have collectively developed an application that will enable divvy delivery drivers to complete their tasks in a timely manner and simultaneously provide tasks that are of high priority that is based on the excess or deficit levels of any specific station within a set radius distance.

The application will only enable verified divvy drivers to log into the system. Once the driver logs in they will have the option to choose the radius from which all stations will be displayed, after which the application will compute a list of tasks from which the driver can select and complete the delivery of bikes that are required for each specific station. Along with this, the driver can view a map that shows the directions the delivery driver must follow to complete the task. All of these processes run quickly and without any lag or interruptions which makes the application more friendly and easier to use.

Finally, a Report Generation feature was implemented to track all the delivery driver's activities which include the number of tasks completed, the distance traveled, the number of logins done per day, and the aggregation of tasks completed and bikes transferred. All of this information can then be used for performance analysis of drivers and can be used to calculate the efficiency and performance of the application.

After working on the coding project for about 3 months we as a group have collectively developed an application that will enable divvy delivery drivers to complete their tasks in a timely manner and simultaneously provide tasks that are of high priority that is based on the excess or deficit levels of any specific station within a set radius.

The application which was developed will only enable verified divvy drivers to log into the system. Once the driver logs in they will have the option to choose the radius from which all stations will be displayed, after which the application will compute a list of tasks from which the driver can select and complete the delivery of bikes that are required for each specific station. Along with this, the driver can view a map that shows the directions the delivery driver must follow to complete the task. All of these processes are executed quickly without any lag or interruptions which makes the application more friendly and easier to use.

Finally, a Report Generation feature was implemented to track all the delivery driver's activities which include the number of tasks completed, the distance traveled, the number of logins done per day, and the aggregation of tasks completed and bikes transferred. All of this information can then be used for performance analysis of drivers and can be used to calculate the efficiency and performance of the application.

a. First Release

The first release was scheduled on the 2nd october at 11:59pm.

For the first release of the coding project the group and the members devised a plan to incorporate a certain number of features to get the ground running on the coding project.

A scenario document was created which contained a specific list of features as follows:

- **Basic User Interface:** Will contain different options for the user (Divvy van driver). The user can navigate through these different icons to retrieve certain information, take on a task, send a report/message to the office, request assistance if needed and so on.
- **Search icon**: Search icon can be used to search a number of things just by entering the zip code of a particular location.
- **Getting data from different Divvy Stations**: Data which depicts the number of bikes in use.
- **Login Credentials page** :Will have username and password icons with usual boilerplate

functions.

- **Help Page:** Will be redirected to an entirely different page which contains solutions / case studies to most of the queries/ problems faced by the user.

In order to measure the success of the first release a set of constraints were set, based on the results of the constraints, we could come to the conclusion if the first release was properly executed and the set of features which was expected to be delivered on the first release are missing, these specific features are added to the product backlog, which will then need to be developed before the next scheduled release.

The constraints are as follows:

- Drivers should be able login to the system such that the Application login is only available for Divvy Van Drivers.
- The dataset containing all the information about the various station names, addresses and the number of bikes that need to be transported must all be specified in the dataset.
- The database must be able to automatically update all the user values and station information based on the actions of the user.
- An overall dashboard should have been created in order for all group members to work on the project simultaneously.

b. Second Release

The second release was scheduled on the 30th October at 11:59PM.

The main deliverables for the second release were the Task-ranking and the interface.

For the second release of the coding project the group and the members devised a plan to integrate all the features for example login, task-ranking with the GUI and made sure we had a working interface.

A list of tasks/features those were covered in the second release are as follows:

- **File System**: As we had a lot of distinct features developed by all the members of the team, we had to integrate all these features under one roof. So, we made sure seamless flow of our product from start to finish and we did overcome the accessibility conflicts.
- **Task-Ranking**: This functionality generates the list of tasks(Bikes to be transported from one dock to another) and sorts them based on the priority(Excess and deficit bikes in a certain dock) that will be displayed to the user from which the user will have the flexibility to pick up the desired task.
- **Graphical User Interface:** From the basic working GUI to fully functional which included login till the working of the user dashboard which included task-ranking and report generation, we built a sophisticated GUI using the PySimpleGUI package.
- **Report Generation**: Inorder to generate/ fetch information on the tasks, users, no of bikes transported etc. we came up with an idea of generating reports for further analysis. We also made sure only the admin will be able to access these reports.
- **Contact Us:** The GUI had a separate window for the Divvy users to contact IT support incase of any issues with the product.
- **About Us:** The GUI had a window for the Divvy users to know more about the usage and the working on the product which is mainly designed to be used by the Divvy users to

transport the bikes from a deficit dock to an excess dock.

- **Help Page:** Will be redirected to an entirely different page which contains solutions / case studies to most of the queries/ problems faced by the user.

In order to measure the success of the second release a set of constraints were set, based on the results of the constraints, we could come to the conclusion if the second release was properly executed and the set of features which was expected to be delivered on the first release are missing, these specific features are added to the product backlog, which will then be added on to the list of features we could not implemented during the development phase.

The constraints are as follows:

- The file system must be in proper order and must be organized in a clear way so that users can easily access the file system with ease.
- The tasks must be ranked according to excess and deficit values of bikes in a specific station, the highest ranked task must be the station which has the highest deficit number of bikes.
- The Gui that is developed must be simple and intuitive to use as most of the divvy delivery drivers may not have an advanced knowledge of operating a complex app.
- The reports that are generated must contain all the values that are stored in the database, no value must be missing.
- Adequate information must be added in the contact us and the help pages, which will provide drivers with information if they are stuck in a specific place.

c. Comparison with Original Project Design Document

When it comes to the original document design, we have followed the system design and object design but we have also tried to improvise/add more features and functionalities. Below diagram gives us the basic idea of the context of work. Some of the additional features that we have added compared to the original product is the Interactive map for displaying the optimal route.

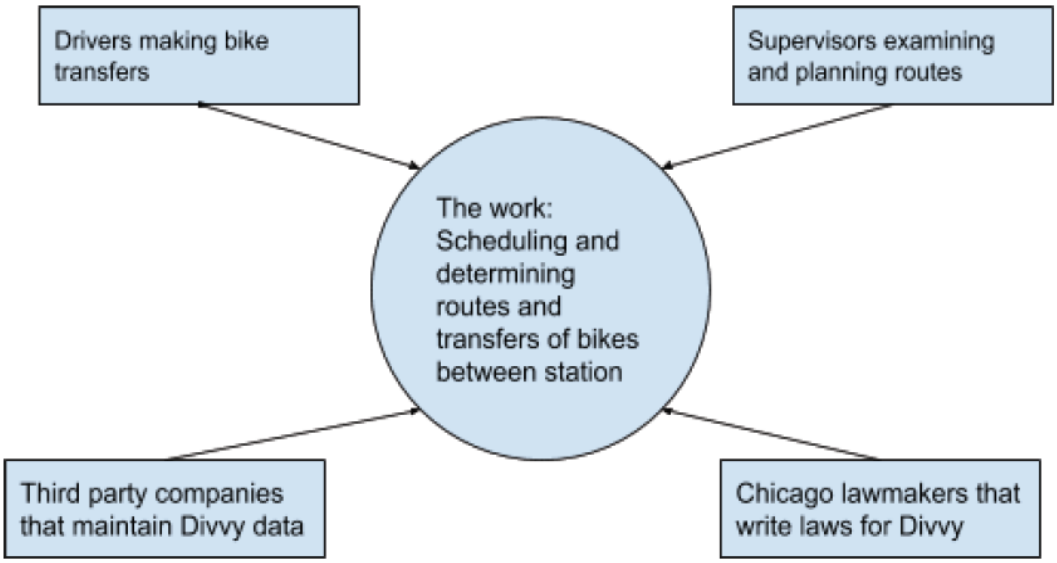

Fig 1: Context of Work Diagram (Depicted as in original report)

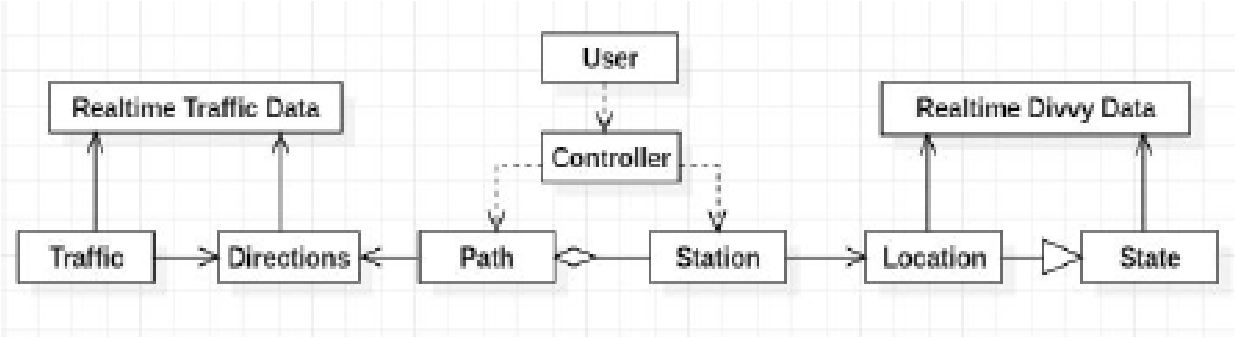

Fig 2: System Design (Depicted as in the original report.)

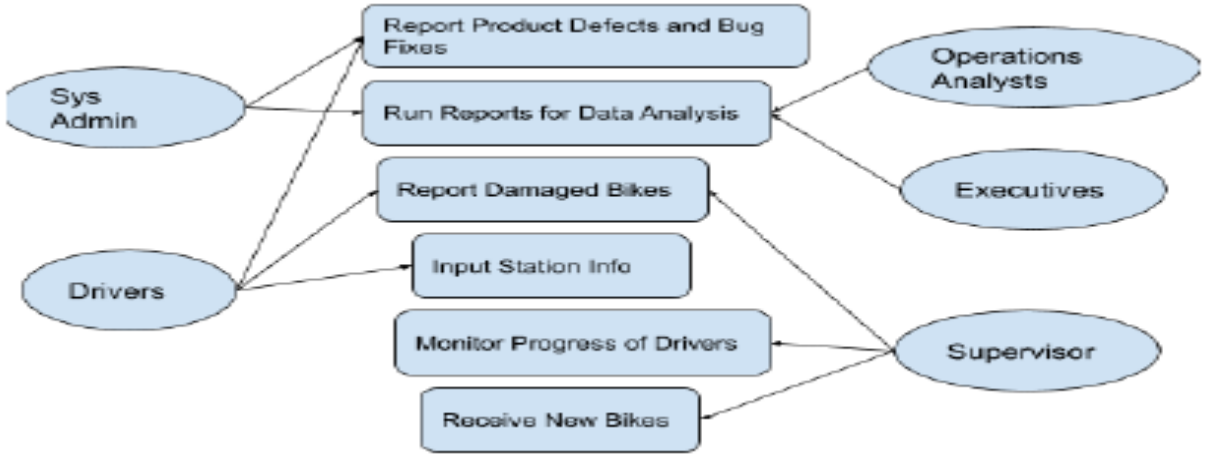

Fig 3 : Use Case Diagram (Depicted as in the original report).

3. TESTING

a. Items to be Tested

- **Emulator:** The stations database is being updated in the background while the code runs in the foreground. Sub-items to be tested include proper creation of the database table where the data will get updated, the random updation of the number of bikes, the creation and existence of the thread. We decided to skip testing if a station would be in excess / deficit / sufficient quantity due to the 40-60% condition on the total number of docks.
- **Task Ranking:** This functionality generates the list of tasks that will be displayed to the user from which the user can then select a task to complete. Sub-items to be tested include proper import of the dataset, test whether the correct subset is generated based upon user selection of the radius, test whether selected stations are being split into excess and deficit stations, test whether task list generated is empty or not, whether station object is generated for each row of the generated subset, whether a task is generated based upon a start and end station and test whether the optimal path is being generated correctly without any issues.
- **Plus Code Values:** A list of plus code values is generated using the addresses of various divvy stations all across Chicago, Divvy has approximately 650+ stations within Chicago and all of these stations have a unique plus code value. Plus codes are values set by google Maps developers which pinpoint a specific location in the world. A test must be done which verifies the accuracy of the plus code values.
- **Report Generation:** Once a driver completes a task his activity is then updated on the database which contains information on driver reports which is stored using a CSV file. This CSV file contains columns like bikes transported, miles driven, the number of times a driver has logged into the system, and various other columns which are present in the report. A simple test must be conducted to ensure that correct and appropriate values are being added to the columns and specifically make sure that there aren't any alterations to the values that are being added.
- **Login:** The main agenda to test here is the Sign-up and Sign-In mechanisms. The new user will be able to add himself to the Divvy users to login into the system. Providing values such as email-id, username and password, the user will be able to create his account on the divvy user database. After creating the user account the same user should be able to login and use the product seamlessly. We can provide invalid user details and try adding the user with random email id, this can be covered as part of the negative test scenario.

b. Test Specifications

- **ID - Test_Creation**

  Description: Tests the creation and import of the divvy stations database using Pymongo
  Items covered:
  - If database table imported is an instance of collection class
  - If database imported is not None

  Requirements addressed: Proper import of the divvy stations database for use in the application.
  Intercase Dependencies: None
  Test Procedures: The script must be run.
  Input Specifications: None
  Output Specifications: Table containing divvy stations data
  Pass / Fail Criteria: If all items are covered by the test, the test will pass

- **ID - Test_Random Updation:**
  Description: Tests the random updation function which will generate a random number for the number of bikes available and assign that value back to the database table.
  Items covered:
  - Check if the value before random update and the value after random update is different.

  Requirements addressed: The database, particularly the number of bikes at each station is being updated in the database to emulate the constantly changing demand of the users.
  Intercase Dependencies: The table / database must exist and the Test - "Test Creation" must pass.
  Test Procedures: The script must be run.
  Input Specifications: None
  Output Specifications: None, although the database updated value must be seen in the database using the MongoDB Compass.
  Pass / Fail Criteria: If all items are covered by the test, the test will pass

- **ID - Test_Thread:**
  Description: Tests whether the thread is created and currently running in the background.
  Items covered:
  - Starts the thread in the background.
  - Tests whether the thread is existing in the thread list.

  Requirements addressed: The database must be updated in the background while the program runs in the foreground. For this to work, a thread must be created which run the updation process. Without the thread, the database will not be updated and new tasks will not be generated.
  Intercase Dependencies: Test Creation and Test Random Updation tests must pass.
  Test Procedures: The script must be run.
  Input Specifications: None
  Output Specifications: None
  Pass / Fail Criteria: If all items are covered by the test, the test will pass.

- **ID - Test_Dataset_Import:**
  Description: Tests whether the dataset is imported correctly.
  Items covered:
  - Tests whether the dataset is not a None object
  - Tests whether the dataset is an instance of pandas dataframe class

  Requirements addressed: The stations database is imported for use as a pandas dataframe. This dataframe is then used to generate tasks based on the user's selection of radius.
  Intercase Dependencies: None
  Test Procedures: The script must be run.
  Input Specifications: None
  Output Specifications: None
  Pass / Fail Criteria: If all items are covered by the test, the test will pass.

- **ID - Test_Subset:**
  Description: Tests if a proper subset based on the selected radius is generated by the function.
  Items covered:
  - Tests if the maximum value of the distance from the base for all rows of the subset generated is less than or equal to the radius selected by the user.

  Requirements addressed: For tasks to be generated based upon the user selection of radius, proper subsets must be generated.
  Intercase Dependencies: The Dataset is imported and ready for use.
  Test Procedures: The script must be run.
  Input Specifications: None
  Output Specifications: An error message will be displayed if the maximum value is greater than the radius selected by the user.
  Pass / Fail Criteria: If all items are covered by the test, the test will pass.

- **ID - Test_Station_Generation:**
  Description: Tests whether for any row of the subset of the dataset, a station object is successfully created.
  Items covered:
  - Tests if the row selected is not None
  - Tests if all attributes of the station object are assigned the corresponding values from the rows.

  Requirements addressed: After generating the subset, a stations object must be created for every row in the subset which can then be used to generate a task ranking.
  Intercase Dependencies: Tests Dataset_Import and Subset must pass for this test to be run.
  Test Procedures: The script must be run.
  Input Specifications: None
  Output Specifications: An error message will be generated if either the row generated is None, or either of the attributes of the object class is not equal to that of the selected row.
  Pass / Fail Criteria: If all items are covered by the test, the test will pass.

- **ID - Test_Task_Generation:**
  Description: Tests whether a task object is generated based upon a start and end station.
  Items covered:
  - Tests if the excess station object generated is not None.
  - Tests if the deficit station object generated is not None.
  - Tests if the attributes associated with the task object are successfully assigned depending on the start and end stations.

  Requirements addressed: After generating a list of excess and deficit stations, a task object must be created which combines the data from the excess and deficit stations. These tasks will then be displayed to the user for selection.
  Intercase Dependencies: Tests Dataset Import, Subset and Station_Generation must pass.
  Test Procedures: The script must be run.
  Input Specifications: None
  Output Specifications: An error message is generated if the tests fails
  Pass / Fail Criteria: If all items are covered by the test, the test will pass.

- **ID - Test_Optimal_Route:**
  Description: Tests if the google maps API is able to find a route between the start and end stations pluscodes.
  Items covered:
  - Tests if the route generated is not None.
  - Tests if the route generated is not empty

  Requirements addressed: After the start and end stations have been assigned along with the number of bikes to be transported, the optimal route between these stations must be calculated which can then be displayed to the user.
  Intercase Dependencies: Tests Dataset Import, Subset and Station_Generation must pass.
  Test Procedures: The script must be run.
  Input Specifications: None
  Output Specifications: An error message is generated if the route generated is None of empty.
  Pass / Fail Criteria: If all items are covered by the test, the test will pass.

- **ID - Test_Task_Ranking:**
  Description: Tests if the task object list is generated once all tests pass.
  Items covered:
  - Tests if the task list that is generated is not None.
  - Tests if the task list that is generated is not Empty.

  Requirements addressed: Once all tests for sub-items pass, i.e. the stations object and task objects are being created, the final task is to generate a list of similar task objects which can be sent to the user.
  Intercase Dependencies: Tests Dataset Import, Subset and Station_Generation, Task_Generation and Optimal_Route must pass.
  Test Procedures: The script must be run.
  Input Specifications: None
  Output Specifications: An error message will be displayed if the task list is empty or None.
  Pass / Fail Criteria: If all items are covered by the test, the test will pass.

- **ID - Login_Sign-up:**
  Description: Test the Sign-Up functionality, where the user will be able to provide the details such as E-mail address, username and password, which can be used for onboarding of new Divvy users.
  Items covered:
  - Check if the user already exists.
  - Check if the username is already registered.
  - Check if the email address is compatible.

  Requirements addressed: Functionality of the onboarding the Divvy user to be addressed.
  Intercase Dependencies: MongoDB server should be up and running, “Test Creation” must pass.
  Test Procedures: This test must be performed right when the divvy GUI starts just before the

user enters the dashboard with his credentials.
Input Specifications: Email-Id, Username, Password.
Output Specifications: User created successfully.
Pass / Fail Criteria: If all the items are covered and if the credentials are successfully committed to the database the test case is marked as “Passed”.

- **ID - Login_Sign-in:**
  Description: Test the Sign-in functionality, where the user will be able to provide his user details such as Username and password to login to his dashboard and start with their tasks.
  Items covered:
  - Check if the user registered.
  - Check the password and login.

  Requirements addressed: Functionality to help a Divvy user to login to the dashboard so that he can start working on his tasks.
  Intercase Dependencies: MongoDB server should be up and running, “Test Creation” must pass.
  Test Procedures: This test must be performed right when the divvy GUI starts just before the user enters the dashboard with his credentials.
  Input Specifications: Username and Password.
  Output Specifications: User logged in successfully.
  Pass / Fail Criteria: If all the items are covered and if the login history is successfully committed to the database the test case is marked as “Passed”.

c. Test Results

- **ID - Test_Creation**
  Date of Execution: 25 November
  Staff conducting tests: Aliasgar
  Expected Results: All tests pass with no error message.
  Actual Results: Tests passed successfully
  Test Status: Pass

- **ID - Test_Random Updation**
  Date of Execution: 25 November
  Staff conducting tests: Aliasgar
  Expected Results: All tests pass with no error message.
  Actual Results: Tests passed successfully
  Test Status: Pass

- **ID - Test_Thread**
  Date of Execution: 25 November
  Staff conducting tests: Aliasgar
  Expected Results: All tests pass with no error message.
  Actual Results: Tests passed successfully
  Test Status: Pass

- **ID - Test_Dataset_Import**

Date of Execution: 25 November
Staff conducting tests: Aliasgar
Expected Results: All tests pass with no error message.
Actual Results: Tests passed successfully
Test Status: Pass

- **ID - Test_Subset**
  Date of Execution: 25 November
  Staff conducting tests: Aliasgar
  Expected Results: All tests pass with no error message.
  Actual Results: Tests passed successfully
  Test Status: Pass

- **ID - Test_Station_Generation**
  Date of Execution: 25 November
  Staff conducting tests: Aliasgar
  Expected Results: All tests pass with no error message.
  Actual Results: Tests passed successfully
  Test Status: Pass

- **ID - Test_Task_Generation**
  Date of Execution: 25 November
  Staff conducting tests: Aliasgar
  Expected Results: All tests pass with no error message.
  Actual Results: Tests passed successfully
  Test Status: Pass

- **ID - Test_Optimal_Route**
  Date of Execution: 25 November
  Staff conducting tests: Aliasgar
  Expected Results: All tests pass with no error message.
  Actual Results: Tests passed successfully
  Test Status: Pass

- **ID - Test_Task_Ranking**
  Date of Execution: 25 November
  Staff conducting tests: Aliasgar
  Expected Results: All tests pass with no error message.
  Actual Results: Tests passed successfully
  Test Status: Pass

- **ID - Login_Sign-up**
  Date of Execution: 25 November
  Staff conducting tests: Sachin
  Expected Results: All tests pass with no error message.
  Actual Results: Tests passed successfully

Test Status: Pass

- **ID - Login_Sign-in**
  Date of Execution: 25 November
  Staff conducting tests: Sachin
  Expected Results: All tests pass with no error message.
  Actual Results: Tests passed successfully
  Test Status: Pass

4. INSPECTION

In this Section we will be talking about the Inspection phase.All the members of our team have reviewed each others' work and then commented on the same to resolve any issues if there was one.We are basically a team of 4 students and hence each of us had to inspect three other items.We had created a branch for each of our implementations and then we reviewed the code other team member wrote and then comments and suggestions for further improvements were given.

a. Items to be Inspected

Some of the items(Features / Functionalities) that had to be inspected are as follows:

- Front-End and Login Page
- Task Ranking
- Report Generation
- Database Updation

b. Inspection Procedures

When it comes to the inspection procedures, we were a team of four and we split the work by inspecting three items each so that all the items will be inspected and there won't be any bias. We basically had four main items that had to be inspected and each of us took three items other than ours and reviewed it and then suggestions and comments were offered for improvements that had good scope.

We made a checklist of all the that a certain feature implementation had to satisfy and keeping that as a reference we reviewed each other's work and improved whatever could have been improved.So we had at least one in person meeting and apart from that we had virtual meetings whenever a collective input was required on a certain aspect of the project work.And during the meeting, we always would clearly state the agenda for that meeting and only those aspects of the project work were discussed which increased our productivity towards efficiently solving the issue in hand.

c. Inspection Results

As explained in the above sections, We followed a procedure for the inspection phase and the result was quite promising. We were able to solve a few computational issues and it was more efficient when we did it collectively as a team.Since each item was reviewed three times,we all got great inputs from our peers and we were able to make improvements to the implemented features and made it more functional.

One of the best improvements was with respect to the computation of the optimal path. Originally we could not compute the optimal path but then we came up with a solution and used plus codes(Google Maps) for obtaining the optimal path.And then re-inspected this again and everything was in order.

5. RECOMMENDATIONS AND CONCLUSIONS

a. Project Issues

- Driver's do not have an interactive map display to use.
- Computing the Task Ranking operation takes a long time to complete.(Similar to the traveling salesman problem : Knapsack Problem.)
- Driver's cannot choose a specific starting location. All operations must start from the base.
- Station distances are computed from the base hub only.
- Absence of a proper encryption system makes the application vulnerable to random attacks.

b. Open Issues

- **Map Display:**

Content:

One of the open issues we are facing right now is with map display, Since we are using PySimpleGUI to develop the UI of the application we are restricted with the flexibility and designing of the user interface.

Motivation:

Since Map display is one of the most desired features of the application, which will significantly improve the user experience. It is one of the most important open issues which must be solved to improve the usability of the application.

Examples:

One We have experimented with various different ways to implement a visually interactive map for divvy delivery drivers to use so that they can reach their destination with ease. But unfortunately we are restricted from implementing an interactive map because of the limited functionality of pysimple GUI which makes it difficult to display web pages or any sort of image on the UI, Therefore even after hours of research done to tackle this problem we are still stuck with no solution available which can be used to resolve this situation.

Considerations:

There are no considerations with regard to map display.

c. Waiting Room

- **Notification System :** One of the potential ideas for our application in the future is to implement a Notification System that will give us live updates of the events(once the task is started) in chronological order.
- **Encryption :** A strong encryption system for our application that will prevent any and all the threat to all the sensitive user data.
- **Tracking the completion time of a specific task :** Tracking all the operations will give us some interesting data for further analysis.
- **Network Connection capability :** Checking the network connection capability is an important feature that can be incorporated.

d. Ideas for Solutions

- **Notification System :**

  Content : One of the potential ideas for our application in the future is to implement a Notification System that will give us live updates of the events(once the task is started) in chronological order.

  Motivation : Giving users the control of the process.

  Consideration : One of the simplest solutions for implementing the live notification system is just to track the updates and send the same to the interface so that it can be printed onto the console.

- **Encryption:**

  Content : A strong encryption system for our application that will prevent any and all the threat to all the sensitive user data.

  Motivation :We had a few different approaches in our mind to implement a strong encryption system so that our application will be free from any malicious threats and attacks.Also that will prevent any attack on the user data which is very important.

  Considerations : We have looked into a few encryption methods that can be implemented for our application. We were considering RSA Public key cryptosystem and Advance encryption standard.

- **Tracking the completion time of a specific task :**

  Content : Tracking all the operations will give us some interesting data for further analysis.

  Motivation : This can be used for further analysis at the business end.

  Considerations : Again similar to the Notification system,we can also implement a feature that will essentially give us the completion time of a specific task.

- **Network Connection capability :**

  Content : Checking the network connection capability is an important feature that can be incorporated.

  Motivation : Seamless transmission of data is very important for us when it comes to our application.At any given point of time, we need to have a reliable network connection for operations.

  Considerations : Operations to be conducted in areas with good network coverage and good bandwidth.

e. Project Retrospective

Content:

This whole semester has been nothing short of completing challenging tasks and submitting work at the last minute of the specified deadline.

During the whole process, we tried different methods and strategies to complete our work. At the start of the semester, we were introduced to Jira which enables members in a group to practice agile methodologies to work on software development. We collectively agree that assigning specific tasks for each user enables all of the team members to put work on their part and complete what is required. Week after week we kept assigning new tasks and details which are supposed to be completed by each member of the group and as weeks passed we were able to integrate all of our features and work into a single program and we successfully demonstrated a working application to our peers.

Motivation

The main motivation for all of the members in the group was to build a unique one-of-a-kind application that can potentially be used in real-life applications and this would actually result in a more significant and efficient way to complete the operation. Additionally, we will be doing all this while working together as a team and emulating real-world situations where all of us will be working together as a group with a shared goal, Furthermore, us being curious students all of us strive to learn something new which will help us become a more complete professional which will guide all of us to the highly sought after career opportunities in software development.

Considerations

There have been plenty of situations where things have not gone our way or as we had planned, to overcome all these problems we had to improvise and create new strategies and plans to solve these problems. Out of all the problems the most challenging one was assigning too many tasks. By doing so we were just overloaded with a lot of tasks and requirements on all the group members and eventually, it went to a point where we could not complete them before the respective project release deadline. We as a group should have been more careful in planning and assigning tasks as the practice we had followed was not correct and ended up causing even more features to be pushed into the product backlog this deterred us from completing the required goals of the project.

6. GLOSSARY

- **Efficiency:** The tool of measurement to determine whether the system has produced a significant reduction in time.
- **UML (Unified Modeling Language)** : The 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.
- **System Design :** System Design is the transformation of the analysis model into a system design model.During the system design , developers define the design goals of the project and decompose the system into smaller subsystems that can then be realized by the concerned teams.
- **Object Design :** During the object design phase , we close the gap between the applications objects and the off the shelf components by identifying additional solution objects and refining existing objects.
- **Graphical User Interface (GUI)** : It is a type of user interface application that allows the users to interact with the application itself on an electronic screen(Hardware device).It is made interactive by adding buttons and other such functionalities.
- **Coupling** : Coupling is basically the degree of interdependence between the created software modules.It is the measure of how closely two routine or software modules are related to each other.
- **Coherence** : It is the degree to which elements inside a software module belong together.
- **Jira Software** : It is a ticketing and project management tool developed by Atlassian. This tool allows agile project management.
- **Version Control** : Version control is the practice of tracking and managing the changes made to the existing software code.
- **Mapping**: The system that will allow delivery workers to locate from one station to another station.
- **Strategy**: This is defined as a series of steps to take by executive management based on the success of the system. This term is mentioned to explain the use cases.
- **Task Ranking**: Based on the excess and deficit bikes in a certain dock, the task is ranked and provided to the user based upon priority.
- **Plus Codes**: We are basically using the Plus Codes to identify a specific location to get the optimal route from Google Maps. Since the codes are simple, we can easily use them for computation. Plus codes are based on latitude and longitude values of any given location.
- **MongoDB**: MongoDB is a document database with the scalability and flexibility that you want with the querying and indexing that you need. We use MongoDB for storing the user/login information, location of the divvy docks, the number of bikes each dock holds and for logging the task completion.
- **Pandas**: Pandas is basically a python package that can be imported and used for data manipulation and data analysis.
- **PySimpleGUI** : This is the python library which we used to implement the GUI for our product. This Python library that wraps tkinter, Qt (pyside2), wxPython and Remi (for browser support), allowing very fast and simple-to-learn GUI programming.
- **Test Cases :** Test cases are basically a set of actions/operations performed on the system(application) to check if it satisfies all the requirements.

7. REFERENCES

[1] Robertson and Robertson, Mastering the Requirements Process.
[2] A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth ed., Wiley, 2013.
[3] J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012.
[4] M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.
[5] Divvy Truck Assistant Project Report: We would like to acknowledge the efforts of Alfonso Arias, Vivek Ganesan, and Juan Zambrano.

8. INDEX

No index entries in the document