Coding Project Report Summary : Divvy Van Assistant

Group 22: Prajwal Athreaya Jagadish, Sachin Srinivas, Aliasgar Merchant, Arjjun Prasaad

First Release

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

Second Release

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

Test Specifications

ID - Test_Creation

Description: Tests the creation and import of the divvy stations database using Pymongo

Items covered:

1. If database table imported is an instance of collection class
2. 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: To be completed before the other tests for the emulator are performed to ensure that database to be used later exists.

Input Specifications: None

Output Specifications: Table containing divvy stations data

Pass / Fail Criteria: If all items are covered by the tests 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:

1. 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: This test must be performed after the database exists to ensure that the data will be changed at random in the background when the thread is created.
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 tests pass.

ID - Test_Thread:

Description: Tests whether the thread is created and currently running in the background.
Items covered:

1. Starts the thread in the background.
2. 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 runs 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:Test Creation and Test Random Updation tests must be completed before the current test is run.
Input Specifications: None
Output Specifications: None
Pass / Fail Criteria: If all items are covered by the tests pass.

Project Retrospective

Conclusion:

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.