I have created a parcel depot software where the Manager handles data regarding the customers and parcels. Based on the Class Diagram in Part 1, the flow of the structure was for the manager to have the liberty to manipulate data regarding the parcels and customers. Manipulations include the addition, removal and search of them by their unique identifiers. The customer queue is sorted according to the customer surnames. To make the software look and feel more like a delivery depo service, I also implemented the receival of parcels by the customers. The manager looks through the customers and since we have created a customer queue, we use the LIFO structure thus the customer with a sequence number of 1 will only be dealt with. Once they agree to receive the parcel, the manager gives them the parcel and the status of the parcel changes from COLLECTING to COLLECTED, the collected parcel list can also be seen when the manager goes to the parcel view. I have also added the implementation of the manager to go to and fro the different views created, keeping in mind of user friendliness, multiple validation checks have been placed to remove incorrect data entry and allow a formulated consistent list of both customers and parcels. The depo center offers discounts to those customers whose parcels end with a ‘5’ and a late The fee is calculated based on the number of days a parcel remains in the depot beyond one week, with a $5 fine applied for each additional day.

The only things missing from my designs in Part 1 include the GUI of the software where through the use of buttons, the Manager is able to interact with the system along with the naming convention of a few functions, this is only because when the coding takes place, sometimes a better preferred function name is chosen but this does not affect the logic of the function. Apart from this the implementation of a waiting list has now been implemented in the part 2 of this project.

The MVC pattern has allowed my code to have a sequence and have a structure of the kinds of classes that deal with specific kinds of data. The Manager in the Model folder since it is the face of the software and where the manager interacts with the software, the csv and txt file has been added in the View since they are the files where i hold my data and the rest of the logic is held in the classes of the Controller package. The MVC clearly gave me a deeper understanding of how the system works in a sequential way thus promoting scalability, testability, and maintainability by keeping the logic, user interface, and data independent yet interconnected.

Screenshot of the Git commits: