Design Document

**A) General overview:**

The system contains a chunk program (project.py) and 4 branches (account\_manager.py, supervisor.py,

dispatcher.py, driver.py)

--Welcome screen:

Running project.py, user can choose to sign up, login, or quit. If user choose to quit (enter “q”), the program will shut down.After user sign up or log in successfully, the application will identify which role the user is, and provide corresponding options as following:

--call account\_manager:

--task 1: list all the information associated with the customer that is selected.

--task 2: create a new master account.

--task 3: add a new service agreement to a customer that user selects.

--task 4: create a summary report of the customer that user selects.

--call supervisor:

--task 1: assign a new account to a manager.

--task 2: create a summary report for a customer.

--task 3: create a summary report for an account manager.

--call dispatcher:

--task 1: create an entry in table service fulfillments.

--call driver:

--task 1: list all the tours that drivers have been assigned to.

The system will show the main options screen for the role after a completed task, until the user explicitly chooses to log out.

**B) Detailed component design:**

Ⅰ. Main (**project.py**): All the steps that the system need to go through and connect a path to sql.

Ⅱ. Welcome\_screen():

1) log\_in():

1. login for registered users

1.1 check\_username(): check if the input username exists in users table

1.2 check\_password(): check if the input username matched the password in users table

2. identify the role (depends on username)

**If call** **object** **account\_manager (account\_manager.py)**

2.1 list\_information(pid,connection,cursor): to list all the information about an account

2.1.1 check\_account(account,pid,connection,cursor):check if the account is managed by this account manager

2.2 create\_service\_agreement(pid,connection,cursor): add new service agreement to a customer

2.2.1 check\_new\_account(): check if the account\_no is already existed

2.2.2 check\_customer\_type():given a customer type, check if it is valid

2.2.3 check\_waste():check the waste types

2.2.4 update\_amount():given a price and a master account, update the price to the total amount in the account

2.3 create\_account(pid,connection,cursor): create a new master account

2.3.1 check\_new\_account(account,connection,cursor):check if the account\_no is already existed

2.3.2 check\_customer\_type(types,connection,cursor):given a customer type, check if it is valid

2.4 summary\_report(pid,connection,cursor): create a summary report for a single customer

2.4.1 check\_new\_account():check if the account\_no is already existed

2.4.2 check\_account():check if the account is managed by this account manager

**If call object supervisor (supervisor.py)**

2. supervisor(pid,connection,cursor):

2.1 assign\_account(pid,connection,cursor):assign a new account to an account manager

2.1.1 check\_manager(): check if this supervisor supervise this account manager

2.1.2 create\_account\_two():create a new master account call

2.2 customer\_report(pid,connection,cursor): select a customer that managed by the account manager who supervised by a supervisor

2.2.1 get\_manager\_name():given a master account, get the account\_manager name

2.2.2 summary\_report\_two():create a summary report for a single customer (called check\_new\_account(),supervisor())

2.3 manager\_report(pid,connection,cursor): generate a summary report for account managers that supervisor supervise

2.3.1 get\_managers():given a supervisor\_id, get the account managers

2.3.2 sort\_report():sor the summary report by the different of the sum of the internal cost and sum of the price

**If call object dispatcher (dispatcher.py)**

2. dispatcher(pid,connection,cursor):

2.1 create\_entry(pid,connection,cursor):create an entry in service\_fulfillments table need to check\_service\_no() check the number is in service\_agreements table, check\_driver(), check\_own\_truck() check if a drive own a truck or not, check\_truck() check if the truck is in trucks table or not.

2.1.1 get\_master\_account(): given a service\_no, to get the master account

2.1.2 get\_pick\_up():given a master account, return the container that he should pick up next time

2.1.3 get\_available\_container():given a list of waste type, return the containers that has the same type with it

2.1.4 calculate\_service\_no():calculate next available service\_no

**If call object driver (driver.py)**

2. driver(pid,connection,cursor):

2.1 list\_information(pid,connection,cursor): list the tour informations about a driver

2.1.1 get\_service\_no(start\_date,end\_date,driver\_id,connection,cursor): given a date range,

return all the service\_no in this range

2.1.2 get\_container\_id(service\_no,driver\_id,connection,cursor):given a service\_no, return the cid\_pick\_up and cid\_drop\_off

2.1.3 get\_information(service\_no,connection,cursor): given a driver id, return the location

After each task the user can choose the logout option.

2)sign\_up(): sign up for new users, store the new user information into table users

2.3.1 check\_pid(): check if the pid is from personnel table

2.3.2 check\_exist\_pid(): check if the pid is registered

2.3.3 check\_role(): check if the user id match the role

2.3.4 check\_username(): check if the username exists in users table

**C) Testing Strategy**:

Our tests start with the welcome screen which is to ask the user to select, whether they want to login or sign up or quit directly.

Welcome screen:

If the user inputs 'l'or "L", the system should accept these commands as the valid commands. And then let the user enter their username. At this point, we tested if the system can recognize the username case-insensitive since the system needs to check if this username existed or not. We also tested the system do not accept the username that does not exist in the database. After the user entered the username, the system asked the user to enter the password. We tested if the password user entered was matched with the password stored in the database. We also test, once the system passed the username test, can it extract the password from the database by the username user just entered. If the user entered the username or password wrong 3 times, we tested if the system will be closed in that situation. As for the sign-up, we first tested if the system can correctly recognize the user id user entered is exist or not and tested if the role entered was matched with the record in the database, if they do not match, the system should ask the user to enter again. Then we tested if the username the user wants to the user is already been occupied. The system should be able to recognize the username in case-insensitive as well, we tested this too.

Account manager:

For function 1: we first tested if the master account entered existed, if exist, we then tested if this account is managed by the user. We also tested if the system will give the right output associated with this account and list all the service agreements that in the database.

For function 2: we first tested if the master account entered is valid, which means cannot include any characters. If it is valid, we tested if this account already existed, if it does not exist, the system can create this account. We also tested if the system can automatically set to the id of the account manager

For function 3: we first tested if the account entered managed by this manager, then tested if the master account can automatically have settled. Then, we tested if the system can correctly set the next available running number.

For function4: We just tested if the account entered is managed by this manager, and if the system can output all the information correctly.

Final, we test if the system can log out correctly with all the changes have been saved to the system.

Supervisor:

For function 1: we did same tests as the account managers, except we also tested if the account manager that assigned is supervised by this supervisor

For function 2: We first tested if the customer selected is managed by the manager that this supervisor supervised. Then, we tested if the system can correctly return the name of the account manager.

For function 3: We first tested if all the manages this supervisor supervised can be listed. Finally, we tested if the report was sorted by the difference between the sum of prices and the sum of internal costs.

We last tested if the system can log out and save all the changes to the database correctly

Dispatcher:

For the only function:

We first tested if the service agreement selected is existed or not, then tested if the driver selected is existed or not. For a driver, we tested if this driver owns a truck, the system should skip the step to ask user enters the truck id. If this driver does not own a truck, the system should do so. We tested if the user selected a truck that owned by someone, the system should ask the user to select again. We also tested if the system can correctly find out which container should be picked, if there is no available container, it should pick up container '0000'. For the drop of id, we tested if the system given a master account can return all the available containers to drop off. If there are no containers to drop off, the system should ask the user to select another service agreements.

Drivers:

For the only function, we just tested given a date range, if the system can find out all the tours for a specific driver, not other drivers. For the tour information, we tested if the information is matched with this driver.

Coverage of test cases: We tried to cover as many scenarios as possible. For each step, we used many pairs valid and invalid input to test if this step is correct or not.

Bugs: During the tests, we did counter many bugs. But most bugs are caused by the incorrectly using the functions in python and some typo errors. For example, when we print the summary report, the float cannot connect with the string, there was also bugs of "index out of range" and "sqlite3 operational error"

**D) Group work description:**

Yuan Wang: Worked on the codes of account manager and supervisor

Time spent total: 5 hours on account manager, 4 hours on the supervisor.

Weiting Chi: Worked on the codes of login, sign up and driver

Time spent total: 7 hours on login and sign up, 2 hours on the driver

Yiyang Wang: Worked on the codes of dispatcher and design and inserted new data into the database

Time spent total: 4 hours on dispatcher, 5 hours on design, 1 hour on insert new data

We all finished our parts, and if someone counter problems, we will discuss in the chat room, sometimes at school or on the internet. We kept updating the progress in the discussion group and report our progress every day. After we finished our separate parts, we spent about a whole day to combine all the codes together and modified the design parts together.

**E) Other**

1.If user login failed 3 times, which means the password or the username is wrong more the 3 times, the application will shut down atomically.

2. user.py we created some user data to test, usernames and passwords are included.

Mini-Project 1

Group 8

Yuan Wang (H06)

Weiting Chi (H07)

Yiyang Wang (H07)