CMPUT 291 Project 1

Julian Stys, Lab Section H07

Kyle Fujishige, Lab Section H02

Alex Li, Lab Section H04

We did not collaborate with any other student outside of our group

(a) General overview:

(i) The waste management database gives a waste management company a simply mechanism to organize and control all the information involved within the company. Permissions are controlled by the user's role in the company (account manager, supervisor, dispatcher, or driver). Once the user is logged in, the system automatically determines the role of user as stored in the database. Then one of the following interfaces is displayed according to his/her role.

(ii) Account manager interface
4 functionalities are provided for account managers, simply select one of the options and follow the prompts:
- Listing information associated with a selected customer.
- Create a new master account.
- Add new service agreement for a selected customer.
- Display a summary report for a selected customer.

(iii) Supervisor interface
The supervisor interface is somewhat similar to the account managers, except they have multiple account managers under them, and can't make service agreements. They have 3 functionalities:
- Create a new customer account using one of the supervisors managers
- Look up a customer's summary report, which includes number of service agreements, the sum of all their service agreements prices, the sum of internal costs, and the managers name who manages the customers account
- Make a summary report from all the current account managers under the supervisor which includes number of master and service agreements and the sum of prices and internal costs. The list is sorted by the difference in price and internal costs.

(iv) Dispatcher interface
A dispatcher only needs to first select an available service agreement, and then they should be given options to select a driver, truck(if not owned by the driver), and an available container to be dropped off and picked up for the customer.

(v) Driver interface
Drivers have a very simple interface. Once a driver has logged in, they are prompted with a start and end date. This is to find what their current schedule is between the given dates. They will be given all the information needed for each job like the date, location, customer contact info, waste type, and the container ids to be picked up and dropped of.

(b) Detailed component design:

(i) project1.py file contains the main function, which controls the main flow of the entire program, and implements a login interface. Depending on user's role, it will initialize the specific classes needed in separate modules.

(ii) When a class gets initialized in any module, it will connect to the database. The main program then runs the interface method that will start up the specific interface.

(iii) Due to the flow of the main program, none of the classes in specific modules will ever interact with a separate module. This is because the flow is dictated in the main interface, and all the data needed by any module is already in the database by simple queries. Thus, there's no need for any interaction between any of the modules and classes.

(iv) Once a user is logged in, and is in their specific interface, they are prompted to enter an input. This input will dictate the flow of their specific module. If the module is the account_managers, supervisor, or dispatcher modules, the interface method will need to call methods that will query certain parts of the database that display the proper information depending on what the user is inputting. The drivers module however does not call any methods because it's a simple query between 2 user inputted dates.

(c) Testing strategy:

- Scenarios:
  - As an account manager, attempted to access accounts under another manager to test it's access permission.
  - As a dispatcher, created schedules for drivers, then logged in as driver to look up these schedules.
  - Supervisors could make a new customer account and the account manager can make service agreements for it, as well as query it.
- Coverage:
  - Some accounts were created, then some service agreements were created under the account for testing purposes. Then attempts were made to access different information against these accounts and service agreements. Different account manager logins were used for testing permissions, to make sure account managers can only access accounts under their management.
  - Account managers and supervisors went well together, as far as testing was concerned. An account manager could make up a whole bunch of customer accounts and service agreements, and then we could use the account managers supervisor to look up the new accounts information.
  - Some erroneous inputs were used to test the interfaces, to make sure different unexpected inputs are handled.
  - Dispatchers would make jobs for drivers, and then drivers would be able to look up their new schedules.

- Bugs:
    - Made cursor and connection global in main program but can’t be referenced in other modules. We had to connect to the database in each module.
    - Passwords entered by a user are hidden to prying eyes thanks to the python getpass module.
    - Lots of foreign key constraints had to be dealt with by using simple error checks by querying the database for the specific columns needed, and then using those to implement infinite loops.

(d) Group work description:

● Julian		
	○ Main program flow and login interface	7 hrs
	○ Dispatchers module	7 hrs
● Kyle		
	○ Supervisor module	5 hrs
	○ Driver module	1 hr
	○ Data preparation for testing	2-3 hrs
	○ Writing of documentation	3 hrs
● Alex		
	○ Account manager module	10 hrs
	○ Data preparation for testing	2-3 hrs
	○ Writing documentation	3 hrs

(e) Other:

- We made sure that there will always be a back option in every possible interface.
- We made sure that the password would be invisible thanks to the python getpass module
- For simplicity sake, we made everything into modules and classes so there’s less code to read over, as well as a nice flow to the program.
- Made sure to check for many different errors that the user could have tried to do. Some errors included proper dating, proper numerical values needed with money inputs, etc.