SWE 321 – Final Project
Spring 2021

Project Title: **Online Car-wash Administration System**

Project Objective and Guidelines:

This project integrates what the student has learned through the semester to analyze a given system requirement using Object Oriented (OO) techniques, design using UML class diagram, extract class relationships and implement the system in Python.

Project submission guidelines

1. The project team will be composed of 1-3 students.
2. Submit a final report, Python code and other files to Blackboard within the given deadline (No extension)
3. Weight: 30% of the total course grade.

Project Description:
This project requires the creation of a computerized online CleanCar administration System for CleanCar.com website. The code will run through a GUI interface. The system is a safe and convenient place to perform online access for users' record, online purchase, top-up (add money) their "WashCards". The system should be able to identify if the web user is an existing user or a new user (new user needs to register).

A user can be an employee or manager or regular user.

When the web user login into his/her account:

- They can check their information (update it if needed as well),
- He/She can perform a purchase of CleanCar products (regular wash, Car Steam Cleaning, Car detailing …etc). You will need to think of a min 5 products for a car wash service.
- Add money to their WashCard using credit cards/cash |(a cash machine should be available in the CleanCar shop to top up using cash, so give directions to the user!).

A new regular-user can register with the following details: username & password, full name, address, telephone, email, WashCard number (Hint: randomly assign a 6 digit number to this card then ask the user to pick the printed card from the shop employee). After the user gets the card, they can add money by credit or cash.

A new employee-user should be able to register as an employee with the following details: employeed-id, full name, username & password, address, manager or regular employee, email, telephone, and WashCard number. If the user doesn't have employee ID they can't register as employee. He/she will be a regular user.

- An employee can access the regular users' details, their cards details, only modify these records, and print these records.

- The manager has extra privileges (explained in the **Assumptions for Implementation section).**

The users can add a dependent card (or more). Each dependent account is linked to the main account. The dependent account will have a card with id-number, full name, and total amount of money left (the money for all accounts together). **Note**: Anyone can top-up the total account money.

According to the category chosen, there will be discounts, and they are:

- A gold member: 35% discount (after 500 points gained, each point is gained after purchase of Dhs1)
- A silver member: 25% discount (after 300 points gained)
- A regular member: 15% discount (They need to Top-up new cards with Dhs100)
- A manager: 50% discount
- An employee: 30% discount.

Assumptions for Implementation:

- The CleanCar.com system uses a File-Manager (FM) software
  - It manages `webuser.txt` file. This file keeps a record of all existing web users and their passwords **only**. It can add a new user to the file and search for an existing user for login purpose.
  - The FM also manages `records.txt` file which has the complete detailed records of all accounts for CleanCar business service.
  - FM can add a new record, delete a record, and view all records in the file and search for an existing record in both the `webuser.txt and records.txt files`.
  - The FM manages `products&prices.txt` file that is connected to each product sold and the user record. Note: the purchase will affect/handle the WashCard usage record.

- Only the manager-user can:
  - use the commands (view-all) and (delete) records from the FM software, which is already linked as mentioned to `records.txt` and `webuser.txt` files.
  - list all employees, gold members, silver members, regular members and all members.
  - view overall day profit.

- Any record should be displayed to the manager as a sorted list (sort by ID or Name)
- Any regular user can add new account, dependent account, modify and view their accounts.

Finally:

- All your inputs should be validated by testing the code using the *try-except* clause.
- All your output screens must be well formatted with sufficient tips to allow users to easily interact with the system.

Final Submission:

1. Report Component

SWE 321: The final submission report should contain the following sections.

• **Title Page:**
Include your names and IDs of team members (max 3 members)

• **Section 1-** Problem Analysis (20%):
In this section, a detailed description of all the requirements is listed. The functionality of your system is explained. The names of the Python modules and the classes in the modules are listed and their purposes are given. A flow chart will be a good addition here.

• **Section 2** - Design (30%):
The UML class diagram, with attributes/data, behavior/function, class relationships, and cardinality for the above business case is provided. All relationships between classes are explained and assumptions are noted.

• **Section 3 –** Plan (20%):
In this section, the algorithm to explain the logical flow that will drive the use of the system is provided. This section describes the different modules, e.g. the module that interface with the employee, the module that manages registration, and the module that administrator view,… etc. This section also provides a snapshot of how the various input and output screens will look like. Mainly this sections is formulated as sudo code.

• **Section 4 –** Implementation and Test Cases (20%):
In this section, provide the Python classes for the identified objects. The classes need to have sufficient documentation and pseudo-code to explain the functions. A fully functional program is required for this section. Also, describe test cases of how the system will be tested. The test cases must ensure that all requirements in the project are tested. Include screen shots of the code while explaining here.

• **Section 5** - Self-reflection (10%):
In this section include a reflection on what was learned in this project, the challenges faced while working on this project, and how they were overcome. Include contribution of each team member:
1. <Student ID> <Name>: <Explain contribution and module worked on >
2. <Student ID> <Name>: <Explain contribution and module worked on >
3. <Student ID> <Name>: <Explain contribution and module worked on >

2. Lab Component

As part of the course completion requirement of SWE 321, the above project must be implemented in Python. The completed Python project is submitted on BlackBoard as part of the SWE 321 course. The code must be well organized and documented. The use of good documentation, the proper naming convention for files, classes, and variables is essential to integrate the modules of the project. A fully functional code with all requirements implemented is required to avail the full score for the lab component. The overall marking scheme rubric for the Python code is given below, which will be used to evaluate a fully functional error-free code.

Marking Scheme:

Name	SWE 321 Project Rubric
Description	
Rubric Detail	

	Levels of Achievement					
Criteria	Exemplary (A)	Accomplished (A-/B+/B)	Developing (B-/C+/C)	Beginning (C-/D+/D)	0 (F)	None
Problem Analysis Weight 20.00%	90.00 to 100.00 % The solution fully meets the criteria	80.00 to 89.00 % The solution mostly meets the criteria	70.00 to 79.00 % The solution moderately meets the criteria	60.00 to 69.00 % The solution is less then average to meet the criteria	0.00 to 59.00 % The solution is lightly meeting the criteria	0.00 to 0.00 % No solution.
Design Weight 30.00%	90.00 to 100.00 % The solution fully meets the criteria	80.00 to 89.00 % The solution mostly meets the criteria	70.00 to 79.00 % The solution moderately meets the criteria	60.00 to 69.00 % The solution is less then average to meet the criteria	0.00 to 50.00 % The solution is lightly meeting the criteria	0.00 to 0.00 % No solution.
Plan Weight 20.00%	90.00 to 100.00 % The solution fully meets the criteria	80.00 to 89.00 % The solution mostly meets the criteria	70.00 to 79.00 % The solution moderately meets the criteria	60.00 to 69.00 % The solution is less then average to meet the criteria	0.00 to 59.00 % The solution is lightly meeting the criteria	0.00 to 0.00 % No solution.
Implementation and Test Cases Weight 20.00%	90.00 to 100.00 % The solution fully meets the criteria	80.00 to 89.00 % The solution mostly meets the criteria	70.00 to 79.00 % The solution moderately meets the criteria	60.00 to 69.00 % The solution is less then average to meet the criteria	0.00 to 59.00 % The solution is lightly meeting the criteria	0.00 to 0.00 % No solution.
Self-reflection Weight 10.00%	90.00 to 100.00 % The solution fully meets the criteria	80.00 to 89.00 % The solution mostly meets the criteria	70.00 to 79.00 % The solution moderately meets the criteria	60.00 to 69.00 % The solution is less then average to meet the criteria	0.00 to 59.00 % The solution is lightly meeting the criteria	0.00 to 0.00 % No solution.