CT60A0203 – Introduction to Programming with Python

Project work: Vehicle Rent System: Part TWO

Due date and time: **3rd of December 2021 11.59 PM (REVISED)**

Submission: Submit your code file and supporting text files in the Moodle- project work page.

Course Plagiarism Statement: Plagiarism in oral, written, or visual presentations is the presentation of the work, idea, or creation of another person, without appropriate referencing, as though it's one's own. Plagiarism is not acceptable and may result in charges of academic misconduct which carries ZERO marks in the assessment task(s). It is also a disciplinary offence for students to allow their work to be plagiarized by another student.

A Car Rental company rent vehicles with a wide range of prices. The rental company rents two types of vehicles namely ordinary (O) and premium (P) for its customers with varied range of prices and discounts. The ordinary cars which are usually normal cars with basic facilities such as A/C, heater, and additional tires (including winter tires). The car rental company also supplies premium cars, which are usually luxury cars (will be dealt here). The premium cars are given a free mileage allowance for each day of rent. The free mileage allowance varies from one premium car to another. After renting a premium car, renters may also choose to rent additional accessories (such as built in mini-fridge, GPS navigator and more). For example, a BMW vehicle renter can rent GPS navigator and a car seat at additional 20.00 Euros for each day of rent. The customers can view the car details including status (A - available for rent / R - rented or not available).

Part II: You are required to update your interactive menu-based program to manage (see below) the renting of vehicles and related transactions done in Part I. The requirements are specified below.

(i) Update a menu driven program that performs the common operations as given here

Vehicle Menu

Display Cars	1
Add/delete Vehicle*	2
Rent Vehicle : Ordinary or Premium	3
Complete Rent	4
Reporting vehicle information*	5
Exit	6

*** Marked are new addition to the menu; and red colored options should be updated as per specifications given below.**

Option 1: Display Cars

The procedure **displayCars()** should be updated with user input that should prompt the user to enter whether the user/staff wants to print/display vehicle details that are available for rent ("A") or rented ("R"). That is, print the details of vehicles based on user input (A/R) from the text file "**Vehicle.txt**". [add some vehicle details via option 2 – Add/delete vehicle in the "**Vehicle.txt**"].

User input must be checked, and program should not be crashed due to invalid input given by the user. Wrong input should be taken care by displaying appropriate error messages (exception handling) and return to main menu.

Option 2: Add/delete Vehicle*

This option should prompt the user to input whether the user wants to add a new vehicle or wants to delete a vehicle exist in **"Vehicle.txt"**. The following sub programs should be called based on user input.

(i) **Add_Vehicle()** that prompts the user to enter *vehicle Id* (which should be unique), name, type, mileage allowance, daily rent rate and status. All user input must be appended to "**Vehicle.txt".** Then user must be asked whether the user wants to add one more vehicle or not. If so, **Add_vehicle()** must be invoked until user wants to. Else, should return to main menu.

(ii) **Delete_Vehicle():** This subprogram takes vehicle ID only as parameter to check if such vehicle exists. If no such vehicle exists, then it should throw an appropriate error message (exception handling) and back to main menu. Else the identified vehicle information must be removed from the file **"Vehicle.txt"** but should be written/added into a new file called "**deletedVehicles.txt**". After successful deletion of vehicle, it should ask whether user wants to delete another vehicle info. If so, the deletion of vehicle process must be continued until the user wants to. Else, it should be returned to main menu.

Option 3: Rent Vehicle (to be updated)

Prompt the user to enter the *Vehicle ID*. Then invoke the procedure **rentVehicle(VehicleID)** that takes Vehicle ID as parameter to check whether the vehicle is available for rent. Search the vehicle "**Vehicle.txt**". If no such vehicle exists or the vehicle is on rent, print an appropriate error message (exception handling) and return to main menu.

If the vehicle is available for rent then ask for *renter ID* (usually it is a social security number /national id), odometer reading at the time of renting the vehicle and display the details for renter to proceed further. However, it should be noted, if the vehicle type is "P" then system should display available additional accessories (at least 3) and ask user to select those before printing the rent vehicle info [user selection is optional]. The rent for additional accessories is normally 20.00 € per day.

Example: renting Premium vehicle [it is same as Ordinary vehicle but accessories]

Car BKV-535 is rented to 310790-171A

******************* Vehicle Details ********************

vehicle ID=BKV-535|Description= Ford Ikon| Daily Rate=60.0|accessories = 20.0|Status=R| Renter ID=310790-171A|Date/time of rent=21/10/202110:46|rent starting odometer=124500

Then the status of the vehicle that was rented should be updated as **"R"** at "**Vehicle.txt".** In addition, **rentVehicle()** should call another procedure **rentDetails()** which should perform the following operations.

rentDetails(): It opens a file **"rentVehicle.txt"** and append the rental details once the **rentVehicle()** execution is successfully completed. Each successful rent vehicle process must be written in a separate row. The details that should be written are, *Vehicle ID, Renter ID, date, and time the vehicle was rented*, (should be current date you are not allowed enter manually), *initial odometer* (int) of the vehicle and *accessories info* if taken. After successful completion of text file update, it should display the message "Renting vehicle is successful" then should return to main menu.

Option 4: Complete Rent [changes in the calculation]

Selecting this option should prompts the user to enter *Vehicle ID* as input. Then invoke the procedure **RentComplete()** which takes Vehicle ID as parameter to check whether the vehicle is actually rented. Search the status of the vehicle at "**Vehicle.txt**".

If no such vehicle exists or the vehicle is not on rent, print an appropriate error message and return to main menu.

If rented, then rental charges must be calculated based on number of days the vehicle was rented (return date which should be a current date – start date + 1). In addition, number of kms the vehicle run (end odometer – initial odometer) must also be calculated. Apart from these, if the rented vehicle was premium and user obtained any accessories then those charges should also be calculated. The rent charge for the vehicle that returned will be:

(Daily rent rate * number of days) + (number of days * 20.00[if any accessories used] +(no. Kms run * 0.020 [O]/0.025[P]).

The status of the returned vehicle must be changed as **"A"** at **Vehicle.txt**. Then the results must be displayed as given below.

Car BKV-535 is returned from 310790-171A

******************* Vehicle Details ********************

vehicle ID= BKV-535|Description=Ford Ikon|Daily Rate=60.0|Accessories=20.0*

Renter ID=310790-171A| Date/time of return =24/10/2021 12:14| rent starting odometer=124500|rent end odometer=128500*| Kms.run=4000|Rental charges=420.00 €

In addition, the afore displayed details must be appended into a file called **"Transactions.txt".** Once the appending is successful it should display the message "Car BKV-535is returned" and return to main menu.

* It should be noted user may or not choose accessories. If not, then it should be registered as 0. Similarly, the end odometer value must always be greater than the initial odometer. Otherwise, error exception must be caught and ask user to enter correct value to proceed further.

Option 5: Reporting vehicle information* [Self-study]

Data visualization is the discipline of trying to understand data by placing it in a visual context so that patterns, trends, and correlations that might not otherwise be detected can be exposed.

Python offers multiple great graphing libraries that come packed with lots of different features. You can use any library that supported by Python to display the data stored in the files created for this project.

By selecting option 5 the user is prompt to choose the following options in order to report in graphical form. That is the results of option should be viewed as charts.

(i) Display the number of vehicles of each type: Ordinary and Premium – Bar chart
(ii) Display the income earned by vehicle company at least for last 3 months as line graph.

It should be noted, data for aforementioned tasks must be obtained from text files created for this project only. The computed data can be saved as csv/xls/dat or simple text file for data visualization.

Option 6: Exit

Exit from the menu by displaying "Thanks for using Car Rental System. Bye! Bye!".

Additional requirements:

User input must be checked, and program should not be crashed due to invalid input given by the user. Wrong input should be taken care by displaying appropriate error messages and return to main menu. You can define your own sub programs for better coding style. You can also initially change the rent dates in the file to check your rent complete calculations.

Items	Check if completed	Possible points	Actual points
Main menu display – using procedure [for example Displaymenu()] - Updated		2	
Option 1: displayCars() – based on user selection		4	
Option 2: Add/delete Vehicle (i) Ask for input and check for vehicle id is unique + writing info (ii) repetition of add vehicle – loop (iii) Check vehicle id if exists for deletion + exception handling (iv) deleting vehicle and updating in Vehicle.txt and Writing into new file- deletedvehicles.txt		 4 2 4 5	
Option 3: Rent Vehicle (updated) (i) If the vehicle info is Premium, then list options for user to choose and proceed further (ii) Invoking *rentDetails()* and writing/appending rent information into a file "*rentVehicle.txt*" and return to main menu + displaying all details including accessories info		 4 4	
Option 4: Complete Rent (i) Changes in the rent charges calculation (ii) Checking odometer reading and related input (iii) Updating into file with accessories info if premium (iv) Displaying with all details including, kms run and accessories rate		 2 4 2 2	
Option 5: Reporting Vehicle information- Self study		10	
Option 6: Exit		1	
(i) Input check – displaying appropriate message and return to main menu if the input is mistyped. (ii) used simplified coding: for example, defined some more subprograms for file handling to avoid repetition.		2 3	
Total (for Part II only)		55/100	

Part II is focused on updating current menu with some more options and calculations + Exception handling + Visualizing the results in graphics form (self-study)