Sid : 1820376

MOD003212

App User Guide

**I attempted a high level of functionality.**

**Objectives:**

The main objectives of Petrol Truly Unlimited Ltd. Petrol Station’s Management App are to manage the details of vehicles, fuels, sales and transactions. It identifies vehicles waiting on queue and monitors vehicles fueling on the pumps while keeping count of all the liters sold, how much of each fuel type, how many vehicles got service and the revenue made by the Petrol Station while keeping a detailed transaction list with the serviced vehicles.

**Synopsys:**

You work for a petrol station? Are you tired of keeping a hand-written record of the transactions? Those days are over, we introduce the new Petrol Station Management App, where we keep a count of the vehicles that visited you every day, we keep an eye on what fuel they used and how much, we will tell you which pumps get used the most and we even calculate your salary for you. If you’re interested give us a call and we will invite you to test it for yourself, bring your manager and we promise he’ll love it and you won’t have to worry about transactions anymore.

**How it works:**

Everything is automatically recorded from queue to pump then to transactions as each client fuels his own car then pays at the pump. The attendant is there to help the clients and make sure they don’t leave the pump without paying, but everything in the app is done automatically. The Management App is here to record and calculate the attendant’s salary and commission as well as the amount of money received each day and, the number of liters of each type of fuel sold per day. This way we can see how much the pumps are used so we know when they’ll need to be serviced this way, we assure a smooth functionality of the petrol station.

Now that the busy season is here Petrol Truly Unlimited Ltd has increased its sales to the point of continuous work, whenever a pump becomes available someone else starts using it. Because of this huge amount of transactions, we decided that our pump attendant will receive 1% of each transaction added to his salary of £2,49 per hour.

**Limitations:**

The fuel prices will be automatically updated every day by our support team which means one of the petrol stations team members has to send an email every morning with the new prices as we do not calculate each petrol station commission on each fuel. This makes it really easy to use by any member of staff because they don’t have to interact with the app themselves.

We decided to not keep a list of the vehicles that left without service as none of their information’s would be important to us, but we keep a count so we can try and give an idea of how much a station should expand in order to service everyone in time.

We did not include a ‘logger’ option yet, therefore the staff must copy the totals at the end of the day somewhere safe, because once they close the management app the information will be lost.

**Activity diagrams:**

A close up of a device

Description automatically generated

Here is a simple representation of the activity of the vehicles in this application, showing that vehicles either get fueled or leave without service and in both cases, they’re recorded to the total numbers.

A close up of a device

Description automatically generated

The pumps activity diagram is a bit more complex first checking availability in the queuing system which does not let vehicles pass to a further pump if the one/s in front are being used. After assigning a vehicle the pump will automatically fuel the vehicle with the specified fuel type, adding the litters to both totals the individual fuel and total fuels. Moving the information to the transaction section where is displayed the number of the vehicle ,vehicle type, fuel type, liters bought, price paid and the pump that it got serviced at.

**Display map:**

A screenshot of a computer

Description automatically generated

I decided to go for a simple and clear display, no distracting colors as some members of the staff can be color blind therefore, I won’t be able to color-code elements. It has a nice flow from left to right, from creating a vehicle to fueling it, adding the totals and then listing the transactions.

A close up of a logo

Description automatically generated

On the left of the screen we have the vehicles queue which displays the number of the vehicle for this day, the vehicle type and the fuel type.

A screen shot of a computer

Description automatically generated

On the right side of the screen we can see the transaction list showing the number of the vehicle, the vehicle type, fuel type, liters fueled, the amount paid and the pump they used. They are displayed in the order they left the pump in, the limit of transactions displayed is 30, after that the first transaction will be removed from the screen and the last transaction will be displayed.

A screenshot of a cell phone

Description automatically generated

In the center top we have the pumps, here you have an example of both stages of the pumps, being “Available” or “Busy”.

The pumps have a queuing system not allowing cars to go for the second or third pump is the first is being used, or use the third if any of the previous ones are in use. As we can see the 3rd and 9th pumps cannot be used as the 1st ,2nd, 7th and 8th are being used.

A screenshot of a cell phone

Description automatically generated

And in the center bottom we have all the totals divided in 4 categories.

The 1st is a count of the vehicles serviced and the ones that left without service.

The 2nd category in an overall count of the liters sold and the money gained from these liters.

The 3rd category is an individual total of each fuel type showing how much of each was sold.

And 4th is the attendant’s salary and commission, clearly seeing the commission is £2.13 out of £212.58 the total revenue today, the commission gets rounded by the system.

**Pseudocode:**

* **Create vehicles:**

//The vehicle is created inside the Data class in a timer adding the vehicle to the vehicle list

Create new timer;

The timer will refresh between a randomly generated time between 1500 and 2200 milliseconds;

When the timer refreshes it will create a new vehicle and add it to the vehicle list;

Start timer;

//Each vehicle will be different as they have many attributes as shown in Vehicle class

Get vehicle type ()

From a string array with multiple options;

Randomly choose one index from the string array;

Return one type of vehicle;

Get fuel type ()

If

The vehicle type is “CAR” then the fuel type will be “DIESEL”,”UNLEADED, or “LPG”

From a string array, the fuel will be randomly chosen and returned as the car’s fuel;

If

The vehicle type is a “VAN” then the fuel will be “UNLEADED” or “LPG”

From a string array ,then the fuel will be randomly chosen between the two;

Else

//In case the vehicle type is a “HGV”

The fuel type will be “DIESEL”;

Get fuel needed()

//Each vehicle is created with some fuel already in the tank, here we decide how much are they going to need to fill up the tank, therefore how much fuel they’ll buy from us

// The old fuel cannot be more than 25% of the tank capacity

If

The vehicle type is “CAR”;

The old fuel will be randomly chosen between 1 liter and 10 liters;

The fuel needed will be calculated out of the tank capacity of a car (40 liters) – the old fuel;

Returning the fuel needed;

If

The vehicle type is “VAN”;

The old fuel will be randomly chosen between 1 and 20 liters;

The fuel needed will be calculated from the tank capacity (80 liters) – the old fuel;

Else

The vehicle type is “HGV”;

The old fuel will be randomly chosen between 1 and 37 liters;

The fuel needed will be calculated from the tank capacity (150 liters) – the old fuel;

Get fuel time()

Taking the number of liters needed to be fuel times 1.5 (the pumps can dispense only 1.5 liters per second);

Returning the number of liters that can be fueled each second;

* **Assigning a vehicle to a pump and releasing after fueling:**

//I assign vehicles to pumps in the Pump class

Create a new timer;

The timer interval is the vehicle’s fueling time (fuel needed \* 1.5 liters per second) \* 1000 (per second);

The timer will not reset;

The timer will release the vehicle after fueling(adding to the serviced total, adding to the total revenue, the total liters sold and adding the transaction to the list);

The current vehicle is null so it will be realized from the pump;

* **Console display:**

Drawing the vehicle queue list(vehicle ID, vehicle type, vehicle fuel type);

The totals for each fuel type, total fuel, total vehicles serviced, vehicles left without service, total revenue;

Displaying the pumps and their availability;

Listing the transactions and making a limit of 30 transactions displayed at a time;

**Software specifications:**

This app was written using the .NET core developed by Microsoft providing a fast platform for creating server apps that run on Windows, Linux, and macOS making it easily accessible. Its written in C# language in in Visual Studio.