

**Introduction to Programming**

Student ID: 1839616

ANGLIA RUSKIN UNIVERSITY

Date of submission: 18/05/2020

MOD003212 TRI1-2 F01CAM

**Table of Contents**

**User Guide and Design………………………………………3**

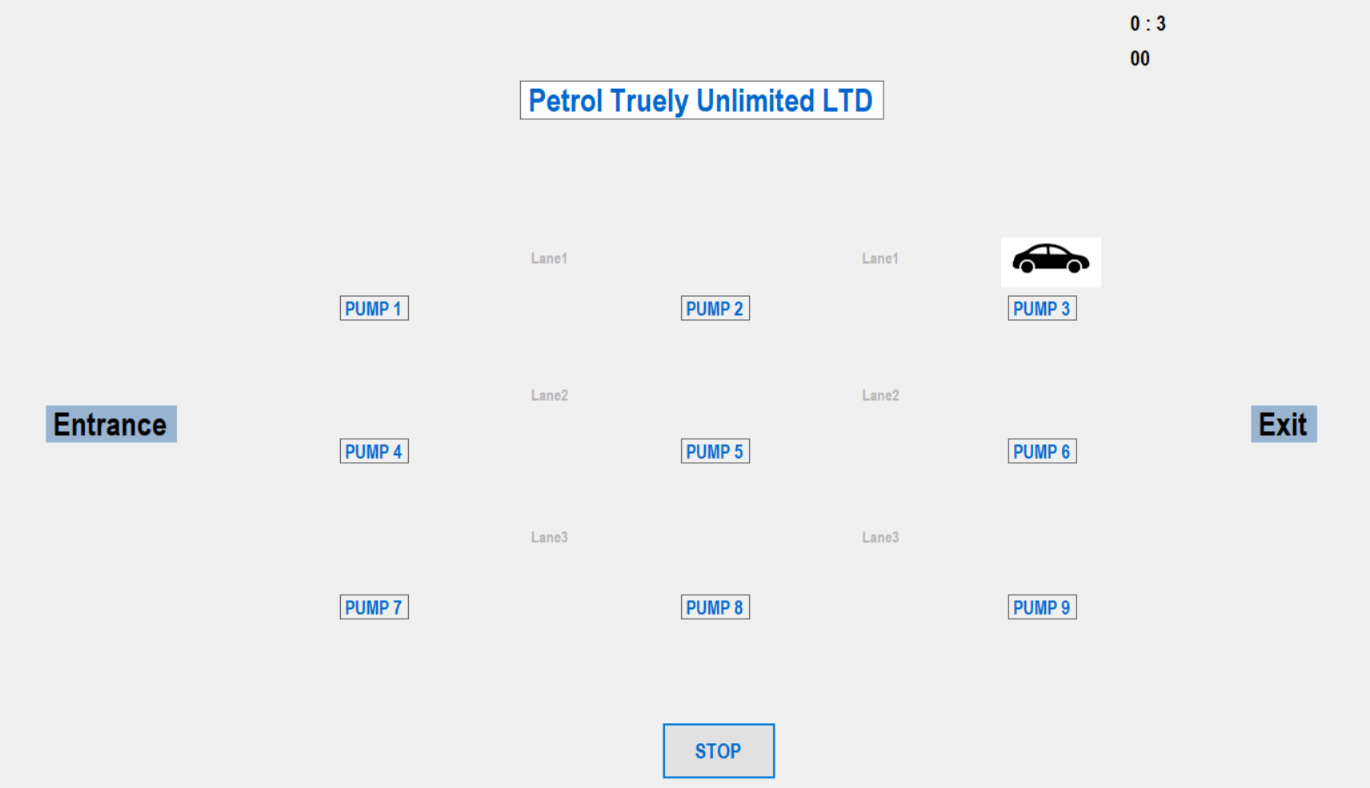
**Testing…………………………………………………………..6**

**Pseudocode…………………………………………………….9**

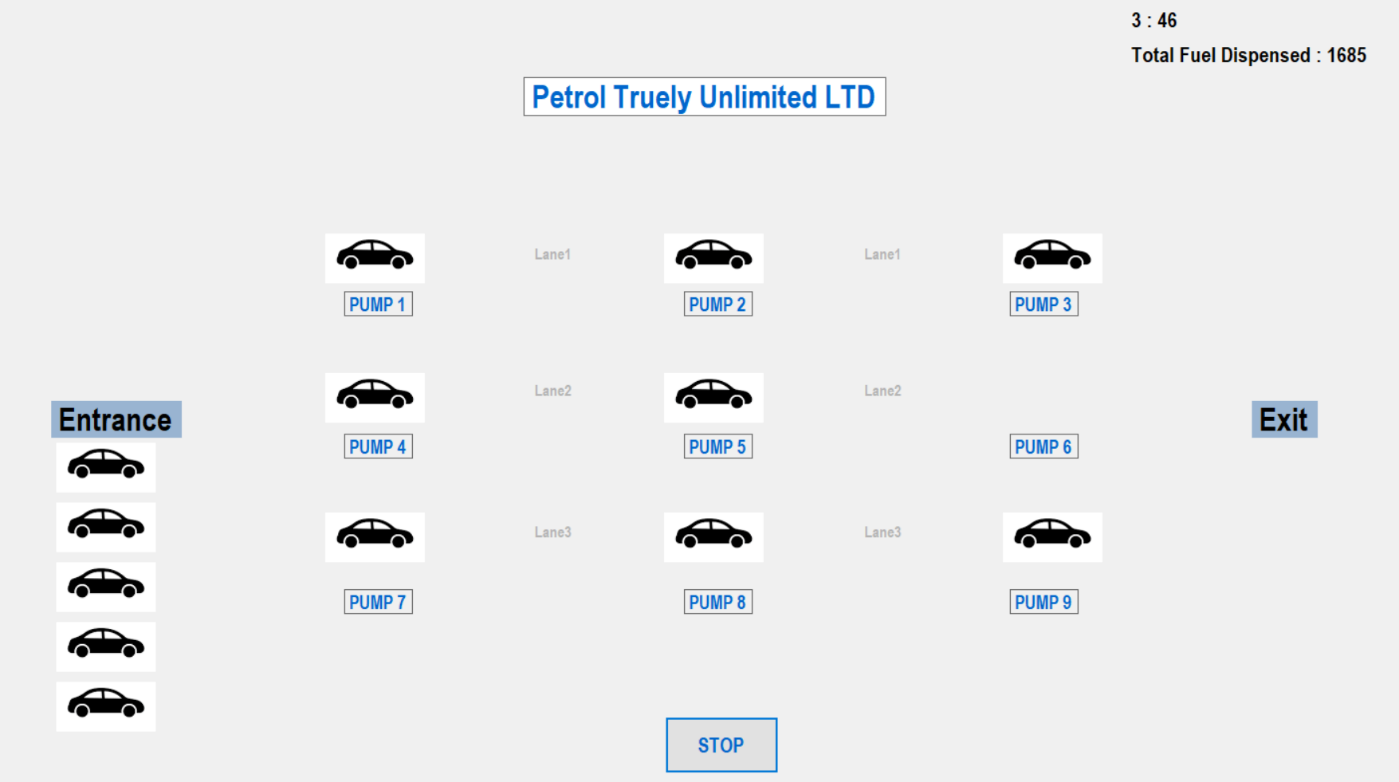
**Functionality Level Implemented: Very High functionality**

**Component1\_User Guide and Design -**

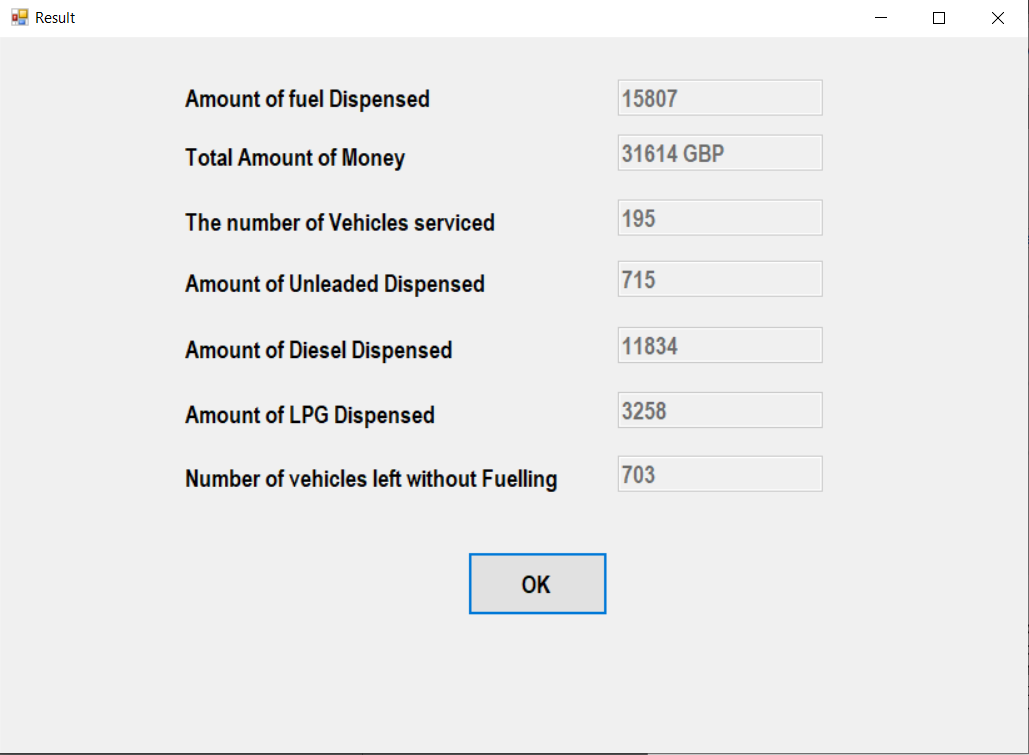
Hello, I am the owner of Petrol Truly Unlimited Ltd. We have recognised the previous difficulties you have faced while your visit to our petrol station. To help better our services we have built an application that demonstrates how our petrol station works with the at most efficiency and we will also provide you with a detailed description on how the application works and how you can use it. Hope to see u at Petrol Truly Unlimited Ltd soon.



The screenshot above shows that layout of our application. Our petrol station consists of 9 fuel pumps which are clearly labelled and the pumps are distributed among 3 lanes as shown in the screenshot. There is a counter on the top right of out application that shows how long the application has run for and also displays the Total Fuel Dispensed.



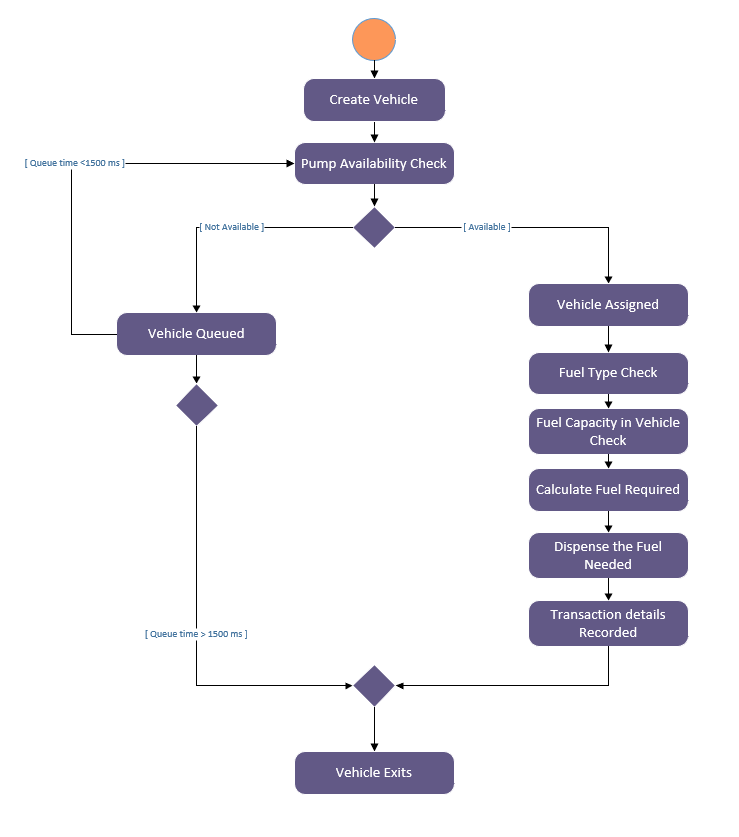
This screenshot shows how the application looks like when it is being run. When our pumps are full, the vehicles are required to queue up. The vehicles are queued up under the “Entrance” tab as shown in the screenshot above and the cars who have left without fuelling will be removed from the queue and those who wait are assigned to their respective pumps based on the fuel type they desire to fill.



After the user has clicked on “STOP”. This displays a result tab that shows the user all the information such as total amount of fuel that was dispensed, each fuel type that was dispensed and also how much of it, it also shows the number of vehicles that were serviced and also shows the vehicles that left without fuelling.

That was a basic outline of how the program functions and the instructions that need to be followed for it to work efficiently.

Activity Diagram -



The above screenshot shows a simple activity diagram for the application created.

**Component2\_Testing –**

|  |  |
| --- | --- |
| **Test** | **Actual Outcome** |
| Interface displays and updates correctly | The interface shows all the required information and it also updates every 50 seconds. |
| Vehicles are assigned to available pumps | Vehicles are being assigned to available as the vehicles are queueing up and waiting to be assigned. |
| Vehicles are removed from queue after waiting too long | Vehicles who waited for too long have left the queue as the vehicles are beginning to queue up again |
| The Transaction details are recorded and displayed on a form | The result form is displayed and information shown is correct. |

**Component3\_Pseudocode –**

* Vehicle Class

A class named “Vehicle” is created

{

A string named “type” is created for vehicle type

An object is created in the “Fuel” class

A method called Vehicle is created.

{

As a new vehicle is created a new fuel type is also ……...created.

}

A vehicle has been assigned a parameter “v”

{ //Vehicle is assigned with attributes

The vehicle has been given a type such as car, van. ……….etc.,

The vehicle has been assigned a fuel type.

}

{

Creating a method called “GetCarType” to use it in the ……….form design later on.

The cartype is returned to the form.

}

}

* Vehicle being added to pumps

Here we look into the class named “Lane”

A method called “CheckAvailability” is created

{

a for loop is created to keep checking the pumps

{

If pump availability is “Y”

// pump 1 and 2 are checked so car can drive to pump 3

{

For loop is created to keep checking pump 3

{

If pump is not available

Break the program

}

If pump 2 is available

Vehicle is assigned to pump 2

Else

return 0;

}

}

return 0; as no pumps are available

}

* Globals Class

A class named Globals is created

{

Number of pump in lanes to be kept constant – 3

Number of lanes to be kept constant – 3

Amount of fuel dispensed = 0;

Number of vehicles that left without fuelling = 0;

Amount of unleaded fuel dispensed = 0;

Amount of diesel dispensed = 0;

Amount of LPG dispensed = 0;

Number of vehicles = 0;

}

* Forms

The Windows generated form design has been used

The pump layout and the lanes are all labelled

The images of vehicles are used

Button “STOP” is created to end the application

* Transaction class

Class Transaction is created

{

A string data type is used to assign the type is created

A double data type is used to assign the No of litres of fuel;

An Integer data type is used to assign the lane ID

An Integer data type is used to assign pump ID

Public Transaction

{

The variable type is assigned a parameter t

The variable NoOfLitres is assigned a parameter no

The variable LaneID is assigned lID

The variable pumpId is assigned pID

}

}

* Waiting Queue

A new vehicle is generated

If pump is not available

Vehicle is put in a queue

Else

Vehicle is assigned a queue

If vehicle is waiting for too long

Vehicle removed from queue

Else

Vehicle remains in queue

* Result form

Windows generated form design

All the transaction details are recorded and displayed

Text boxes are created to show what information is being displayed

Button “OK” is created to end the application.