**Jacob. S. Lummis**

**SID: 2213276**

MOD008365 – Introduction to programming

Contents

[Introduction 1](#_Toc120103178)

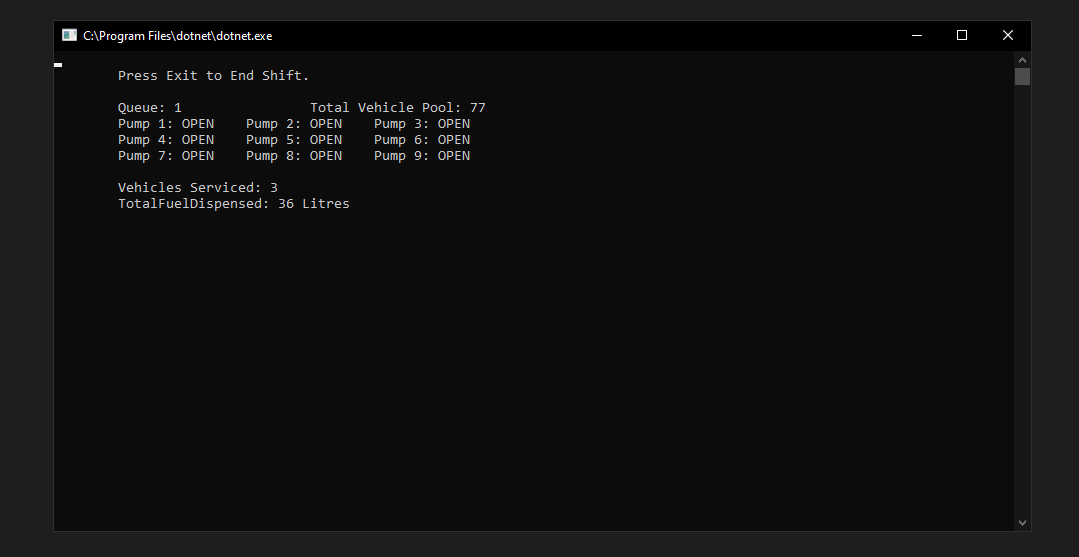
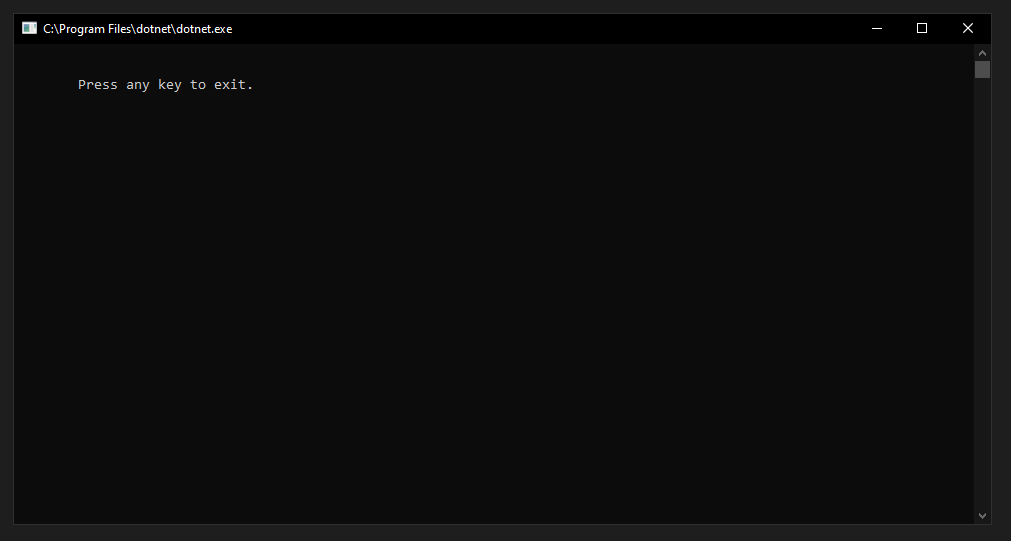
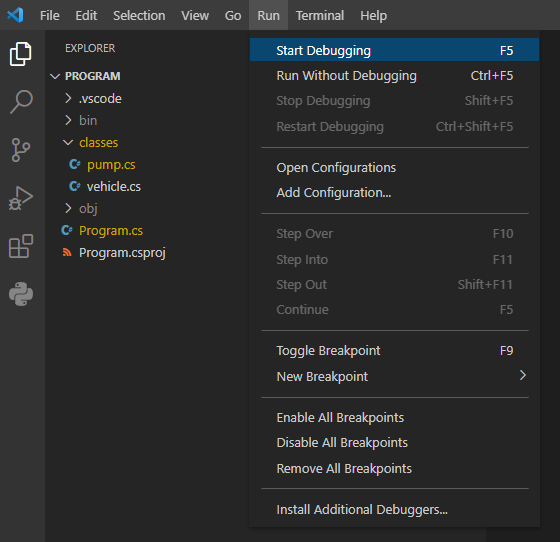
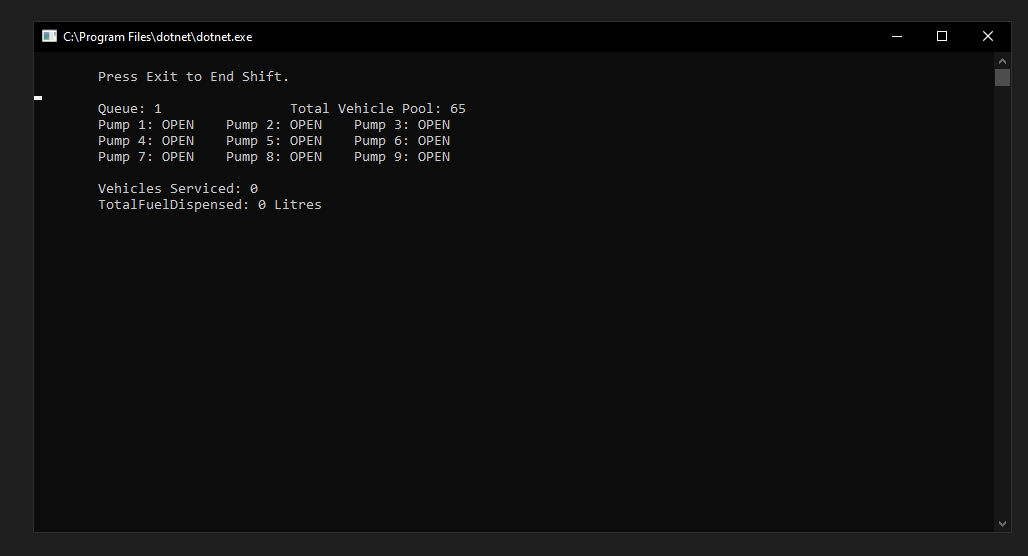
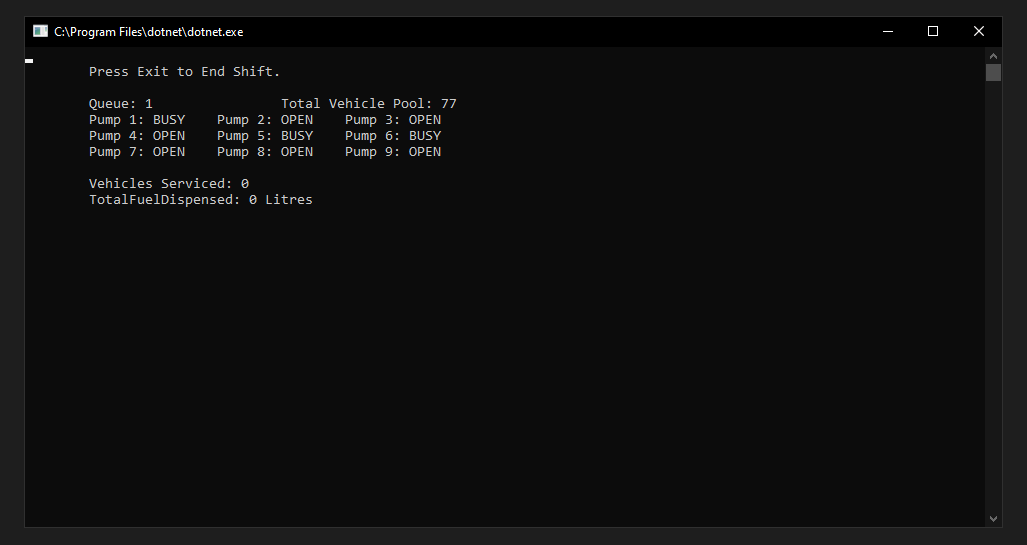
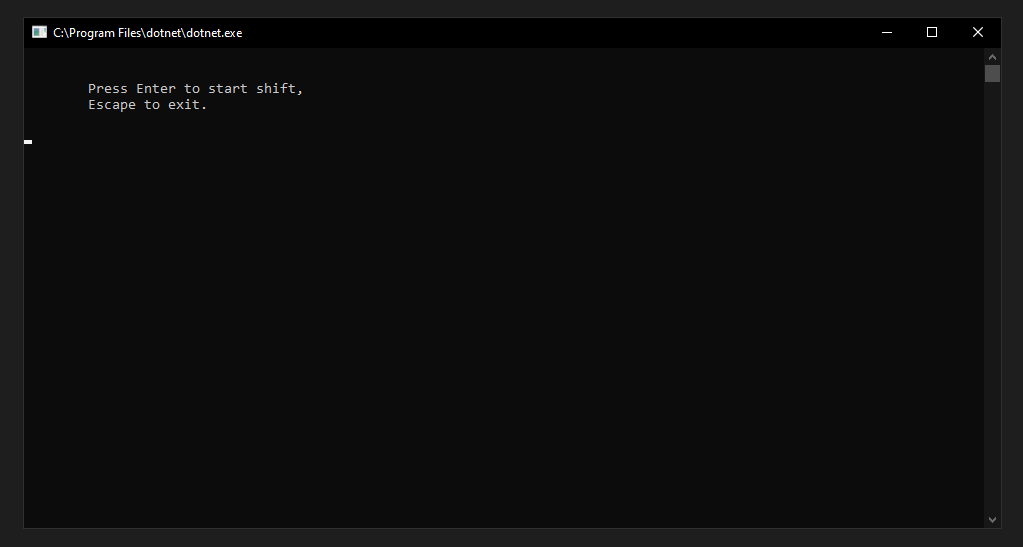
[User Guide 2](#_Toc120103179)

[Test Plan 3](#_Toc120103180)

[Bibliography (Resources used to develop program) 4](#_Toc120103181)

# Introduction

This report describes the ‘Petrol Station Management’ application written to satisfy the brief provided in “MOD008365 Assignment 0102 2022.docx”. The application written was aimed and succeeds at meeting the Core functionality criteria.



After loading the program, either by double left-clicking on its executable icon in a full program release, or by running the “start debugging” option from within VS code, the user; a simulated ‘Fuel Attendant’, is presented with a basic start screen, asking that the user presses the enter key to continue to their ‘shift’, or that they press the escape key to exit.

The limited-scope implementation of the application generates a pool of 81 vehicles, generating 1 every 1.5 seconds. Vehicles from this pool are then added to the “Fuel Station’s” waiting queue, which at this level, is 1 vehicle.

By pressing one of the corresponding number keys on their keyboard (or numpad keys with Numlock on), the Fuel Attendant can send a waiting vehicle to the pump associated with that number. The Vehicle will then be passed from the Queue to the Pump, causing its status to change from “OPEN” to “BUSY”, wherein the vehicle will ‘Fuel’ for 8 seconds, before leaving the pump and thus, the fuel station. The pump will then revert to “OPEN”, while the fuel station will update the number of vehicles serviced and associated litres pumped. Because at this level, the fuel time is fixed, and the pump rate is assumed to be a constant, that means for every vehicle serviced, 12 litres of fuel are dispensed. Additionally, as there is no differentiation between fuel or vehicle types, only 1 kind of vehicle & 1 kind of fuel are being served or dispensed at the fuel station, respectively.

# User Guide

Program users should be familiar with standard keyboard layouts and usage. The program only registers inputs from the ‘Enter’, ‘Escape’ and number keys 1 through 9. Any erroneous inputs will not cause program errors.

Due to how the vehicle pool population timer is implemented, swift users may find themselves waiting up to 3 seconds before the first vehicle is generated and transferred to the Queue for pump assignment.

# Test Plan:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **TEST** | **DESCRIPTION OF TASKS** | **STEPS TO EXECUTE** | **EXPECTED RESULTS** | **PASS** | **FAIL** | **DEFECT / COMMENTS / ADDITIONS** |
| **APPLICATION 1:** | |  |  |  |  |  |
| 1 | Does the Program Start? | Perform “Start Debugging” from within VS Code | Console Window Opens, displaying start screen. |  |  | Pressing the Escape key (Esc) will progress the program directly to the exit screen, where any keypress will close the console window. |
| 2 | Will the Program Progress to a “Shift”? | Press the Enter key | Console displays “Shift” Screen, the “Vehicle Queue” and “Vehicle Pool” being to fill up. |  |  | For this limited scope demonstration, only 81 vehicles are generated in total. |
| 3 | Does Pump Assignment Work? | Press **any** of the number keys, from 1 to 9 | If the Vehicle Queue has at least 1 car in it, it will be assigned to the corresponding pump, which will switch from “OPEN” to “BUSY” |  |  | After 8 seconds from the pump becoming “BUSY”, it will switch back to “OPEN” and the vehicles serviced, and fuel dispensed counters will increase. |
| 4 | Can you exit program at any point? | Press the Escape (Esc) key | Program will go to exit screen, where any keypress will close the console window |  |  | N/A |

|  |  |
| --- | --- |
| **GENERAL QUESTIONS / COMMENTS** | |
| Test 2: | Due to how the program loads, it may take longer than 1.5 seconds for the first vehicle to generate and be transferred to the que, if the user progresses from the basic start screen particularly quickly. Vehicle generation after that point has been consistent during in-house testing. |

# Bibliography:

Many thanks are given to the editor(s) and contributors of the learn.microsoft.com site, for their copious if sometimes opaque examples and explanations.

I mainly frequented the C# and .Net documentation, with this link as an initial starting point: [C# docs - get started, tutorials, reference. | Microsoft Learn](https://learn.microsoft.com/en-us/dotnet/csharp/).

Additional thanks are given to my Father, my course lecturer, and my module/course-mates, for the support, patience and brainstorming opportunities & sanity checks they provided when writing my code for this Assignment.