**CST1510 Final Coursework**

## 1. Brief Task Description

Design and implement a 'Vending Machine' for drinks, snack, masks, train tickets, fruits, ice-cream or some other product of your choice. Client/ Server-side software system should be built in Python to interact with the user on the client-side and the data storage should be process on the server-side. The client program should have an intuitive user interface to view available items and prices, create order, (add, remove item), manage order and payment. The server program should allow one client (multiple clients for the advance version of the software) interact with the current stock (database) of a product and interact, update the stock based on the orders done by the client.

## 2. Submission

The submission will comprise of two parts: First, submission of the code and pdf file with design in week 23, second part will be zoom presentation of the code in week 24.

You must submit a single zip file of all required source code and a single PDF file with a UML diagram and running instructions by Friday, end of week 23.

The zip file should include:
Separate python file for Client, separate for Server, any other files you wish to implement based on your design, a CSV or txt or (SQL to create and populate a MySQL database) file.

*Note:*
*Code must be build using python3 and it must be able to compile from the terminal or CMD. Running instructions should be provided in the PDF file.*
*As anonymous marking will be applied, you should not include your name in your source code or pdf.*

## 3. Scenario

A vending machine is an automated machine that is intended to provide users with a diverse range of products: snacks, beverages, cupcakes, newspapers, tickets, etc. A vending machine dispenses a product to the users based on the selection of the product and payment.
In a nutshell, its basic function is to flawlessly provide users with a diverse range of products at any time.
*Scope: please choose a selection of one product.*

**The Vending machine must display available product including product name, price and ID number**
(Extra: it could contain a graphical representation of stock being available in the Vending Machine. Graph or pie chart)

When **generating an order**, the process should allow the user to:

- Start with a welcome message:

- Enter the code of the item they wish to buy followed by a number of chosen items.
- The display should display the name of the item, number of the items, individual price and total price.
- After each operation the machine should allow a list of options: “add another”, “finish and pay” or “cancel”.
- “Add another” option should allow the user to add a new item and again display the above operations.
- “Finish and pay” should display the final receipt containing the list of items, individual price, total price and lead the user to make a payment.
- Please note that the payment process should be just mimicking the actual process by providing the client with payment options: cash or card and calculate the change if cash payment is made.
- Finally, “Thank you” and “Goodbye” message should be displayed
- The “Cancel” option should display a message with apology “Sorry, we could not provide you with your choice today. We hope to see you soon again. Have a good day!”

## 4. Software Description

Start by designing your Vending machine using a UML diagram or similar design. (Please note that there should be a demonstration of some thought process put into the design of the software. The UML diagram does not have to be using precise UML notations. For that some explanations should be provided)

**Data storage:**

You could have a simple .txt file or .csv file or full implementation of MySQL data containing all available stock. The file should contain the product id number, name, price, number of the same products.

You should also have another .txt or .csv file for storing orders/transactions

**Server**

Design and implement a (command line) server program which connects to the, previously implemented, database (in a form of text or CSV or MySQL). The server should have methods to achieve the required functionality:

- Get the transaction and stored it in the transactions.txt or transactions.csv file.
- Get the items from the transaction and remove and/or update the available_stock.txt accordingly.
- Update the client with the currently available stock.

**Client**

Design and implement Python file to be used as the client. The Client should contain the Vending Machine user Interface (UI), in a form of GUI or text-based.

This GUI or text-based program should allow the user to see all available products with their numbers, prices and availabilities, generate the order and send the transaction to the server.

After each transaction, the GUI should be updated with the current stock availability.

To generate the user order, steps listed in the scenario above (Part 3) should be taken into consideration.

## 5. Marking

The code and pdf file will be marked according to the below-attached marking scheme. Marking will be anonymous, i.e. person marking the code will not see the name of the student while marking.

The grades will be published before week 24. The second part of the submission will take place in week 24. The 2nd part of the submission would be a Zoom demonstration of your work during which you will have the opportunity to prove your knowledge about the coursework python code.

| **Item:** | **Mark:** |
|---|---|
| • Some design<br>• .txt or .csv file containing product stock information's<br>• Server reading the data from the data storage<br>• Sending the data to the client<br>• Displaying the data on the client-side<br>• Generating a basic order / Text-based design | 40% |
| • Good design<br>• .txt or .csv file containing product stock information's<br>• Server reading the data from the data storage<br>• Sending the data to the client<br>• Displaying the data on the client-side<br>• Generating an order/ GUI design<br>• Sending transaction back to the server<br>• Storing the transaction into a separate .txt file<br>• Subtract the "Sold" items from the stock available text or CSV file<br>• Updating the client with the "Still available" items<br>• Allow for a new transaction to take place | 80% |
| In addition to the above functionality, there should be an implementation of a material that we have not covered, such as:<br>MySQL or visualise current stock item using matplotlib or some additional implementation such as all software using OOP or multiple clients. (One of the listed above would be enough) | 100% |

## 6. Academic Misconduct

This is an individual work, and you should complete it by yourself. You should not work as a group and any student should not submit the same or similar work (with only minor changes) as their own. Any material or ideas found online, in textbooks, etc should be properly referenced.

You should familiarise yourself with the university's academic integrity https://unihub.mdx.ac.uk/study/academic-integrity
and misconduct policy: https://www.mdx.ac.uk/about-us/policies/university-regulations

## 7. Extenuating Circumstances

There may be difficult circumstances in your life that affect your ability to meet the assessment deadline or affect your performance in the assessment. These are known as extenuating circumstances or 'ECs'. Extenuating circumstances are exceptional, seriously adverse and they are outside of your control. Please see link below for further information and guidelines:
https://unihub.mdx.ac.uk/study/assessment/extenuating-circumstances