Course Learning Outcomes:

Upon completion of this assignment, you should be able to:

CLO1	Translate simple problem statements into programmable solutions using flow chart/pseudo code (C3, PLO2).
CLO2	Comprehend knowledge of basic and advanced programming concepts (C2, PLO1).
CLO3	Show the ability to write computer programs for a given problem statement (P4, PLO3).

ASSIGNMENT DESCRIPTION

GROCERY STORE INVENTORY SYSTEM

Scenario: You are given a task to program a simple inventory system for a grocery store. The system will track the flow of items in the stores and prompt the user to replenish the stock when it is lower than a minimum quantity.

System Requirements: The system will be able to keep track of the item code, description, category, unit, quantity (available in stock), minimum (threshold allowed), and price (cost) of the goods+ in the grocery store. At the end of every business day, the inventory checker will perform a stock-taking to record the available quantity of items in the store. The purchaser will check the system to retrieve the list of items that is lower than the minimum threshold, to make orders to replenish the items. The example data of the items are listed in the table below.

Code	Description	Category	Unit	Price	Quantity	Minimum
10000	Milk 1L	Dairy	Box	7.00	30	5
20000	Beef 200g	Freezer	Pack	20.00	10	7
21000	Chicken thigh	Freezer	Pieces	10.00	15	5
30000	Apple	Fruits	Pieces	2.00	50	30
32000	Orange	Fruits	pieces	2.50	50	30
41000	Broccoli	Vegetables	pieces	2.50	23	10

The system should have the following functions:

1. **INSERT NEW ITEM.** The user will be able to insert one or more new item(s) into the system. Each item inserted will be temporarily stored in a list, and subsequently written in a text file. i.e. inventory.txt. The initial data you create should have at least 5 categories and 40 items. For example, your data may be stored in the following format:

10000	Milk 1L	Dairy	box	7.00	30	5
20000	Beef 200g	Freezer	pack	20.00	10	5
30000	Apple	Fruits	pieces	2.00	50	30
32000	Orange	Fruits	pieces	2.50	23	30

2. **UPDATE ITEM.** The user will be able to update the specific details of the item. For example, the user may update/change either the code, description, category, unit, price or minimum (threshold) of the item. *Hint: You may read the whole data to a master list, use change the specific details of an item, then rewrite the list to the file.*
3. **DELETE ITEM.** The user may be able to delete any of the items from the inventory. The user will key in the item code, the system will remove the item from the data file (inventory.txt). *Hint: You may read the whole data to a master list, use remove() function to remove the item, then rewrite the list to the file.*
4. **STOCK TAKING.** The user (inventory checker) can perform stock-taking for the items by keying in the item code. The system will print the quantity available, and the user may confirm or change the quantity accordingly.
5. **VIEW REPLENISH LIST.** The user (purchaser) will be able to view the list of the items that they should replenish. This function is supposed to list down the items with a **quantity lower than the minimum threshold** of the items.
6. **STOCK REPLENISHMENT.** The user (purchaser) will key in the item code, the system will show the quantity. Then, the user may add the quantity of the newly purchased items.
7. **SEARCH ITEMS.** This function allows the user to search the items based on the following few methods:
    a. Search item by description. The user will input the item description, the system will show the item details based on the description entered.
    b. Search the item by code range. i.e., code between 30000 and 39999
    c. Search items in a specific category. i.e., search all items in fruits.
    d. Search items in a specific price range. i.e., between RM10 to RM 100.
    For example:
    Price min: 5.00
    Price max:10.00
    Output:
    10000 Milk 1L Dairy box 7.00 30 5

EXTRA MARKS: LOGIN AUTHENTICATION AND ACCESS CONTROL MODULE.

Expand your system with user login and password authentication, the system should have 3 user types, which are **admin, inventory-checker, and purchaser**.

ACCESS CONTROL: Each user type will be able to access or perform different functions of the system as follows:

ADMIN: ALL Tasks.

INVENTORY-CHECKER: STOCK TAKING, SEARCH ITEMS.

PURCHASER: VIEW REPLENISH LIST, STOCK REPLENISHMENT, and SEARCH ITEMS**.**

The user won't be able to access the functions if he/she is not given the access right.

LOGIN AUTHENTICATION: To perform the user authentication, you must create a few more functions to input the username, password, and user type as follows:

1. ADD NEW USER: Only the Admin can perform this task. This function will allow the user to input the username, password, and user type. The data will be stored in a separate file. i.e. userdata.txt. For example:

   1 ahmad password admin

   2 lee abcd1111 inventory-checker

2. USER AUTHENTICATION: This function will allow the user to key in a username and password. The function will read from the userdata.txt file and check the username and password. Please take note the username and password are case-sensitive. Users will only be able to log in with the correct username and password entered. Failed login will prompt the message: **Incorrect password or username.**

1.0REQUIREMENTS

i. You are required to carry out extra research for your system and document any logical assumptions you made after the research.

ii. Validations need to be included to ensure the accuracy of the system. State any assumptions that you make under each function.

iii. You are required to store all data in text files.

iv. You are expected to use lists and functions in your program. Your program must embrace modular programming techniques and should be menu driven.

v. You may include any extra features which you may feel are relevant and that add value to the system.

vi. There should be no need for graphics (user interface) in your program, as what is being assessed, is your programming skill, not the interface design.

vii. You should include good programming practices such as comments, variable naming conventions, and indentation**.**

viii. In a situation where a student:

- ***Failed to attempt the assignment demonstration, overall marks awarded for the assignment will be adjusted to 50% of the overall existing marks.***
- ***Found to be involved in plagiarism, the offense will be dealt with in accordance with APU regulations on plagiarism.***

ix. You are required to use Python programming language to implement the solution. Use of any other language like C/C++/Java is not allowed. A global variable is not allowed.

x. Results of comprehensive testing are to be included in your document. The tests conducted shall take into consideration of all valid inputs and negative test cases.

2.0 DELIVERABLES

You are required to submit a softcopy of:

i. Program coded in Python – submitted as .py file.
   - Name the file under your name and TP number (e.g. KATHY_SIERRA_TP123456.py)
   - Start the first two lines in your program by typing your name and TP number (e.g. as follows):
     #KATHY SIERRA
     #TP123456

ii. Text files created through test data – submitted as .txt files.

iii. A documentation of the system (1000 words) – submitted as NAME_TPNUMBER.pdf file - that incorporates basic documentation standards such as header and footer, page numbering and includes:
   - Cover page
   - Table of contents
   - Introduction and assumptions
   - Design of the program – using pseudocode **and** flowcharts – which adheres to the requirements provided above
   - Programming Concepts with source code for explanation
   - Screenshots of sample input/output and explanation
   - Conclusion
   - References (if any) using APA Referencing

3.0 ASSESSMENT CRITERIA

i. Design (Pseudocode and Flowchart) 30%

Detailed, logical, and accurate design of a programmable solution.

ii. Coding / Implementation (Python code) 30%

Application of Python programming techniques (from basic to advanced); good programming practices in implementing the solution as per design; and adequate validation meeting all system requirements with all possible additional features.

iii. Documentation 30%

Adherence to document standard format and structure; screen captures of input/output with explanation; and inclusion of generated text files.

iv. Demonstration 10%

Ability to run, trace code, explain work done and answer questions.

The marking scheme for the assignment has been provided so that you clearly know how the assessment for this assignment would be done.

4.0PERFORMANCE CRITERIA

Distinction (75% and above)

This grade will be assigned to work that meets all the requirements stated in the question. The program runs smoothly when executed. There is clear evidence and application of Python concepts up to an advanced level. The program solution is unique with excellent coding styles and validation. The program implemented maps completely against the design (pseudocode and flowchart) as seen in the documentation. The design of the solution varies in style and has unique logic with hardly any errors/omissions. The documentation does not have any missing components. Sample inputs/outputs documented have a clear explanation. Students must be able to provide an excellent explanation of the codes and work done, show additional concepts / new ideas used in the solution, able to answer all questions posed with accurate/logical answers/explanation provided with sound arguments and clear discussion. Overall, an excellent piece of work submitted.

Credit (65%-74%)

This grade will be assigned to work which is considered to be of a good standard and meets most of the requirements stated in the question. The program runs smoothly when executed. There is clear evidence and application of Python concepts up to at least an intermediate level. The program solution is unique with good coding styles and validation. The program implemented maps well against the design (pseudocode and flowchart) as seen in the documentation. The design of the solution varies in style and has unique logic with minor errors/omissions. The documentation does not have any missing components. Sample inputs/outputs documented with some explanation. Students must be able to provide a good explanation of the codes and work done and answer most questions posed with mostly accurate/logical answers/explanations. Overall, a good assignment was submitted.

Pass (50%-64%)

This grade will be assigned to work which meets at least half of the basic requirements (approximately 50%) stated in the questions. The program runs smoothly when executed. There is clear evidence and application of Python concepts at the basic level. The program solution is common with basic coding styles and validation. The program implemented somewhat maps with the design (pseudocode and flowchart) as seen in the documentation. The design of the solution is average in terms of logic and style with some errors / omissions. The documentation has some missing components. Sample inputs/outputs documented but without

any explanation. Students must be able to explain some codes and work done and able to answer some questions posed with some accurate/logical answers/explanations. Overall, an average piece of work was submitted.

Fail (Below 50%)

This grade will be assigned to work which achieved less than half of the requirements stated in the question. The program is able to compile but not able to execute or with major errors. The program solution has only basic coding styles with no validation. The program solution has little or no mapping with the design. The design of the solution has major/obvious errors/omissions. The documentation has some missing essential components. A student is barely able to explain the codes/work done and answer given on the questions posed but with mostly inaccurate / illogical answers / explanation. Overall, a poor piece of work submitted.