**Course Learning Outcomes:**

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

|  |  |
| --- | --- |
| CLO1 | Explain the fundamental of software development and programming concepts (C2, PLO1) |
| CLO2 | Construct a programmable solution using appropriate problem-solving methods and programming concepts to the given scenario. (C3, PLO2) |
| CLO3 |  |

1. **GROUP ASSIGNMENT DESCRIPTION**

**INVENTORY MANAGEMENT SYSTEM FOR**

**PERSONAL PROTECTIVE EQUIPMENT (PPE)**

The Department of Health in your state needs a computer program to manage the inventory of PPEs1 that it receives from multiple suppliers2 and distribute them to the hospitals3 that it manages. List of PPE items the department receives are listed in Table 1 below:

Table 1: PPE Items

|  |  |
| --- | --- |
| **Item Code** | **Item Name** |
| HC | Head Cover |
| FS | Face Shield |
| MS | Mask |
| GL | Gloves |
| GW | Gown |
| SC | Shoe Covers |

The inventory system must be programmed in Python. Users of the program need to have a valid user ID and password. Your team have been recruited for the same. Both members are required to put in equal efforts and therefor, following will be the Workload Matrix for both the members:

|  |  |
| --- | --- |
| **Member** | **Functionalities** |
| Member 1 | 1. User Management and Initial Inventory Creation & Management 2. Searching Functionalities |
| Member 2 | 1. Hospital and Supplier Creation, & Item Inventory Management 2. Item Inventory Tracking |

You are required to write the Python program with following features and submit one Python File named as MEMBER1\_NAME\_TPNO\_MEMBER2\_NAME\_TPNO.py.

1. **User Management & Initial Inventory Creation**

All users and their details must be stored in *users.txt* file. The details for each user should include user ID, name, password, userType (admin or staff). Other relevant details can also be recorded.

All items and its details need to be recorded in *ppe.txt* file. The details for each item should include item code, supplier code and quantity in stock (measured in number of boxes). Other relevant details can also be recorded.

* All users must have valid (unique) user ID and a password. Admin should be able to add new users, modify, search and delete users.
* All PPE items are measured in boxes, i.e., they are received, recorded and distributed in boxes.
* Each item is supplied by exactly one supplier. However, one supplier can supply more than one type of item.
* Assume that there should be minimum of three hospitals in your state.

Important Note:

1. Inventory creation should be done **only once and during the very first time** the program is executed. Initial quantity of each item need to be recorded as 100 each in *ppe.txt* file during this time. The program should prompt for all inputs for creating this file.
2. You can only have **3 or 4 suppliers**. You should not include supplier details in *ppe.txt* file. Only supplier code need to be stored in *ppe.txt* file.
3. The program should have a feature to create *suppliers.txt* file for storing and updating supplier details.
4. **Item Inventory Update**

The program should have a feature for user to update the item quantities every time after receiving from suppliers (increase in quantity) or distributing to the hospitals (decrease in quantity). You should record the details of all the updates in a text file *transactions.txt*. Details of transactions (i.e. items received and distributed) should include item code, supplier code/Hospital code, quantity received/quantity distributed, and date-time.

Important Note:

1. Details of suppliers need to be stored in *suppliers.txt* file.
2. When testing the program, you should perform adequate updates on each item. This is to prove whether the feature is correctly functioning.
3. Before distributing any item to hospitals, the program should check for available quantity in stock. User need to be notified if the quantity in stock is insufficient. The program should also indicate the current quantity in stock for the user to retry with appropriate quantity.
4. The program should have a feature to create *hospitals.txt* file for storing and updating hospital details. Include hospital code for each of them. You can only have **3 or 4 hospitals**.
5. Record all updates in *transaction.txt* file.
6. **Item Inventory Tracking**

The program should have options to track items and print:

1. Total available quantity of all items sorted in ascending order by item code.
2. Records of all items that has stock quantity less than 25 boxes.
3. Track available quantity for a particular item.
4. Track item received during a specific time period (startDate to endDate)
5. **Search Functionalities**

The program should have options to search and print the filtered list

* For details of items distribution for any particular item.
* For details of items received from any particular item.

Important Note:

1. The search should be done by using item code.
2. The list should include suppliers/hospital codes and quantity distributed with date received.
3. If the item has been received/distributed to the same supplier/hospital for more than once, then their quantities have to be summed up together.
4. **REQUIREMENTS**
5. You are required to carry out extra research for your system and document any logical assumptions you made after the research. You are required to research about datetime module and allowed only datetime module in you assignment.
6. Your program should use symbolic constants where appropriate. Validations need to be included to ensure the accuracy of the system. State any assumptions that you make under each function.
7. You are required to store all data in text files indicated under the system requirements.
8. You are expected to use list and functions in your program. Your program must embrace modular programming technique and should be menu-driven.
9. You may include any extra features which you may feel relevant and that add value to the system.
10. 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.
11. You should include the good programming practice such as comments, variable naming conventions and indentation**.**
12. 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 offence will be dealt in accordance to APU regulations on plagiarism.***
13. You are required to use Python programming language to implement the solution. Use of any other language like C/C++/Java is not allowed. Global variable is not allowed.
14. Results of a comprehensive testing is to be included in your document. The tests conducted shall take into consideration of all valid inputs and negative test cases.
15. **DELIVERABLES**

You are required to submit a softcopy of:

1. Program coded in Python – submitted as .py file.

* Name the file under your name and TP number (e.g. DAVID\_JONES\_TP012345\_KATHY\_SIERRA\_TP123456.py)
* Start the first two lines in your program by typing your name and TP number (e.g. as follows):

# DAVID\_JONES, KATHY SIERRA

# TP012345, TP123456

1. Text files created through test data – submitted as .txt files.
2. A documentation of the system (1000 words) – submitted as NAME1\_TPNUMBER\_NAME2\_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
* Program source code and explanation
* Screenshots of sample input/output and explanation
* Conclusion
* References (if any) using APA Referencing

1. **ASSESSMENT CRITERIA**
2. Design (Pseudocode and Flowchart) 30%

Detailed, logical and accurate design of programmable solution.

1. 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.

1. Documentation 25%

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

1. Demonstration 15%

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.

1. PERFORMANCE CRITERIA

Distinction (80% and above)

This grade will be assigned to work which meets all of the requirements stated in the question. The program runs smoothly when executed. There is clear evidence and application of Python concepts up to 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 styles and has unique logic with hardly any errors / omissions. The documentation does not have any missing components. Sample inputs/outputs documented have clear explanation. Student must be able to provide 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 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 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 styles and has unique logic with minor errors / omissions. The documentation does not have any missing components. Sample inputs/outputs documented with some explanation. Student must be able to provide good explanation of the codes and work done, answer most questions posed with mostly accurate / logical answers / explanation. Overall a good assignment 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 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. Student must be able to explain some codes and work done and able to answer some questions posed with some accurate / logical answers / explanation. Overall an average piece of work 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. 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.