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).

1.0 INDIVIDUAL ASSIGNMENT DESCRIPTION

COVID-19 VACCINATION RECORD MANAGEMENT SYSTEM

Coronavirus or severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) is a deadly virus that has caused global pandemic. The virus causes respiratory tract infections that can range from mild to lethal. The World Health Organization (WHO) has named the disease caused by coronavirus as coronavirus disease (COVID-19).

As of now, there is no specific medication available nor drugs discovered for treating COVID-19 patients. Thus, hospitals usually provide supportive care to COVID-19 patients as part of their treatment procedure. This includes treatment to relieve symptoms, fluid therapy, oxygen support and prone positioning as needed, and medications or devices to support other affected vital organs. The COVID-19 vaccination is found to be a safer way to help build protection against the virus and end the pandemic. This has made many countries to quickly vaccinate their people, and in order to vaccinate mass population of their people as quickly as possible, the COVID-19 vaccination process is simultaneously done in many vaccination centres in the country.

Assume that the place you are living in has **two (2) vaccination centres (VC1 and VC2)** and they require a computer program to keep the records of all vaccination done. The details of the vaccines administered in these centres are shown in Table 1.

Table 1: Details of Vaccines Administered in VC1 and VC2

Vaccine Code	Dosage Required	Interval Between Doses	Age Group
AF	2	2 weeks (or 14 days)	12 and above
BV	2	3 weeks (or 21 days)	18 and above
CZ	2	3 weeks (or 21 days)	12 - 45
DM	2	4 weeks (or 28 days)	12 and above
EC	1		18 and above

You are required to write a Python program with the following features to manage the vaccination records in the above-mentioned centers:

1. New Patient Registration

Individuals who come to the VCs will be required to register as a patient before the vaccination. New patient registration is only done once that is before the first dosage of vaccination. Upon registration, each individual will need to be given a **unique patient id**. All relevant details about the patient and patient id need to be recorded in a text file named as ***patients.txt***. Among these details, the **VC the patient belongs to** and **the vaccine they have selected based on the age group** shown in Table 1 are required to be stored in this text file. Patient address is not required, however, their **contact number and/or email address** will need to be recorded instead.

Important Note:

i. The registration process **should start by prompting for VC**.
ii. **Age should be prompted prior to vaccine selection**. Only eligible vaccines (based on the age entered) need to be displayed for selection.
iii. When testing the program, you should register a good number **(15 – 20) and mixture of patients from different age groups, vaccines and VCs**.
iv. The patient id needs to be **unique and in sequence**.
v. When recording the vaccine and VC, you may choose to use the abbreviations as shown in Table 1 for simplicity.
vi. Details on medical history, blood group, height, weight and alike are optional.

2. Vaccine Administration

The program should record all vaccine administered in ***vaccination.txt*** file. The program may prompt for inputs like **patient id and dose number (D1 or D2)** before the vaccine is administered.

Important Note:

i. All patients should go through first dosage before the second dose unless if they have opted for EC vaccine.
ii. After completing dose 1, the program should advice when the patient should come for dose 2 (either by indicating **next vaccination date or number of weeks later**).

iii. When testing the program, you need to have a **good mixture of patients completed dose 1 and dose 2**.

iv. The patients **need not** have to be vaccinated in sequence as per their patient id.

v. The program **should allow selection between New Patient Registration and Vaccine Administration** features in any sequence. That is, not all (15 – 20) patients that you planned to vaccinate need to be registered in one-go before the vaccine administration.

3. Search Patient Record and Vaccination Status

The program should have an option to search patient record using patient id. Vaccination status of the searched patient should be displayed along with the patient record. Vaccination status that can be used in your program in shown in Table 2 below:

Table 2: Patient Vaccination Status

Vaccine Code	**Dosage Required**	**Status**		
		Before Dose 1	**After Dose 1**	**After Dose 2**
AF	2	NEW	COMPLETED-D1	COMPLETED
BV	2			
CZ	2			
DM	2			
EC	1		COMPLETED	

4. Statistical Information on Patients Vaccinated

The program should have option to print the total number of patients vaccinated by each VC. These numbers should be broken down into **people who are waiting for dose 2** and **people who have completed vaccination**.

2.0 REQUIREMENTS

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

ii. 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.

iii. You are required to store all data in text files indicated under the system requirements.

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

v. You may include any extra features which you may feel 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 the good programming practice 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 offence will be dealt in accordance to 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. Global variable is not allowed.

x. 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.

3.0 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
   - Program source code and explanation
   - Screenshots of sample input/output and explanation
   - Conclusion
   - References (if any) using APA Referencing

4.0 ASSESSMENT CRITERIA

i. Design (Pseudocode and Flowchart) 30%

Detailed, logical and accurate design of 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 25%

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

iv. 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.

5.0 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.