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

Title

Online Pharmacy Management System (OPMS)

Section A: Program Specification

You are required to develop OPMS which will be used by "OCEAN Sdn Bhd" a Malaysian leading company to manage pharmacy functionalities.

Due to covid pendamic government decided to go for lockdown in country due to which all peoples are restricted to come out of their house. In this situation it become difficult for customers to purchase mediciens also. So, OCEAN decided to develop OPMS.

A good System allows you to effectively manage all transactions of customers easily.

Functionality of OPMS

There are 3 users for this system, 1. Admin 2. New Customer 3. Registered Customer. The following characteristics are important to be included in your system:

Admin

1. Can login to access system.
2. Can upload Medicine detail in system. (Medicine name, exp date, price, specification, etc...)
3. Can view all uploaded Medicines.
4. Can update/modify Medicine information if required.
5. Can delete Medicine information.
6. Can search specific Medicine detail.
7. Can view all orders of customers.
8. Can search order of specific customer.
9. Exit

New Customer

1. Can view Medicine detail.
2. Can do registration by providing their detail like Name, Address, Email ID, Contact Number, Gender, Date_Of_Birth, User ID, Password, Rewrite Password, etc…
3. Exit

Registered Customer

1. Can login to the system.
2. View all Medicines detail.
3. Place order of medicines and do payment.
4. Can view own order.
5. Can view personal information.
6. Exit

The application has to be developed using python programming language. Data may be stored in lists, files, etc.

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. There is no limit on the number of text files that can be used but they should be kept minimum.

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 in your program, as what is being assessed, is your programming skill not the interface design. The marking scheme for the assignment has been provided so that you clearly know how the assessment for this assignment would be done.

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 plagiarism, the offence and 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.

x. Global variables, build in functions like min, max, sort, etc… are not allowed.

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 – 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 Harvard Name 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 advance); 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.

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. Marking S

Marking Scheme

	Fail	Marginal Fail	Pass	Merit	Distinction
	0 - 5	**6 - 7**	**8 - 9**	**10 - 11**	**12 - 15**
Design (30%) CLO1 – PLO2	• Inappropriate or no pseudocode submitted. • Pseudocode covers less than 40% of system requirements with correct logic/solution.	• Pseudocode covers between 40% - 50% of system requirements with correct logic/solution.	• Pseudocode covers between 50% - 65% of system requirements with correct logic/solution.	• Pseudocode covers between 65% - 80% of system requirements with correct logic/solution.	• Pseudocode covers more than 80% of system requirements with correct logic/solution.
	• Inappropriate or no flowchart submitted. • Flowchart covers less than 40% of system requirements with correct logic/solution and shapes.	• Flowchart covers between 40% - 50% of system requirements with correct logic/solution and shapes.	• Flowchart covers between 50% - 65% of system requirements with correct logic/solution and shapes.	• Flowchart covers between 65% - 80% of system requirements with correct logic/solution and shapes.	• Flowchart covers more than 80% of system requirements with correct logic/solution and shapes.
					Total: / 30

	Fail	Marginal Fail	Pass	Merit	Distinction
Coding (Implementation) (30%) CLO3 – PLO3	**0 - 5**	**6 - 7**	**8 - 9**	**10 - 11**	**12 - 15**
	• No program submitted. • Incomplete / illogical solution. • Program has major errors, does not compile and/or does not run when executed. • Solution/output meets less than 40% of system requirements. • No / least mapping between program design and solution.	• Program compiles with no/some errors and runs when executed. • Solution/output meets between 40% - 50% of system requirements. • Little or no mapping between mapping between program design and solution.	• Program compiles with no/some errors and runs when executed. • Solution/output meets between 50% - 65% of system requirements. • Average/some mapping between program design and solution.	• Program compiles with no/minimum errors and runs when executed. • Solution/output meets between 65% - 80% of system requirements. • Good mapping between program design and solution.	• Program compiles with no errors and runs when executed. • Solution/output meets more than 80% of system requirements. • Excellent mapping between program design and solution.
	0 - 3	**4**	**5 - 6**	**7**	**8 - 10**
	• No program submitted. • Only basic programming techniques applied to build program solution. • Application of programming techniques like modular programming, menu-driven application development, nested lists, nested control structures and File I/O is not evident.	• Program solution shows application of programming techniques like modular programming, menu-driven application development, nested lists, nested control structures and File I/O is minimum or below average level.	• Program solution shows average use of programming techniques like modular programming, menu-driven application development, nested lists, nested control structures and File I/O.	• Program solution shows adequate application of programming techniques like modular programming, menu-driven application development, nested lists, nested control structures and File I/O.	• Program solution shows mastery level of candidate in using programming techniques like modular programming, menu-driven application development, nested lists, nested control structures and File I/O.
	0 - 1	**2**	**3**	**4**	**5**
	• No program submitted. • Very poor coding style. Adherence to good programming practices like commenting, variable naming and indentation is less than 40%. • No validation.	• Poor coding style. Adherence to good programming practices like commenting, variable naming and indentation is between 40% - 50% only. • Poor/minor validation.	• Basic coding style. Adherence to good programming practices like commenting, variable naming and indentation is between 50% - 65% only. • Average/some validation.	• Adherence to good programming practices like commenting, variable naming and indentation is between 65% - 80% only. • Good validation.	• Excellent adherence to good programming practices like commenting, variable naming and indentation. • Excellent validation.
					Total: / 30

Documentation (25%) CLO2 – PLO1	**Fail**	**Marginal Fail**	**Pass**	**Credit**	**Distinction**
	0 - 5	**6 - 7**	**8 - 9**	**10 - 11**	**12 - 15**
	• No source code included. • No / poor / inaccurate explanation on the internal working of codes in program solution.	• Major printout of source codes not included. • Insufficient explanation on the internal working of codes in program solution.	• Document missing some minor printout of source codes. • Moderate explanation on the internal working of codes in program solution.	• Most of the source code printout included in the documentation. • Provided detail and accurate explanation on the internal working of codes in program solution.	• All source codes included in documentation. • Provided detail and accurate explanation on the internal working of all codes in program solution.
	0 - 3	**4**	**5 - 6**	**7**	**8 - 10**
	• No documentation submitted or shows no attention to documentation format and structure. • No or very minimum sample input/output screenshots attached in documentation without any explanation. • No text file attached in documentation.	• Shows least attention to documentation format and structure. • Insufficient sample input/output screenshots attached in documentation without or with minimum explanation. • Not all created text files are attached in documentation.	• Shows moderate attention to documentation format and structure. • Moderate amount of sample input/output screenshots attached in documentation with some explanation. • Almost all created text files are attached in documentation.	• Shows sufficient level of attention to documentation format and structure. • Sample input/output screenshots attached in documentation is almost comprehensive with adequate explanation. • All created text files are attached in documentation.	• Shows high degree of attention to documentation format and structure. • Sample input/output screenshots attached in documentation is comprehensive and explained in detail. • All created text files are attached in documentation.
Demonstration (15%) CLO2 – PLO1	**0 - 5**	**6 - 7**	**8 - 9**	**10 - 11**	**12 - 15**
	• Did not turn up for presentation. • Not able to trace any of the codes / work done. • Unable or barely able to answer any of the question asked.	• Barely able to trace the codes / work done. • Mostly inaccurate / illogical answers / explanation provided or barely able to answer some of the questions asked	• Able to trace some codes / work done with hesitation. • Able to answer some questions posed accurately or logically.	• Able to trace the codes and work done. • Able to answer most questions posed accurately and shows a good understanding of how the program works.	• In depth understanding of the codes / work done. • Able to answer all questions posed with minimal omissions. • Show additional concepts / new ideas used in the solution.
					Total: / 40

	Fail	Marginal Fail	Pass	Merit	Distinction
Coding (Implementation) (30%) CLO3 – PLO3	**0 - 5**	**6 - 7**	**8 - 9**	**10 - 11**	**12 - 15**
	• No program submitted. • Incomplete / illogical solution. • Program has major errors, does not compile and/or does not run when executed. • Solution/output meets less than 40% of system requirements. • No / least mapping between program design and solution.	• Program compiles with no/some errors and runs when executed. • Solution/output meets between 40% - 50% of system requirements. • Little or no mapping between mapping between program design and solution.	• Program compiles with no/some errors and runs when executed. • Solution/output meets between 50% - 65% of system requirements. • Average/some mapping between program design and solution.	• Program compiles with no/minimum errors and runs when executed. • Solution/output meets between 65% - 80% of system requirements. • Good mapping between program design and solution.	• Program compiles with no errors and runs when executed. • Solution/output meets more than 80% of system requirements. • Excellent mapping between program design and solution.
	0 - 3	**4**	**5 - 6**	**7**	**8 - 10**
	• No program submitted. • Only basic programming techniques applied to build program solution. • Application of programming techniques like modular programming, menu-driven application development, nested lists, nested control structures and File I/O is not evident.	• Program solution shows application of programming techniques like modular programming, menu-driven application development, nested lists, nested control structures and File I/O is minimum or below average level.	• Program solution shows average use of programming techniques like modular programming, menu-driven application development, nested lists, nested control structures and File I/O.	• Program solution shows adequate application of programming techniques like modular programming, menu-driven application development, nested lists, nested control structures and File I/O.	• Program solution shows mastery level of candidate in using programming techniques like modular programming, menu-driven application development, nested lists, nested control structures and File I/O.
	0 - 1	**2**	**3**	**4**	**5**
	• No program submitted. • Very poor coding style. Adherence to good programming practices like commenting, variable naming and indentation is less than 40%. • No validation.	• Poor coding style. Adherence to good programming practices like commenting, variable naming and indentation is between 40% - 50% only. • Poor/minor validation.	• Basic coding style. Adherence to good programming practices like commenting, variable naming and indentation is between 50% - 65% only. • Average/some validation.	• Adherence to good programming practices like commenting, variable naming and indentation is between 65% - 80% only. • Good validation.	• Excellent adherence to good programming practices like commenting, variable naming and indentation. • Excellent validation.

Total: / 30

Documentation (25%) CLO2 – PLO1	**Fail**	**Marginal Fail**	**Pass**	**Credit**	**Distinction**
	0 - 5	**6 - 7**	**8 - 9**	**10 - 11**	**12 - 15**
	• No source code included. • No / poor / inaccurate explanation on the internal working of codes in program solution.	• Major printout of source codes not included. • Insufficient explanation on the internal working of codes in program solution.	• Document missing some minor printout of source codes. • Moderate explanation on the internal working of codes in program solution.	• Most of the source code printout included in the documentation. • Provided detail and accurate explanation on the internal working of codes in program solution.	• All source codes included in documentation. • Provided detail and accurate explanation on the internal working of all codes in program solution.
	0 - 3	**4**	**5 - 6**	**7**	**8 - 10**
	• No documentation submitted or shows no attention to documentation format and structure. • No or very minimum sample input/output screenshots attached in documentation without any explanation. • No text file attached in documentation.	• Shows least attention to documentation format and structure. • Insufficient sample input/output screenshots attached in documentation without or with minimum explanation. • Not all created text files are attached in documentation.	• Shows moderate attention to documentation format and structure. • Moderate amount of sample input/output screenshots attached in documentation with some explanation. • Almost all created text files are attached in documentation.	• Shows sufficient level of attention to documentation format and structure. • Sample input/output screenshots attached in documentation is almost comprehensive with adequate explanation. • All created text files are attached in documentation.	• Shows high degree of attention to documentation format and structure. • Sample input/output screenshots attached in documentation is comprehensive and explained in detail. • All created text files are attached in documentation.
Demonstration (15%) CLO2 – PLO1	**0 - 5**	**6 - 7**	**8 - 9**	**10 - 11**	**12 - 15**
	• Did not turn up for presentation. • Not able to trace any of the codes / work done. • Unable or barely able to answer any of the question asked.	• Barely able to trace the codes / work done. • Mostly inaccurate / illogical answers / explanation provided or barely able to answer some of the questions asked	• Able to trace some codes / work done with hesitation. • Able to answer some questions posed accurately or logically.	• Able to trace the codes and work done. • Able to answer most questions posed accurately and shows a good understanding of how the program works.	• In depth understanding of the codes / work done. • Able to answer all questions posed with minimal omissions. • Show additional concepts / new ideas used in the solution.