ONLINE EVENT MANAGEMENT SYSTEM (ODOMS)

ASIAN EVENT MANAGEMENT SERVICES(AEMS) is one of the fastest growing event management agency in Malaysia which helps clients in organizing and joining events. Considering the increase in demand for their services, AEMS has decided to enhance their services by developing an interface to manage their business processes

SOFS requires you to develop Python program for AEMS which will have 2 types of users (Admin and Customer) and it should contain the following features:

Functionalities of Admin

- i. Login to access system.
- ii. Add event in the relevant category
- iii. Modify event record
- iv. Display all records of
    - a. Each category
    - b. Events in each category
    - c. Customer Registration
    - d. Customer Payment
- v. Search Specific record of
    - a. Customer registration
    - b. Customer payment
- vi. Exit

Functionalities of All customers (registered / not registered)

- i. View all events as per category
- ii. New customer registration to access other details
- iii. Exit

Functionalities of registered customer

- i. Login to access system
- ii. View detail of
    - a. Event categories
    - b. Events
- iii. Select Event item and add to cart
- iv. Make payment to confirm order.
- v. Exit

Python (25-27)
Pseudo/Flow (28-29)
Documentation (30-31)

1.0 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.

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 – 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 with explanation
   - Additional features source code with explanation
   - Screenshots of sample input/output with explanation
   - Conclusion
   - References using Harvard Name Referencing

3.0 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 20%

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.

v. Q & A 10%

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

4.0 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.

Teachers' expectation

1. Login
    a. Must be in a loop
    b. Should be kept as a separate file
2. Add events
    a. You should have a file that contains all the category (sports, music, games, etc)
    b. Add name, time, date, venue, organizer name, cost, and any other remarks
3. Modify record
    a. You should be able to modify the time, venue, and all other attributes
4. Display all records of
    a. In this case, there should be a feature that fetches all information of each category first, then each event in the category, then each customer's registration and their payment status.
5. Search for specific records of registration
    a. If you have a registration number, you should be able to fetch it directly
    b. It should display their name and payment
6. Exit

   a. You should have an exit feature that enables the admin to exit the program

Files:

1. "event", "customer registration & payment", "category" etc should be saved in separate files and fetched accordingly.

Expectation of customers:

1. Login
   a. allow customer to login with username and password
2. View details of
   a. All the event categories
   b. All the available events
3. Select event item and add to cart
   a. They can choose a certain category, and the event inside the category
   b. They should be able to add multiple events into their shopping cart
4. Make payments and confirm order
   a. There should be a payment section where the program collects the credit card number from customer to make payment
   b. When it's done, it should be saved and a "payment successful" message should be printed.
   c. Payment should have a primary key "payment_id" so we can track customer purchase history and give them discount for repeated patronage
5. Exit
   a. A customer should be able to exit the program afterwards

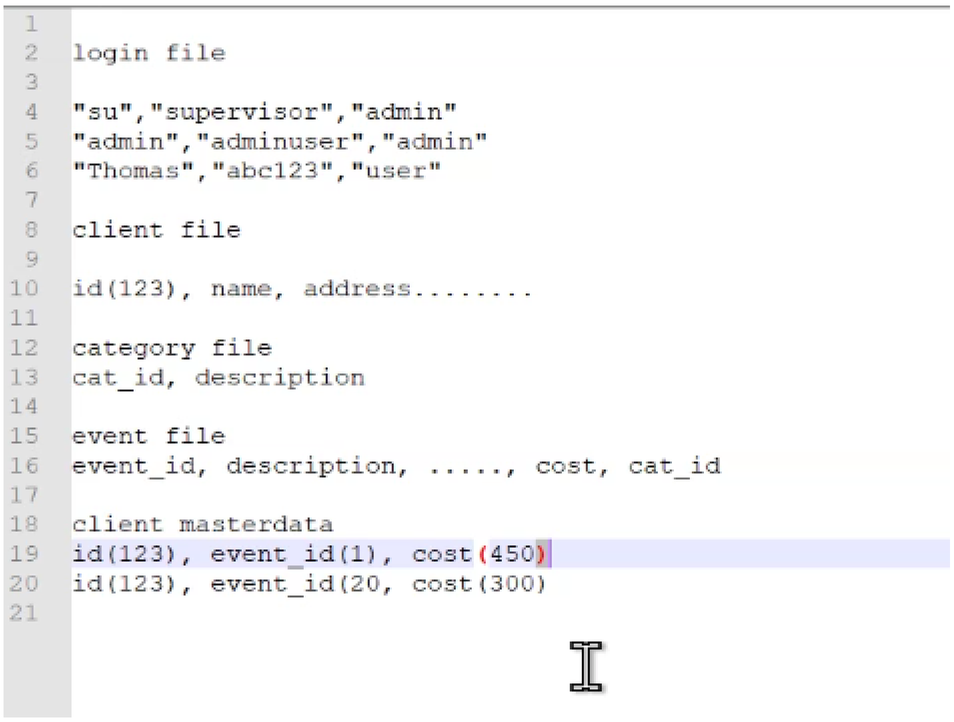
Don't use classes, don't use OOP (object-oriented programming)

Don't import anything from python libraries except time and date

Protip from lecturer

1. Ask more questions during class, it's not a video, it's a livestream
2. Do it now! Don't wait for motivation, create your own motivation

Example of login file



```

login file

"su","supervisor","admin"
"admin","adminuser","admin"
"Thomas","abc123","user"

client file

id(123), name, address........

category file
cat_id, description

event file
event_id, description, ....., cost, cat_id

client masterdata
id(123), event_id(1), cost(450)
id(123), event_id(20, cost(300)

```

tip:
# make sure you understand split and join if you wanna be good at this assignment
https://www.w3schools.com/python/ref_string_join.asp
https://www.programiz.com/python-programming/methods/string/join
https://www.python.org
https://www.stackoverflow.com

# use validation
you won't get distinction without validation
eg of validation: "Garbage in Garbage Out"