PYQ OCT 2025

Question 1

Question

a) You are developing an online student portal for a college. The portal allows students to view their personal profiles, update their contact information, and check their exam results. Additionally, the system includes:

- Real-time validation of the student's phone number format as they type during profile updates.
- Automatic retrieval of the student's latest exam results from the database after they logged in.

Based on the scenario above, analyse the features and determine which one is best implemented using client-side scripting and which requires server-side scripting. Provide a clear justification for your decisions based on the nature and requirements of the task. *(10 marks)*

Answer

- The real-time validation of student's phone number format should be applied using client-side scripting.
  - Since the phone number format validation need to be real-time and having immediate response along with user typing, the JavaScript script can be applied for validating the format.
  - The client-side scripting does not involve the process of sending request and receiving response when validating the phone number format. So, it can ensure a minimum delay for responding to the user input. All the validation actions will only be done within the client web browser once the web pages have been downloaded.
  - Meanwhile, the format of phone number format are always the same or seldom be changed. Thus, no database interaction or complex validation should be required. The client-side scripting using JavaScript is sufficient enough for validating the student's phone number format.
- The automatic retrieval of student's latest exam results from database should be done using server-side scripting.

- The student's latest exam results may change dynamically along with time or based on the semester the student is located in.
- Since the client-side scripting is not able and not allowed to interact with the database directly, the client browser must send request to the server and let the server-side script to interact and access to the database. Eventually, the server fetches the student's latest exam results from the database and respond back to the client browser.
- Meanwhile, the authentication function must be done in server side since the system is only allowed to retrieve the student's latest exam results after the they have logged in.
- Server-side scripting can ensure a more secure implementation and also hide the implementation from the public. In contrast, client-side scripting can be easily exploited and visible by the attackers. It is not suitable for leaving the authentication features inside the client-side scripting as it may reveal the sensitive information to public.

Question

b) You are working on a large-scale banking application where accuracy, security, and reliability are crucial. Your team is considering whether to use a strongly typed or weakly typed programming language. Analyse **FOUR (4)** reasons why a *strongly typed* language would be more suitable for this project. *(8 marks)*

Answer

- Strongly typed language will explicitly define the data type for each variable declared. This can reduce the risk of type-related errors such as assigning incorrect values to variables. This improves the code accuracy and prevents critical mistakes in financial calculations.
- Strong typing also improves the code readability and maintainability in large-scale systems. Developers can clearly understand how data flows through the system, making it easier for teams to maintain and extend complex banking applications.
- Strongly typed languages perform type checking at compile time, allowing man errors to be detected before the application is deployed. This increases the system reliability and reduces the likelihood of runtime failures in production.

- Strongly typed languages provide more predictable execution and better compiler optimizations, resulting in stable and efficient system behavior, which is important for performance-critical banking operations.

Question

c) A developer stores user credentials directly in the source code for convenience and uses MD5 to hash passwords. The login system performs only basic username and password verification, and no secure transmission or storage practices are applied. Assess **ONE (1)** weakness in this system based on authentication best practices. *(2 marks)*

Answer

- One of the weakness in this system is the use of MD5 for password hashing.
- MD5 is insecure although it is a fast method to hash the password.
- It is vulnerable to brute-force and rainbow table attacks, making it easy for attackers to recover the user password if the hashes are exposed.
- Storing user credentials directly in the source code is also insecure because anyone with access to the codebase can easily obtain the sensitive authentication information, leading to unauthorized access.

Question

d) A government e-services portal allows users to submit feedback through a text form. However, some users have discovered that they can inject JavaScript code into the feedback, which then can be executed when viewed by an admin. Evaluate **ONE (1)** security threat present in this scenario and recommend **ONE (1)** suitable mitigation strategy. *(5 marks)*

Answer

- Cross-Site Scripting (XSS)
- The attackers may easily inject the malicious JavaScript code into the web pages, then let the users browser to accidentally execute the JavaScript code.
- This is a critical security threat which may allow the attackers to steal the session

cookies and hijack user accounts or redirect the users to phishing websites and so on.

- In order to mitigate this issue, the input sanitization must be taken.
- The input sanitization should escape, neutralize or remove the special characters before passing the data into server side. For example, the executable code `<script>` will be neutralized into `&lt;script&gt;`.
- This can make sure all the input data is in plain text form instead of executable code.

Question 2

Question

Suppose you are developing an online shopping platform that enables customers to browse products, add them to a shopping cart, and choose different types of promotions or discounts during checkout. Each customer may select a discount option such as:

- flat discount (e.g., RM10 off),
- percentage discount (e.g., 10% off),
- or a membership-based discount (e.g., free shipping or bundle deals).
  Based on the selected discount type, the final price is calculated before proceeding to payment.

Note: You may make any reasonable assumptions to answer the following questions.

a) Identify **ONE (1)** design pattern that can be applied for the above application. Provide an applicable scenario where you can apply the selected design pattern. *(5 marks)*
b) Briefly describe the general purpose of the selected design pattern in **Question 2a)**. *(4 marks)*
c) Create a class diagram to illustrate how the selected design pattern can be implemented for the scenario that you had specified in **Question 2 a)**. *(10 marks)*
d) Illustrate **TWO (2)** key advantages of applying the selected design pattern for the scenario that you had specified in **Question 2 a)**. *(6 marks)*

Answer

(a)

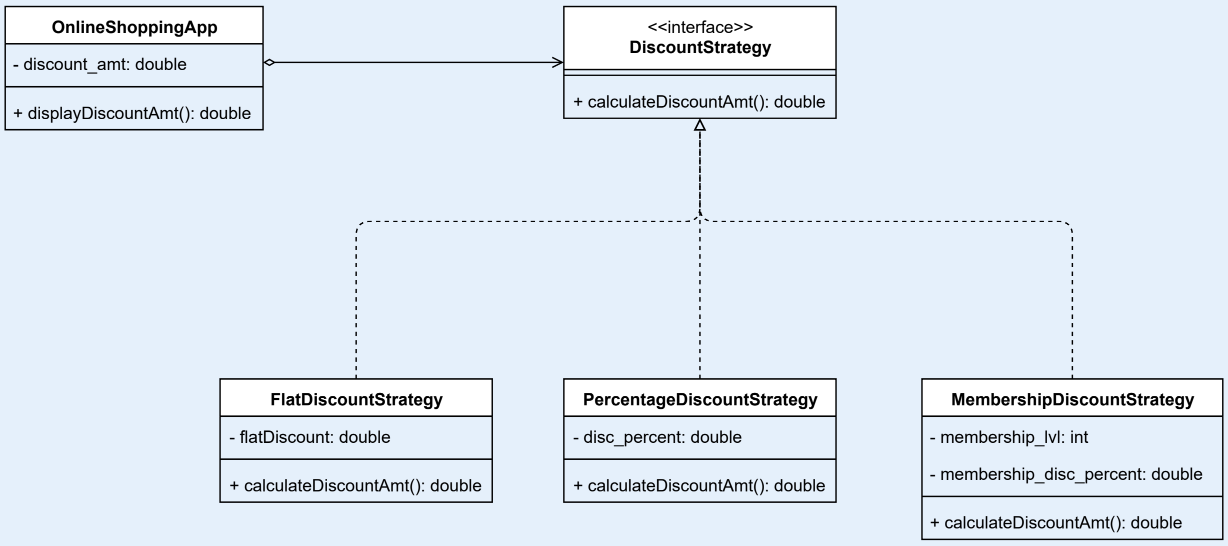
- Strategy design pattern
- Since there are different types of promotions or discounts available in the platform, each type will apply different strategy of calculating the discounted amount.

- The Strategy design pattern can help for creating a centralized Strategy interface with reference with the ConcreteStrategy classes to provide different discount calculation strategy to the Context object.
- It defines a family of algorithms and encapsulate each strategy in one class, then make them interchangeable. So, the Context object can delegate the work to a selected strategy without using complex conditional logic.
- For example, when the customer has selected the flat discount, the Context object will delegate the work to the FlatDiscountStrategy class and access the calculateDiscountedAmount() method in that class. Thus, it can find out the discount amount accurately. For other discount such as percentage discount and membership-based discount, they will also apply the same workflow to get the correct discount amount.

(b)

- The Strategy design pattern enables for accessing different algorithms at runtime based on the user selected behavior without modifying the Context object or the existing code.
- It encapsulates each algorithm into one ConcreteStrategy class. When there is necessary for changing the context behavior, it will change the strategy object instead of the context code.

(c)


OnlineShoppingApp
- discount_amt: double
+ displayDiscountAmt(): double
<<interface>>
DiscountStrategy
+ calculateDiscountAmt(): double
FlatDiscountStrategy
- flatDiscount: double
+ calculateDiscountAmt(): double
PercentageDiscountStrategy
- disc_percent: double
+ calculateDiscountAmt(): double
MembershipDiscountStrategy
- membership_lvl: int
- membership_disc_percent: double
+ calculateDiscountAmt(): double

(d)

- Strategy design pattern has successfully fulfilled the Open-Closed Principle which means open for extension, closed for modification.
  - When there is a behavioral changes in the context code, it can just delegate the work to the specified strategy class for different behavior, instead of changing the

context code.
    - No modification will be needed in the context code.
    - The Strategy design pattern also flexible and scalable to extend its behavior by adding more ConcreteStrategy class and support more behavior in future. Meanwhile, no modification on context code needed also.
- Strategy design pattern can also support behavior changes and algorithm switches at runtime based on the behavior chosen by the client side.
    - When switching from one behavior to another such as flat discount to membership discount, the Context object only need to delegate the work to the specific ConcreteStrategy class to work with it and using different formulas and constants.

Question 3

Question

Figure 1 shows a Document Type Definition (DTD) file named *library.dtd* that defines the structure of a library catalog containing multiple books.

```
<!ELEMENT library (book*)>
<!ELEMENT book (title, author, year)>
<!ATTLIST book
    ISBN ID #REQUIRED
    category (Fiction | NonFiction | Reference) #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT year (#PCDATA)>
```

Figure 1: library.dtd (Library Catalog)

```
<?xml version="1.0" encoding="UTF-8">
<!DOCTYPE library SYSTEM "library.dtd">
<!-- Your answer goes here -->
```

Figure 2: library.xml

a) Complete the Extensible Markup Language (XML) file in **Figure 2** by adding **TWO (2)** valid book records that follow the rules in the DTD file shown in **Figure 1**. *(9 marks)*
b) Create an XML Schema (XSD) file based on the completed XML file from **Question 3 a)**. *(10 marks)*

c) When an XML parser processes a document, it performs several important functions to ensure the document is valid and usable by applications. Examine **THREE (3)** responsibilities of an XML parser and their importance in handling XML data. *(6 marks)*

Answer

a)

```
<?xml version="1.0" encoding="UTF-8"?>
<library>
    <book ISBN="AJS123" category="Fiction">
        <title>Mathematics</title>
        <author>Jason Liow</author>
        <year>2001</year>
    </book>
    <book ISBN="SDL294" category="NonFiction">
        <title>Life is good</title>
        <author>Johnson Tan</author>
        <year>2010</year>
    </book>
</library>
```

b)
Russian Doll Design

```
<?xml version="1.0"?>
<xs:schema>
    <xs:element name="library">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="book" type="bookType"
maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="title" type="xs:string" />
                            <xs:element name="author" type="xs:string" />
                            <xs:element name="year" type="xs:integer" />
                        </xs:sequence>
                        <xs:attribute name="ISBN" type="xs:ID"
use="required" />
                        <xs:attribute name="category" use="required">
                            <xs:simpleType>
                                <xs:restriction base="xs:string">
```

```
                                    <xs:enumeration value="Fiction" />
                                    <xs:enumeration value="NonFiction" />
                                    <xs:enumeration value="Reference" />
                                </xs:restriction>
                            </xs:simpleType>
                        </xs:attribute>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="bookType">
        <xs:complexContent>
            <xs:attribute name="ISBN" type="xs:string" use="required" />
            <xs:simpleType name="category">
                <xs:restriction base="xs:string">
                    <xs:enumeration value="Fiction" />
                    <xs:enumeration value="NonFiction" />
                    <xs:enumeration value="Reference" />
                </xs:restriction>
            </xs:simpleType>
        </xs:complexContent>
        <xs:sequence>
            <xs:element name="title" type="xs:string" use="required" />
            <xs:element name="author" type="xs:string" use="required" />
            <xs:element name="year" type="xs:integer" use="required" />
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Salami Slice Design

```
<?xml version="1.0" ?>
<xs:schema>
    <xs:element name="library">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="book" type="bookType"
maxOccurs="unbounded" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="bookType">
        <xs:sequence>
```

```
                <xs:element name="title" type="xs:string" />
                <xs:element name="author" type="xs:string" />
                <xs:element name="year" type="xs:integer" />
            </xs:sequence>
            <xs:attribute name="ISBN" type="xs:ID" use="required"/>
            <xs:attribute name="category" type="categoryType" use="required"
/>
        </xs:complexType>
        <xs:simpleType name="categoryType">
            <xs:restriction base="xs:string">
                <xs:enumeration value="Fiction" />
                <xs:enumeration value="NonFiction" />
                <xs:enumeration value="Reference" />
            </xs:restriction>
        </xs:simpleType>
</xs:schema>
```

c)

- Well-formed Checking
    - An XML parser checks whether the XML document is well-formed, ensuring that all tags are properly nested, closed and follow correct XML syntax.
    - This is important because only well-formed XML documents can be parsed and processed correctly by applications, preventing syntax errors that may cause system failures.
- Validation Against a DTD or XML Schema
    - An XML parser can validate the document against a DTD or XML schema (XSD) to ensure the structure, elements, attributes, and data types follow predefined rules.
    - This is important because validation guarantees that the XML data is structurally correct and consistent, allowing applications to safely rely on the data format.
- Data Parsing and Access for Applications
    - An XML parser reads the XML document and converts it into a form that applications can process, such as a tree structure (DOM) or event stream (SAX).
    - This is important because it enables applications to extract, manipulate, and use XML data efficiently for further processing.

Question 4

Question

a) Compare **THREE (3)** advantages of Representational State Transfer (RESTful) services over Simple Access Object Protocol (SOAP) that explain their popularity in modern web and mobile applications. *(9 marks)*

Answer

- RESTful services are lightweight and typically use JSON, which is less verbose than XML used by SOAP. This results in smaller message sizes and faster parsing, making REST more suitable for modern web and mobile applications.
- RESTful services are easier to maintain and scale because they are stateless and follow a clear client-server separation. This reduces coupling between components and allows systems to handle large numbers of concurrent users more efficiently.
- RESTful services are more flexible and easier to implement, as REST is an architectural style rather than a strict protocol. In contrast, SOAP requires strict adherence to standards and message formats, which increases complexity and development overhead.

Question

b) Suggest **TWO (2)** situations where SOAP would be a better choice than RESTful for developing a web service. Justify your answers. *(4 marks)*

Answer

- SOAP is a better choice for enterprise-level, legacy or government systems where strict security and interoperability standards are required. SOAP provides standardized security specifications such as WS-Security, which support message-level encryption, authentication and integrity, making it suitable for regulated environments.
- SOAP is also preferable in systems that require high reliability and transactional support such as financial or payment-related services. SOAP supports features like reliable messaging and strict message validation, ensuring that requests are delivered and processed correctly even in complex and distributed system.

Question

c) Daniel is a software developer working on a multiplayer online game that requires fast and real-time communication between players. Initially, the system was built using Transmission Control Protocol (TCP) for reliability. However, as the player base grew, latency and synchronisation issues became more noticeable during gameplay. After testing, Daniel considers switching the communication protocol to User Datagram Protocol (UDP) to better suit the game's fast and real-time needs.

(i) Analyse **THREE (3)** characteristics of TCP that may cause performance issues in Daniel's multiplayer game. *(6 marks)*
(ii) In your opinion, evaluate **THREE (3)** reasons why using UDP may help improve the performance of Daniel's multiplayer game. *(6 marks)*

Answer

(i)

- Connection-oriented communication
  - TCP requires a connection setup (3-way handshake) before data transmission.
  - In a multiplayer game with many players, maintaining multiple TCP connections increases latency and overhead.
  - This delays real-time updates such as player movement and actions.
- Reliable delivery with retransmission
  - TCP guarantees reliable delivery by retransmitting lost packets.
  - In fast-paced games, retransmission can cause delays, as the game must wait for missing packets before processing new data.
  - This results in lag and synchronization issues during gameplay.
- Ordered delivery of packets
  - TCP ensures packets are delivered in sequence.
  - If one packet is delayed or lost, subsequent packets are held back until the missing packet arrives.
  - This can cause temporary freezes or stuttering, which negatively impacts real-time player experience.

(ii)

- Connectionless communication
  - UDP does not require connection setup, allowing data to be sent immediately.
  - This reduces communication overhead and enables faster data transmission.
  - As a result, player actions can be updated in real time with minimal delay.

- Lower latency with no retransmission
  - UDP does not retransmit lost packets.
  - In real-time games, losing a small amount of data (e.g. a single position update) is preferable to delaying the game.
  - This helps maintain smooth gameplay and responsiveness.
- Suitable for real-time, time-sensitive data
  - UDP allows newer packets to replace outdated ones.
  - This ensures players always receive the latest game state instead of waiting for old data.
  - It improves synchronization and reduces visible lag during gameplay.