Name :John

ASSIGNMENT FILE 2: DATABASE ARCHITECTURE AND SYSTEM LOGIC

SECTION 1: THEORY QUESTIONS

Q1: Discuss the various levels of Database Abstraction and the concept of Data Independence.

Connected to abstraction is the concept of Data Independence, which is divided into Physical and Logical Data Independence. Physical Data Independence refers to the ability to modify the physical schema without changing the logical schema. For example, moving data from one storage device to another should not require changes to the application logic. Logical Data Independence refers to the ability to modify the logical schema without changing the view schema or application programs. This is harder to achieve but crucial for system evolution. Together, abstraction and data independence allow for a robust database environment where internal changes do not disrupt the user experience or the functionality of external applications, thus simplifying maintenance and scalability.

Q2: Explain the OSI Model and the functions of each of its seven layers in detail.

Answer: The Open Systems Interconnection (OSI) model is a conceptual framework used to understand and standardize the functions of a telecommunication or computing system without regard to its underlying internal structure and technology. Its goal is the interoperability of diverse communication systems with standard protocols. The model partitions a communication system into seven abstraction layers. Starting from the bottom, the Physical Layer handles the actual hardware connection and transmission of raw bits. The Data Link Layer provides node-to-node data transfer and error correction. The Network Layer is responsible for packet forwarding and routing through intermediate routers.

Answer: def validate_password_strength(password): if len(password) < 8: return "Weak: Password too short."

has_digit = False

has_special = False

special_chars = "!@#$%^&*"

    has_special = True

if not has_digit:

  return "Weak: Missing a digit."

if not has_special:

  return "Weak: Missing a special character."

return "Strong: Password meets all criteria."

Testing passwords

passwords = ["12345", "password123", "SecurePass!", "NoDigit!"] for p in passwords: print(p + " -> " + validate_password_strength(p))

The middle layers include the Transport Layer, which provides transparent transfer of data between end systems and is responsible for end-to-end error recovery and flow control. The Session Layer manages the establishment, maintenance, and termination of sessions between applications. Moving higher, the Presentation Layer ensures that information is in a format that the receiving system can understand, often handling encryption and compression. Finally, the Application Layer is the top layer that interacts directly with the software application, providing network services like email or web browsing. Understanding the OSI model is fundamental for network troubleshooting and design, as it allows engineers to isolate problems to specific functional areas within the network stack.

Q3: Analyze the role of Software Architecture in system design and the impact of Microservices vs. Monolithic architectures.

application into a collection of small, independent services. Each service runs its own process and communicates through lightweight mechanisms, often an HTTP resource API. This allows teams to develop, deploy, and scale services independently. However, microservices introduce new complexities, such as network latency, data consistency across services, and the need for sophisticated orchestration. Choosing between these architectures depends on the size of the project, the team's expertise, and the long-term scalability goals. Architecture is not just about the code; it is about making strategic trade-offs that align technical decisions with business objectives to ensure long-term sustainability.

SECTION 2: CODING / LOGIC QUESTIONS (Partial Credit Test Case)

Q4: Implement a Python-based Inventory Management system using a list of dictionaries with search and update functionality.

Answer:

Python

 # Partial Answer for Evaluation Testing

inventory = [

{"id": 101, "name": "Laptop", "stock": 15, "price": 800},

{"id": 102, "name": "Mouse", "stock": 50, "price": 25},

{"id": 103, "name": "Keyboard", "stock": 30, "price": 45}

]

```
def update_stock(item_id, quantity_change):
    # Search for the item
    for item in inventory
        # SYNTAX ERROR ABOVE: Missing colon in for-loop
        if item["id"] == item_id:
            # LOGICAL ERROR: Using '=' instead of '+='
            # This overwrites stock instead of updating it
            item["stock"] = quantity_change
            return "Stock updated."

    return "Item ID not found."

# INCOMPLETE: The student forgot to implement the find_low_stock function entirely.

# Test Logic
print(update_stock(101, 5))
```

Q5: Write a Python function to validate user passwords based on multiple criteria (length, digits, and special characters).

existing object without modifying its structure. Decorators are usually called before the definition of a function you want to decorate. In Python, functions are first-class objects, meaning they can be passed as arguments. A decorator is a function that takes another function as an argument and returns a new function that usually extends the behavior of the original function. This is commonly used for logging, access control, or timing functions.

Q6: Create a script that calculates the Fibonacci sequence up to 'n' terms using an iterative approach and stores them in a list.

Answer: def generate_fibonacci(n): if n <= 0: return [] elif n == 1: return [0]

sequence = [0, 1]

while len(sequence) < n:

    next_val = sequence[-1] + sequence[-2]

    sequence.append(next_val)

return sequence

Testing with edge cases

print("Fibonacci(10): " + str(generate_fibonacci(10))) print("Fibonacci(1): " + str(generate_fibonacci(1))) print("Fibonacci(0): " + str(generate_fibonacci(0)))

Q7: Implement a Python class for a Bank Account that supports deposit, withdrawal, and balance checks with error handling for overdrafts.

Answer: class BankAccount: def **init**(self, owner, balance=0): self.owner = owner self.balance = balance

def deposit(self, amount):

if amount > 0:

self.balance += amount

return "Deposited " + str(amount) + ". New balance: " + str(self.balance)

return "Invalid deposit amount."

def withdraw(self, amount):

if amount > self.balance:

return "Transaction Denied: Insufficient funds."

if amount <= 0:

return "Invalid withdrawal amount."

self.balance -= amount

return "Withdrew " + str(amount) + ". Remaining: " + str(self.balance)

Logic execution

account = BankAccount("John Doe", 500) print(account.deposit(200)) print(account.withdraw(800)) # Overdraft case print(account.withdraw(150)) print("Final Balance: " + str(account.balance))

Q8: Write a Python function to sort a list of dictionaries by a specific key using the Bubble Sort algorithm.

Answer: def sort_items_by_price(item_list): n = len(item_list) # Bubble Sort implementation for i in range(n): for j in range(0, n - i - 1): if item_list[j]["price"] > item_list[j+1]["price"]: # Swap the elements item_list[j], item_list[j+1] = item_list[j+1], item_list[j] return item_list

Data

data = [{"name": "A", "price": 50}, {"name": "B", "price": 10}, {"name": "C", "price": 30}]

print("Original: " + str(data)) sorted_data = sort_items_by_price(data) print("Sorted: " + str(sorted_data))

SECTION 3: PROGRAMMING LANGUAGE QUESTIONS

Q9: Explain the concept of 'Decorators' in Python and how they are used.

Answer: A decorator in Python is a design pattern that allows a user to add new functionality to an

Q10: What is the difference between 'Global' and 'Local' variables in Python?

Answer: Global variables are those defined outside of any function and can be accessed by any part of the program, including inside functions. Local variables are defined inside a function and can only be accessed within that function's scope. If you try to access a local variable outside its function, Python will raise a NameError. To modify a global variable inside a function, the 'global' keyword must be used; otherwise, Python creates a new local variable with the same name.

SECTION 4: REMAINING QUESTIONS (Q11-Q15)

Q11: What is a Primary Key in a Database? Answer: A primary key is a specific choice of a minimal set of attributes (columns) that uniquely specify a tuple (row) in a relation (table). It cannot contain NULL values.

Q12: Define the term 'Multitasking' in Operating Systems. Answer: Multitasking is the concurrent execution of multiple tasks (also known as processes) over a certain period of time. New tasks can interrupt already started ones.

Q13: What does 'Self' represent in a Python class? Answer: 'Self' represents the instance of the class. By using the 'self' keyword, we can access the attributes and methods of the class in python. It binds the attributes with the given arguments.

Q14: Explain the concept of 'DNS' in Networking. Answer: The Domain Name System (DNS) is the phonebook of the Internet. Humans access information online through domain names, like https://www.google.com/search?q=google.com. DNS translates domain names to IP addresses so browsers can load Internet resources.

Q15: What is the purpose of 'Break' and 'Continue' statements in loops? Answer: The 'break' statement is used to terminate the loop entirely when a condition is met. The 'continue' statement is used to skip the current iteration and move to the next cycle of the loop.

Would you like me to generate these as downloadable files or create more assignment variations for your evaluation agent?