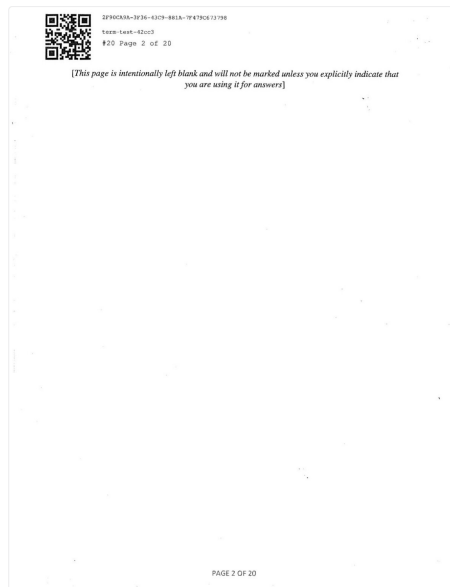


My grades for Term Test



Q1 16

25802608-3821-4389-A3E1-96081-083A731
Year: 2023
#20 Page 3 of 20

Question 1. [20 points]

The following diagram represents a simplified *Microservices* architecture. The API gateway used in this architecture routes requests from clients to services and it is designed to handle two types of client applications only. To support other types of client applications, new functionalities should be added to the gateway.

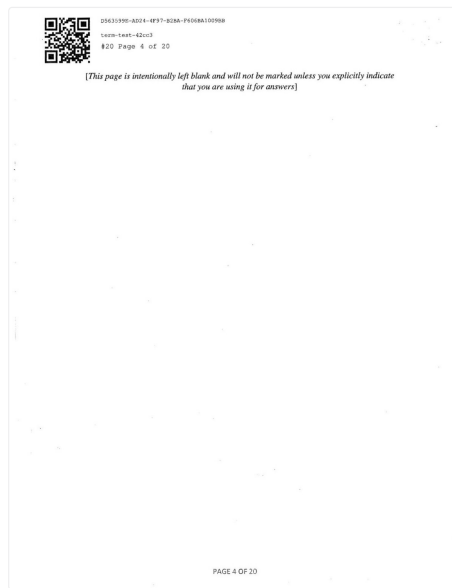
a) Explain two aspects of this design that promote maintainability. [8 points]

The API gateway routes the clients to the microservices. This allows new microservices to be added without the clients needing to change. This provides the coupling which is important for maintainability. Each microservice also does a specific task which allows high cohesion which helps improve the maintainability.

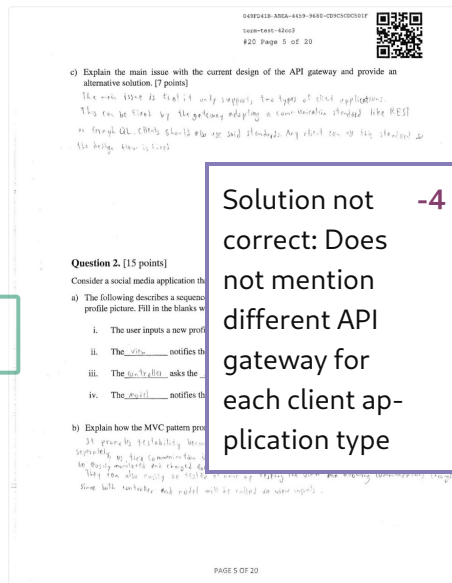
b) Explain the role of the service registry. [5 points]

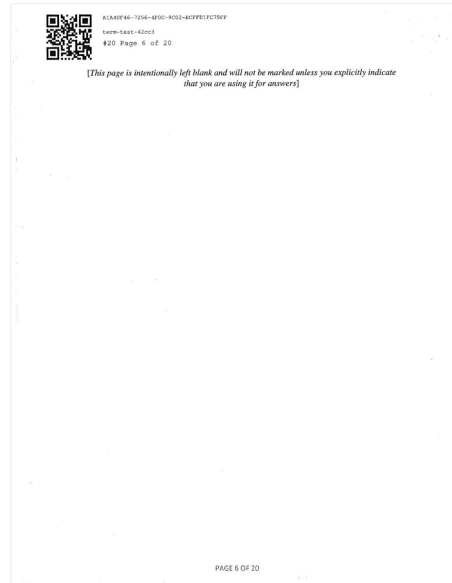
The service registry keeps track of which services are available and how to reach them. When a new service is added, it can be registered with the service registry and the API gateway can simply check the registry to find the service.

PAGE 3 OF 20



Q2 15






Q3

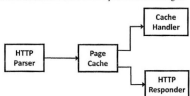
10

0203446-0336-4676-82A6-40706220F10
exam-test-42603
#20 Page 7 of 20



Question 3. [10 points]

The following diagram represents the architecture of an HTTP server consisting of several components that are connected by event queues. Each component processes the incoming events and dispatches zero or more events on the queues of other stages.



```
graph LR;
    HP[HTTP Parser] --> PC[Page Cache];
    PC --> CH[Cache Handler];
    PC --> HR[HTTP Responder];
    CH --> HR;
```

Identify the architectural pattern used and explain one advantage and one disadvantage of this pattern.

This is page-filling answer that can prove a little bit more useful.
One advantage is that the pattern can be used to handle multiple requests in parallel.
One disadvantage is that the pattern can be used to handle multiple requests in parallel.
which are always changing the results.

Correct 10

Question 4. [25 points]

Consider the following code:

```
class ElectronicDevice {
    String deviceType;
    double price;

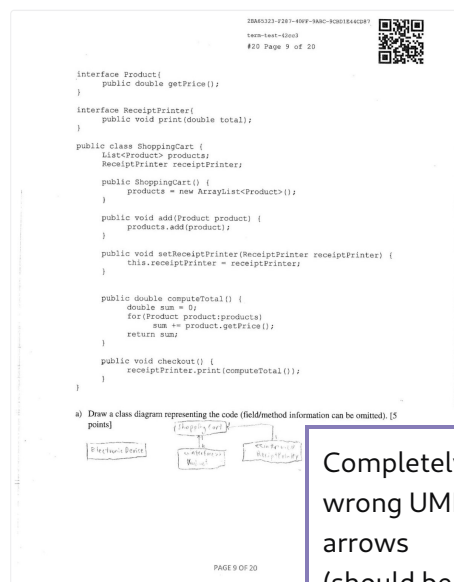
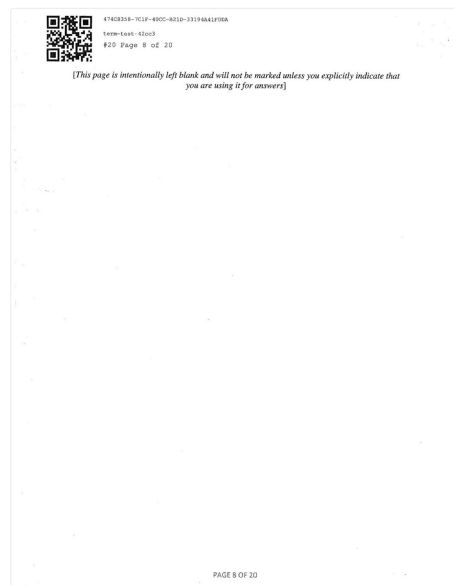
    public ElectronicDevice(String deviceType, double price) {
        this.deviceType = deviceType;
        this.price = price;
    }

    public double getPrice() {
        return price;
    }
}
```

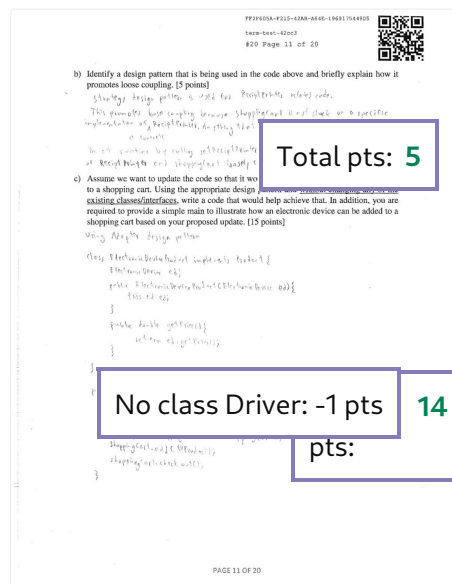
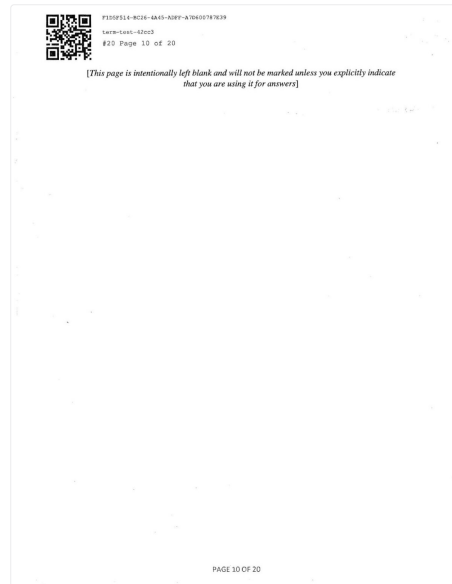
PAGE 7 OF 20

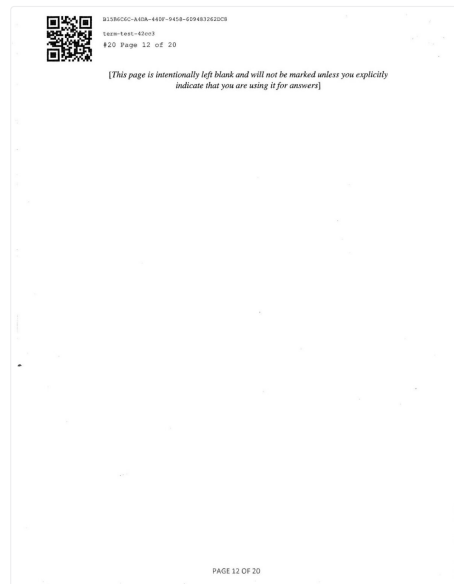
Q4

22

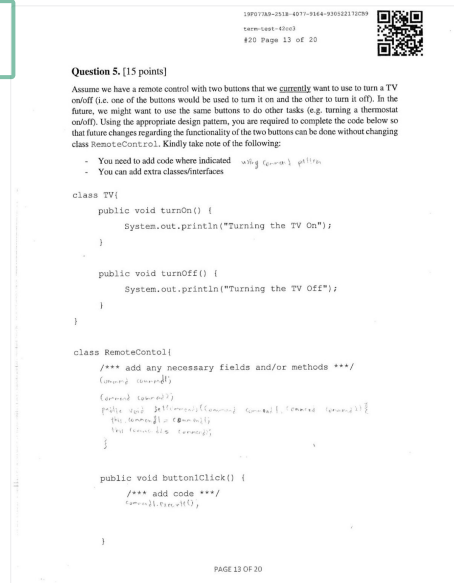


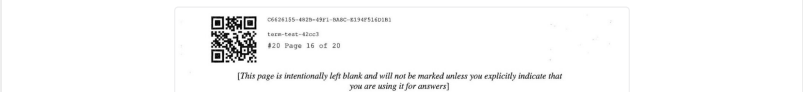
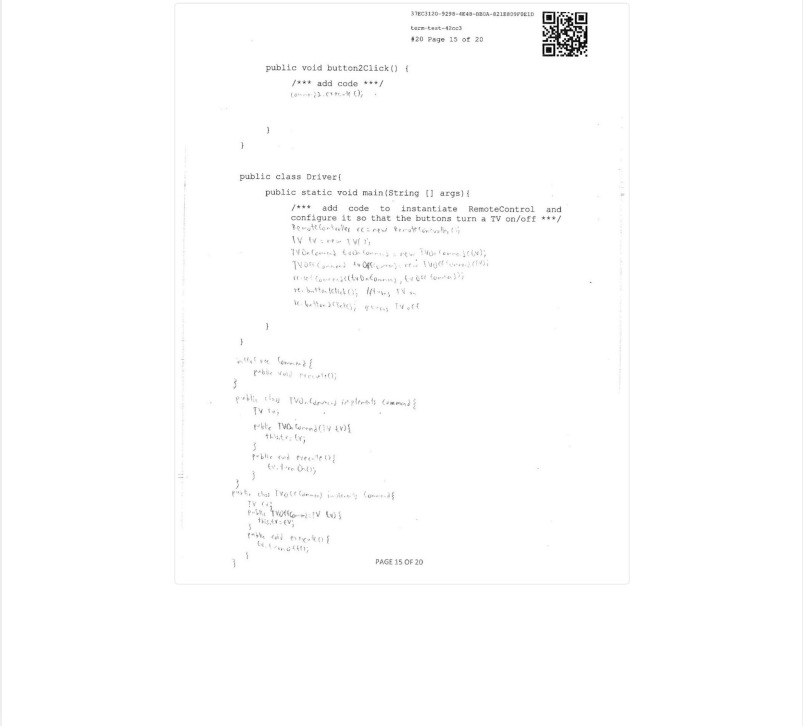
Completely wrong UML arrows (should be weak aggregations): -2 pts





Q5 15








Q6 12

CS308FA-1280-0880-8020-566037040803
term-test-02m3
#20 Page 17 of 20



Question 6. [15 points]

a) Briefly explain whether the following statement is true or false. [3 points]
"If software system with a proper initial design would never require refactoring."
 $\forall x, y, z, \text{ if } x \text{ is a } \text{Design} \wedge y \text{ is a } \text{System} \wedge z \text{ is a } \text{Refactoring} \wedge x \text{ is not } z \wedge y \text{ is not } z \wedge x \text{ is not } y$
 $\exists x, y, z \text{ s.t. } x \text{ is a } \text{Design} \wedge y \text{ is a } \text{System} \wedge z \text{ is a } \text{Refactoring} \wedge x \text{ is not } z \wedge y \text{ is not } z \wedge x \text{ is not } y$

b) Identify the design flaws in the following code and refactor it accordingly. [12 points]

```
abstract class Account {
    String accountId;

    public Account(String accountId) {
        this.accountID = accountId;
    }

    public abstract double computeInterestRate();

    public String getInfo() {
        if (this instanceof SavingsAccount)
            return "SavingsAccount " + accountId;
        else if (this instanceof CheckingAccount)
            return "CheckingAccount " + accountId;
        else
            return "Unsupported account type";
    }
}

class CheckingAccount extends Account {
    double balance;

    public CheckingAccount(String accountId, double balance) {
        super(accountID);
        this.balance = balance;
    }

    public void deposit(double amount) {
        balance += amount;
    }

    public void withdraw(double amount) {
        balance -= amount;
    }

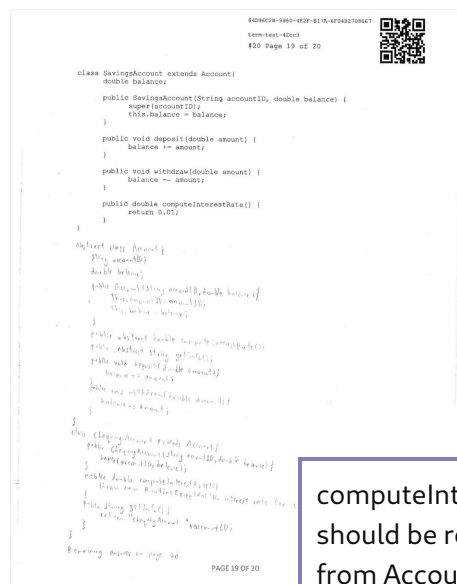
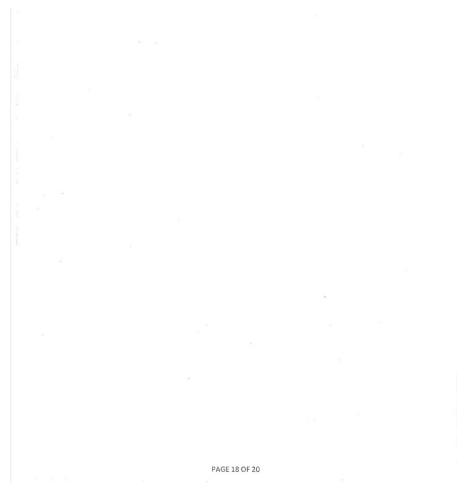
    public double computeInterestRate() {
        throw new RuntimeException("No interest rate for checking accounts");
    }
}
```

PAGE 17 OF 20

CS308FA-1280-0880-8020-566037040803
term-test-02m3
#20 Page 18 of 20



[This page is intentionally left blank and will not be marked unless you explicitly indicate that you are using it for answers]



computeInterestRate -3
should be removed
from Account and
ChequingAccount

