

**OO Analysis & Design Project 2025**

****

**SDH2-B Group 6**

**Ice Ybanez, Matas Orliukas**

**Table of Contents**

[Part A 3](#_Toc196653101)

[Question 1 - Analysis Model Domain Model 3](#_Toc196653102)

[Question 2 - System Sequence Diagram 3](#_Toc196653103)

[Question 3 – Identify and List Operations Signatures 4](#_Toc196653104)

[Question 4 i. Find the delivery address of an order 6](#_Toc196653105)

[Question 4 ii. Find the status of an order 6](#_Toc196653106)

[Question 4 iii. Find the journey time of a particular delivery schedule 7](#_Toc196653107)

[Question 4 iv. Find the total cost of an order for a particular customer 7](#_Toc196653108)

[Question 4 v. Find the current location of a driver who is out on a delivery of a customer order 8](#_Toc196653109)

[Question 5 Design Model Design Class Diagram 8](#_Toc196653110)

[Question 6 Design Model Design Class Diagram 9](#_Toc196653111)

[Part B 10](#_Toc196653112)

[Question 1 i. 10](#_Toc196653113)

[Question 1 ii. 10](#_Toc196653114)

[Question 1 iii. 11](#_Toc196653115)

[Question 1 iv. 11](#_Toc196653116)

[Question 2 12](#_Toc196653117)

[Question 3 i. 13](#_Toc196653118)

[Question 3 ii. 13](#_Toc196653119)

[Question 3 iii. 14](#_Toc196653120)

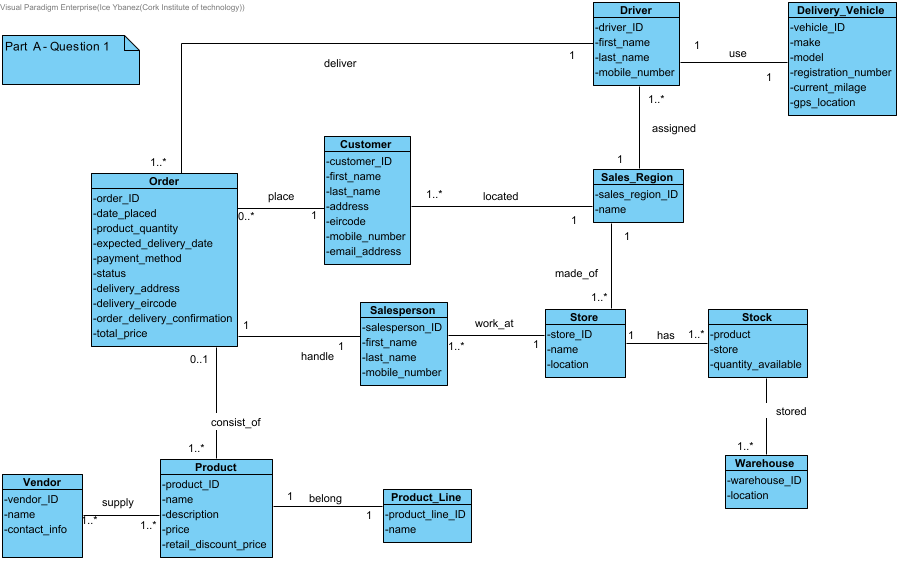
[Student Declarations 15](#_Toc196653121)

[Ice Ybanez 15](#_Toc196653122)

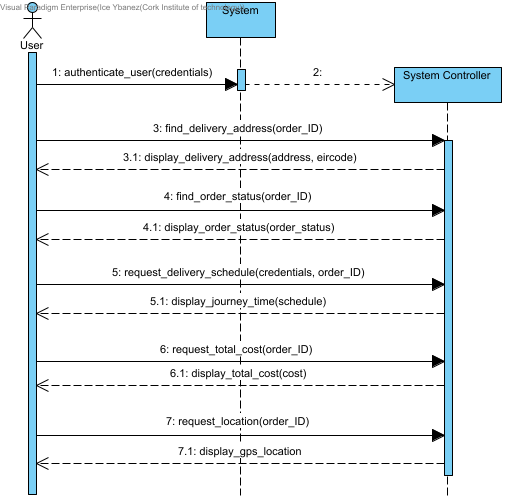
[Matas Orliukas 15](#_Toc196653123)

# Part A

## Question 1 - Analysis Model Domain Model



## Question 2 - System Sequence Diagram



## Question 3 – Identify and List Operations Signatures

i) authenticate\_customer(credentials) - Customer

input\_order(order\_ID) - Customer

request\_delivery\_address(order\_ID) - Customer

find\_order\_delivery\_address(order\_ID) - Order

display\_delivery\_address(address, eircode) - Order

ii) authenticate\_customer(credentials) - Customer

input\_order(order\_ID) - Customer

request\_order\_status(order\_ID) - Customer

find\_order\_status(order\_ID) - Order

display\_order\_status(order\_status) - Order

iii) authenticate\_driver(credentials) - Driver

input\_order(order\_ID) - Driver

request\_delivery\_schedule(credentials, order\_ID) - Driver

calculate\_total\_delivery\_schedule(order\_ID) - Order

display\_journey\_time(schedule) - Order

iv) authenticate\_customer(credentials) - Customer

search\_order(order\_ID) - Customer

retrieve\_order\_info(order\_ID) - Order

request\_total\_cost(order\_ID) - Customer

calculate\_product\_cost(price, product\_quantity) - Order

display\_total\_cost(cost) - Order

v) authenticate\_user(credentials) - Customer

request\_location(order\_ID) - Customer

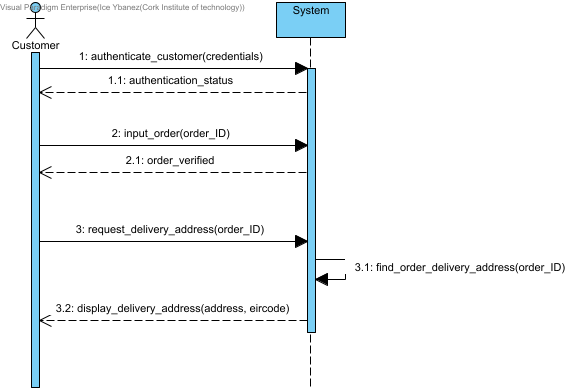
find\_driver(order\_ID) - Order

find\_vehichle(driver\_ID) - Driver

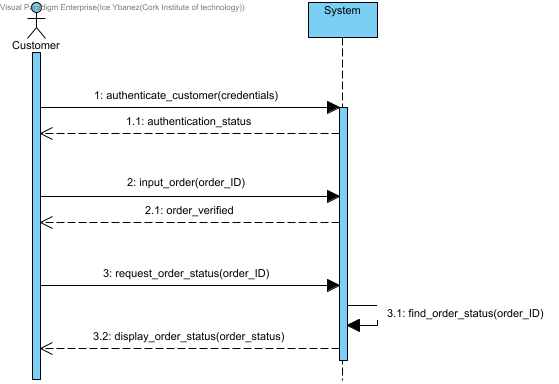
locate\_gps(vehicle\_ID) – Vehicle

display\_gps\_location(location) - Vehicle

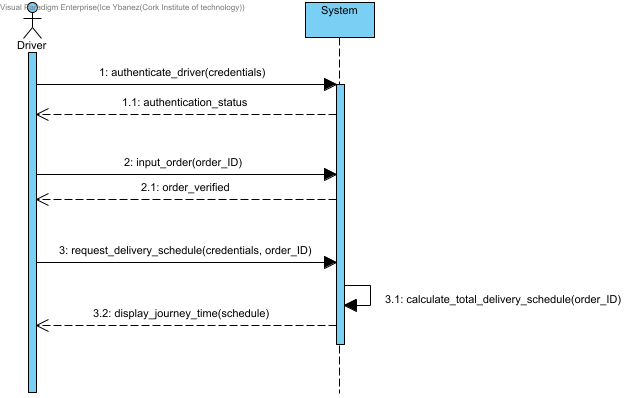
## Question 4 i. Find the delivery address of an order



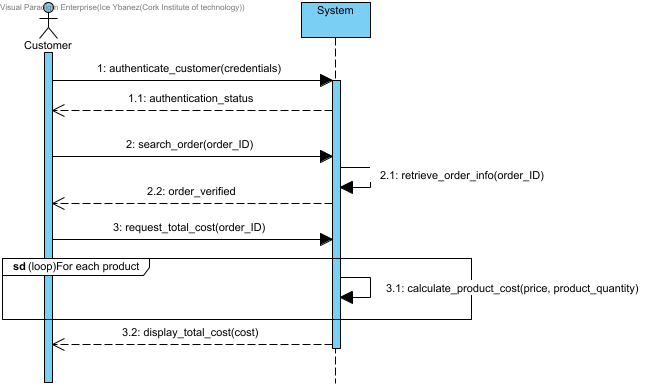
## Question 4 ii. Find the status of an order



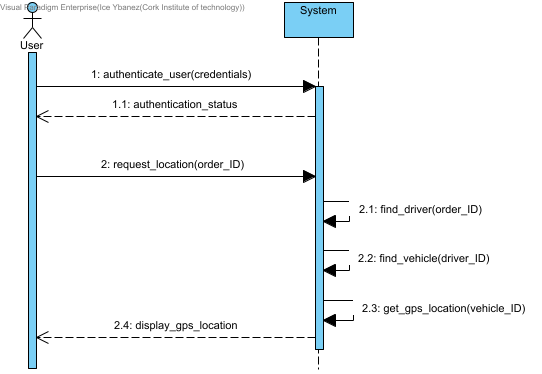
## Question 4 iii. Find the journey time of a particular delivery schedule



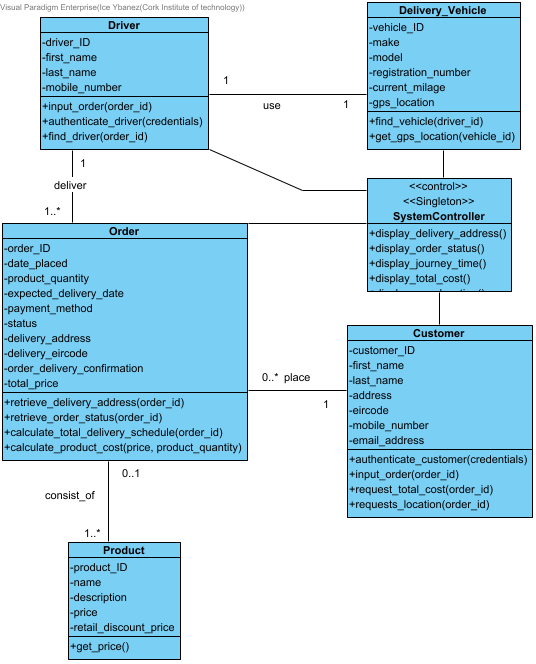
## Question 4 iv. Find the total cost of an order for a particular customer



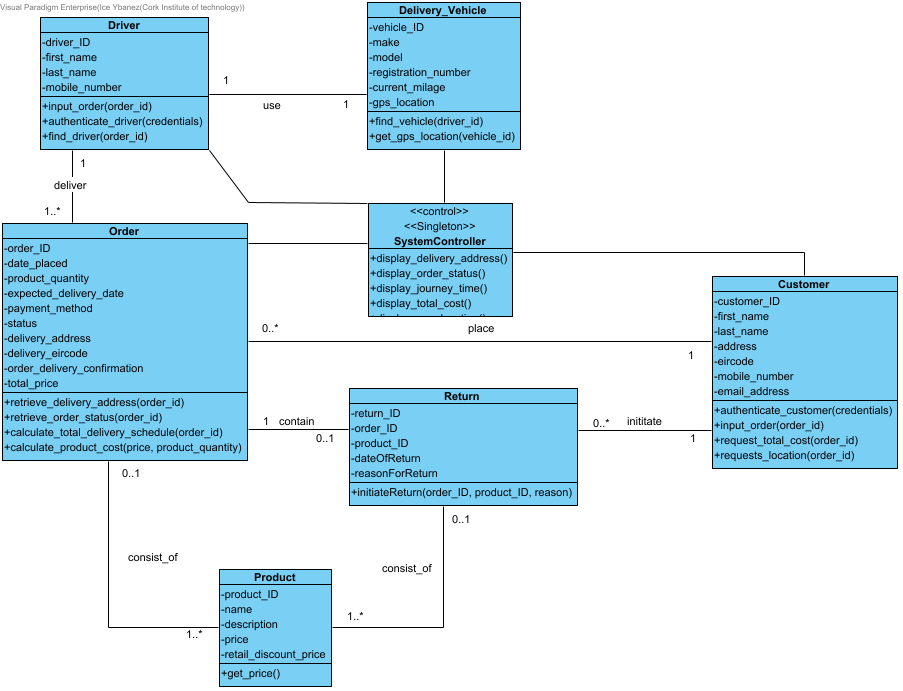
## Question 4 v. Find the current location of a driver who is out on a delivery of a customer order



## Question 5 Design Model Design Class Diagram

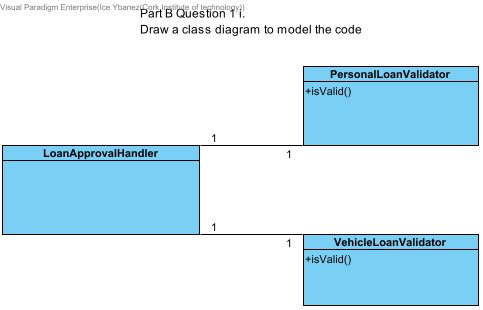


## Question 6 Design Model Design Class Diagram



# Part B

## Question 1 i.



## Question 1 ii.

The code violates the open closed principles because in the LoanApprovalHandler class, whenever a new type of loan needs to be supported, the class must be modified to add new approval methods for the new type of loan.

This modification of the LoanApprovalHandler class violates the open closed principle since software entities should be open for extension and closed for modification.

## Question 1 iii.

public interface LoanValidator {

boolean isValid();

}

public class PersonalLoanValidator implements LoanValidator {

@Override

public boolean isValid() {

// Personal loan validation logic

}

}

public class VehicleLoanValidator implements LoanValidatorx {

@Override

public boolean isValid() {

// Vehicle loan validation logic

}

}

public class LoanApprovalHandler {

public void approveLoan(LoanValidator validator) {

if(validator.isValid()) {

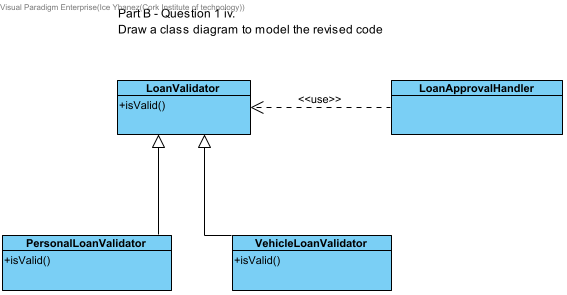
// Process the loan

}

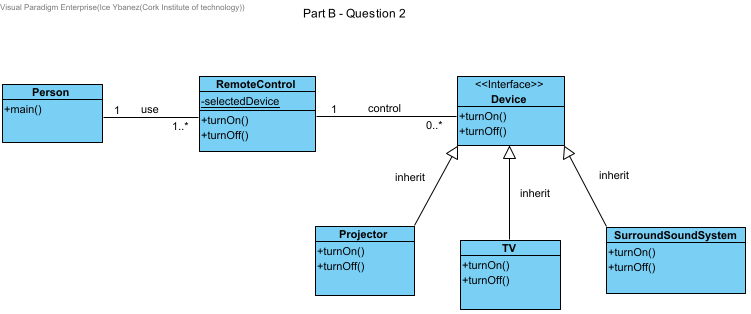
}

}

## Question 1 iv.



## Question 2



We have designed the class diagram in a way that the Person class interacts with the RemoteControl class to operate devices found in the main method of the Person class.

The RemoteControl class has a static method, selectedDevice, that allows the Person to choose a device and the turnOn and turnOff methods call the selected device’s methods.

This follows the open - closed principle.

The Device class is a common interface that all devices will implement. It also has turnOn and turnOff methods to be used in the RemoteControl class.

Since the Device class is an interface, it allows for more devices to be added in the home entertainment system without breaking the code because you only need to create a new class that implements the Device class for a new device to be created.

Finally, the Projector, TV and SurroundSoundSystem classes are classes that implement the Device class with each having their own specific implementation of the methods turnOn and turnOff.

## Question 3 i.

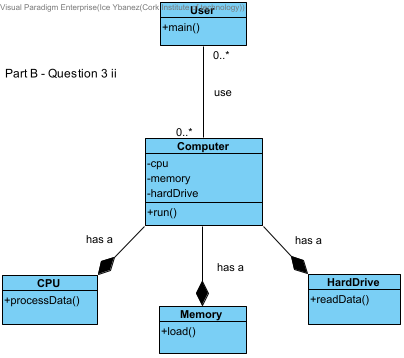
The facade design pattern provides a simplified interface for a larger and more complex sub-system.

Instead of interacting with multiple classes directly, the pattern introduces a single-entry point that encapsulates the functionality of multiple classes, making the system easier to use.

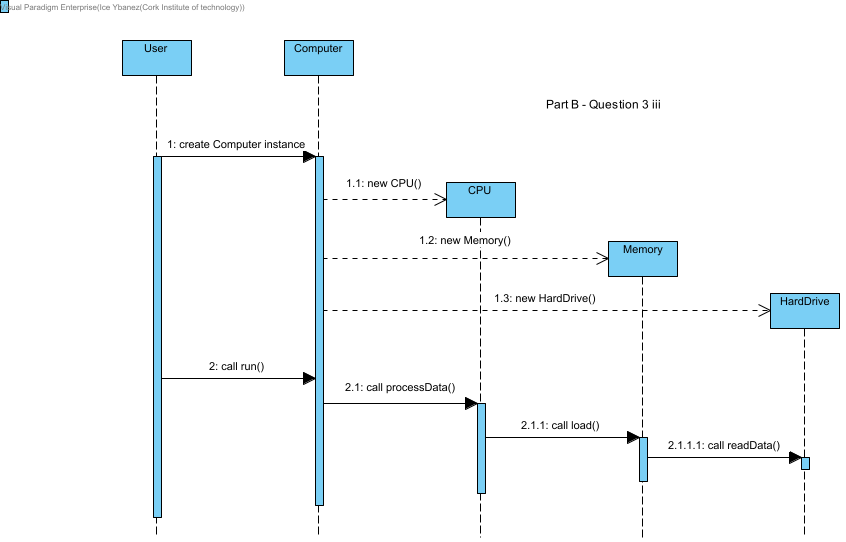
This is seen in how the sub-system classes like CPU, Memory and HardDrive classes are encapsulated in the facade design class Computer and used in the run method.

The User class then uses the Computer class without knowing the complexities of the sub-system classes with all the functionality still available.

## Question 3 ii.



## Question 3 iii.



# Student Declarations

## Ice Ybanez

**Student declaration**

I can confirm the following details:

**Student ID/Registration number:** \_\_\_\_\_\_\_\_R00176611\_\_\_\_\_\_\_\_\_

**Name:** \_\_\_\_\_Ice Ybanez\_\_\_\_\_\_

**Module Name:** OO Analysis and Design Lab

**Lecturer:** Nasir Ahmad/ Vincent Emeakaroha

**Due Date:** April 27, 2025 @midnight

## Matas Orliukas

**Student declaration**

I can confirm the following details:

**Student ID/Registration number:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Name:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Module Name:** OO Analysis and Design Lab

**Lecturer:** Nasir Ahmad/ Vincent Emeakaroha

**Due Date:** April 27, 2025 @midnight