Department of Computing

Module Name: **Software Development and Application Modelling**

Module Code: **COMP40003**

Title of Assignment: **Assignment 2**

Module Learning Outcomes for This Assignment

3. Demonstrate, apply and document to the appropriate standards the key techniques of business analysis and application modelling.	Analysis Enquiry
4. Implement Object-Oriented application models in a suitable programming language.	Application

Submission deadline

Submit by: Friday, 19 May 2023 no later than 23:59 (via Blackboard)

Demonstration dates

Week beginning 22 May 2023

Demonstrations will be held by appointment with an SDAM tutor

Feedback deadlines

Part 1: 20 working days after submission deadline (via Blackboard)

Part 2: 20 working days after your demonstration (via Blackboard)

Assignment overview

This document describes the entire assignment.

In the tutorial worksheets for weeks 3, 5, 6, 7, 8 and 9, there are assignment tasks that guide you in small steps through the assignment. Do not try to rush ahead with the assignment, just do what each assignment task tells you.

Part 1 – Group work (Assignment task in weeks 7 and 8)

This part of the assignment must be done in groups of 3-6 students.

Stationery management system

A company wants a computer system for tracking stationery usage. Stationery items (pens, pencils, paper, etc.) are added to stock and taken from stock.

When items are added to stock, the quantity and price paid are recorded in the system. When an item is taken from stock, the person taking the item is recorded.

Each event is stored in a transaction log.

At any time, the following reports can be requested:

- An inventory of all items currently held in stock;
- Expenditure on each item and all items;
- The transaction log;
- Personal usage.

The computer system is to provide the following use cases:

Use case	Notes
Add to stock	Read item code. If the code is not already in stock, read name, quantity, price paid, and date added. Create (or update) the stock item details. Make an entry in the transaction log.
Take from stock	Read item code, quantity taken, name of person removing stock, and date taken. If the item is in stock and there is sufficient quantity in stock, deduct the quantity taken from the stock item details and make an entry in the transaction log.
Inventory report	Output list of all items. For each item, show its code, name, and current quantity in stock
Financial report	Output total expenditure for each item, and total money spent on all items.

Display transaction log	Output the list of all transaction log entries. Each record should show:
	• Date of transaction
	• Type of transaction
	  ○ Add (show item code, name, and price paid)
	  ○ Remove (show item code, name, and person's name)

Report personal usage	Read person name. Output all items taken out of stock by that person. Show item code, item name, date taken

Your task

NB: This part of the assignment must be modelled as an Object-Oriented system.

Model

As a group, follow the Unified Software Development Process (USDP) to develop an object-oriented model (using UML) for the computerised system described above. Use Microsoft Visio for your models.

Each person in the group designs one use case using all the following diagrams. The diagrams should be developed in the order listed:

- Use case diagram (one for the entire application)
- Activity diagrams (one per use case)
- Analysis diagrams (one per use case)
- A class diagram with relationships between the classes (one for the entire application)
- Sequence diagrams (one per use case)

You might also write pseudo-code for any method that you think would benefit from the additional information.

The number of use cases to design will depend on the number of students in the group:

- 3 students: Add to stock; Take from stock; Inventory report.
- 4 students: As above, plus Display transaction log.
- 5 students: As above, plus Report personal usage.
- 6 students: All use cases.

Before submission, collate your individual contributions into a single, coherent design document as a PDF file, meeting the following requirements:

1. All diagrams of the same type must be grouped together.
2. Indicate clearly in your document who has developed which diagrams.
3. If anyone did not contribute, state this clearly in your document.

Part 2 – Individual work (Assignment tasks in weeks 3, 5, 6 and 9)

This part of the assignment must be done on your own. You may only ask your SDAM tutor for help.

Scenario: Taxi management system

A taxi company wants a computer system for tracking the fares taken during the day. A fare can only be collected from a taxi rank. There are three taxi ranks. A taxi always joins the back of a rank, and the taxi at the front of the rank is always used for the next fare.

When a fare is taken, the agreed price is recorded in the system. When the fare is dropped at the destination, the payment of the price is recorded.

Each event is stored in a transaction log.

The computer system is to provide the following operations:

Operation	**Information**
Taxi joins a rank	Read taxi number and rank id. If the taxi is on the road (i.e. not in any rank) and the rank has space available, Store the taxi number at the back of the selected rank Make an entry in the transaction log.
Taxi leaves a rank	Read rank id, fare destination, agreed price. Remove the front taxi from the rank (if there is one) and store the fare information. Make an entry in the transaction log.
Taxi drops a fare	Read taxi number, price paid or not. If the taxi is on the road, Record whether price has been paid. Make an entry in the transaction log.
Financial report	Output total money paid to each taxi, and total money paid to all taxis.

Display transaction log	Output the list of all transaction log entries. Each record should show:
	• Date/time of transaction
	• Type of transaction
	  ○ Join (show taxi number and rank id)
	  ○ Leave (show taxi number, rank id, destination, agreed price)
	  ○ Drop fare (show taxi number, price paid or not)

Report taxi locations	Output all taxis showing:
	• Taxi number
	• Location
	  ○ In rank *<show rank id>*
	  ○ On the road to *<show destination>*

Dates and times are to be input and displayed as `dd/MM/yyyy HH:mm` using 24-hour clock.

Appropriate validation is to be done on all input data.

Your task

NB: This part of the assignment must be written as an Object-Oriented application in C#.

Download from Blackboard the design diagrams for this part of the assignment.

As directed by the tutorial assignment tasks, download the next stage of the assignment and write code to pass the unit tests included in the download.

Remember: You must not ask anyone other than your SDAM tutor for help.

Hint: If you are not sure what to write for a given method, try to work it out by studying the corresponding tests. If you are still not sure, please discuss it with your SDAM tutor.

Recommended way of working

All tests except the first in "_01_Taxi_Tests.cs" have been commented out. As you develop your code, gradually uncomment each test, file by file, to increase the level of testing.

For example…

1. Write the Taxi class so that it matches the UML class diagram. Then use the Test Explorer panel to run the first test in "_01_Taxi_Tests.cs". If the test does not pass, correct your code and run the test again. Keep doing this until the test passes.

2. Uncomment the second test in "_01_Taxi_Tests.cs" and run all tests. If the tests do not pass, correct your code and run the tests again. Keep doing this until all tests pass. Then uncomment the third test, and so on until you have completed the current stage.

3. Take the same approach for each of the other stages, as directed in the tutorial assignment tasks.

Submit your assignment

Submit your assignment using the following links on Blackboard:

- **"Assignment 2, Part 1 group submission"** for:
  - Your group's UML model as one PDF document.

- **"Assignment 2, Part 2 individual submission"** for:
  - A ZIP file containing your Visual Studio project folder (clean the solution before compressing it).

Mark Scheme

You should refer frequently to the marking scheme during your work on this assignment. You should ensure that your work presents evidence to support the awarding of marks in each criterion of the rubrics.

The rubrics will give you quantitative and qualitative feedback on your work.

You are reminded of the university's policy about plagiarism, as described at:
http://www.staffs.ac.uk/support_depts/info_centre/handbook/academic-life.jsp

Marks for this assignment will be allocated as follows:

Part 1 (30%)

Your group's model will be assessed using Rubric 1 (see Appendix A).

If you did not contribute to the model, you will score zero marks for Part 1 of the assignment. All contributors to the model will receive the same mark.

Part 2 (70%)

Your software will be assessed in your demonstration using Rubric 2 (see Appendix A).

After submitting your work, you will be required to demonstrate your software to a module tutor. First, you will be asked to run the 100 unit tests to show how many tests pass, and then you will be asked to run the application to show your user interface. You may be asked questions about your code. Incorrect answers might result in your marks being reduced.

If you do not demonstrate your software, you will score zero marks for Part 2 of the assignment.

Instructions for the demonstration

Book your demonstration appointment time with your SDAM tutor.

To prevent delays in the demonstration, it is essential that you are ready to begin **at least** five minutes before for your appointment. You should do the following before your demonstration start-time:

1. Log on to a PC (or your own laptop) in the demonstration room at a location indicated by your assessor
2. Open Visual Studio, and prepare your application
3. Be ready to show the tutor that the dates on your files are the same as those submitted via Blackboard
4. Make sure that your system is in its initial state (i.e. ready to run without delay)
5. Make sure you have uncommented only those tests that will pass
6. Await instructions from the tutor who will assess your demonstration

Appendix A: Rubrics

Part 1 – Group work (30%)

Criterion (weighting)	Mark				
	0	**1-3** Unsatisfactory	**4-6** Satisfactory	**7-9** Excellent	**10** Exceptional
Use case diagram (5%)	No use case diagram	Another developer would not know which activity diagrams to prepare	Good use of notation; clear layout; another developer would know which activity diagrams to prepare with several supplementary questions	Presented to a professional standard; very minor mistakes in the use of the notation; another developer would know which activity diagrams to prepare with few supplementary questions	No mistakes in the use of the notation; another developer would know which activity diagrams to prepare without any supplementary questions
Activity diagrams (15%)	No activity diagrams	Another developer could not use the diagrams to derive the analysis diagrams	Good use of notation; clear layout; another developer could use the diagrams to derive the analysis diagrams with several supplementary questions	Presented to a professional standard; very minor mistakes in the use of the notation; another developer could use the diagrams to derive the analysis diagrams with few supplementary questions	No mistakes in the use of the notation; another developer could use the diagrams to derive the analysis diagrams without any supplementary questions
Analysis diagrams (20%)	No analysis diagrams	Another developer could not use the diagrams to derive the class diagram	Good use of notation; clear layout; another developer could use the diagrams to derive the class diagram with several supplementary questions	Presented to a professional standard; no mistakes in the use of the notation; another developer could use the diagrams to derive the class diagram with few supplementary questions	No mistakes in the use of the notation; another developer could use the diagrams to derive the class diagram without any supplementary questions
Class diagram (30%)	No class diagram	Many significant errors; Entirely inadequate for writing the program code	Good use of notation; another developer could use the diagram to write the program code with several supplementary questions; does not use inheritance for the transactions	Presented to a professional standard; no mistakes in the use of the notation; another developer could use the diagram to write the program code with few supplementary questions; uses inheritance for the transactions	No mistakes in the use of the notation; another developer could use the diagram to write the program code without any supplementary questions
Sequence diagrams (30%)	No sequence diagrams	Many significant errors; Entirely inadequate for writing the program code	Good use of notation; another developer could use the diagrams to write the program code with several supplementary questions	Presented to a professional standard; no mistakes in the use of the notation; another developer could use the diagram to write the program code with few supplementary questions	No mistakes in the use of the notation; another developer could use the diagrams to write the program code without any supplementary questions

Part 2 – Individual work (70%)

Criterion (weighting)	Mark				
	0	**1-3** Unsatisfactory	**4-6** Satisfactory	**7-9** Excellent	**10** Exceptional
Unit tests (100%)	The mark given here will be the number of unit tests that pass. The test classes have the following number of tests: • _01_Taxi_Tests: 21 • _02_Rank_Tests: 8 • _03_TaxiManager_Tests: 7 • _04_RankManager_Tests: 20 • _05_Transaction_Tests: 10 • _06_TransactionManager_Tests: 4 • _07_UserUI_Tests: 30				