University of Westminster
Department of Computer Science

7SENG001C.2 Enterprise Development – Coursework One (2020/2021)	
Module leader	M. Cassim Farook
Unit	Coursework 1 Refer / Defer – Analysis design and evaluation.
Weighting:	40%
Qualifying mark	35%
Description	Use of UML for the analysis and design of a domain problem. A report on the design decisions and the suitability of the design for implementation.
Learning Outcomes Covered in this Assignment:	This assignment contributes towards the following Learning Outcomes (LOs): - LO3 Demonstrate a thorough understanding of the design and implementation of applications, together with a critical review - LO4 To be able to critically assess the need for local and remote persistent storage
Handed Out:	8th February 2021
Due Date	15th March 2021, 1 pm UK time
Expected deliverables	PDF report uploaded to Turnitin as pdf file.

Method of Submission:	Electronic submission on BB via a close to the submission time. The file you upload should have the following naming format:
	        \<Cw1_StudentNumber_FullName.pdf>
	        E.g. Cw1_w123456_JohnSmith.pdf

Type of Feedback and Due Date:	Feedback and marks 15 working days (3 weeks) after the submission deadline. All marks will remain provisional until formally agreed by an Assessment Board.

Assessment regulations

Refer to section 4 of the "How you study" guide for undergraduate students for a clarification of how you are assessed, penalties and late submissions, what constitutes plagiarism etc.

Penalty for Late Submission

If you submit your coursework late but within 24 hours or one working day of the specified deadline, 10 marks will be deducted from the final mark, as a penalty for late submission, except for work which obtains a mark in the range 40 – 49%, in which case the mark will be capped at the pass mark (40%). If you submit your coursework more than 24 hours or more than one working day after the specified deadline you will be given a mark of zero for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid.

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University

Assessment Regulations, please refer to the following website:**http://www.westminster.ac.uk/study/current-students/resources/academic-regulations**

Coursework Description

Problem Statement

You are to design a personal financial management and prediction tool for personal use. This tool will enable you to record your income and expenses, produce reports, and can also be used to predict your financial status at a future date based on your expense patterns.

For extra credit, add time management functionality to your system. Enable the user to maintain events for a given day. Events may be one-off or recurring, and will be either appointments or tasks. You should also provide a weekly view showing the events and expenses for each day of the week.

Develop a design document for this application. You are also to produce a report describing the reasons for your design decisions and how suitable you deem your design is for implementation. You will then use this design to implement your application for CW2.

The design document will consist of these sections:

Part A – Requirements

List the main functions that you will build into your product. Use Appendix A to get started. You should also produce designs for your forms.

Your objective is to list the functionality of your application. The requirements should answer the questions:

What does your product do? What activities can users perform while using it?

State the software/system requirements for the system using this style:

R1 The software shall allow the user to enter expenses for a certain date.
R1.1 The software must allow for the setting up of a payer or a payee.
R1.2 The software will provide a form for the entry of expenses.
R1.2.3 An option will be given to enter recurring or one off expenses.
…
R2 Allow the user to produce a financial report.
R3 Allow the user to produce and expense prediction for a certain date .
Description: For each you may put a sentence or paragraph further describing the requirement.
… and so forth.

Record as many as you can. These are requirements so do not discuss how it shall be done just **what** needs to be done.
Additionally you will need to list non-functional requirements:

NF1. The application must run on windows.
NF2 The application must be implemented using c# and windows forms.
…
Marks Available 10%.

Part B – Use Case Diagrams

1)   Based on your requirements document, draw one or more use case diagrams for the system. Note: The use of correct UML is important here. ***Marks Available 5%.***

2)   For each use case – create a use case description for the main flow (see Appendix B **for the template on which to base these)** *Marks Available 10%.*

Part C – Classes

1)   In consideration of your use case descriptions, discover as many classes as possible and record these in a CRC (Classes, Responsibilities, and Collaborations) table. (See **Appendix C** for a template). ***Marks Available 10%.***

2)   Draw a domain model for the system (**see lecture notes for an example**). You do not need to put in any attributes or methods but you must label all associations and use correct UML when describing generalisations and aggregations etc. ***Marks Available 15%.***

Part D – Collaboration

For each Use Case draw an analysis sequence diagram (see Appendix D as an example) using the Model View Controller as a design pattern.
Marks Available 15%.

Part E – Activity

Derive an algorithm for the prediction of your finances at a given date based on your current usage. Draw a *detailed* activity diagram to describe the calculation. This should be detailed enough for you to produce a coded algorithm for your CW2. ***Marks Available 15%.***

It is important that you be as concise as possible with your design as you will be required to implement and deploy your design for the second coursework.

Part F – Report

1. A 500 to 1000 word report on the reasons for the design decisions you took. 10%

2. A 500 to 1000 word evaluation of the suitability of your design for implementation. What are the strengths and what is lacking. Think about how you would react if you were a manger and someone gave you this design. 10%

Marks Available 20%.

Coursework Marking scheme

The Coursework will be marked based on the following marking criteria:

Criteria	Component Mark	Mark Provided	Comments
PART A Requirements	10		
PART B Use cases			
- Diagram	5		
- Descriptions	10		
PART C Classes			
- CRC	10		
- Domain model	15		
PART D Collaboration	15		
PART E Activity	15		
PART F Evaluation			
- Design Desicions	10		
- Suitability for implementation	10		
Total	**100**		

Appendices to assist are found in the next page.

APPENDICES

Appendix A

This tool will allow an individual user to keep a record of their expenses and also to predict financial status at a certain date.

User Interface

This will be built as a .NET C# windows forms desktop application. You are free to design the user interface as you wish but here are the minimal requirements. The interface **must at least have these views**:

1) A contacts view for entering and updating the details of contacts (payers or payees).
2) An expenses entry view for entering and updating expense details for a certain day.
3) A financial report view – showing all expenses for a chosen date range.
4) A view that enables the user to see their predicted financial state at a certain date.

For extra credit:
5) A view for entering events: appointments and tasks.
6) A weekly view displaying daily events and expenses.

It is up to you how you design your forms. We are purposely not giving you a design example to avoid everyone having the same design. You are advised to create mockups and storyboards and modify them iteratively as you develop your design document.

Your design decisions should be included in your report.

Persistent storage of run time data

The data for the expenses will be created by a view that allows the specification of the expenses to be entered for a date, and this should be a **programmatic dynamic interface**. Once the user has finished you need to save the expense data as an XML file and in a database of your choice. When the application is run again (after closing) the system shall use the XML data to populate the data on your interface. It should use the database data for the financial report. When writing to or reading from the database the activity should be threaded (to enable the interface to be useable while writing to an external database)

Appendix B - Use Case Scenario Template

Use Case: **<Enter Use Case name here>**
**<Enter a short name for the Use Case using an active verb phrase.
e.g. Withdraw Cash, Register Customer, Rent Video, Calculate Sales Tax, etc.>**

Id: UC-- <Enter value of Id here>
Note: Enter a unique numeric identifier for the Use Case. e.g. UC-001>

Description
<Enter description here>

**Note: Briefly describe this use case. e.g. Enter a recurring
expense.**

Primary Actor
<List the Primary actor here>

**<List the Actor who's goal is being satisfied by this Use Case and has the primary interest in the
outcome of this Use Case. e.g. Student>**

Supporting Actors (if any)
<List supporting actors here>

**<List the Actors who have a supporting role in helping the Primary Actor achieve his or her goal.
e.g. Customer, Store >**

Stakeholders and Interests (if any)
<List Stakeholders and their interests here>

**<List the various entities who may not directly interact with the system but they may have an interest in
the outcome of the use case. Identifying stakeholders and interests often helps in discovering hidden
requirements that are not readily apparent or mentioned directly by the users during discussions.**

Pre--Conditions
<List Pre--Conditions here>

**<List the system state/conditions that must be true before this Use Case can be executed. e.g. User must
have at least one expense entry.>**

Post Conditions

Success end condition
<List success end condition here>

**Note: Enter the successful end condition of the Use Case where the Primary Actor's goal is satisfied.
e.g. Expense amount is updated.**

Trigger

Note: The event that starts this Use Case Example: the user wishes to enter date for expense prediction.

<List Use Case trigger here>

Main Success Scenario

Note: Enter the Main flow of events. i.e. The steps narrating/illustrating the interaction between Actors and the System. Describe Actor's actions/stimuli and how the system responds to those stimuli. Describe the 'happy path/day' scenario, meaning the straight and simple path where everything goes 'right' and enables the primary actor to accomplish his or her goal. Main flow/path should always end with a success end condition.

1. User selects add expense entry.
2. User enters date. System checks valid date.
3. User enters number of entries.
4. User enters details for each entry.
5. System updates total for expenses.
6. System displays updated total

Variations

<Enter variations here>

A variation would be if the user has the option to enter expenses one at a time with an 'Add' button for example.

End of template

Appendix C - This an example CRC

Class Name	Type	Responsibility	Collaborations
Payee	Model	Holds all Payee properties such as name, address, …	Expense
PayeeDetailsView	View (Form)	View for creating or updating Payee	CreatePayeeViewController

Appendix D