**INTI International College Penang School of Computing**

**3+0 Bachelor of Science (Hons) in Computer Science, in collaboration with Coventry University, UK 3+0 Bachelor of Science (Hons) in Computing, in collaboration with Coventry University, UK**

# Coursework cover sheet

**Section A - To be completed by the student.**

|  |  |
| --- | --- |
| Full Name: TAN YAW PIN | |
| CU Student ID Number: 14196276 | |
| Semester: 1 | |
| Session:  **April 2023** | |
| Lecturer:  **Puteri Nursyawati Azzuri (puteri.azzuri@newinti.edu.my)** | |
| Module Code and Title:  **4067CEM Software Design** | |
| Assignment No. / Title:  **Continuous Assessment** | % of Module Mark:  **50** |
| Hand out Date:  **12 May 2023** | Due Date:  **Task 1: 02 June 2023, by 11.59pm.**  **Task 2: 07 July 2023, by 11.59pm**  **Task 3: 23 June 2023, by 11.59pm.**  **Task 4: 23 June 2023, by 11.59pm.**  **Task 5: 23 June 2023, by 11.59pm.** |
| Penalties: No late work will be accepted. If you are unable to submit coursework on time due  to extenuating circumstances, you may be eligible for an extension. Please consult the lecturer. | |
| Declaration: I/we the undersigned confirm that I/we have read and agree to abide by the University regulations on plagiarism and cheating and Faculty coursework policies and procedures. I/we confirm that this piece of work is my/our own. I/we consent to the appropriate storage of our work for plagiarism checking.  Signature(s): | |

# Section B - To be completed by the module leader

|  |  |  |
| --- | --- | --- |
| Intended learning outcomes assessed by this work:   1. Understand and apply appropriate concepts, tools, and techniques to each stage of the software development. 2. Understand and apply design patterns to software components in developing new software. 3. Demonstrate an understanding of project planning and working to agreed deadlines, along with professional, interpersonal skills and effective communication required for software production.   5. Demonstrate an awareness of, and ability to apply, social, professional, legal, and ethical standards as documented in relevant laws and professional codes of conduct such as that of  the Malaysian National Computer Confederation. | | |
| Marking scheme | Max | Mark |
| 1. User Story Mapping | 20 |  |
| 2. Setting up a GitHub |  |
| Repository | 10 |
| 3. Creating a Class diagram and |  |
| design pattern selection | 30 |
| 4. Creating a Prototype User |  |
| Interface and Usability Testing | 20 |
| 5. Discuss the ethical issue |  |
| related to the software | 20 |
| Total | 100 |  |

**The 4067CEM assessment should be completed as a full individual work over the course of the module. The assessment output are only judged at the end of the module and not by the expectations during that week. The assessment should be undertaken individually. All submissions will be checked against each other and the internet for possible plagiarism.**

Activities – These activities consist of **50%** of your coursework marks. It will be run throughout the semester and there will be a final submission at the end of the semester. These activities consist of activities that will be done in a software design phase.

# System

Student Business System for College.

# Task 1 – User Story Mapping (20 marks)

The first thing that you need to do is ask the user what they wished for in a system. The user here can be your friends as the system is related to them. Get at least 10 real users to get their feedback. Document their feedback. Use software like Trello to complete this activity.

Output – All the user stories, and backlog with goals, activities, and tasks. In Word format, uploaded it to GitHub.

Due – Week 9 of the semester. 02 June 2023, by 11.59 pm.

# Task 2 – Setting up a GitHub Repository (10 marks)

This is where the output of the tasks will be stored, Make sure you register an account, create a repository and your files are uploaded here and it is in an organized manner and can be easily found.

Output – GitHub Repository with Task 1, Task 3, Task 4 and Task 5 documents. Take note the date of the files will be shown so you must follow the due date of each task.

Due – It will be accessed on Week 14 of the semester. 07 July 2023, by 11.59 pm

# Task 3 – Creating a Class diagram and design pattern selection (30 marks)

Create a simple Class diagram which should consists of the Classes that might be used to represent the system and the association between them. You don’t have to declare the attributes and operations for this activity. You do have to explain the class responsibility of each class declared. You can use software like StarUML to complete this activity.

Output – A class diagram containing classes and associations. In Word format, uploaded it to GitHub.

Consider the problem and select a suitable design pattern that can be implemented on the problem. Give justification on why the design pattern was chosen. Draw the UML diagram representing your class diagram as a design pattern UML. Include all the abstract class/interface, concrete class, and inheritance (if any) used to represent the problem.

Output – UML diagram representing the design pattern. In Word format, uploaded it to GitHub. Due – Week 12 of the semester. 23 June 2023, by 11.59 pm.

# Task 4 – Creating a Prototype User Interface and Usability Testing (20 marks)

Create a Prototype User Interface (hand drawn/digital) of TWO (2) important functions of the proposed system. Come up with usability testing questions. You don’t have to carry out the test, just prepare the questions. You should indicate what you are testing for in the Usability Testing.

Output – A Prototype and Usability Testing Questions. In Word format, uploaded it to GitHub. Due – Week 12 of the semester. 23 June 2023, by 11.59 pm.

# Task 5 – Discuss the ethical issue related to the software (20 marks)

Discuss and do a critical analysis of your software in these areas, privacy concerns, intellectual property rights, and effects on society.

Output – A report in Word format, uploaded to GitHub.

Due – Week 12 of the semester. 23 June 2023, by 11.59 pm.

# Submission

All tasks needed to be documented in Word format and submitted for SafeAssign checking (Links will be provided before the due date).

Upload the document and the SafeAssign report to your GitHub repository by each task due date. Due – It will be accessed on Week 14 of the semester. 07 July 2023, by 11.59 pm

# Marking Rubric for Continuous Assessment

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Marks Below 40%** | **Marks in the range 40 – 49%** | **Marks in the range 50 – 59%** | **Marks in the range 60 – 69%** | **Marks 70% and above** |
| **User Story** | User Story Mapping | User Story Mapping | User Story Mapping | User Story Mapping | User Story Mapping done and does capture most important activities of the system. The breakdown of the user story mapping is excellent and uses software that can assist that process (For example Trello compared to Ms.  Word). |
| **Mapping** | not done or User | done at a minimum | done and does | done and does |
| **(20 marks)** | Story copied/does  not match the exact | level and does not  capture the | capture several  important activities of | capture several  important activities of |
|  | system. | important activities of | the system. The | the system. The |
|  |  | the system. | breakdown of the | breakdown of the user |
|  |  |  | user story mapping | story mapping is good |
|  |  |  | can be improved. | and uses software that |
|  |  |  |  | can assist that |
|  |  |  |  | process (For example |
|  |  |  |  | Trello compared to |
|  |  |  |  | Ms. Word). |
| **Setting up a** | GitHub repository | GitHub repository | GitHub repository | GitHub repository exist | GitHub repository |
| **GitHub** | does not exist or | exist and some of | exist and most of the | and all of the required | exist and all of the |
| **Repository** | cannot be accessed | the required files are | required files are | files are available at | required files are |
| **(10 marks)** | or the required files  are not available at | not available at the  time of access. | available at the time  of access. However | the time of access.  However the dates for | available at the time  of access. The dates |
|  | the time of access. |  | the dates does not | some files does not | on the files follows |
|  |  |  | follow the required | follow the required | the required |
|  |  |  | deadline. | deadline. | deadline. |
| **Creating a** | The Class diagram | The Class diagram | The Class diagram | The Class diagram | The Class diagram |
| **Class diagram** | does not represent | and design pattern | and design pattern | and design pattern | and design pattern |
| **and design pattern selection (30 marks)** | the required solution (contains generic or non- related classes  such as admin), the design pattern | represent the required solution but in a very general and incomplete way.  Required classes in | represent the required solution in a partial way. A few  required classes in the design are not | represent the required solution in a satisfactory way. Most  required classes are declared. | represent the required solution in an excellent way. All  required classes are declared. |
|  | suggested is not | the design are not | declared. |  |  |
|  | suitable for the given | declared. |  |  |  |
|  | problem. |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Creating a Prototype User Interface and Usability Testing**  **(20 marks)** | No prototype were available or the measurement for the usability testing is not clear. | The prototype cover minimalist and trivial design (such as login) and the measurements for the usability testing are not clear. | The prototype cover adequate design and several measurements for the usability testing are not clear. | The prototype cover good design and most measurements for the usability testing are clear. | The prototype cover excellent design and all measurements for the usability testing are clear. |
| **Discuss the ethical issue related to the software**  **(20 marks)** | There is no discussion on the ethical issue or only the theories are pasted back for this component. | There is an attempt to discuss on the ethical issue but no critical  analysis was done | There is an attempt to discuss on the ethical issue with some critical analysis was done | There is an attempt to discuss on the ethical issue with good critical analysis. | There is an attempt to discuss on the ethical issue with excellent critical analysis. |

**Task 3: Class Diagram and Design Pattern**

**3.1 Introduction**

The purpose of this task is to design a class diagram and explore design patterns for a student business system. In order to develop an efficient and well-structured system, a class diagram will be created to illustrate the relationships and interactions between key entities. This diagram will serve as a visual representation of the system's architecture, showcasing the classes, their attributes, and their associations. Furthermore, design patterns will be explored to address common design challenges and provide reusable solutions. By utilizing design patterns, we can enhance the system's flexibility, maintainability, and scalability while adhering to established best practices. The combination of a well-designed class diagram and the implementation of appropriate design patterns will contribute to the creation of a robust and extensible student business marketplace system.

**3.2 Class Diagrams**

A class diagram is a visual representation of the structure and relationships of the classes in a system. It provides an overview of the main components, including classes, attributes, methods, and their relationships, offering a blueprint of the system's structure.

The main components of a class diagram include:

Class: It represents a blueprint for objects, defining the attributes and behaviors that objects of that class will possess.

Attributes: They represent the properties or characteristics of a class. Attributes are typically represented as variables within a class and provide data storage.

Methods: They define the behaviors or actions that objects of a class can perform. Methods are represented as functions within a class and can manipulate the class's attributes or interact with other objects.

Relationships: They represent the associations between classes and describe how classes interact with each other. Common types of relationships include associations, aggregations, compositions, and inheritances.

Multiplicity: It specifies the number of instances of one class that can be related to a single instance of another class. It is denoted using numbers or special symbols, such as "\*" for many or "1" for one.

Inheritance: It represents the hierarchy of classes, where a subclass inherits attributes and behaviors from a superclass. Inheritance allows the reuse and extension of existing classes, promoting code reusability and maintainability.

Associations: They represent the relationships between classes, indicating that objects of one class are connected to objects of another class. Associations can be one-to-one, one-to-many, or many-to-many, and they help define the structure and interactions within the system.

By understanding these main components of a class diagram, one can effectively model the structure and relationships of classes in a system, facilitating better communication, design, and development of software applications.

This is a simple class diagram I came up with for the Student Business System

A diagram of a computer

Description automatically generated

Figure 3.2.1

The class diagram above captures the basic structure and relationships between classes in a simplified system for student-based online commerce. It includes the following classes:

Student: Represents a student participating in the online commerce system. It has attributes such as ID, name, and email.

Product: Represents a product available for sale. It has attributes such as ID, name, and price.

Order: Represents an order placed by a student for one or more products. It has attributes such as ID, status, and date.

Seller: Represents a seller who offers products for sale. It has attributes such as ID and name.

Payment: Represents a payment made for an order. It has attributes such as ID, amount, and payment status.

The class diagram depicts the relationships between these classes:

A student can have multiple orders, forming a one-to-many association between Student and Order.

An order can include multiple products, forming a one-to-many association between Order and Product.

Each order has a payment associated with it, forming a one-to-one association between Order and Payment.

A seller can offer multiple products, forming a one-to-many association between Seller and Product.

Many payments can be made by one student, forming a many-to-one association between Payment and Student.

Many payments can be received by one seller, forming a many-to-one association between Payment and Seller.

This class diagram provides a visual representation of the entities involved in the student-based online commerce system and their relationships, helping to understand the structure and interactions within the system.

**3.3 Design Pattern**

A design pattern is a reusable solution to a common problem that occurs during software design. It is a proven approach that provides a template for solving specific design issues in a consistent and efficient manner. Design patterns encapsulate best practices and help software developers create flexible, maintainable, and scalable code. Design patterns are not specific algorithms or frameworks, but rather higher-level concepts that guide the overall architecture and structure of a software system. They provide solutions for common design problems, such as object creation, communication between objects, and managing complex relationships.

There are various types of design patterns, including:

Creational Patterns: These patterns focus on object creation mechanisms, providing flexibility and decoupling the creation process from the client code. Examples include the Singleton pattern, Factory pattern, and Builder pattern.

* Factory Method: Creates an instance of several derived classes
* Singleton: A class of which only a single instance can exist
* Builder: Separates object construction from its representation

Structural Patterns: These patterns deal with the composition of classes and objects to form larger structures while ensuring flexibility and ease of use. Examples include the Adapter pattern, Composite pattern, and Decorator pattern.

* Adapter: Match interfaces of different classes
* Decorator: Add responsibilities to objects dynamically
* Composite: A tree structure of simple and composite objects

Behavioral Patterns: These patterns address the interaction and communication between objects, defining how they collaborate and distribute responsibilities. Examples include the Observer pattern, Strategy pattern, and Command pattern.

* + - * Observer: A way of notifying change to a number of classes
* Strategy: Encapsulates an algorithm inside a class
* Command: Encapsulate a command request as an object

This is the Design Pattern I came out with the observer pattern for the Student Business System to check stock in the student business system.

A diagram of a computer

Description automatically generated

Figure 3.2.2

In this diagram:

The Inventory class maintains the stock information and a list of observers (in this case, Student objects). It provides methods to get and set the stock, as well as register and remove observers. When the stock changes, the notifyObservers() method is called to inform all registered observers.

The Student class implements the Observer interface and provides the updateStock(stock: int) method, which will be called by the Inventory object when the stock changes. The Student object can then update its own state or take any necessary action based on the updated stock value.

This observer pattern allows the Student to receive updates about the stock in the inventory and react accordingly. By registering as observers, the Student can stay informed about stock changes and make decisions based on the current stock level.

**3.4 Summary**

In summary, a class diagram is a visual representation of the classes, attributes, and relationships in a system, while design patterns are reusable solutions to common software design problems. By combining the usage of class diagrams and design patterns, we can achieve a clear understanding of the structure and behavior of the system and implement proven solutions to design challenges.