INTI International College Penang **School of Engineering and Technology**
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: Matthew Loh Yet Marn	
CU Student ID Number: 13445539	
Semester: 1	
Session: **August 2022**	
Lecturer: **Nadhrah Abdul Hadi (nadhrah.abdulhadi@newinti.edu.my)**	
Module Code and Title: **4067CEM Software Design**	
Assignment No. / Title: **Continuous Assessment**	% of Module Mark: **50**
Hand out Date: **6th September 2022**	Due Date: **Task 1: 30 September 2022, by 11.59pm.** **Task 2: 18 November 2022, by 11.59pm** **Task 3: 4 November 2022, by 11.59pm.** **Task 4: 4 November 2022, by 11.59pm.** **Task 5: 4 November 2022, 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 appropriate storage of our work for plagiarism checking. Signature(s): [illegible]	

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	

Task 3 – Creating a Class diagram and design pattern selection (20 marks)

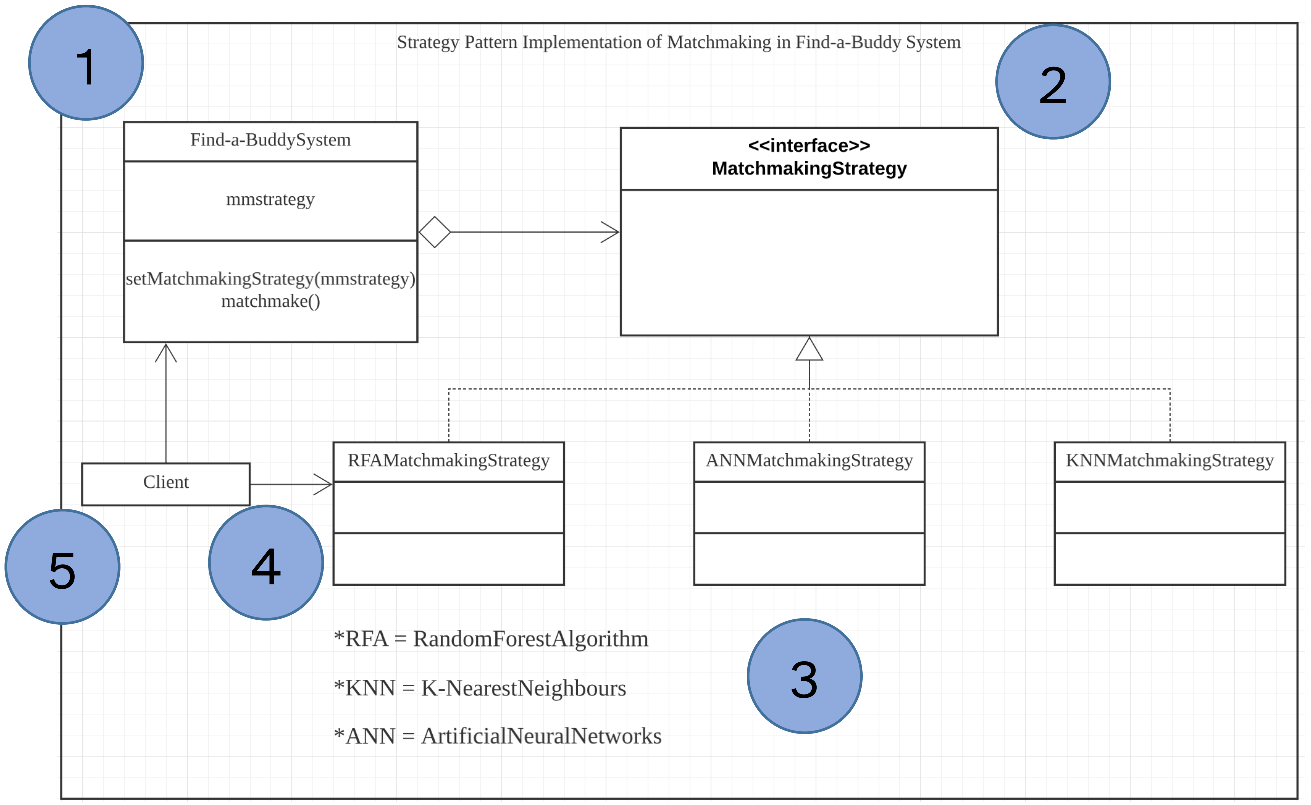
Matthew Loh Yet Marn – P21013568

BACKGROUND

In developing good software, it is crucial that we try to write good quality code. But what is good quality code and how do we write it? In an object-oriented approach, above all, a developer should never try to reinvent the wheel. Software engineering is much akin to problem-solving, and just like in problem-solving, pattern recognition is key to saving time and money when writing code. This is where design patterns come in, where pattern recognition and problem-solving intertwine to help developers design more reusable, robust and extendable object-oriented software. They are not a finished design that can be transformed directly into code but instead, are a general description or template for how to solve a problem that can be used in many different situations. Design patterns are formalized best practices that the programmer can use to solve all sorts of problems when designing an application or system. Design patterns are not specific to any one language, but are language independent as well.

Focusing now back to the College Buddy System itself, it’s important that we select the right design pattern to ensure we can set the foundation right for scalability and expandability of our software. Firstly, an introduction to the feature that is prime for switching to a design pattern and forgoing the traditional style of coding. The Find-a-Buddy feature of the College Buddy System allows users to be matched by various metrics. When two students are to be matched, it is important that we select the right algorithm to maximize the compatibility between the individuals. Problems inevitably arise with the design of the module, that is, as a result of the fact that matchmaking often needs to adapt to various social circumstances and unpredictable metrics. Whenever new algorithms for matchmaking need to be added, great difficulties become apparent in upkeeping and maintaining the code for the client, which is the Find-a-BuddySystem class. Sometimes, some algorithms can be phased out of use, or more complicated algorithms can affect the implementations of other ones. In these cases, where many of the algorithms used are related in output and only differ in under-the-hood behavior, the Strategy design pattern fits the bill. The Strategy design pattern is a behavioral design pattern that allows the encapsulation of families of algorithms, by transforming the implementations into separate classes to make objects which they execute on interchangeable. The following section will include a Class diagram for the proposed implementation of the Strategy pattern in the Find-a-Buddy module of the College Buddy System.

CLASS DIAGRAM



Strategy Pattern Implementation of Matchmaking in Find-a-Buddy System
1
2
Find-a-BuddySystem
mmstrategy
setMatchmakingStrategy(mmstrategy)
matchmake()
<<interface>>
MatchmakingStrategy
Client
RFAMatchmakingStrategy
ANNMatchmakingStrategy
KNNMatchmakingStrategy
5
4
3
*RFA = RandomForestAlgorithm
*KNN = K-NearestNeighbours
*ANN = ArtificialNeuralNetworks

EXPLANATION

In the above class diagram, Find-a-BuddySystem, MatchmakingStrategy, RFAMatchmakingStrategy, ANNMatchmakingStrategy and KNNMatchmakingStrategy represent classes.

1. In the above class diagram, the Find-a-BuddySystem class acts as the Context, which is configured with a mmstrategy (MatchmakingStrategy) object. The Find-a-BuddySystem class maintains a reference to a MatchmakingStrategy object. As a Context, it also has the ability to allow data to be passed through.
2. MatchmakingStrategy is a class that acts as an interface for all the concrete strategies. While not depicted, this interface is able to declare a method which can be called by Find-a-BuddySystem to execute a matchmaking algorithm.

3. RFAMatchmakingStrategy, ANNMatchmakingStrategy and KNNMatchmaking strategy represent variations of a matchmaking algorithm that the Find-a-BuddySystem desires. These classes are Concrete Strategies.
4. On running of the matchmaking functions, the Find-a-BuddySystem class calls the matchmaking method on the connected strategy object every time an algorithm is needed. Note that the context, Find-a-BuddySystem does not need to know how the algorithm is implemented and in fact does not possess knowledge of the MatchmakingStrategy object, including what type of strategy each one is. The Find-a-BuddySystem class only creates a reference.
5. The Client represents the class that includes the creation of the specific MatchmakingStrategy object and the passing of it to the Find-a-BuddySystem.

Using the Strategy design pattern, we are able to add more algorithms that we can matchmake the users by without having to change any parts of the Find-a-BuddySystem class itself. On runtime itself, the Strategy pattern allows for the switching of variants of an algorithm. By using the Strategy pattern, we also cut down on the need of massive conditional operators by choosing to encapsulate the algorithm's code in classes, effectively isolating the implementation details and making maintenance have stronger cohesion. The Strategy pattern also has very clear delegation of class responsibilities. The MatchmakingStrategy interfaces and is responsible for collaboration with the Find-a-BuddySystem to actually implement the chosen algorithm in the Client. The family of matchmaking algorithms are only responsible for executing their respective code as a result of interacting with MatchmakingStrategy exclusively. In summary, The MatchmakingStrategy is an abstract base class that acts as an interface for the Find-a-BuddySystem (which has the method(s) that does the actual matchmaking). Abstract methods in the MatchmakingStrategy class carry out the matchmaking depending on the strategy and thus, allow the different implementations.

CONCLUSION

Design patterns are an integral component of being a programmer that writes clean code. Overall, in this task, we successfully realize the power of design patterns and applied them in the concept of matchmaking buddies. It is important that the app presents the option to allow the selection of different algorithms for matchmaking. Furthermore, there must be an understanding of the drawbacks each design pattern has.

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 Mapping (20 marks)	User Story Mapping not done or User Story copied/does not match the exact system.	User Story Mapping done at a minimum level and does not capture the important activities of the system.	User Story Mapping done and does capture several important activities of the system. The breakdown of the user story mapping can be improved.	User Story Mapping done and does capture several important activities of the system. The breakdown of the user story mapping is good and uses software that can assist that process (For example Miro compared to Ms. Word).	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 Miro compared to Ms. Word).
Setting up a GitHub Repository (10 marks)	GitHub repository does not exist or cannot be accessed or the required files are not available at the time of access.	GitHub repository exist and some of the required files are not available at the time of access.	GitHub repository exist and most of the required files are available at the time of access. However the dates does not follow the required deadline.	GitHub repository exist and all of the required files are available at the time of access. However the dates for some files does not follow the required deadline.	GitHub repository exist and all of the required files are available at the time of access. The dates on the files follows the required deadline.

Creating a Class diagram and design pattern selection (30 marks)	The Class diagram does not represent the required solution (contains generic or non- related classes such as admin), the design pattern suggested is not suitable for the given problem.	The Class diagram and design pattern represent the required solution but in a very general and incomplete way. Required classes in the design are not declared.	The Class diagram and design pattern represent the required solution in a partial way. A few required classes in the design are not declared.	The Class diagram and design pattern represent the required solution in a satisfactory way. Most required classes are declared.	The Class diagram and design pattern represent the required solution in an excellent way. All required classes are declared.
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.