

# Declaration of Original Work for CE/CZ2002 Assignment

We hereby declare that the attached group assignment has been researched, undertaken, completed and submitted as a collective effort by the group members listed below.

We have honored the principles of academic integrity and have upheld Student Code of Academic Conduct in the completion of this work.

We understand that if plagiarism is found in the assignment, then lower marks or no marks will be awarded for the assessed work. In addition, disciplinary actions may be taken.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Course  (CE2002 or CZ2002) | Lab Group | Signature /Date |
| Audrey Ho Hai Yi | CZ2002 | FSP2 | C:\Users\AUDREY\Documents\GitHub\oodp.ass\Audrey.png |
| Ong Wei Feng Kelvin | CZ2002 | FSP2 | Kelvin.png |
| Wong Kin Sum | CZ2002 | FSP2 | KinSum.png |
| Toh Jian Hao | CZ2002 | FSP2 | JianHao.png |

Important notes:

1. Name must **EXACTLY MATCH** the one printed on your Matriculation Card.

# Design Considerations

## System Conceptual Design

The purpose of the MySTARS system is to automate and streamline the process for Student’s registering for Courses and for Administrators to manage the system. It provides a simple interface for both Students and Administrators to perform the tasks that they are required to do. It automates the steps required to perform them and also prevents the student and administrator from making mistakes while doing so.

The system has been made to accommodate both user types with reference to the different tasks they need to perform and how they can be coordinated with one another so that changes and made can be reflected into the system.

Several assumptions were made about how the system that has influenced its development, from what features to provide and also what the student and administrator are expected to know. Due to the fact that it is a console application there are certain design flaws that are unable to be fixed and have to be accepted. These are stated in point form in the list below.

## Issues that have been considered

For Student

1. System automatically keeps track of Courses that students are currently registered to or in waiting list for.
2. Detect and alert students of clashes in time table.
3. Prevents students from registering into the same Course twice.
4. Alerts the student about if they have managed to successfully been registered or they have been put into the waiting list.
5. Only allows shifting of Course Group if students are registered under a Course of that same group and if only there is a place for the student in the Course Group they wish to switch to or if there is student who has agreed to shift with them.

For Administrator

1. Ensure that Administrators do not put in an invalid entry period (characters instead of numbers and dates that are out of range such as year 1200) which would lock out all of the students from the system
2. Alerts Administrator of them making invalid inputs into the system when entering a course or student

## Assumptions and Unsolvable Design Issues

For users in general

1. As based on the assignment document these are the issue that have been considered of
   1. Need not consider multi-users concurrent login.
   2. Need not consider pre-requisite conditions when registering course.
   3. The passwords for login will be stored a flat file in hashed format and not clear text.
   4. External source implies pre-existing records and can be loaded from the file/s.
   5. Course and Students are to be stored in files. The format of storage is up to individual group’s considerations.
2. As concurrent login has no need to be considered the system will only accommodate one person at a time and has no need to provide real time alerts to certain events, such a requests to swap courses, and management of updates to the flat file system from multiple sources

For Students

1. SMS alert has not been implemented and instead a place holder method giving the message that SMS has been sent has been use to simulate the affect.
2. It is assumed that users know or can find out the Group Index number of the Course of their choice and will not be provided on the screen.
   1. Since it is a Console Application it will be unwieldy to print out all of the possible Groups that a user can join and provide a way for the student to select them.
   2. The growth of the number of possible groups available is exponential to the number of Courses that can be added to the system, as each Course has on average seven groups, therefore making it more improbable to print out every Course Group a student can join.
   3. The user is assume to already know the Course Group Index Code of their choice and done prior research before the application date and therefore manual insertion should not prove to be an issue..

For Administrators

1. Administrators Accounts are added into the system by another party affiliated with NTU
2. All Administrators have documents to let them know about what to put in for entry date, Courses and Student to input.
3. Initially the System would be loaded in with all the Courses and Students inside, any changes or errors that have been made would be made or corrected through the system manual update prior to the initial upload
4. Since concurrent login has not been considered, it has been taken into account of what will happen if an Administrator is attempting Add, Delete or change any courses during the access period where the user will be using the system.

## Model, View and Control (MVC)

The system design uses the MVC Framework in order to implement Object Oriented Programming principal of SRP. How this is achieved is that it establishes the domain of interest that each class works on and how are they supposed to work with each other. In the MVC framework, the components of a program can be categorised as Model, View and Controller based on their function in respect to the system’s purpose.

The Model represents the real-life entities that are part of the process of Course Registration. They hold attributes that describe these entities and provide external interfaces to allow processing of information.

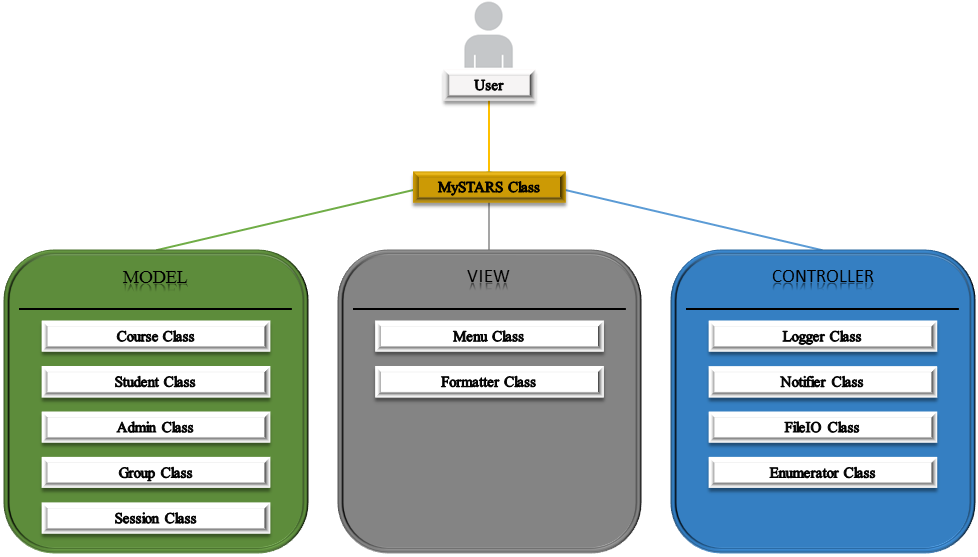
The View formats the data before displaying it to the user such as controlling the form and content of the User Interface (UI) displayed onto the console.

The Controller processes any of the user’s commands and translate it down to working on information or outputting the proper response to the user.

This segregation of duties can be taken further by making classes take more specific roles in those areas. For example the Menu class and Formatter class that fall under View both format information into away that it can be displayed in a meaningful manner to the user but does so in two different areas. The Menu Class focuses on forming UIs for the user to input commands into the system while the Formatter decides how data should be arranged on screen so as to give the user meaningful output.

In this way SRP is easier to establish and maintain. Classes can be categorised as a Model, View or Controller and have their scope narrowed down further into a single purpose. It also provide a mental map that can be used during development to check if a class that we wish to insert is either not well established or we placing too heavy a responsibility on it. If we are unable to determine which category a class should be placed into, likely the above two issues have occurred.

Abstraction of details becomes easier to be applied as the MVC clearly defines what interfaces need to be given to the other classes in order for them to use their services. The Model only has to provide interfaces for the other classes to manipulate data that they store. The View only has to provide an interface for information to be passed into them and returned into their formatted form. Lastly the Controller only has to provides interfaces to be invoked so that the User’s requests can be processed



The diagram above illustrates the architecture of our system in context to the MVC architecture. From the diagram we can see that all User interaction with the system is performed at the MySTARS class. The purpose of the MySTARS class is to translate user input into actions by invoking the services of the other classes as they are needed. Since the inner working of the system are never exposed to the user, changes to the system are transparent to them.

## Student and Administrator relationship with Course, Group and Session

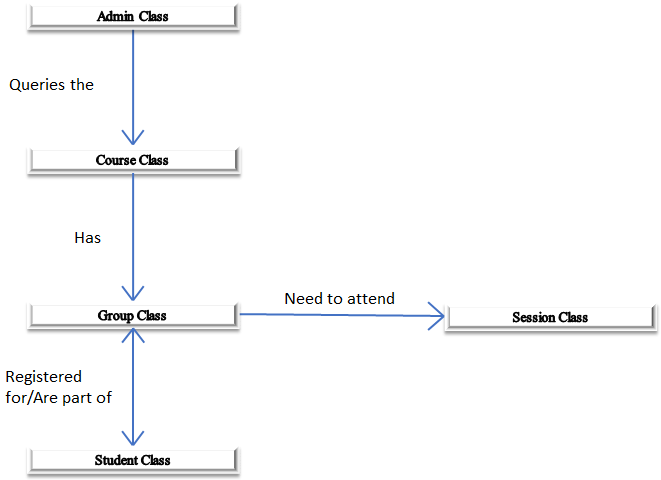
In the system, there are two groups of users who need to access the system and fulfil differing obligations. These two groups are Students and Administrators, the classes that represent them both require different set of attributes to describe them which differ so much that they very little common ground to establish. Therefore we do not consider it appropriate to make these two classes inherit from a common class as doing so would simply slow down the system and provide little benefit in return.

For Students their main purpose is to register for the Courses of their choice. By doing so they would also establish their time table based on the lessons they are required to take and also the number of AUs they will be taking the Semester. All of this information is held in both the Course Class and the Session Class.

What provides the link between Student and the Course Class and Session Class is an intermediary class known as Group. The Group class keep track of the number of students registered to the course and those who are in the waitlist, waiting for a free slot to be inserted into. It also establishes the differing time slots that a student can register for in order for them to register that Course in respect to their time table.

Therefore when a Student wishes to register for a particular course they would need to enter the appropriate Group Number of the Course they want to register for. This would then be used by the system to determine if there is a slot for the user to be inserted into, or if they need to be placed into the waiting list. The relationship with Course and Session would then enable the system to check if the user is already registered to the Course or if there is a time table clash.

Administrator would also rely on the same relationship in order to determine the number of users in each Course. This can be taken further by giving the Administrator the ability to print out the students based on what Course they are taking or by their Index number. The system will simply manipulate the relationship and draw out the necessary information to process the request.



From what has been stated and the diagram above, we can see that the relationship of Course, Group and Session remains the same regardless of whether or not Student or Admin is using the system. Therefore the different ways the classes can be used to serve both type of users are dependent on only the usage of their interfaces.

This shows that by establishing this relationship of classes we have achieved Dependence Injection Principal (DIP). The classes do not depend on any rigid implementation of code but instead utilized flexible interfaces that can be used to fulfil to any manner of tasks as is seen fit. This is allows for a highly decoupled system tzhat can be rearrange with minimal changes to solve any issue as well as allowing changes to implementation to be made with disrupting of the system processes.

# Detailed UML Class Diagram.

* Further Notes, if needed

# Detailed UML Sequence Diagram of stated function.

* Further Notes, if needed

# Testing.

* Test Cases and Results