

**NANYANG
TECHNOLOGICAL
UNIVERSITY**





SINGAPORE

Declaration of Original Work for SC/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 (SC2002)	Lab Group	Signature
Marcus Chen Zirui	SC2002	SCEX	
Muhammad Ayub Bin Ibrahim	SC2002	SCEX	
Muhammad Iqshan Bin Mohd Sa'ad	SC2002	SCEX	
Puah Rong Qi	SC2002	SCEX	

Design Considerations

Approach Taken

Looking at the entire assignment, the task can be quite daunting as there is a lot of information to decipher. Therefore, we start by understanding and analyzing the requirements that are already provided in the assignment. After that, we proceed to identify the key functionalities and user roles (**Staff, Student, Camp Committee Member**). We also discuss with one another as well as with the Course Coordinator to clarify any doubts or ambiguities that we may have upon reading the assignment before officially starting on it.

We then started to define the data models for the User, Camp, Staff, and Student as well as the entities that the user roles will use such as Enquiry and Suggestion. After that, we'll specify the relationships that they have with one another, whether it be an association, aggregation, composition or another type of relationship as well as the different key functionalities that each user role may have due to the different privileges that these classes may have such as Student and Camp Committee Member. An example of such would be a Student who is able to withdraw from a camp that they've registered themselves in but a Camp Committee Member could not.

Principles Used

Throughout working on our assignment, we adopt the SOLID Design Principles,

- Single Responsibility Principle (SRP)

This helps us ensure that each class or module that we have should only have a clear and single responsibility. For example, separate classes for camp management, report generation, enquiry, suggestion and more. This also ensures that the classes can only be changed for a single reason and their responsibilities will not be coupled as well, thus eliminating the possible ripple effect. This simplifies the other processes of running tests, modifying and reusing a portion of the code somewhere else.

- Open-Closed Principle (OCP)

This principle states that classes should be able to be open for extension but closed for modification, allowing for the addition of new functionalities through the use of

extension instead of modification. We can achieve this by using Object Oriented Programming (OOP) methods through inheritance, polymorphism and abstraction.

An example of applying OCP in our project is through the use of our database. We created an abstract class, “Database” that is extended to create other different types of databases such as “StudentDatabase”, “StaffDatabase”, “CampDatabase”, “EnquiryDatabase” and “SuggestionDatabase”. Each of these subclasses overrides the “getFilePath()” method which is an extension of the abstract class itself. This is because each of these subclasses has different files and in order to reach all of them, all of these subclasses can modify the “getFilePath()” method to retrieve their own file respectively without modifying the existing code. This also means that if we want to add a different kind of database, we simply can just extend from the abstract class itself and won’t have to modify the existing code.

- Liskov Substitution Principle (LSP)

This principle states that the objects of a superclass shall be replaceable with objects of its subclasses without breaking the application. It also states that functions that use pointers to base classes must be able to use objects of derived classes without knowing it. We can achieve this by ensuring the objects of the subclasses behave in the same way as the objects of the superclass.

An example of LSP that we have applied is the implementation of the User superclass by Student and Staff subclass. In any future addition to the User object, all the methods in the User object are applicable to all types of users beyond Student and Staff. With this, the developer would not need to modify the code to accommodate an instance of, say, an Alumni user type.

- Interface Segregation Principle (ISP)

This principle states that many specific interfaces are better than one general-purpose interface. This is so that we can avoid designing a fat interface. By adopting this approach, implementing classes only needs to be concerned about the methods that are

relevant to them and reduce the ripple effect when modifying our system. With that in mind, we concluded that the ‘Model’ interface must be split down into ‘User’, ‘Camp’, ‘Suggestion’ and ‘Enquiry’. This allows the different entities to implement the interfaces that suit their specific requirements.

- **Dependency Injection Principle (DIP)**

This principle states that high-level modules must not depend on lower-level modules and both should rely on abstraction. This means that we can depend on interfaces rather than concrete classes as interfaces have the lowest chance to undergo modifications once the system is developed. We’ve incorporated this into our design. For instance, when obtaining the ID of a staff member, instead of relying on the specific ‘Staff’ concrete class, we depend on the ‘User’ interface. This approach allows us to also add other user types such as ‘Student’ with minimal effort in the future by simply extending from the ‘User’ interface itself.

Use of OO Concepts

Some of the OO concepts that we use for the classes.

Encapsulation for the attributes for classes. Classes such as suggestion, have their attributes declared as private. Access to these attributes will be accessed via public methods such as setters and getters. This is to allow reuse of data and attributes while preventing open access to the attributes and data.

Abstraction is embedded into our CAMs system. From the start of the application till the end and everything in between, the user does not need to know the private instance variables and methods. All options given to users are presented in an orderly and hierarchical fashion such as Login → Main page → Specific page (Register for camp, Submit enquiry, etc). The complexities of error-checking and validation are hidden from the user at every decision point, enabling a simple and easy-to-use application.

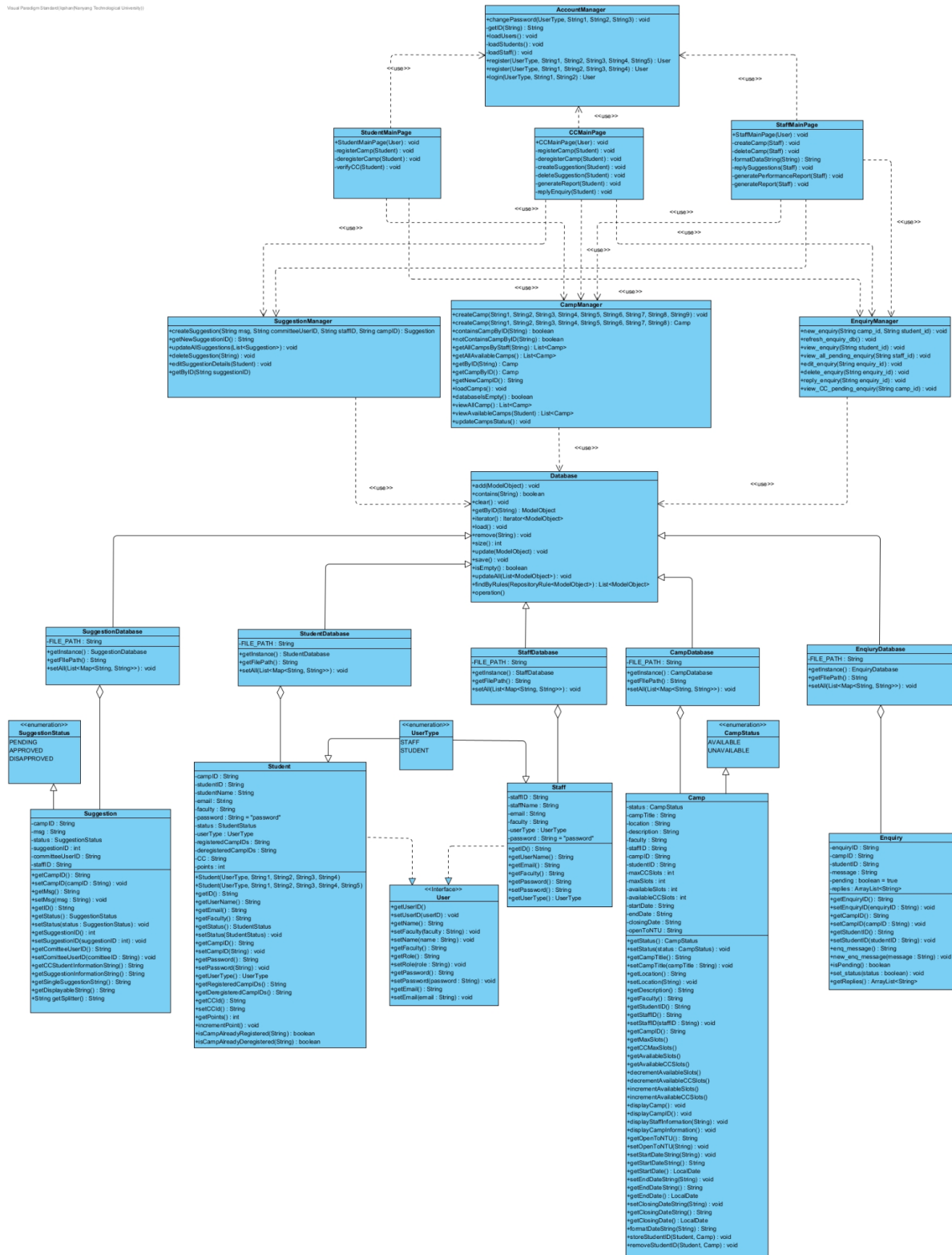
Inheritance is applied on interfaces. The interface, model, inherits abstract methods from the parent interface mappable which allows the conversion of an object to map and vice versa while adding its own abstract method toString. This allows us to build on interfaces or classes without recreating the original class or methods.

Polymorphism is used in classes that implement displayable. Two such classes such as suggestion and camp, both use getDisplayableString(); However, the output results of the classes are different as they depend on the attributes of their respective classes. It allows the code to become more flexible and maintainable.

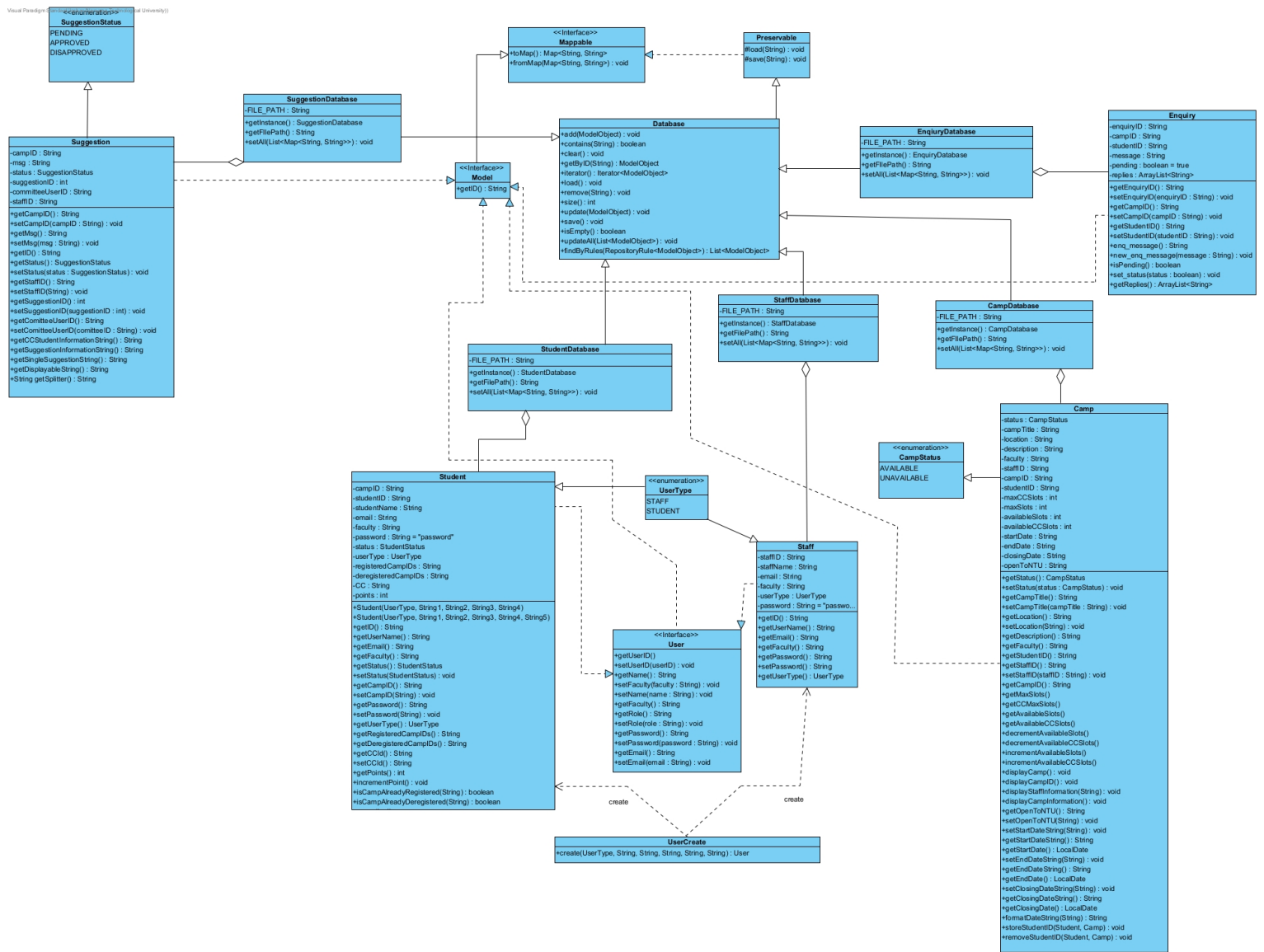
Assumptions Made

- We assume that users know their user ID. (the field before the domain of their email)
- We assume that registration of camp and camp committee is automatic as long as there is a vacancy.
- We assume that the number of camp committees is not counted in the total slots.
- We assume that the students can withdraw from camps without requesting permission from the staff in charge.
- We assume that the password will not be hashed for easier testing purposes
- We assume that the camp's end date extends until 2359 hours, encompassing the whole day. This means that if the end date of a camp is the same as the start date of another camp, they will not be able to register for that camp.

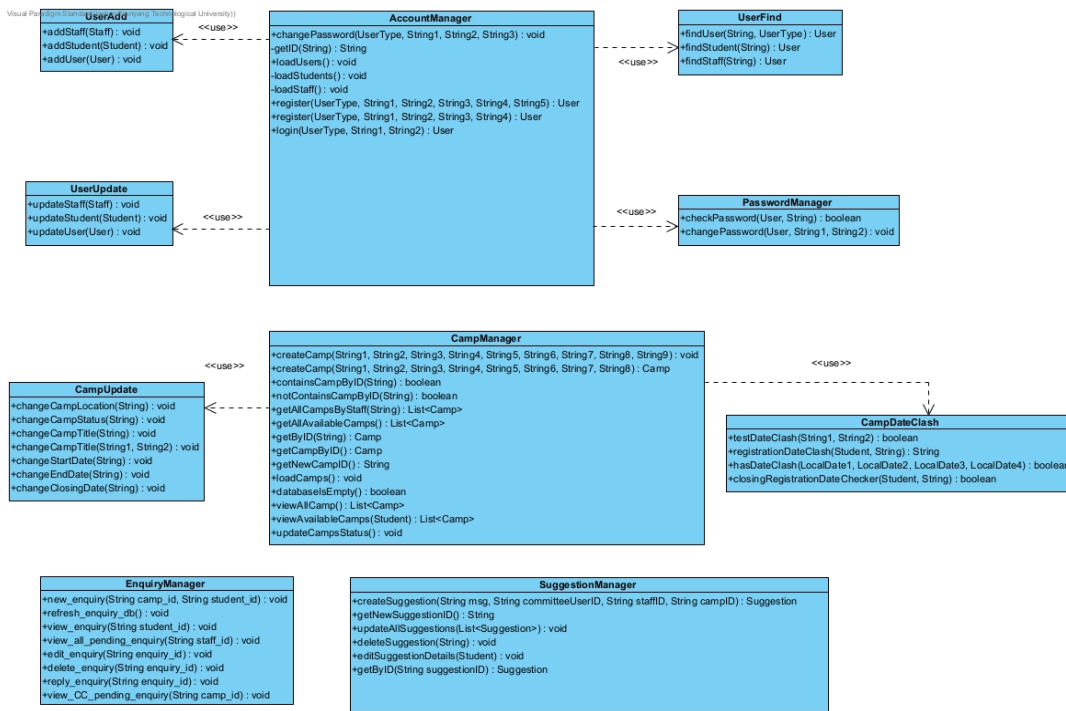
Main Diagram



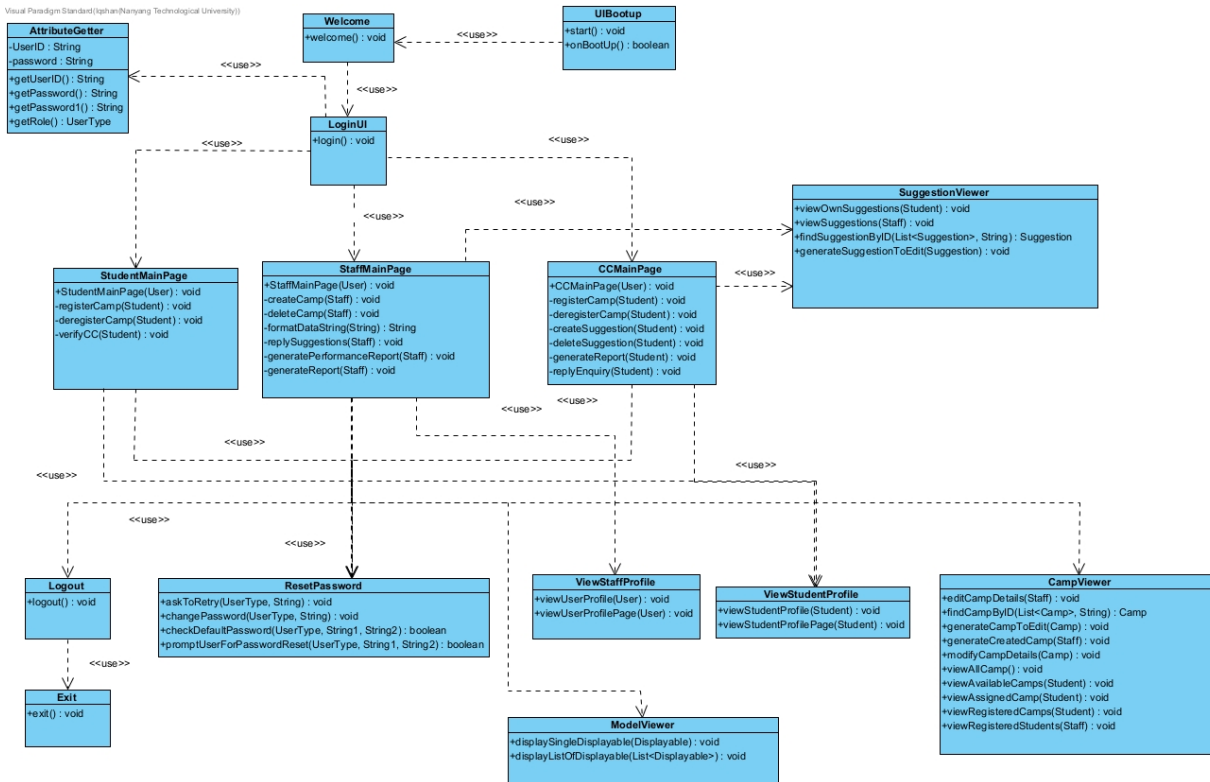
Entity Sub Diagram



Controller Sub Diagram



Boundary Sub Diagram



Testing

Login Page

<pre>Welcome to the Camp Application and Management System (CAMSs) Please enter your choice to continue. 1. Login 2. Exit Your choice (1-2): </pre>	<pre>1. Student 2. Staff Please select your domain (1-2): 1 Please enter your UserID: YCHERN Please enter your password: password</pre>
Upon entering the system, they can pick either to login into the Student/Staff domain	
<pre>Welcome to Student Main Page Hello, CHERN! 1. View my profile 2. Change my password 3. View camps 4. View registered camps 5. Register for a camp 6. Withdraw from a camp 7. Create enquiry 8. View enquiry 9. Edit enquiry 10. Delete Enquiry 11. Switch to CC Main Page 12. Logout Please enter your choice: </pre>	
Here is an example of a Student's Main Page	

Prompt Reset Password

<pre>Welcome to Student Main Page Hello, CHERN! Your password is still the default password. Would you like to change it? [y]/[n] : y</pre>	<pre>Please enter your old password: password Please enter a new password: skyblue1308 Please enter a new password again: skyblue1308 Password changed successfully. Press [Enter] to go back to the main page. </pre>
<pre>1. Student 2. Staff Please select your domain (1-2): 1 Please enter your UserID: ychern Please enter your password: skyblue1308 Login Successful Press enter to continue. </pre>	
This is a successful attempt to reset a password if the user enters [y]	

```

Please enter your old password: password
Please enter a new password: pass
Please enter a new password again: pass
Password must be more than 8 characters.
Press Enter to try again or [b] to go back.

```

```

Please enter your old password: skyblue1308
Please enter a new password: pass
Please enter a new password again: asdad
The two passwords are not the same.
Press Enter to try again or [b] to go back.

```

New password must follow the requirements as above in order for it to be valid

Staff create camp

```

===== FOC SCSE =====
Camp ID          | P6
Staff Name       | MADHUKUMAR
Staff Email Address | HUKUMAR@NTU.EDU.SG
Staff Faculty    | SCSE
Location         | North Spine
Available Slots  | 20
Available CC Slots | 10
Camp Status      | AVAILABLE
Start Date       | 2023-12-01
End Date         | 2023-12-03
Closing Registration Date | 2023-11-25
=====
Camp Description
=====
For Freshies
=====
Are you sure you want to create this camp? (Y/N)
y

```

```

===== FOC SCSE =====
Camp ID          | P6
Staff Name       | MADHUKUMAR
Staff Email Address | HUKUMAR@NTU.EDU.SG
Staff Faculty    | SCSE
Location         | North Spine
Available Slots  | 20
Available CC Slots | 10
Camp Status      | AVAILABLE
Start Date       | 2023-12-01
End Date         | 2023-12-03
Closing Registration Date | 2023-11-25
=====
Camp Description
=====
For Freshies
=====
Enter <Enter> to continue

```

The newly created camp will be displayed here

Staff toggle visibility of camp to students

```

=====
Camp ID          | P6
Staff Name       | MADHUKUMAR
Staff Email Address | HUKUMAR@NTU.EDU.SG
Staff Faculty    | SCSE
Location         | North Spine
Available Slots  | 20
Available CC Slots | 10
Camp Status      | AVAILABLE
Start Date       | 2023-12-01
End Date         | 2023-12-03
Closing Registration Date | 2023-11-25
=====

```

Camp Description

For Freshies

Editing camp with CampID: P6

1. Edit Camp Title
2. Change Camp Status
3. Edit location
4. Edit Start Date
5. Edit End Date
6. Edit Closing Registration Date

Please enter your choice:

1. AVAILABLE
2. UNAVAILABLE
Current Camp Status: AVAILABLE
Please enter your choice for the new status of your camp:
2
Camp details edited successfully.
Press enter to go back.

FOC SCSE

Camp ID	P6
Staff Name	MADHUKUMAR
Staff Email Address	HUKUMAR@NTU.EDU.SG
Staff Faculty	SCSE
Location	North Spine
Available Slots	20
Available CC Slots	10
Camp Status	UNAVAILABLE
Start Date	2023-12-01
End Date	2023-12-03
Closing Registration Date	2023-11-25

Camp Description
For Freshies
Press Enter to go back.

The newly created camp is now unavailable

Students viewing camps only available to them (Option 3 of the Student Main Page)

Staff Email Address	HUKUMAR@NTU.EDU.SG
Staff Faculty	SCSE
Location	LT1
Available Slots	20
Available CC Slots	10
Camp Status	AVAILABLE
Start Date	2023-11-26
End Date	2023-11-28
Closing Registration Date	2023-11-22

Camp Description

Explore the cutting-edge world of Artificial Intelligence in this retreat-style camp.

VRVerse Voyage

Camp ID	P4
Staff Name	DATTA
Staff Email Address	ANMIT@NTU.EDU.SG
Staff Faculty	SCSE
Location	TCT-LT
Available Slots	20
Available CC Slots	10
Camp Status	AVAILABLE
Start Date	2023-11-19
End Date	2023-11-22
Closing Registration Date	2023-11-15

Camp Description

Embark on a virtual reality adventure at VRVerse Voyage
Press Enter to go back.

Since P6 has been toggled to unavailable, the student is unable to see that camp in their list

Camp Registration

<pre>===== Join the FutureFinance Festival to unravel the mysteries of the financial world. ===== Please enter the Camp ID that you would like to register: p5</pre>	<pre>===== Join the FutureFinance Festival to unravel the mysteries of the financial world. ===== Are you sure you want to register for this Camp? ([y]/[n]) y The number of available slots now are : 19 Register Success! Press enter to go back.</pre>
--	---

When a student registers for a camp, it will update the number of available slots accordingly

Camp Deregistration

```
=====
You are a Camp Attendee of this camp
Press Enter to go back
Please enter the Camp ID that you would like to remove yourself from:
p5
The number of available slots now are : 20
Deregistration is successful. Please do note that you are unable to register yourself to this camp again.
Press enter to go back.
```

Registered camps will be displayed in their registered camp list that the student can view from.

The student is also unable to register themselves into a camp that they have previously deregistered

Reflection

Throughout this assignment, we have recognised the significance of adhering to the SOLID Design Principles as much as possible. During the assignment's early stages, we faced uncertainties regarding how to separate the implementation of the codes and laughed at the idea of splitting a single class into many multiple subclasses. This meant that any changes that we had, regardless of their scale, could potentially trigger a ripple effect. Any code related to the changes that we have made would be modified very easily. To address this challenge, we decided to adopt the SOLID Design Principles and incorporate them into our assignment. Thus, a software system that can achieve high cohesion and loose coupling possesses high flexibility, making it more maintainable and extendable.

We've also added a display of the camps and suggestions for the user depending on whether the user is a Student, Staff or a CC member. This is to prevent them going back and forth, making our system a user-friendly experience.