SC2002 Object Oriented Design & Programming

Camp Application and Management System (CAMs)

Report of Project Design Structure and Functionality

AY 23/24 Sem 1 | SCS3, Group 5

Github Main Page: https://github.com/ye-chuan/sc2002-project

Project Documentation:

Name	Matriculation Number
Belvedere Song Zheng Yi	U2222304L
Edmund Ser Wei En	U2221617C
Lee Ye Chuan	U2221917E
Ong Yi Ren, Elaine	U2222435D

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
Belvedere Song Zheng Yi	SC2002	SCS3	
Edmund Ser Wei En	SC2002	SCS3	
Lee Ye Chuan	SC2002	SCS3	
Ong Yi Ren, Elaine	SC2002	SCS3	

1 Design Approach

1.0 Entity Controller Boundary Model (ECB)

ECB is a software architectural pattern that divides a system into three main components - entities, boundaries and controllers. ECB is implemented for this project to further promote the separation of classes and interfaces which increases modularity, making it easy to modify its behaviour. This enables the project to be reusable, extensible and maintainable in the future. In this project, Entity classes are classes related to databases, Boundary classes are classes related to the user interface and Controller classes are bridges between the Entity and Boundary classes.

During the design phase of our project, we aim for the three areas of the ECB to be minimally dependent on each other in terms of the code structure and internal functions.

1.1 Design Consideration: Loose Coupling and High Cohesion

Loose coupling means to reduce the dependencies between components. This makes it easier to modify or replace one component without affecting others. Changes in one part of the system are less likely to ripple into the rest of the entire codebase, improving the system's flexibility and maintainability. High cohesion enhances reusability as modules with high cohesion are often more well-defined towards a centralised goal, forcing us to rethink our design and split large components into smaller ones.

For the Boundary class, to create a loose coupling. We make use of an interface called IUserInterFace where the UIMain class will be calling the only method in IUserInterFace (showUI( )) with the return type IUserInterface. This allows for each UI created to be destroyed at the end of its own respective “page” before the next UI gets created. This adheres to the idea of loose coupling as a UI page's existence will not be tied to another UI page. Furthermore, each UI page is free to call any UI page.

Next up, to implement high cohesion design in our UI implementation. We decided to use an Association class for the interaction between the UIMain and the classes that implement the IUserInterface. The association class is called UIInformation. This class contains information such as userID, campID, enquiryID, suggestionID. All this information is then used in the UI

classes to be passed to the controller classes to obtain the necessary details. As such, the boundary classes can work hand in hand while being independent from one another.

The many-to-many association between Students and Camps is hard to model without increasing the coupling between them. The naive approach is to have each Student contain a list of all their registered Camps and each Camps containing a list of all their registered Students. However, this goes against the Single Source of Truth architecture as it increases the risk of desynchronisation between the 2 lists.

Hence, we decided against this design and created a CampMembership class which represents the association between a Student and a Camp. We then have a separate CampMembershipDatabase to store the list of all such associations.

Furthermore, this process is in line with database design principles such as database normalisation - by separating the many-to-many associations to 2 one-to-many associations, we have satisfied the first normal form.

1.2 Solid Design Principle

1.2.1 Single Responsibility Principle (SRP)

SRP suggests that each class should only have one clear responsibility so as to avoid needing to edit a lot of classes when there is a slight change in the project requirements.

For the Boundary class, to consider SRP in our design, we first identify the general responsibility of a Boundary. We have identified that Boundary classes have the following responsibilities: being able to print menu, being able to select from the menu, being able to print details on the screen and lastly controlling what the UI will show with each user's choice. After finalising these 4 main responsibilities. We created one class for each respective responsibility of each possible UI we needed.

Our design choice of creating a separate CampMembershipDatabase to store the membership information of Students and Camps as mentioned earlier also adheres to SRP. It relieves the

CampDatabase of the responsibility of registering students, so it can focus just on storing the Camps. Similarly, the responsibility of the Student class is to keep information on the Students, regardless of the presence of school camps. The single responsibility of this new CampMembershipDatabase class is to store and manage the registrations of Students to the various Camps.

For our controller classes, each controller class such as CampController and SuggestionController has its distinct responsibility. A separate class like CampListController, and SuggestionListController is created which handles the sorting and filtering of a list of camp objects separately.

1.2.2 Open-Closed Principle (OCP)

OCP states that classes should be open for extension but closed for modification, this is to allow for future addition of new functionality without having to modify the existing code which may cause a domino effect, affecting classes that rely on this existing code and classes that rely on affected classes.

Next to ensure that our Boundary class obeys OCP. Our main UI runs on with only one function as mentioned above (showUI( ) from IUserInterface). This allows the main UI to be open for extensions, for example, a new possible UI can still run on this UIMain class even if it has a completely different role and responsibility through implementing IUserInterface.

The filtering system is made to be modular and loosely coupled, conforming to OCP. The CampFilterer class takes in a collection of Camps, CampFilters can be added or removed from the CampFilterer object, and the Filterer will then output a filtered list of Camps from the given Filters. The Fitlerer base abstract class and the CampFilterer specialised class are both built to be closed for modification, but open for extensions. Extending the function of a Filterer requires creating more Filter classes that inherit from Filter, but no modification to the Filterer class is required to use it. All Filters will implement a .pass(object) method that determines whether a specific object can pass through this filter. This allows us to create any arbitrary filter types we might need in the future, and filter any attribute of a Camp in any way it sees fit.

1.2.3 Liskov Substitution Principle (LSP)

LSP suggests that subtypes must be substitutable for base types without breaking or changing the behaviour of the program.

We have many different types of users such as Staff, Student and Camp Committee with each having its differences and similarities. The hierarchy for most of the UI is as follows: Student -> Camp Committee -> Staff, with staff having the most privilege in terms of their view. With that, since most UI have similar general responsibilities as mentioned above, we can have student UI class be the root and camp committee UI class to inherit student UI and Staff to inherit camp committee UI class. This obeys the LSP since all our sub-classes will inherit the methods from their superclass and do a bigger responsibility.

1.2.4 Interface Segregation Principle (ISP)

ISP is similar to SRP, it is essentially SRP for interfaces. ISP states that a program should have many specific interfaces rather than a fat interface (one general interface with many abstract methods). This is to ensure that classes are not forced to implement methods they do not use when implementing an interface.

As mentioned above, our Boundary class has 4 general responsibilities. We apply ISP by creating an interface for each of the 4 responsibilities. This allows even more modification for our UI class allowing more OCP since now UIs will be able to run any methods of those classes that override the interface method without changing the code.

1.2.5 Dependency Inversion Principle (DIP)

DIP states that high-level modules should not depend upon low-level modules. Both should depend upon abstraction. Abstraction should not depend upon details. Details should depend upon abstraction.

To adhere to DIP, our UI classes will only have attributes of the interface of IUserInterface. This allows for loose coupling and high cohesion to occur since this allows our program to further be

improved and extended in the future by creating new classes that implement the 4 interfaces and having the UIs to run those programs.

The Database class represents a database system where each entry has a unique Primary Key attached to it. Data retrieval from a Database is done using the Primary Key via the .getItem(String primaryKey) method.

The Primary Key generator is implemented using a very simple LongIncrementalIDGenerator, which simply increments a Long every time a new ID is requested. In view of the Dependency Inversion Principle, users of this class access this implementation through the IUniqueIDGenerator interface with the .generateUID() abstract method. This allows us to easily change the implementation of how we generate unique IDs in the future.

1.3 Reflection (Challenges and future improvements/modifications)

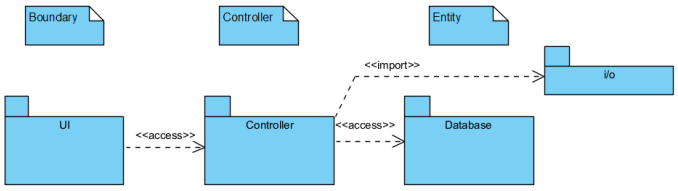
The most painful experience in this project would be during the combination of codes. We underestimated the difficulty that will be faced when integrating our code together even with rather well-defined interfaces. There needs to be a huge overhaul to our workflow as a group in the future, where constant integration and testing should be done as the project progresses.

We learnt that even though interfaces provide a way to split a huge project up into individual parts that can be easily delegated, constant merging and frequent testing of the project is still paramount to producing a high-quality program.

2 Detailed UML Class Diagram

Full UML diagram can be accessed on:

Class Diagram (based on Entity, Boundary and Class model)

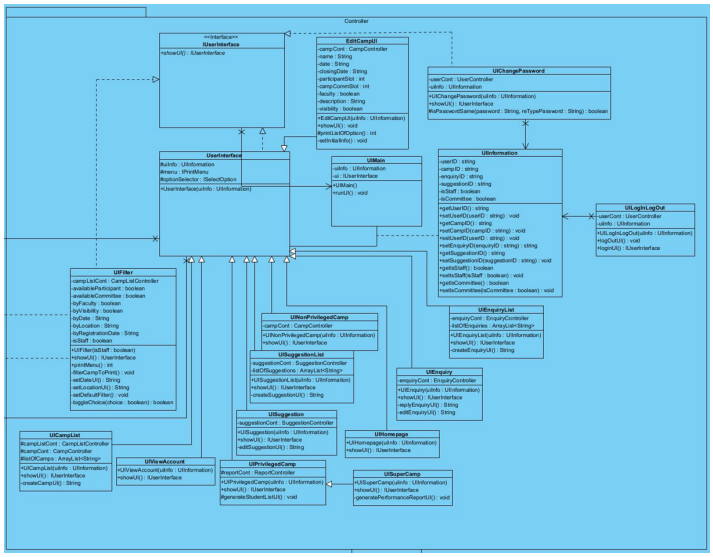


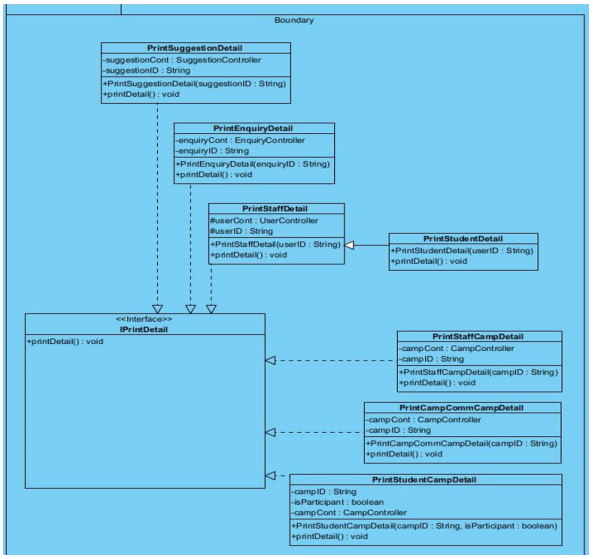
Program Core

Main Boundary: UI package

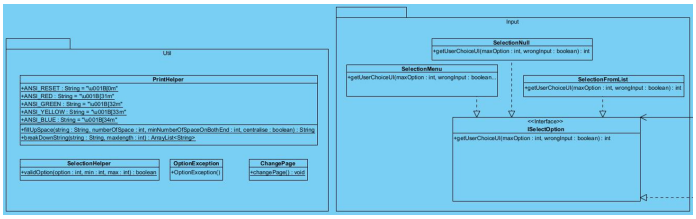
The UI package will access the program's main controller's package. Within the UI package itself, classes are also segregated into smaller packages using ECB. The following are packages and classes within the main UI package.

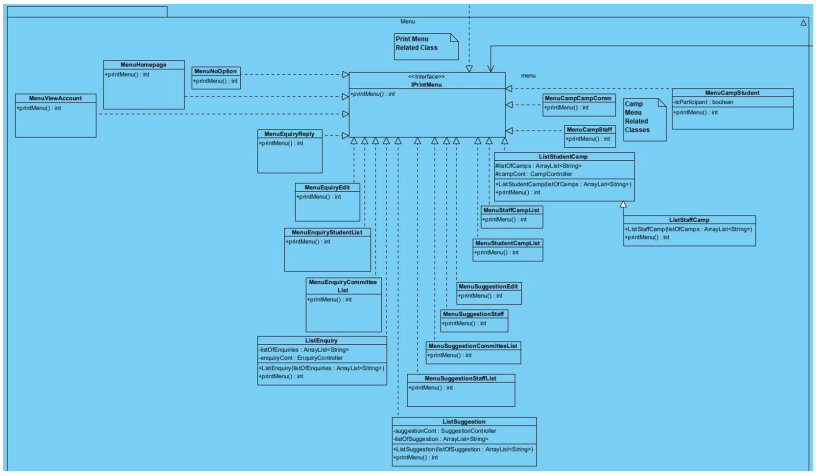
Boundary and controller sub-packages within the UI package are responsible for communication between the UI and the program's main controller (View below).



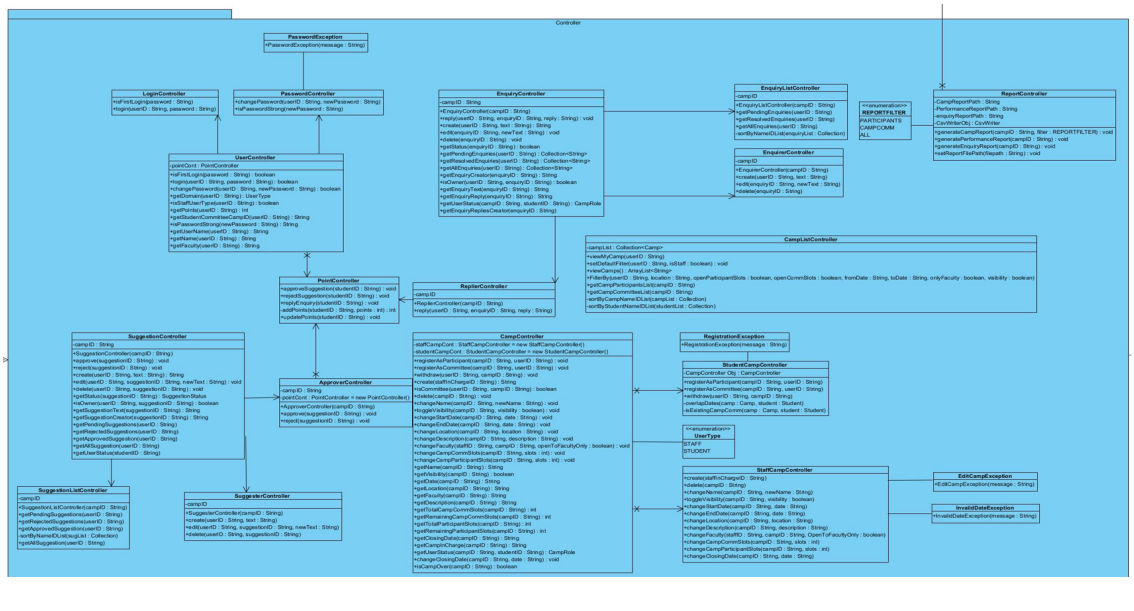


Other Packages within UI package

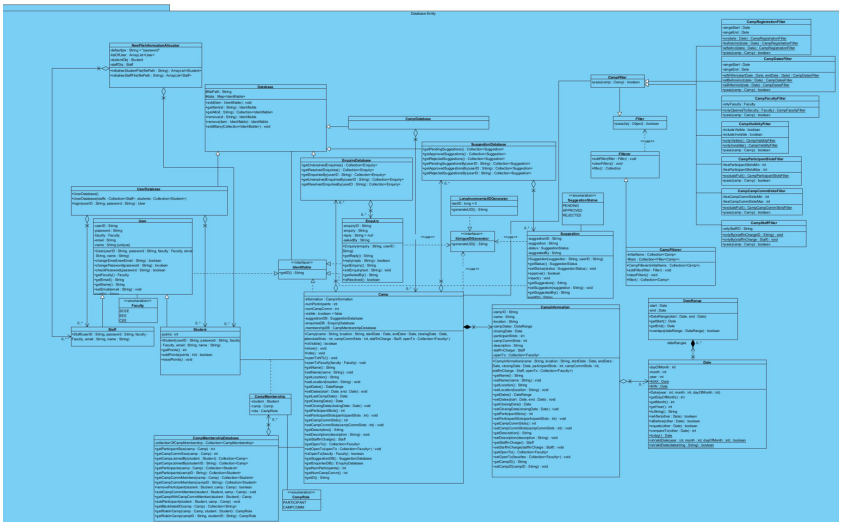




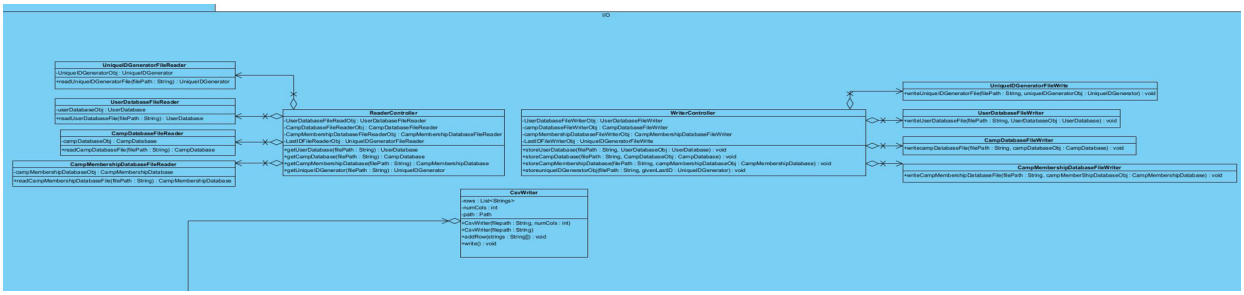
Main Controller: Controller package



Main Entity: Database



I/O (Additional package)



Testing

Upon running our program. The login page is as shown below, users will be prompted to enter their userID and password. If the password is default, they will be prompted to change their password.

```
  .g8""bgd     db       `7MMM.     ,MMF' .M""bgd
.dP'     `M    ;MM:        MMMb    dPMM  ,MI    "Y
dM'           ,V^MM.       M YM   ,M MM  `MMb.
MM            ,M  `MM      M  Mb  M' MM    `YMMNq.
MM.           AbmmmqMA     M  YM.P'  MM  .     `MM
`Mb.     ,'  A'     VML    M  `YM'   MM  Mb     dM
  `"bmmmd'.AMA.   .AMMA..JML. `'   .JMML.P"Ybmmd"


Welcome to Camp Application and Management System (CAMs)
UserID: ARVI
Password: password
Login successful...

Password is default password
Please change to a strong password

Change of Password Menu
Enter new password:
```

```
Change of Password Menu
Enter new password: password
Password must contain an upper character!
Password is not strong
Enter new password: Password
Password must contain a special character
Password is not strong
Enter new password: Password@
Password must contain a digit
Password is not strong
Enter new password: Password@123
Re-type new password: Password@123
```

This is an example of how the Account Detail of a staff and student will look.

```
ACCOUNT DETAIL

| Name    | ARVIND          |
| Faculty | NBS             |
| Email   | ARVI@NTU.EDU.SG |

(press any key to go back)
```

```
ACCOUNT DETAIL

| Name               | CHAN                          |
| Faculty            | NBS                           |
| Email              | YCN019@e.ntu.edu.sg           |
| Camp Committee of  | NANYANG SCHOOL OF BUSINESS CAMP FRESHM... |
| Number of Points   | 3                             |

(press any key to go back)
```

When we view camps, Staff and Student menu is as follows

```
CAMP LIST MENU
        (1) View All Camps
        (2) View My Camps
        (3) Create a new Camp
        (4) Go to Home Page
        (0) Exit Program
-----------------------------------------------------
Select option:
```

```
CAMP LIST MENU
        (1) View All Camps
        (2) View My Camps
        (3) Go to Home Page
        (0) Exit Program
-----------------------------------------------------
Select option:
```

The Staff List of camps is as follow:

```
LIST OF CAMPS
| (1) | NANYANG SCHOOL OF BUSINESS CAMP FRESHMAN ORIENTATION 2023
                                                  05/12/2023-20/12/2023
        PARTICIPANT SLOT: 0/20
        CAMP COMMITTEE SLOT: 0/9
        VISIBILITY: ON
| (2) | WELFARE SERVICE CLUB WSC ANNUAL ORIENTATION CAMP
                                                  10/12/2023-20/12/2023
        PARTICIPANT SLOT: 0/5
        CAMP COMMITTEE SLOT: 0/1
        VISIBILITY: ON
-----------------------------------------------
(press any non-numeric key to go to Camp List Menu)
(press 0 to go to filter)
Select option:
```

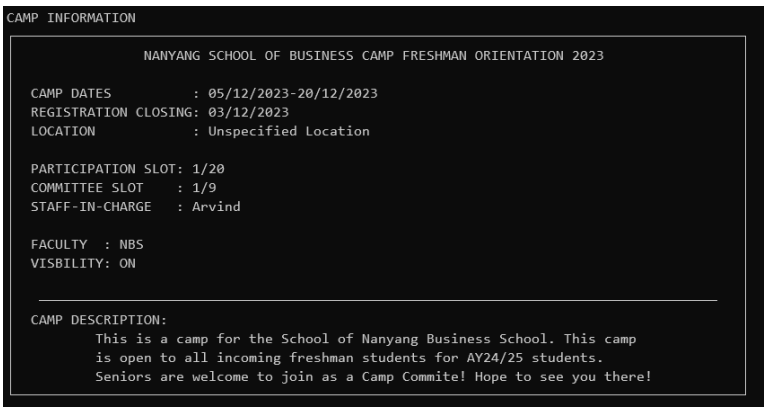
The student list is camp is as follows

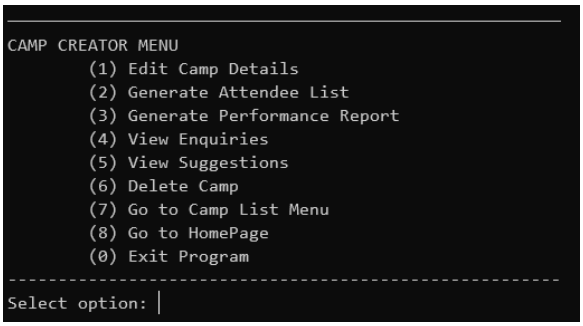
```
List Of Camps
| (1) | NANYANG SCHOOL OF BUSINESS CAMP FRESHMAN ORIENTATION 2023
                                                  05/12/2023-20/12/2023
        PARTICIPANT SLOT: AVAILABLE
        CAMP COMMITTEE SLOT: AVAILABLE

-----------------------------------------------
(press any non-numeric key to go to Camp List Menu)
(press 0 to go to filter)
Select option:
```

The filter is as shown for filtering the list of camps:

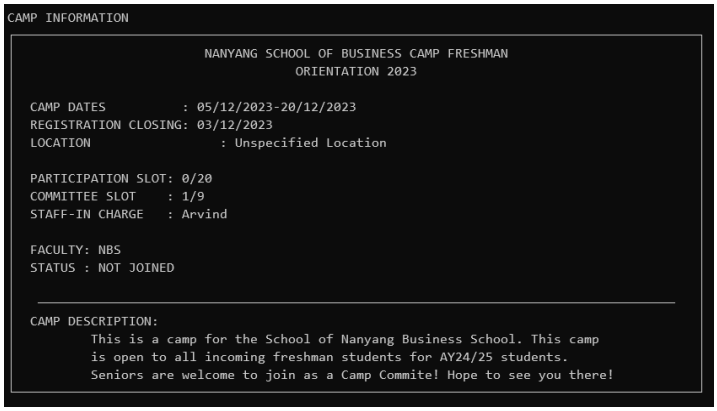
```
Filter Selection
        (1) By available participants slot: true
        (2) By available camp committee slot: true
        (3) By faculty: false
        (4) By Location:
        (5) By Date: 1/12/23-30/12/23
        (6) By visibility: true
        (7) Reset Filter
        (0) Confirm filter selection
-----------------------------------------------
(press any non-numeric key to go back to Camp List)
Select option: 0
```

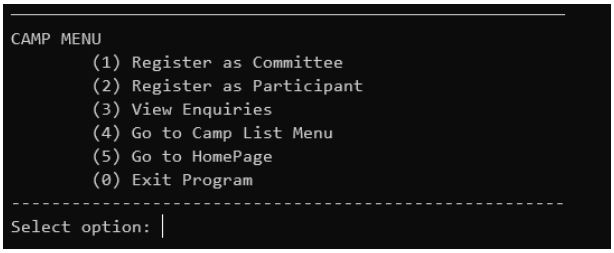
When users select a camp, they will get different camp view depending on their status in the camp:
Staff:



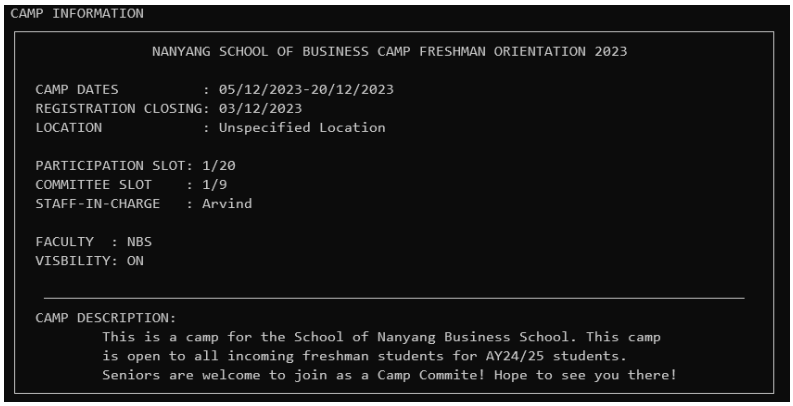


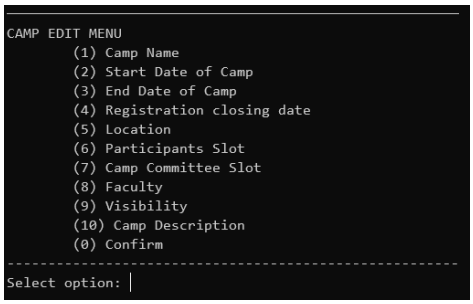
Camp Committee / Student / Staff that is not IC: The status will be PARTICIPANT if the student is a joined while status will be CAMP COMMITTEE // Staff cant join





For staff, they are able to create and edit camp, both interface is the same but with input being UNSET / UNSPECIFIED / 0 where staff can force to set all unset information before they can create a camp





Enquiries List and the interface when they choose

```
LIST OF ENQUIRIES
+-----+-----------------------------------------------------------------------+
| (1) | Is the camp free ?                                                    |
|     |                                                                REPLIED |
| (2) | I am currently recovering from a sickness and I cannot really participate in phy... |
|     |                                                                PENDING |
+-----+-----------------------------------------------------------------------+
(press any non-numeric key to go to Enquiry List Menu)
Select option:
```

```
                                ENQUIRY INFORMATION

     Asked by: LIU
     +-------------------------------------------------------------------+
     |    Is the camp free ?                                             |
     +-------------------------------------------------------------------+
     Replied by: CHAN
     +-------------------------------------------------------------------+
     |    Yes!! the camp is free!!!                                      |
     +-------------------------------------------------------------------+
(press any key to go back)
```

Suggestion List and the interface when they choose

```
LIST OF SUGGESTION
+-----+-----------------------------------------------------------------------+
| (1) | Can the camp be for be for as LONGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG as ... |
|     |                                                               APPROVED |
| (2) | And it will be better if we can have more guys than girls!! cause i like guys :D |
|     |                                                               REJECTED |
+-----+-----------------------------------------------------------------------+
(press any non-numeric key to go to Suggestion List Menu)
Select option:
```

```
                              SUGGESTION INFORMATION

     Asked by: YCN019
     +-------------------------------------------------------------------+
     |    And it will be better if we can have more guys than girls!! cause i |
     |   like guys :D                                                    |
     +-------------------------------------------------------------------+
     STATUS: REJECTED
(press any key to go back)
```