Introduction

Camp Application and Management System (CAMs) is an application designed to facilitate registering and viewing Camps within NTU, as well as camp management for the staff. In this report, we will show various design considerations and principles as well as object-oriented concepts which we have implemented during the development of this application. A Detailed UML Class diagram and a few important test cases are included to illustrate the dependencies and functionalities of the application.

Application Overview

The main features of CAMs involve verifying users, creating camps, registering camps, providing detailed reports, and facilitating camp inquiries. Our app primarily targets NTU students and staff, classifying students as participants or Camp Committee Members.

- *For Students:* Easily access, register for, or inquire about available camp details.
- *For Camp Committee Members:* Enjoy student functionalities with added privileges to access comprehensive camp details they oversee, ensuring a tailored experience.
- *Staff:* Possess full control over camp-related tasks, including creation, editing, deletion, and management of camp inquiries and suggestions. They can also control camp visibility and handle attendee lists.

CAMs prioritises user-friendliness, enabling intuitive navigation and straightforward interactions. Security is paramount, with measures like personalised user accounts and password protection in place

Design Considerations

Object-Oriented Concepts

1. Inheritance

Inheritance refers to the ability of one class to inherit the attributes and methods from another. It reduces code duplication as it allows two similar classes that require similar features but have different purposes. Both of these classes can be generalised and inherit the code from a superclass.

- For this project, both the staff and student objects have the same type of attributes (ID, name, password, faculty) and should have the same methods available to them such as logging in, logging out, and changing passwords. Hence, we created the User class that has basic attributes, which the Student and Staff classes inherit from.

2. Encapsulation

Encapsulation involves bundling data and methods that operate on the data into a single class. Access to the data is restricted to methods within the class, enhancing data security.

- Encapsulation was used in our CAMs project to help protect data from unwanted viewers. All attributes are made to be private and can only be accessed and modified through accessor methods by the appropriate entities. This allowed us to create instances and functions that are only usable by some classes but not others.

3. Polymorphism

Polymorphism refers to an object's ability to take on multiple forms, allowing us to execute a task differently depending on the parameters or where it is in the inheritance hierarchy. Method overriding, which comes with polymorphism, was useful in the design of our code. By having multiple implementations in one interface or class, the code is much easier to read and it increases reusability. Method overloading is also present throughout our source code as different methods are used depending if a staff or student object is passed to it.

SOLID Design Concepts

*1. **S**ingle Responsibility Principle (SRP)*

This principle ensures that classes focus on a single task, and should have only one reason to change. Having only one job or responsibility enhances the maintainability of the code.

- For example, in our source code, the PasswordFileHandler and PointsFileHandler classes are separate and given a single task each, to manage the passwords for each user and the keeping track of the points for committee members respectively.

*2. **O**pen/Closed Principle (OCP)*

The open/closed principle promotes designing code that allows new functionality to be added without altering the existing code. This extendibility can be achieved by making sure that modules are open for extension and closed for modification.

- An example that adhered to OCP in our project is the addition of interfaces such as MenuDisplayable which decouples the menu display logic from the User class. This would allow us to add new types of users such as camp counsellor if we wish to by

simply creating a new class that implements MenuDisplayable and extends from User without modifying the existing class.

*3. **L**iskov Substitution Principle (LSP)*

LSP states that a user of a base class should continue to function properly if a derivative of that base class is passed to it. In other words, subtypes must be substitutable for their base types without affecting CAMS's correctness.

- An example of this that can be seen in our code is the Student and CampCommitteeMember class, where the derived CampComitteeMember class can be a substitute for the base Student Class but it additionally enhances the Student class without causing more trouble for it.

*4. **I**nterface Segregation Principle (ISP)*

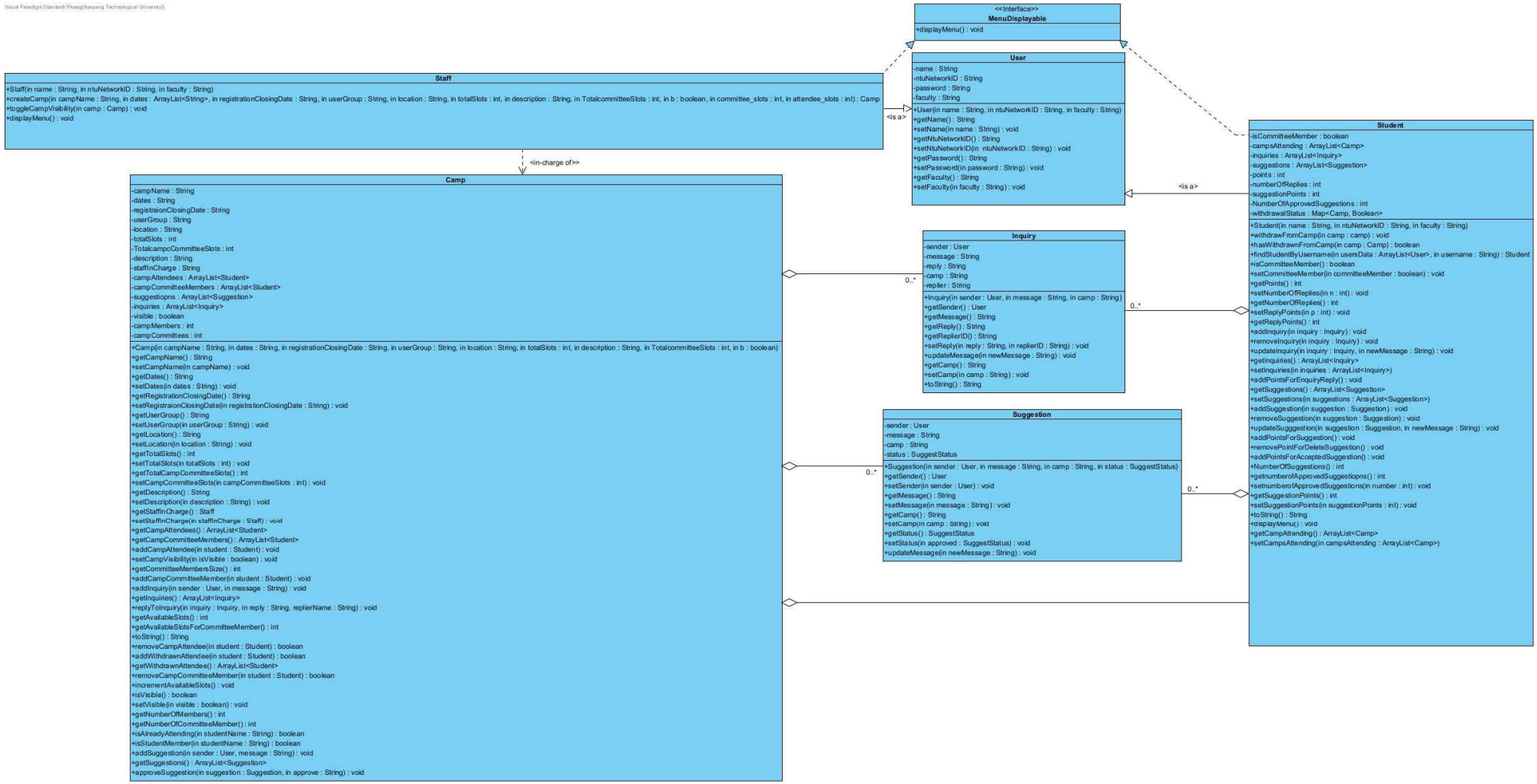
It is generally better to have many client specific interfaces rather than one general purpose interface. However, our group did not think custom interfaces were necessary for the execution of our project. Splitting large interfaces into smaller ones for hyper specific use cases is a good practice in general but for a small scale project like the CAMs, we thought it would over complicate our design.

*5. **D**ependency Injection Principle (DIP)*

High level modules should not depend upon low level modules, both should depend on abstractions. This way, there is a loose coupling between the high level modules and the low level modules, which enables reusability of high level modules.Basically we pass in instances of "lower level" classes into "higher level" classes instead of creating instance of these "lower level" classes in the "higher level" class.

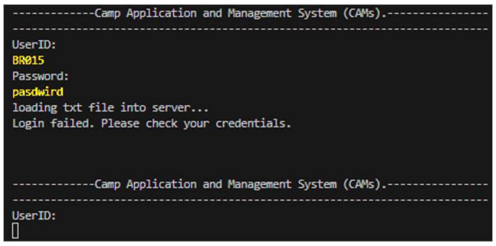
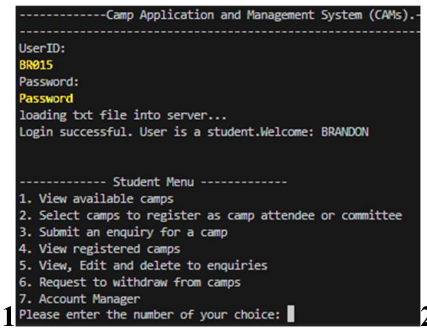
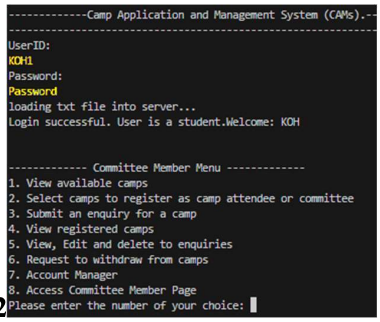
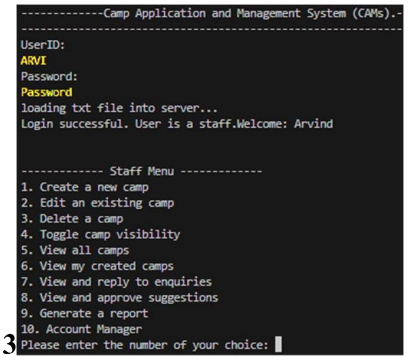
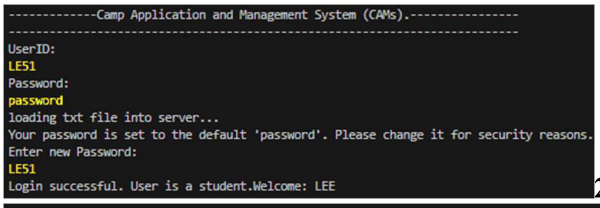
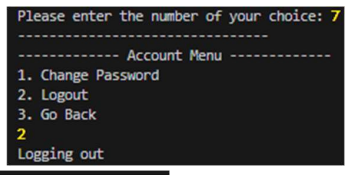
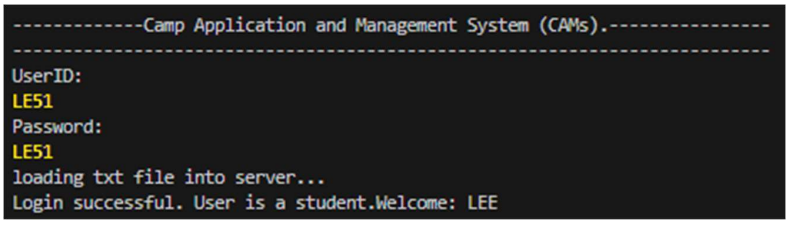
- One example would be a method named updateSuggestion in SuggestionManager,which is a class responsible for handling suggestions for committee members.This method takes in an instance of student and an instance of suggestion in order to handle and update the specific suggestion passed into the method.

Detailed UML Class Diagram

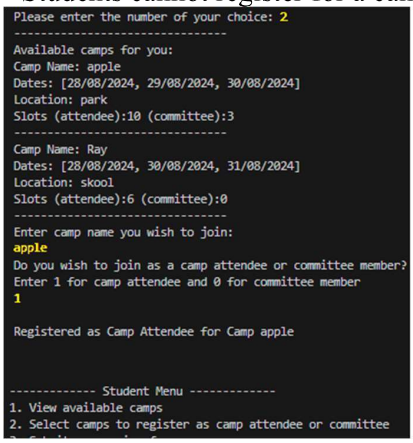
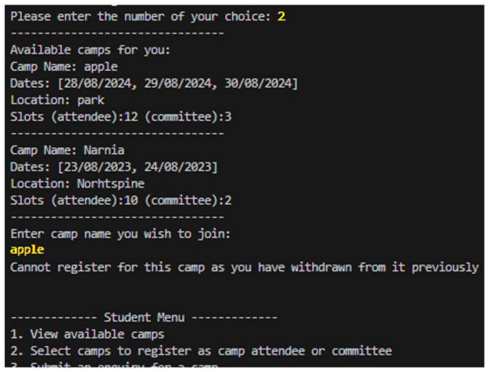
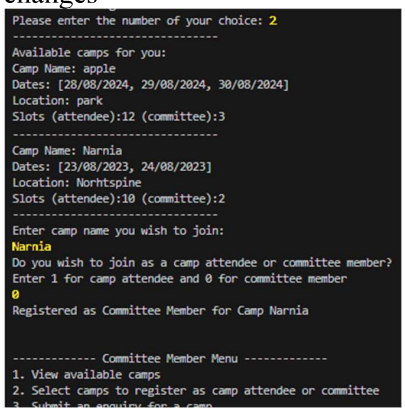


Essential Test Cases

a. Login

Test Cases	Screen Capture (User Inputs highlighted in yellow)
Invalid ID Or Valid ID with wrong password	
Login Successful Different menu list 1. Staff 2. Camp Committee 3. Student	1  2  3 
Change Password (default password) Option. 1. First log in 2. Log out 3. Log in again	1.  2.  3.  - user remains logged in after password change and can choose to continue or log out.

b. Main Page for Student

Test Cases	Screen Capture (User Inputs highlighted in yellow)
View available camps In the txt file, only camps with "Visibility: true" AND matching faculty camps are visible to the student	
Register a camp as attendee or committee	- Students cannot register for a camp he/she has withdrawn from   - Students become committee member when they register as one, and their main menu changes 

<table>
<tr>
<td>Submit enquiries for camp</td>
<td>

```
Please enter the number of your choice: 3
-------------------------------
Available camps for inquiry:
chop
apple
party
Narnia
Enter the name of the camp you want to inquire about: Narnia
Enter your inquiry message: Is lunch  provided?
Inquiry added successfully.
```

</td>
</tr>
<tr>
<td>View registered camps</td>
<td>

```
Please enter the number of your choice: 4
-------------------------------
Camps you are attending:
Camp Name: Pear
Dates: [15/05/2024]
Location: NS2
-------------------------------
Camps you are a committee member of:
Camp Name: BlueFire
Dates: [24/05/2024, 25/05/2024]
Location: NS3
-------------------------------
```

</td>
</tr>
<tr>
<td>View reply to enquiry/
Edit enquiry</td>
<td>

```
Please enter the number of your choice: 5
-------------------------------
-------------
a. View Inquiries
b. Edit an Inquiry
c. Delete an Inquiry
d. Exit
a
1. will there be transport?
Camp:  apple
Reply: yes
2. Is lunch  provided?
Camp:  Narnia
Reply: null
```

```
Please enter the number of your choice: 5
-------------------------------
-------------
a. View Inquiries
b. Edit an Inquiry
c. Delete an Inquiry
d. Exit
b
Enter the index of the inquiry you want to edit:
2
Enter the updated message:
Is dinner provided?
Inquiry updated successfully.
```

- replies will be updated whenever the staff respond
</td>
</tr>
<tr>
<td>Request to withdraw from camps</td>
<td>
- withdraw as camp attendee is allowed

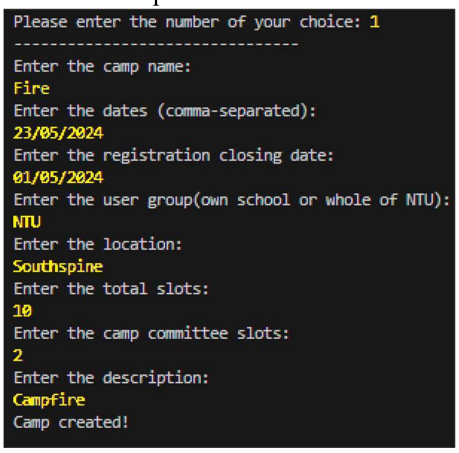
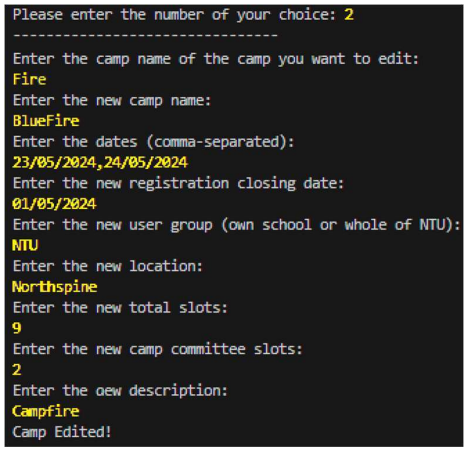
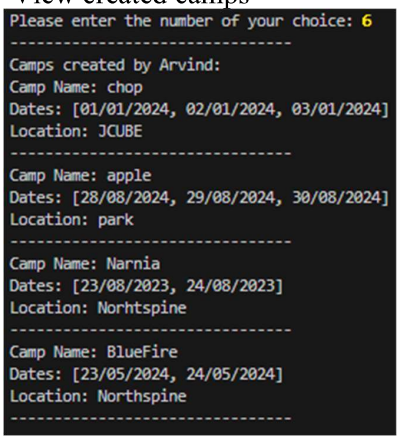
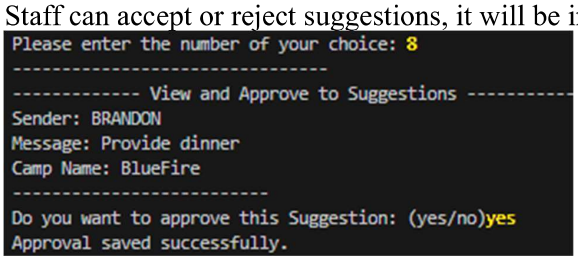
```
Please enter the number of your choice: 6
-------------------------------
Enter the camp name to withdraw from:
apple
You have successfully withdrawn from the camp.
```

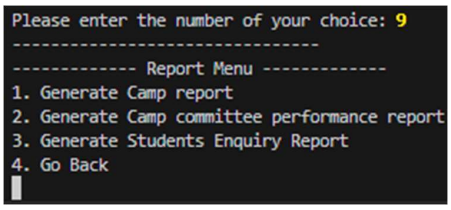
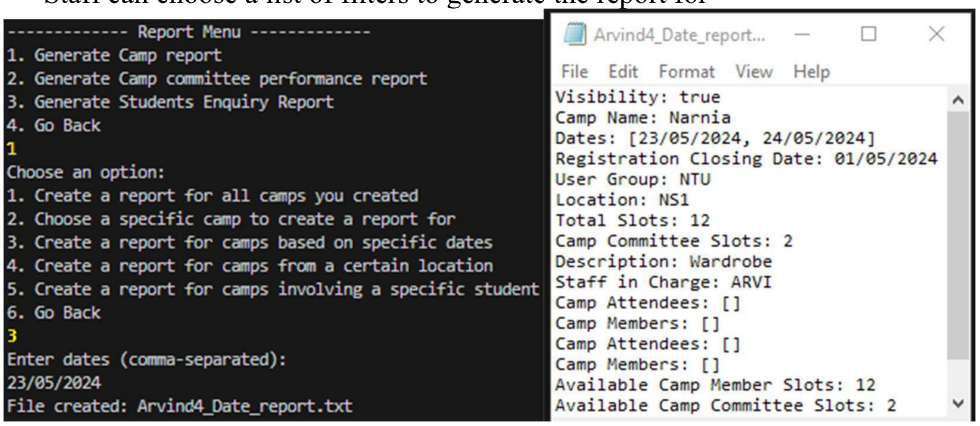
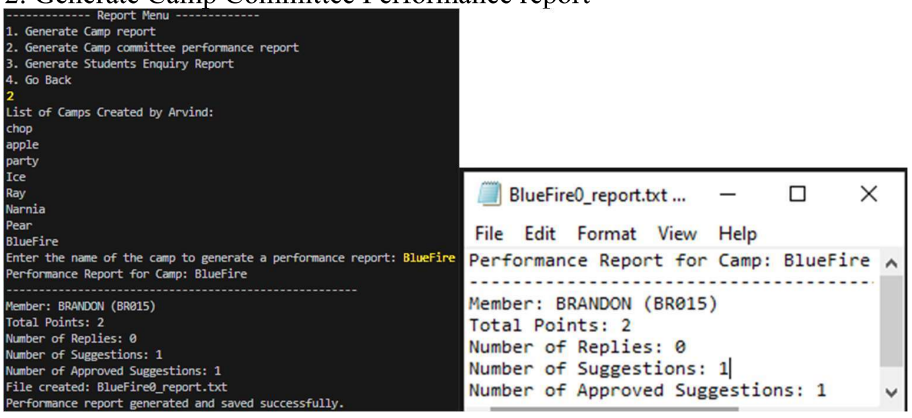
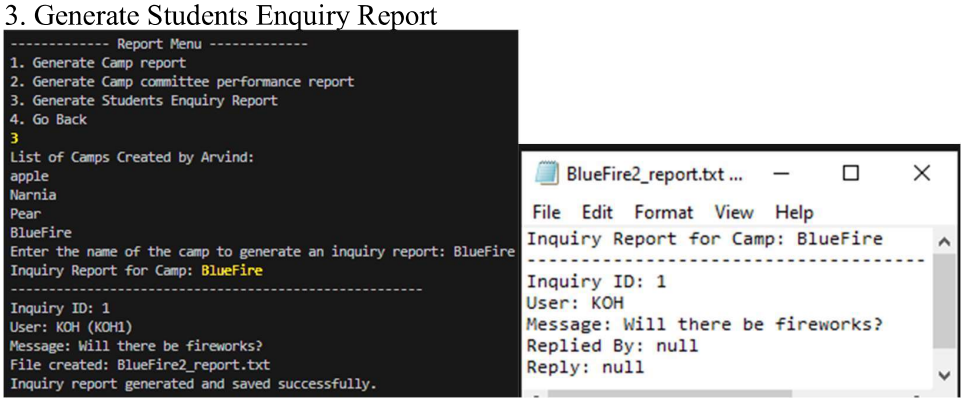
- withdraw as camp committee member not allowed

```
Please enter the number of your choice: 6
-------------------------------
Enter the camp name to withdraw from:
Narnia
You are a Member of the camp
Failed to withdraw from the camp. Please check the camp name.
```

</td>
</tr>
</table>

c. Main Page for Staff

Test Cases	Screen Capture (User Inputs highlighted in yellow)
Create/Edit/View camps	Create camps - Prompt to enter/edit the details for the camps  Edit camps  View created camps 
View/React to suggestions	Staff can accept or reject suggestions, it will be in “pending” state before that 

Generate report	
	1. Generate Camp Report - Staff can choose a list of filters to generate the report for  2. Generate Camp Committee Performance report  3. Generate Students Enquiry Report 

d. Main Page for Committee Member

Test Cases	Screen Capture (User Inputs highlighted in yellow)
Submit suggestions	Please enter the number of your choice: 8 ------------------------------ ------------ Committee Member Menu ------------ 1. Submit suggestions 2. View and Reply enquiry 3. View, Edit and delete to Suggestions 4. Generate report 5. Go back 1 Available camps for suggestions: BlueFire Enter the name of the camp you want to submit suggestions about: BlueFire Enter your Suggestion message: Free gifts Suggestion added successfully.
View/edit/delete own suggestions	View ------------ Committee Member Menu ------------ 1. Submit suggestions 2. View and Reply enquiry 3. View, Edit and delete to Suggestions 4. Generate report 5. Go back 3 -------------- a. View Suggestion b. Edit an Suggestion c. Delete an Suggestion d. Exit a 1. Free gifts Camp: BlueFire Approved: pending Edit ------------ Committee Member Menu ------------ 1. Submit suggestions 2. View and Reply enquiry 3. View, Edit and delete to Suggestions 4. Generate report 5. Go back 3 -------------- a. View Suggestion b. Edit an Suggestion c. Delete an Suggestion d. Exit b Enter the index of the Suggestion you want to edit: 1 Enter the updated message: Free bottle Suggestion updated successfully. Delete ------------ Committee Member Menu ------------ 1. Submit suggestions 2. View and Reply enquiry 3. View, Edit and delete to Suggestions 4. Generate report 5. Go back 3 -------------- a. View Suggestion b. Edit an Suggestion c. Delete an Suggestion d. Exit c Enter the index of the Suggestion you want to delete: 1 Are you sure you want to delete this Suggestion? (Y/N) Y Suggestion deleted.

View/Reply to enquiries

```
------------ Committee Member Menu ------------
1. Submit suggestions
2. View and Reply enquiry
3. View, Edit and delete to Suggestions
4. Generate report
5. Go back
2
------------ View and Reply to Inquiries --------
Sender: KOH
Message: Will there be fireworks?
Camp Name: BlueFire
-------------------------
Enter your reply: No
Reply saved successfully.
```

Generate report

```
------------ Committee Member Menu ---------
1. Submit suggestions
2. View and Reply enquiry
3. View, Edit and delete to Suggestions
4. Generate report
5. Go back
4
------------ Report Menu -------------
1. Generate Camp report
2. Generate Students Enquiry Report
3. Go Back
```

1. Generate Camp Report

- Committee members can generate report filtered by attendees or committee members

```
------------ Report Menu -------------
1. Generate Camp report
2. Generate Students Enquiry Report
3. Go Back
1
BlueFire
Enter the Camp to write a report: BlueFire
0. Generate the whole report
Filter by:
1. Camp Attendees
2. Camp Members
1
File created: BlueFire0_report.txt
Report generated successfully.
```

*BlueFire0_report.txt - Notepad

File Edit Format View Help

```
Visibility: true
Camp Name: BlueFire
Dates: [24/05/2024, 25/05/2024]
Registration Closing Date: 01/05/2024
User Group: NTU
Location: NS3
Total Slots: 8
Camp Committee Slots: 2
Description: Campfire
Staff in Charge: ARVI
Camp Attendees: [Name: KOH, Student ID: KOH1]
Available Camp Member Slots: 7
Available Camp Committee Slots: 1
```

2. Generate Students Enquiry Report

```
------------ Report Menu -------------
1. Generate Camp report
2. Generate Students Enquiry Report
3. Go Back
2
List of Camps BRANDON is a member of:
BlueFire
Enter the name of the camp to generate an inquiry report:
BlueFire
Inquiry Report for Camp: BlueFire
-----------------------------------------------------
Inquiry ID: 1
User: KOH (KOH1)
Message: Will there be fireworks?
File created: BlueFire3_report.txt
Inquiry report generated and saved successfully.
```

BlueFire3_report.txt - ...

File Edit Format View Help

```
Inquiry Report for Camp: BlueFire
-----------------------------------------
Inquiry ID: 1
User: KOH
Message: Will there be fireworks?
Replied By: BR015
Reply: No
```

Reflection

The development of our CAMs was a challenging yet rewarding experience that provided us a deep understanding of object-oriented programming concepts, SOLID design principles, and also gave us a sense of what it is like to work in a group on a small-scale coding project. However, it was without a few hiccups.

The initial challenges we encountered were delegating tasks among each group member. Deciding who would code which part of the program based on the requirements demanded careful consideration. We ended up taking a brute-force approach for the coding process, where each member would simply continue to modify where the previous group member stopped working on the code. After which, the team is informed of what changes were made to the code. A list of tasks that needed to be done gave us some direction and kept us on track for the completion of the project.

While our project showcased strengths in adhering to SRP and OCP in some areas, challenges arose in implementing the other aspect of SOLID design principles. We encountered difficulties in breaking down our functioning code into smaller, more manageable classes, which resulted in 1 or 2 “god” classes that contradicted the principles that we aimed to follow.

In conclusion, this group project taught us invaluable lessons in teamwork, task delegation, and the practical application of OOP concepts and SOLID design principles. While we successfully created a functional program, the journey also highlighted areas for improvement, particularly in achieving a more granular class structure and fine-tuning adherence to SOLID principles. As we move forward, addressing these challenges will be instrumental in refining our skills as Java developers and enhancing the robustness of our future projects.