SC2002 OBJECT-ORIENTED

DESIGN AND PROGRAMMING

AY23/24 Semester 1 Group Assignment

Camp Application and Management System (CAMS)

Lab Group: SCS2 Group 1

Group Members		
S/N	**Matriculation No.**	**Student Name**
1.	U2222332A	Kalidoss Madhumitha
2.	U2222549G	Praveena Vijayan
3.	U2222761C	Manasarow Gunasekaran
4.	U2222429C	Augustine Jesuraj Senchia Gladine
5.	U2222685K	Seet Tze Shin, Cheyenne

Date of Submission: 26/11/2023

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

NANYANG TECHNOLOGICAL UNIVERSITY

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 honoured 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
Manasarow Gunasekaran	SC2002	SCS2	Manasarow (26/11/2023)
Kalidoss Madhumitha	SC2002	SCS2	Madhumitha (26/11/2023)
Praveena Vijayan	SC2002	SCS2	Praveena (26/11/2023)
Augustine Jesuraj Senchia Gladine	SC2002	SCS2	Senchia (26/11/2023)
Seet Tze Shin, Cheyenne	SC2002	SCS2	Cheyenne (26/11/2023)

Important notes:

1. Name must **EXACTLY MATCH** the one printed on your Matriculation Card.
2. Student Code of Academic Conduct includes the latest guidelines on usage of Generative AI and any other guidelines as released by NTU.

1.1 Application Overview

The Camp Application and Management System (CAMs) is an application for staff and students to manage, view and register for camps within NTU. The application will act as a centralised hub for all staff and students.

We employ array lists to organise and store all camp information, suggestions and enquiries. User data from an Excel file, encompassing names, emails, and faculty details, is organised into role-specific array lists. During this process, user IDs are also assigned, and passwords are securely stored into the array lists.

The utilisation of array lists alongside secure user data handling reinforces the system's efficiency, and organisation in managing camp-related functionalities and user interactions. Additionally, the application features a main application class for login and tailored menus for the different user groups, ensuring seamless navigation and access to specific functionalities, thereby enhancing overall usability.

1.2 Assumptions

Below are the assumptions taken into consideration when building the CAMS:

1) The totalSlots in Camp class refer to the total slots for camp attendees. The campCommitteeSlots in Camp class refer to the total slots for camp committee members. Both attributes are disjoint.
2) Camps can start and end on the same day.
3) Every camp will have at least one attendee signed up (camp will not only consists of committee members)
4) The system is sensitive to whitespace. Eg. “CampA” and “Camp A” are different camps. We will assume the user inputs the camp name accurately with accurate whitespaces.
5) The staff cannot make any more changes to camp details once there are students enrolled in a camp (either camp attendees or camp committee members)
6) A staff can only edit, delete and view the details of their own camps.
7) A staff need not necessarily create a camp for their own faculty, they can create a camp for any faculty, and they can create as many camps as they want.
8) The performance report generated by the staff for camp committee members only contains the name, userID and the number of points collected by the committee member.

9) When staff create camp, we assume the staff will input a valid duration for the camp. Start and end dates for camp show a reasonable camp duration.
10) Once a student withdraws from camp (as a camp attendee) they cannot re-register as a camp attendee or camp committee member for that same camp
11) Students will still be able to submit enquiries for the camp even if they are a committee member of that camp. We assume that camp committee members will not answer their own enquiries.

2. Design Considerations

2.1 SOLID Approach

2.1.1 Single-Responsibility Principle

The Single-Responsibility Principle (SRP) states that a class should have only one reason to change, having only one responsibility.

Our code demonstrates this through the following two ways:

1. Using an *Object* class and corresponding *ObjectList* class
    - *Object* classes represent individual entities in the system, like Student, Staff, and Camp. They contain specific attributes and methods for manipulation and access.
    - *ObjectList classes* manage the collections of these entities, providing functionalities like adding, removing, and organising instances, such as StudentList or CampList.
2. Using two broad types of classes for each user - *Manager* and *Service* classes
    - *Manager* classes often present the functionalities available to users in a more generalised manner. For instance, CampManager provides methods to handle camp like createNewCamp and editExistingCamp
    - *Service* support managers by implementing actions necessary to fulfil their requests, such as CampRegistrationValidation verifying student eligibility for camp sign-up.

2.1.2 Open-Closed Principle

The Open-Closed Principle promotes building systems in a way that allows for extension without modifying the existing code.

Our classes exhibit this principle through inheritance and the use of interfaces.

1. Through Inheritance:

    - For example, the Student class is closed for modification as it does not modify the existing behaviour inherited from the User class. But allows for extension without altering the existing behaviour of the User class. It introduces a new ArrayList “withdrawnCampNames” without modifying the behaviour of the inherited User properties and methods.
2. Through Interfaces:
    - For example, UserList provides a base set of functionalities such as setting userID and initial default password. When StudentList and StaffList implement the interface, additional methods such as findStudentByID and findStaffByID are introduced.

2.1.3 Liskov Substitution Principle

The Liskov Substitution Principle states that objects of subclasses must be substitutable for their base types without affecting the correctness of the program.

For example, our codes implemented this principle through inheriting Student (Sub Class) from User Class (Base Class).

- The Student class inherits functionalities from the User class without overriding any base methods or modifying its behaviour. Therefore, objects instantiated from the Student class can seamlessly replace objects of the User class wherever objects of the User class are expected without altering the program's behaviour or correctness, though the objects of the Student class possess other student functionalities.

2.1.4 Interface Segregation Principle

The Interface Segregation Principle (ISP) emphasises segregating interfaces into smaller, specific ones, rather than having a single large interface that encompasses multiple functionalities.

We implemented this principle through usage of interfaces such as UserList and FeedbackList interfaces, which define specific functionalities related to feedback management (FeedbackList) and user data handling (UserList), promoting a more focused and specific interface.

1. SuggestionList and EnquiryList Implement FeedbackList:
    - These classes implement the FeedbackList interface, providing methods specific to managing suggestions or enquiries.
2. StudentList and StaffList Implement UserList:

- These classes implement the UserList interface and provide methods related to managing student and staff data, such as setting default passwords, assigning user IDs, etc.

2.1.5 Dependency Injection Principle

The Dependency Injection Principle states that high-level modules should not depend upon low-level modules, both should depend upon abstractions.

For example, the CampRegistration and CampRegistrationValidation classes are high-level modules that do not directly depend on low-level details but rely on abstractions such as CampList and ObjectFinder. This ensures that the high-level modules remain unaware of detailed implementations, fostering modularity and flexibility. By injecting dependencies, we achieved a design that is easily extensible, allowing for changes in the underlying implementations without affecting the core functionality of our modules.

2.2 Object-Oriented Principles

2.2.1 Inheritance

Inheritance is a key concept in our application that allows the subclasses to inherit attributes and functions from a superclass.

Using inheritance in our application allows us to achieve:

1. Reusability: Student and Staff inherit from User class, reuse coding, ensuring consistency and reducing redundancy across user functionalities.
2. Common Functionality Encapsulation: CCMSuggestionManager and StudentEnquiryManager inherit from abstract base class FeedbackManager which enforce a unified structure for managing feedback, ensuring a consistent approach to handling feedback-related functions.
3. Polymorphism and Flexibility: This inheritance enables specialised behaviour in subclasses, providing flexibility to extend or modify functionalities specific to students, staff, or feedback management without altering the base behaviour defined in the User or FeedbackManager classes.

2.2.2 Encapsulation

Encapsulation conceals a class's internal workings by keeping implementation details private and exposing only a public interface for interaction by declaring instance variables as private, limiting their access to within the class. This approach ensures data integrity, controlled access, and promotes flexibility and reusability in code.

In our application, the instance variables for almost all of our classes are private, a demonstration of the concept of encapsulation. For example, in the Camp class, all its instance variables such as campName, registrationClosingDate and listOfAttendees are private.

This allows for:

1. Flexibility: Encapsulation allows us to switch between read-only or write-only modes by defining getter and setter methods for instance variables. This flexibility aids in altering access to data without changing the class's core logic.
2. Reusability: Encapsulation simplifies code reuse and adaptation by encapsulating data and methods within classes. This "black box" approach allows using classes without needing internal knowledge. Hence, when applied in new applications, there's no need to re-implement encapsulated data and methods.

2.2.3 Polymorphism

Our code employs compile-time polymorphism using method overloading in Java, where methods share the same name but have different parameters. This technique allows the compiler to determine the specific method to execute based on the method signature used in the code, enabling multiple behaviours under a single method name. This approach enhances code readability, adaptability, and clarity.

For example, in StaffAndCCMEnquiryManager, we used method overloading for the submitAnswer method based on the user (staff or camp committee member) calling the method, and the method implementations vary accordingly.

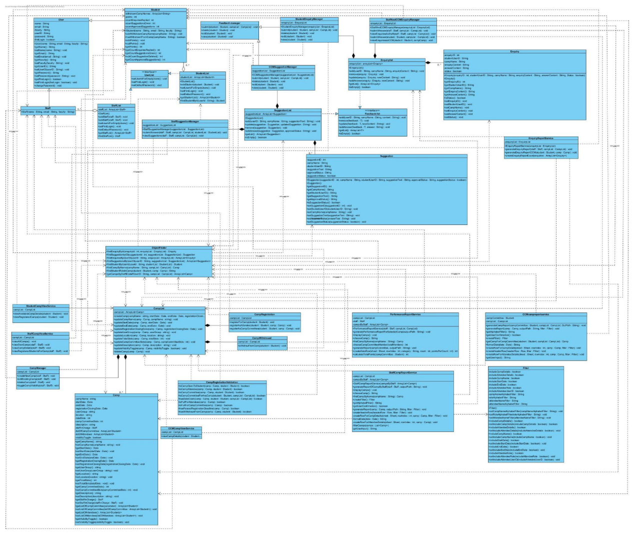
2.2.4 Abstraction

Abstraction refers to the concept of hiding complex implementation details and displaying only the necessary features of an object.

In our code, abstraction is demonstrated through the usage of Abstract Classes and Interfaces.

- For example, the abstract class FeedbackManager defines common methods for managing feedback, without providing a full implementation for all methods. The subclasses CCMSuggestionManager and StudentEnquiryManager then implement these methods as per their specific functionality.
- Interfaces such as FeedbackList or UserList define a contract of methods that different classes implement in their own way, ensuring that classes that implement these interfaces adhere to a set of behaviours.

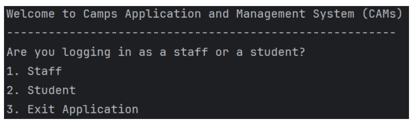
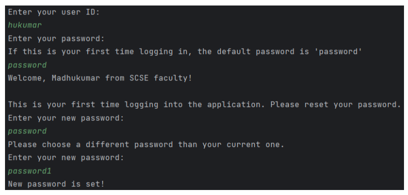
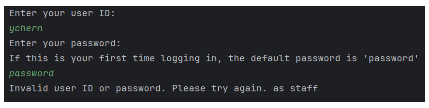
3. Detailed UML Class Diagram



(for a clearer image please view the UML Diagram file attachment)

4. Testing (for more extensive testing refer to the additional testing document given)

4.1 Login

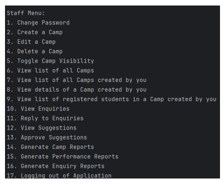
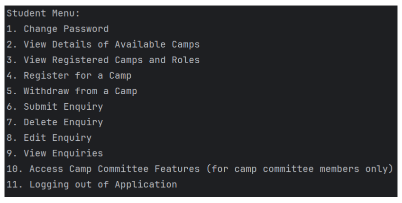
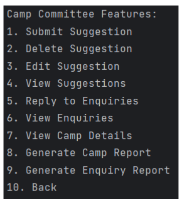
Login Page	**Successful/Unsuccessful login**	
	Successful (able to change password after login)	Unsuccessful
		

```
Welcome to Camps Application and Management System (CAMs)
----------------------------------------------------------
Are you logging in as a staff or a student?
1. Staff
2. Student
3. Exit Application
```

```
Enter your user ID:
hukumar
Enter your password:
If this is your first time logging in, the default password is 'password'
password
Welcome, Madhukumar from SCSE faculty!

This is your first time logging into the application. Please reset your password.
Enter your new password:
password
Please choose a different password than your current one.
Enter your new password:
password1
New password is set!
```

```
Enter your user ID:
ychern
Enter your password:
If this is your first time logging in, the default password is 'password'
password
Invalid user ID or password. Please try again. as staff
```

4.2 Different Menu Pages

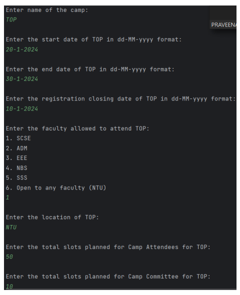
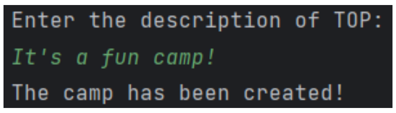
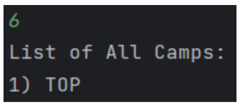
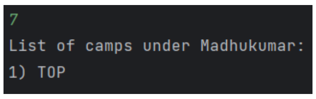
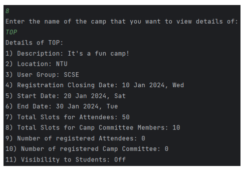
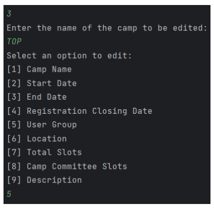
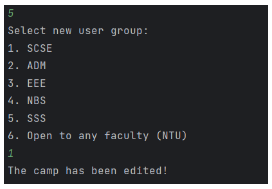
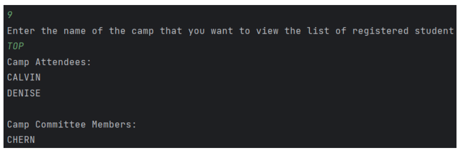
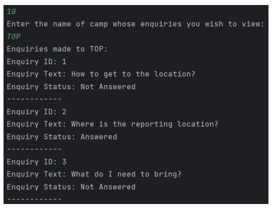
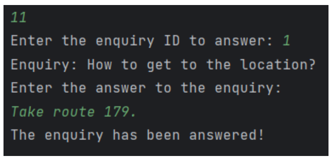
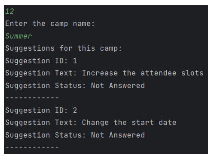
Staff Menu Page	Student Menu Page	Camp Committee Member Menu Page (accessible through student menu)
		

```
Staff Menu:
1. Change Password
2. Create a Camp
3. Edit a Camp
4. Delete a Camp
5. Toggle Camp Visibility
6. View list of all Camps
7. View list of all Camps created by you
8. View details of a Camp created by you
9. View list of registered students in a Camp created by you
10. View Enquiries
11. Reply to Enquiries
12. View Suggestions
13. Approve Suggestions
14. Generate Camp Reports
15. Generate Performance Reports
16. Generate Enquiry Reports
17. Logging out of Application
```

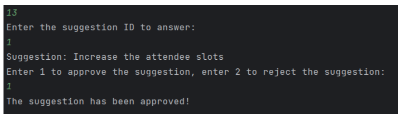
```
Student Menu:
1. Change Password
2. View Details of Available Camps
3. View Registered Camps and Roles
4. Register for a Camp
5. Withdraw from a Camp
6. Submit Enquiry
7. Delete Enquiry
8. Edit Enquiry
9. View Enquiries
10. Access Camp Committee Features (for camp committee members only)
11. Logging out of Application
```

```
Camp Committee Features:
1. Submit Suggestion
2. Delete Suggestion
3. Edit Suggestion
4. View Suggestions
5. Reply to Enquiries
6. View Enquiries
7. View Camp Details
8. Generate Camp Report
9. Generate Enquiry Report
10. Back
```

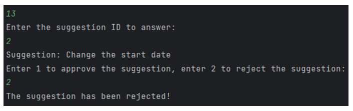
4.3 Staff functions

Create camps	 Enter name of the camp: *TOP* Enter the start date of TOP in dd-MM-yyyy format: *20-1-2024* Enter the end date of TOP in dd-MM-yyyy format: *30-1-2024* Enter the registration closing date of TOP in dd-MM-yyyy format: *10-1-2024* Enter the faculty allowed to attend TOP: 1. SCSE 2. ADM 3. EEE 4. NBS 5. SSS 6. Open to any faculty (NTU) *1* Enter the location of TOP: *NTU* Enter the total slots planned for Camp Attendees for TOP: *50* Enter the total slots planned for Camp Committee for TOP: *10*  Enter the description of TOP: *It's a fun camp!* The camp has been created!		
View camps	View all camps	View camps created by user	View camp details of camp created by user
	 *6* List of All Camps: 1) TOP	 *7* List of camps under Madhukumar: 1) TOP	 *8* Enter the name of the camp that you want to view details of: *TOP* Details of TOP: 1) Description: It's a fun camp! 2) Location: NTU 3) User Group: SCSE 4) Registration Closing Date: 10 Jan 2024, Wed 5) Start Date: 20 Jan 2024, Sat 6) End Date: 30 Jan 2024, Tue 7) Total Slots for Attendees: 50 8) Total Slots for Camp Committee Members: 10 9) Number of registered Attendees: 0 10) Number of registered Camp Committee: 0 11) Visibility to Students: Off
Edit camp	 *3* Enter the name of the camp to be edited: *TOP* Select an option to edit: [1] Camp Name [2] Start Date [3] End Date [4] Registration Closing Date [5] User Group [6] Location [7] Total Slots [8] Camp Committee Slots [9] Description *5*  *5* Select new user group: 1. SCSE 2. ADM 3. EEE 4. NBS 5. SSS 6. Open to any faculty (NTU) *1* The camp has been edited!		
View list of students registered	 *9* Enter the name of the camp that you want to view the list of registered student *TOP* Camp Attendees: CALVIN DENISE Camp Committee Members: CHERN		
View Enquiries	 *10* Enter the name of camp whose enquiries you wish to view: *TOP* Enquiries made to TOP: Enquiry ID: 1 Enquiry Text: How to get to the location? Enquiry Status: Not Answered ------------ Enquiry ID: 2 Enquiry Text: Where is the reporting location? Enquiry Status: Answered ------------ Enquiry ID: 3 Enquiry Text: What do I need to bring? Enquiry Status: Not Answered ------------		
Reply To Enquiries	 *11* Enter the enquiry ID to answer: *1* Enquiry: How to get to the location? Enter the answer to the enquiry: *Take route 179.* The enquiry has been answered!		
View suggestions	 *12* Enter the camp name: *Summer* Suggestions for this camp: Suggestion ID: 1 Suggestion Text: Increase the attendee slots Suggestion Status: Not Answered ------------ Suggestion ID: 2 Suggestion Text: Change the start date Suggestion Status: Not Answered ------------		
	Accept		Reject

Accept/Reject suggestion



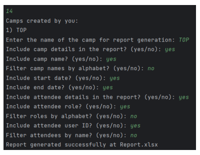
```
13
Enter the suggestion ID to answer:
1
Suggestion: Increase the attendee slots
Enter 1 to approve the suggestion, enter 2 to reject the suggestion:
1
The suggestion has been approved!
```



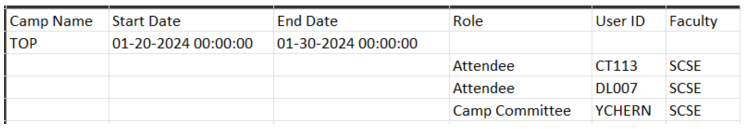
```
13
Enter the suggestion ID to answer:
2
Suggestion: Change the start date
Enter 1 to approve the suggestion, enter 2 to reject the suggestion:
2
The suggestion has been rejected!
```

Generate report

Camp

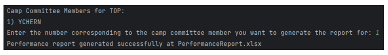


```
14
Camps created by you:
1) TOP
Enter the name of the camp for report generation: TOP
Include camp details in the report? (yes/no): yes
Include camp name? (yes/no): yes
Filter camp names by alphabet? (yes/no): no
Include start date? (yes/no): yes
Include end date? (yes/no): yes
Include attendee details in the report? (yes/no): yes
Include attendee role? (yes/no): yes
Filter roles by alphabet? (yes/no): no
Include attendee user ID? (yes/no): yes
Filter attendees by name? (yes/no): no
Report generated successfully at Report.xlsx
```

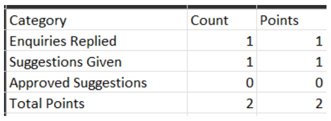


Camp Name	Start Date	End Date	Role	User ID	Faculty
TOP	01-20-2024 00:00:00	01-30-2024 00:00:00			
			Attendee	CT113	SCSE
			Attendee	DL007	SCSE
			Camp Committee	YCHERN	SCSE

Performance

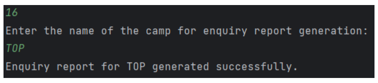


```
Camp Committee Members for TOP:
1) YCHERN
Enter the number corresponding to the camp committee member you want to generate the report for: 1
Performance report generated successfully at PerformanceReport.xlsx
```

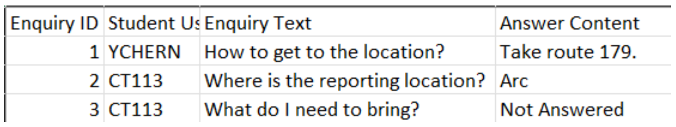


Category	Count	Points
Enquiries Replied	1	1
Suggestions Given	1	1
Approved Suggestions	0	0
Total Points	2	2

Enquiry



```
16
Enter the name of the camp for enquiry report generation:
TOP
Enquiry report for TOP generated successfully.
```

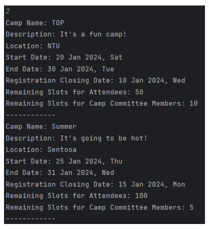


Enquiry ID	Student Us	Enquiry Text	Answer Content
1	YCHERN	How to get to the location?	Take route 179.
2	CT113	Where is the reporting location?	Arc
3	CT113	What do I need to bring?	Not Answered

4.4 Student functions

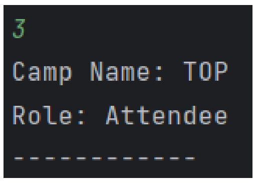
View Camps

View all camps



```
2
Camp Name: TOP
Description: It's a fun camp!
Location: NTU
Start Date: 20 Jan 2024, Sat
End Date: 30 Jan 2024, Tue
Registration Closing Date: 10 Jan 2024, Wed
Remaining Slots for Attendees: 50
Remaining Slots for Camp Committee Members: 10
------------
Camp Name: Summer
Description: It's going to be hot!
Location: Sentosa
Start Date: 25 Jan 2024, Thu
End Date: 31 Jan 2024, Wed
Registration Closing Date: 15 Jan 2024, Mon
Remaining Slots for Attendees: 100
Remaining Slots for Camp Committee Members: 5
------------
```

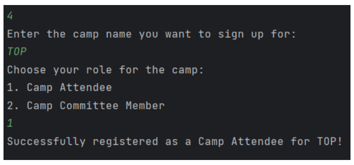
View registered camps



```
3
Camp Name: TOP
Role: Attendee
------------
```

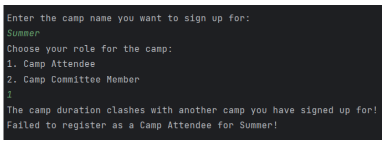
Register for a camp

Successful registration:



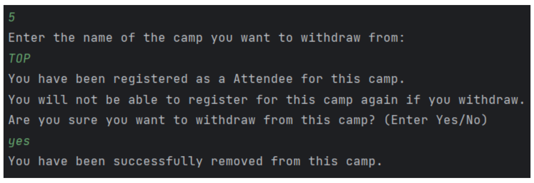
```
4
Enter the camp name you want to sign up for:
TOP
Choose your role for the camp:
1. Camp Attendee
2. Camp Committee Member
1
Successfully registered as a Camp Attendee for TOP!
```

Unsuccessful registration:



```
Enter the camp name you want to sign up for:
Summer
Choose your role for the camp:
1. Camp Attendee
2. Camp Committee Member
1
The camp duration clashes with another camp you have signed up for!
Failed to register as a Camp Attendee for Summer!
```

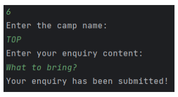
Withdraw from camp



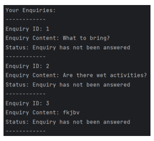
```
5
Enter the name of the camp you want to withdraw from:
TOP
You have been registered as a Attendee for this camp.
You will not be able to register for this camp again if you withdraw.
Are you sure you want to withdraw from this camp? (Enter Yes/No)
yes
You have been successfully removed from this camp.
```

	Submit enquiry	View enquiries	Delete enquiry

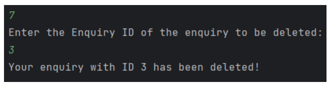
Submit/ view/edit enquiry



```
6
Enter the camp name:
TOP
Enter your enquiry content:
What to bring?
Your enquiry has been submitted!
```

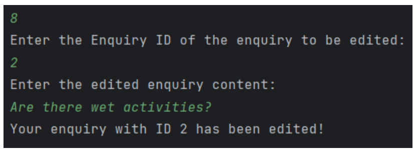


```
Your Enquiries:
------------
Enquiry ID: 1
Enquiry Content: What to bring?
Status: Enquiry has not been answered
------------
Enquiry ID: 2
Enquiry Content: Are there wet activities?
Status: Enquiry has not been answered
------------
Enquiry ID: 3
Enquiry Content: fkjbv
Status: Enquiry has not been answered
------------
```



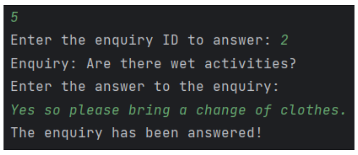
```
7
Enter the Enquiry ID of the enquiry to be deleted:
3
Your enquiry with ID 3 has been deleted!
```

Edit enquiry



```
8
Enter the Enquiry ID of the enquiry to be edited:
2
Enter the edited enquiry content:
Are there wet activities?
Your enquiry with ID 2 has been edited!
```

View reply to enquiry

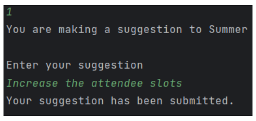


```
5
Enter the enquiry ID to answer: 2
Enquiry: Are there wet activities?
Enter the answer to the enquiry:
Yes so please bring a change of clothes.
The enquiry has been answered!
```

4.5 Camp Committee Member functions

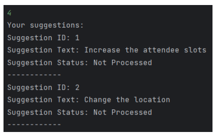
Submit/ view/ edit/ delete/ suggestion

Submit suggestion



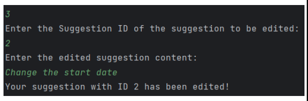
```
1
You are making a suggestion to Summer

Enter your suggestion
Increase the attendee slots
Your suggestion has been submitted.
```

View suggestions



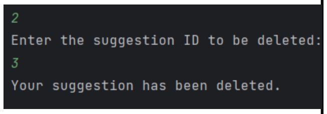
```
4
Your suggestions:
Suggestion ID: 1
Suggestion Text: Increase the attendee slots
Suggestion Status: Not Processed
------------
Suggestion ID: 2
Suggestion Text: Change the location
Suggestion Status: Not Processed
------------
```

Edit suggestions



```
3
Enter the Suggestion ID of the suggestion to be edited:
2
Enter the edited suggestion content:
Change the start date
Your suggestion with ID 2 has been edited!
```

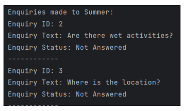
Delete suggestion



```
2
Enter the suggestion ID to be deleted:
3
Your suggestion has been deleted.
```

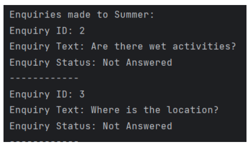
View enquiry

View enquiry



```
Enquiries made to Summer:
Enquiry ID: 2
Enquiry Text: Are there wet activities?
Enquiry Status: Not Answered
------------
Enquiry ID: 3
Enquiry Text: Where is the location?
Enquiry Status: Not Answered
------------
```

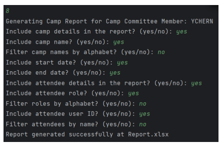
Reply to enquiry



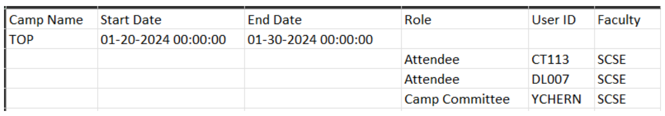
```
Enquiries made to Summer:
Enquiry ID: 2
Enquiry Text: Are there wet activities?
Enquiry Status: Not Answered
------------
Enquiry ID: 3
Enquiry Text: Where is the location?
Enquiry Status: Not Answered
------------
```

Generate report

Camp

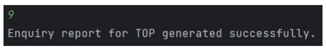


```
8
Generating Camp Report for Camp Committee Member: YCHERN
Include camp details in the report? (yes/no): yes
Include camp name? (yes/no): yes
Filter camp names by alphabet? (yes/no): no
Include start date? (yes/no): yes
Include end date? (yes/no): yes
Include attendee details in the report? (yes/no): yes
Include attendee role? (yes/no): yes
Filter roles by alphabet? (yes/no): no
Include attendee user ID? (yes/no): yes
Filter attendees by name? (yes/no): no
Report generated successfully at Report.xlsx
```

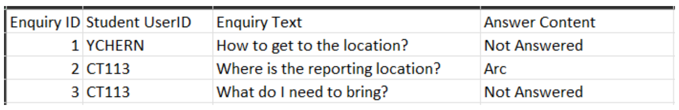


Camp Name	Start Date	End Date	Role	User ID	Faculty
TOP	01-20-2024 00:00:00	01-30-2024 00:00:00			
			Attendee	CT113	SCSE
			Attendee	DL007	SCSE
			Camp Committee	YCHERN	SCSE

Enquiry



```
9
Enquiry report for TOP generated successfully.
```



Enquiry ID	Student UserID	Enquiry Text	Answer Content
1	YCHERN	How to get to the location?	Not Answered
2	CT113	Where is the reporting location?	Arc
3	CT113	What do I need to bring?	Not Answered

5. Reflection

5.1 Knowledge learnt

First, we learned how to handle I/O operations, in particular, reading and writing to an excel file. We used this in order to read in the names of students and staff in order to initialise them in our CAMS system. Secondly, we made use of a new data type; date. This data type stores the calendar date. By importing the library "java.util.Date", users can input the date which we can format into the dd/mm/yyyy format. We also learned about predicates in java, a functional interface that represents a boolean-valued function of one argument. We have implemented predicates in our filter clase to filter the camps.

5.2 Difficulties encountered

In early planning, defining classes, relationships, and structure posed challenges. For instance, creating a separate class for Camp Committee initially overcomplicated other classes. We addressed this by researching object-oriented programs and simplifying ours accordingly.

5.3 Further improvements

Storing passwords in our current system poses a potential security risk. While they are currently kept as protected variables, there's still a vulnerability within the package. Implementing a hashing function would enhance security, making the data unreadable in case of a breach. Currently, user input data like camps, suggestions, and enquiries are stored in program-based arraylists, leading to data loss upon program termination. To improve security and convenience, we can consider storing such data in external Excel files.