

CAMS SC2002 SCSX

Lab Group 1

Wang Yangming (U2222553D)

Liau Zheng Wei (U2222032K)

Fan Tianyu (U2222105H)

Clayton Fernalo (U2220422E)

Christopher Angelo (U2220148L)

Table of contents

01

Introduction

CAMS Overview and Design

02

Demonstration & Test Cases

App Implementation
and Testing

01

Introduction

Camp Application and Management System (CAMs)



02

Demonstration &

Testing





SOLID

S

- Single Responsibility Principle

O

- Open Closed Principle

L

- Liskov Substitution Principle

I

- Interface Segregation Principle

D

- Dependency Injection Principle

S - Single Responsibility Principle

CampEnquiryController	
Method	Description
getEnquiries()	EnquiryList
getEnquiry(String)	Enquiry?
deleteEnquiry(Enquiry)	boolean
createEnquiry(Enquiry)	boolean
answerEnquiry(Enquiry, User, String)	boolean
getEnquiries(Student)	EnquiryList
getEnquiries(Camp)	EnquiryList
updateEnquiry(Enquiry)	boolean

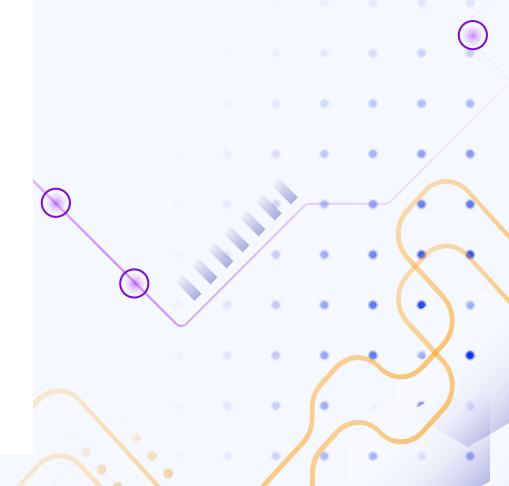
CampSuggestionController	
Method	Description
getSuggestions(Camp)	SuggestionList
getSuggestions(Staff)	SuggestionList
approveSuggestion(Suggestion, Staff, boolean)	boolean
getSuggestions(Student)	SuggestionList
deleteSuggestion(Suggestion)	boolean
createSuggestion(Student, CampDetails, Camp)	boolean

CampRegistrationController	
Method	Description
deregisterCamp(Camp, Student)	OperationResult
registerCampAsCommittee(Camp, Student)	OperationResult
registerCamp(Camp, Student)	OperationResult
checkJoinedAsCommittee(Student)	boolean
checkConflict(Camp, Student)	Camp?

CampViewController	
Method	Description
getCamp()	Camp[]
getCamp(Staff)	Camp[]
getCamp(String)	Camp[]
getCamp(Student)	Camp[]

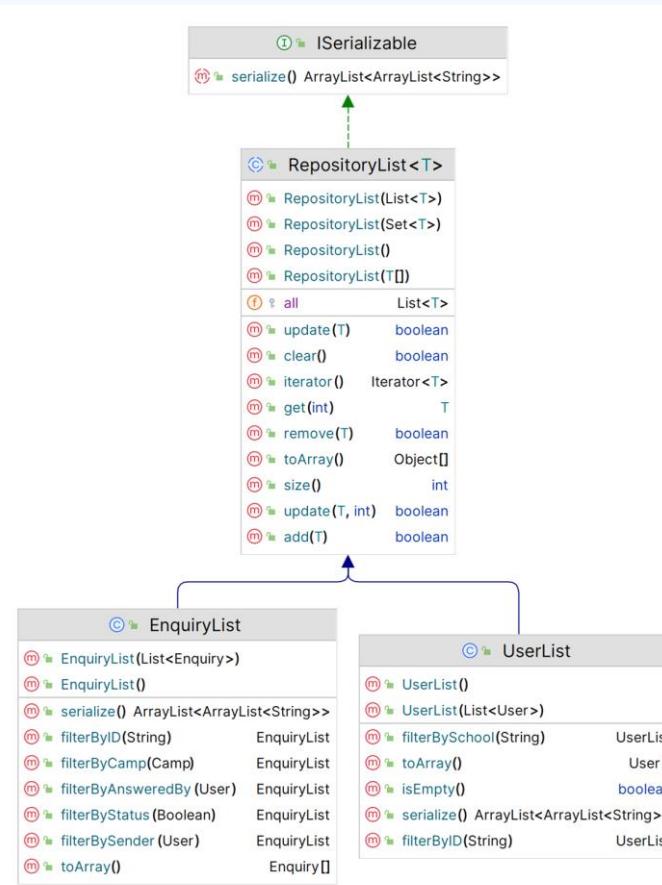
CampStatusMonitorController	
Method	Description
campList	CampList
generatePerformanceReport(Camp)	Map<String, Integer>
getAttendingStudents(Camp)	ArrayList<User>

CampModificationController	
Method	Description
deleteCamp(Camp)	boolean
createCamp(Camp)	boolean
editCamp(CampDetails)	void

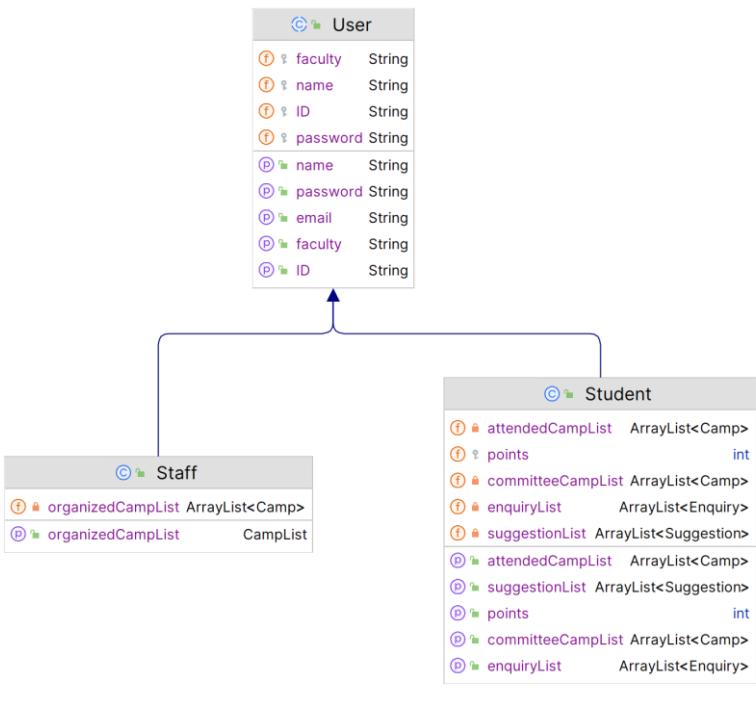




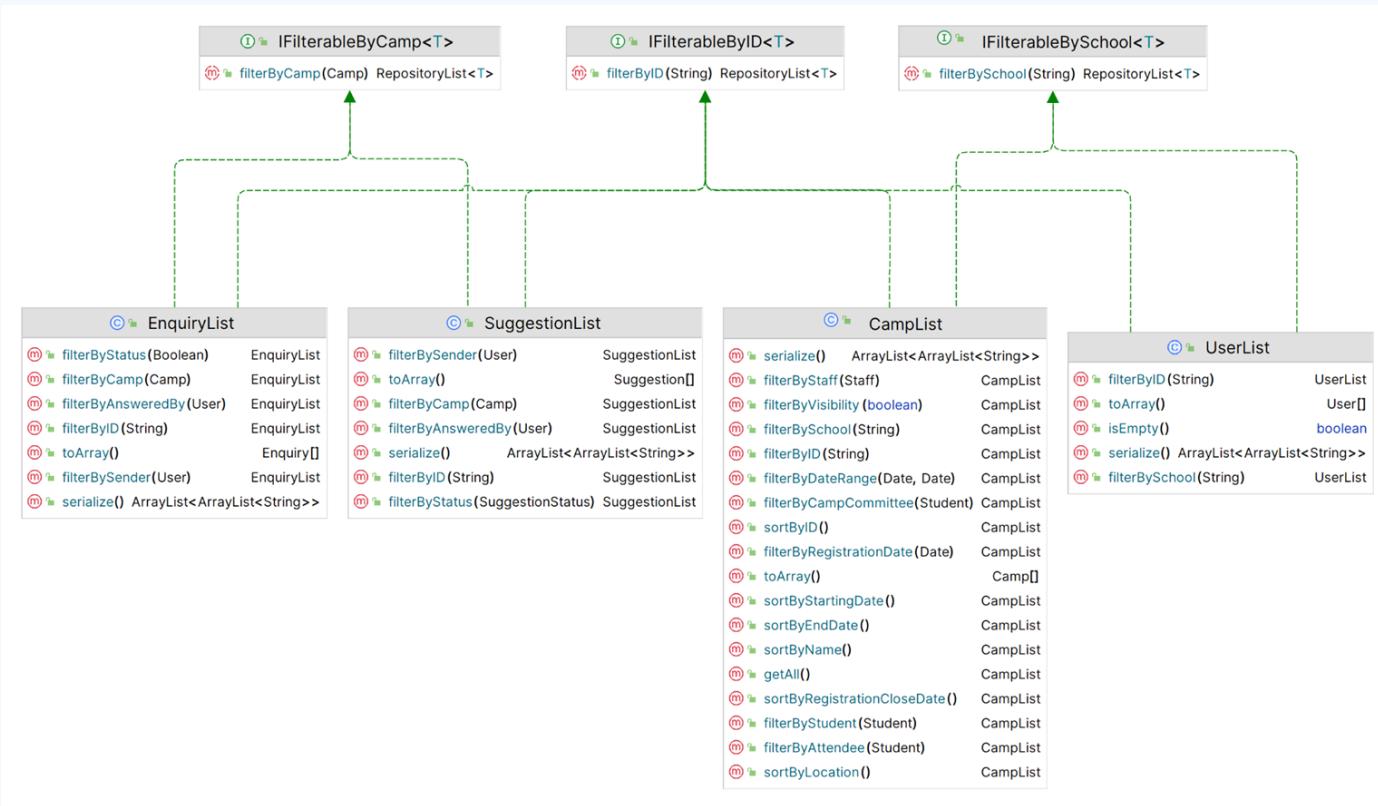
- Open Closed Principle



Liskov Substitution Principle



- Interface Segregation Principle



D

- Dependency Injection Principle

```
public static boolean exportToCSV(String filename, ISerializable serializable) {
    ArrayList<ArrayList<String>> data = serializable.serialize();
    try (CSVPrinter printer = new CSVPrinter(new FileWriter(filename), CSVFormat.DEFAULT)) {
        for (ArrayList<String> row : data) {
            printer.printRecord(row);
        }
    } catch (Exception e) {
        System.out.println(e.toString());
        return false;
    }
    return true;
}
```

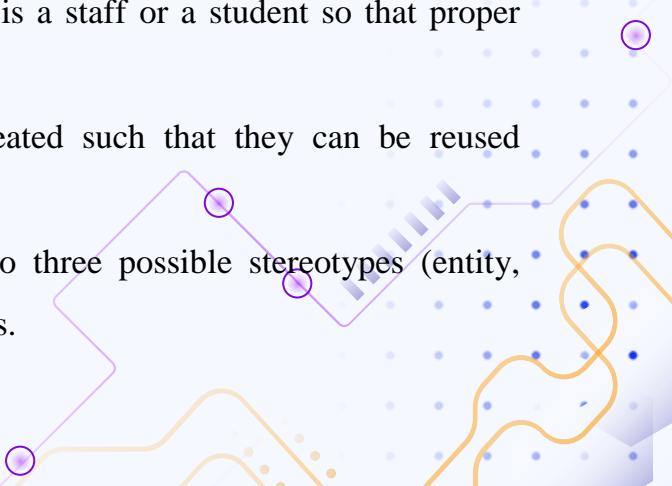
util/CSV.java

```
public static void save() {
    CSV.exportToCSV(Config.USER_REPOSITORY_PATH, userRepository);
    CSV.exportToCSV(Config.CAMP_REPOSITORY_PATH, campRepository);
    CSV.exportToCSV(Config.ENQUIRY_REPOSITORY_PATH, enquiryRepository);
    CSV.exportToCSV(Config.SUGGESTION_REPOSITORY_PATH, suggestionRepository);
}
```

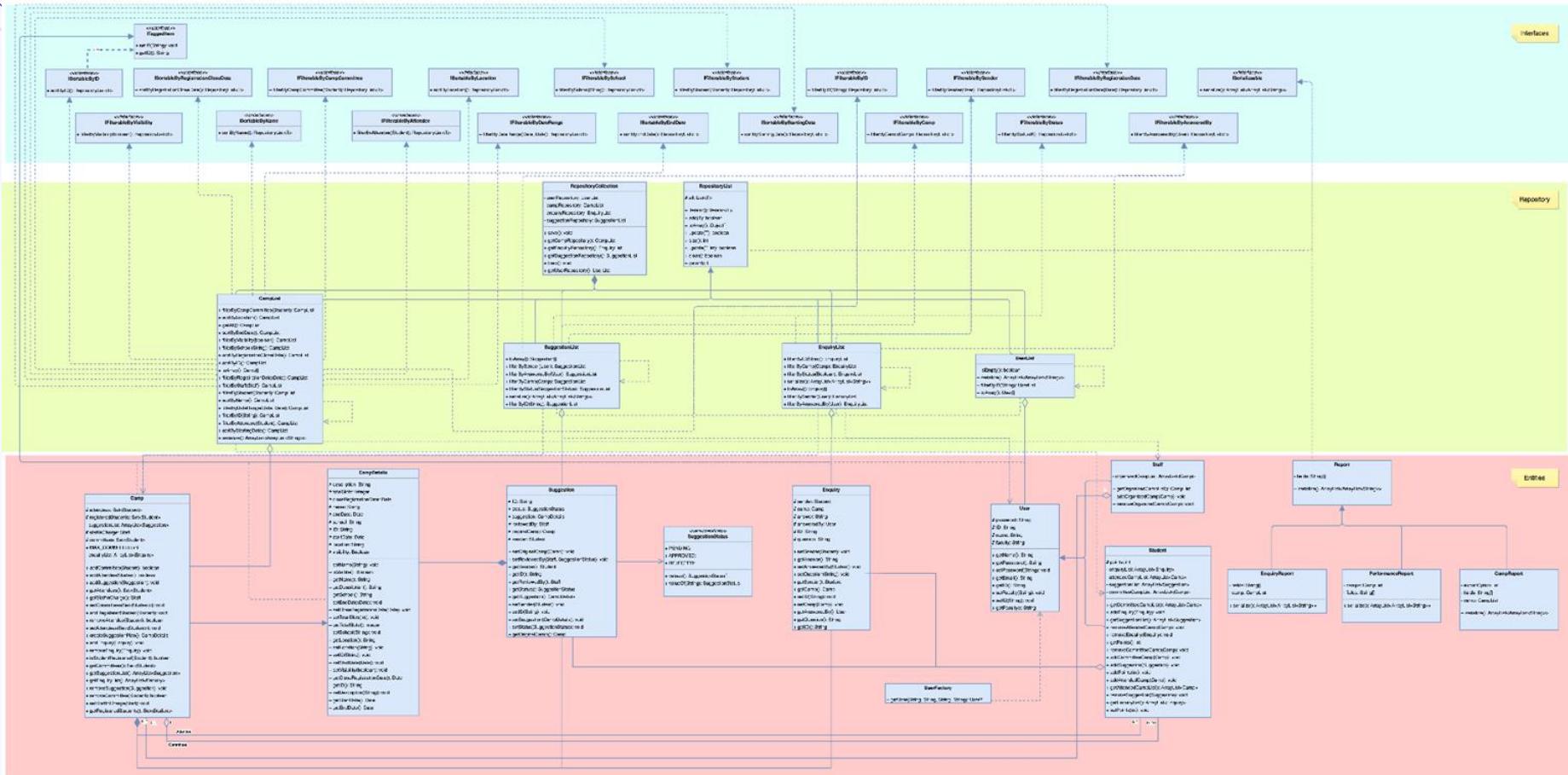
RepositoryCollection.java



Design Highlights

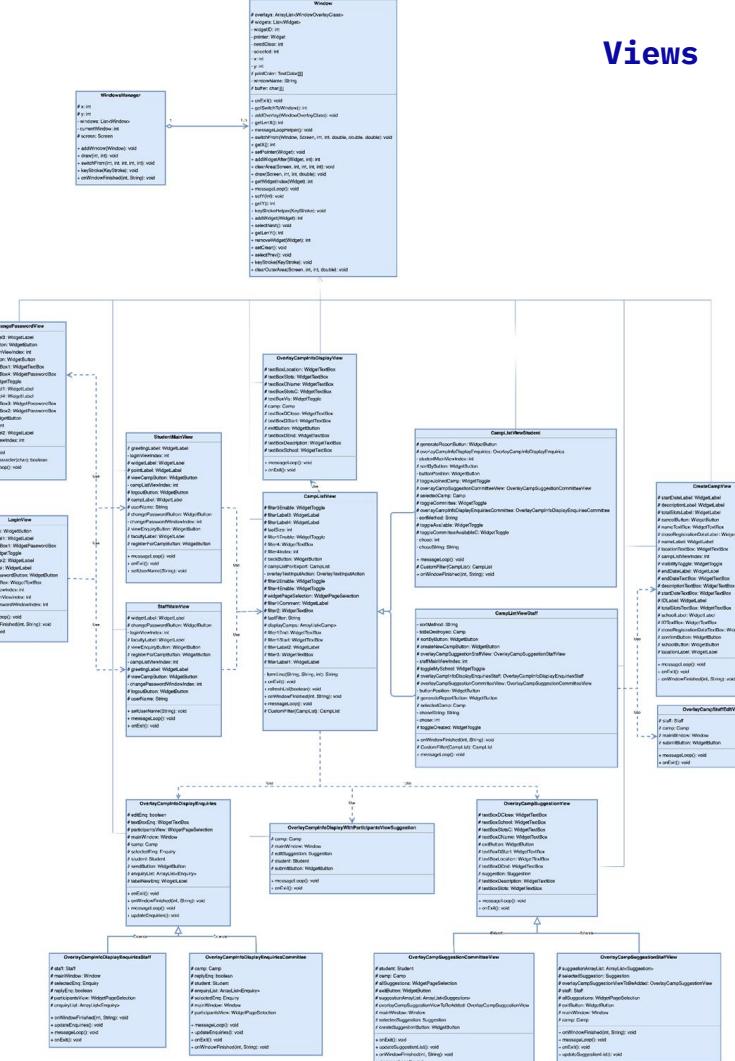
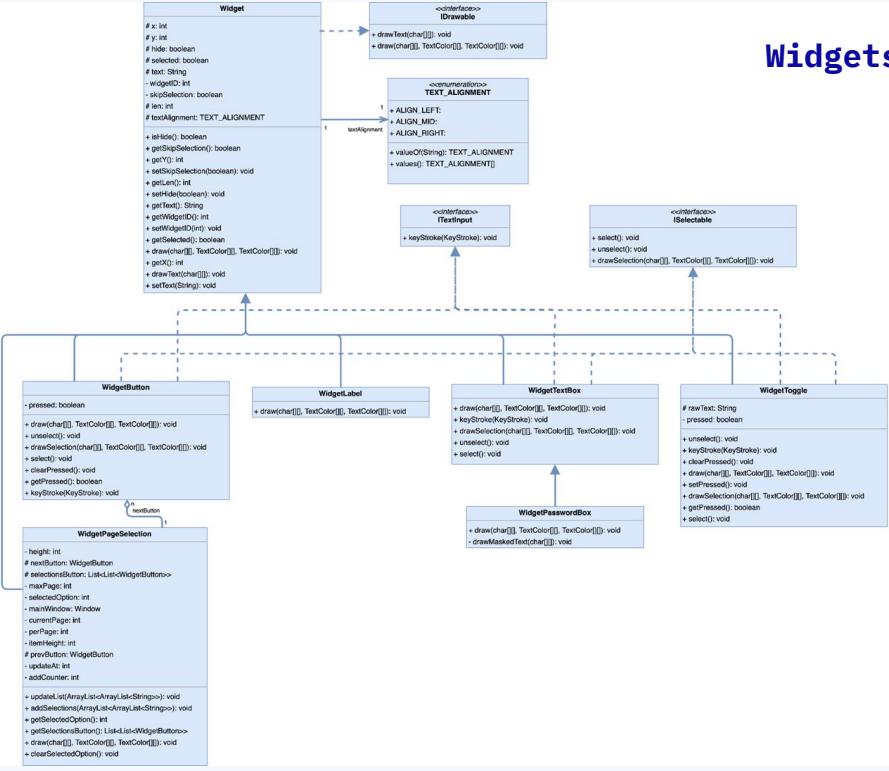
- **Generic Repository Class:** An overall *RepositoryList* abstract class is used to store data and easily perform various filtering and sorting operations.
 - **Configurable Central Repository:** All CSV data are imported and saved together through the *RepositoryCollection* class to ensure persistent data while being easily modifiable by changing the path values in `const/Config.java`.
 - **Factory Design Pattern:** A factory pattern is used to determine if a user is a staff or a student so that proper methods can occur
 - **Extensible and Reusable User Interface (UI):** UI components are created such that they can be reused throughout the entire application.
 - **Entity-Control-Boundary (ECB) Architecture:** Classes are separated into three possible stereotypes (entity, control, and boundary), which provide modularity and separation of concerns.
- 

UML Class Diagram: Entity

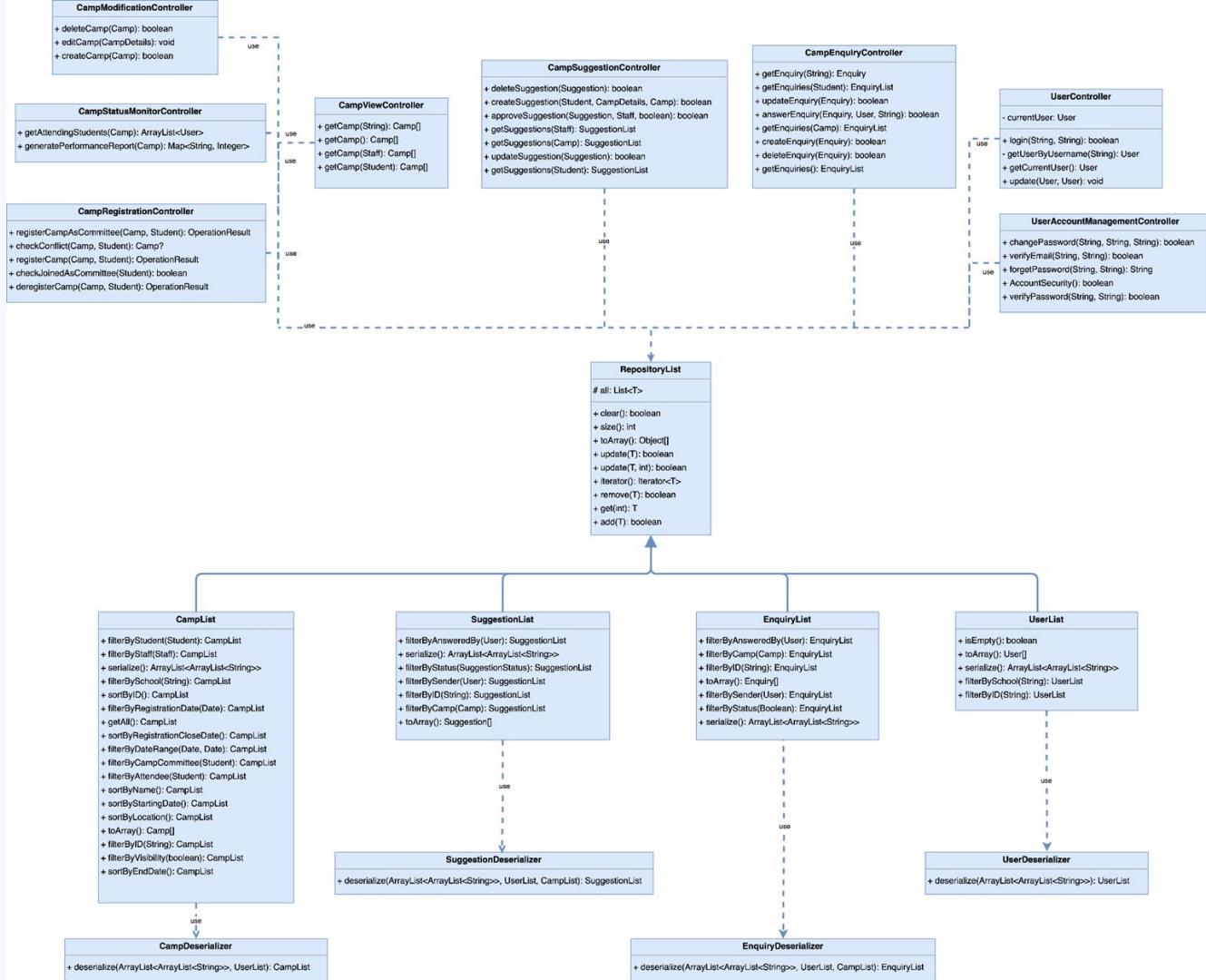


Boundary Diagram

Widgets



UML Class Diagram Controller



03

Demonstration &

Testing



Login

Functionality	✓
1. Cannot Login: invalid credentials	✓
2. Different menu for different users	✓
3. Can change password, re-login to verify effect	✓
4. First login: prompt user to change password, default password: 'password'	✓

Student Main Page

Functionality	✓
1. Change Password	✓
2. Can use filters to view the camp list, default: alphabetical order	✓
3. View available camps based on faculty and visibility & its remaining slots	✓
4. Select camps to register as camp attendee or committee	✓
5. Can't join camp if it's at full capacity	✓

Functionality	✓
6. View/Edit/Submit enquiries for a camp	✓
7. View his/her registered camps & camp role	✓
8. Request to withdraw from camp, cannot rejoin that camp	✓
9. View reply to enquiry	✓

Staff Main Page

Functionality	✓
1. Change Password	✓
2. Create camp from inputting required data	✓
3. Only able to edit camps created by him/her + hide camp	✓
4. View all camps	✓
5. View and Reply Students' Enquiries in camps they created	✓

Functionality	✓
6. View list of students registered for the camp as attendees or committee	✓
7. View Suggestions to camp details from camp committee	✓
8. Accept/Reject Suggestion	✓
9. Generate camp report (student list, camp committee performance report, students' enquiry report)	✓
10. Can use filters to view the camp list, default: alphabetical order	✓

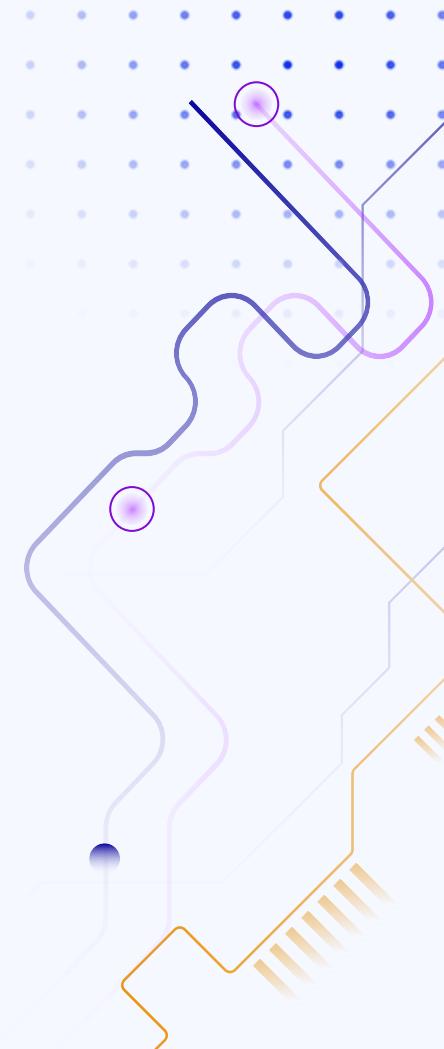
Camp Committee Member Main Page

Functionality	✓
1. Student Main Page functions	✓
2. Submit Suggestions to staff for changes to Camp details + point added	✓
3. View and Reply to Enquiries from Students from Camp he/she oversee + point added	✓

Functionality	✓
4. View/Edit/Delete Suggestions	✓
5. Generate Report for each camp (Student list, Students' Enquiry Report)	✓

04

Reflection



Initial Challenges

- Struggled with an overloaded codebase
- Faced a fragile system where minor changes caused significant issues.
- Difficulty in meeting project requirements effectively.

Impact of Design-Centric Development

- Achieved greater efficiency in workflow.
- Shifted perspective to view design as a foundational element in software development.
- Gained a deeper appreciation for the art of software design.

Future Outlook

- Confidence in design-first methodology for integrating advanced features.
- Anticipate seamless integration of features like password hashing and concurrent user access.
- Expectation of maintaining an adaptable and forward-compatible system.

Thanks!

