NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

SC2002 Object Oriented Programming

Assignment Report

Tutorial Group SCS4 Team 1

Name	Matriculation Number
Hong Sheng Yang	**U2222382B**
Bryan Ng Yuan Sheng	**U2222120E**
Fong Keng Le	**U222351F**
Soh Han Yu Brian	**U2223002H**

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 the 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
Hong Sheng Yang	SC2002	SCS4	[illegible] 11/24/2023
Bryan Ng Sheng Yuan	SC2002	SCS4	[illegible] 11/24/2023
Fong Keng Le	SC2002	SCS4	[illegible] 11/24/2023
Soh Han Yu Brian	SC2002	SCS4	[illegible] 11/24/2023

1. Design Considerations

2.1 Approach

The Camp Application and Management System (CAMs) is an application for staff and students to manage, view and register for camps within NTU. Our system was designed to ensure reusability, maintainability and scalability, therefore the application was split into different packages based on functionality. Since each package has its own core functionality, it allows us to achieve loose coupling and high cohesion. Furthermore, we have implemented a Model-View-Controller (MVC) architectural pattern to support rapid development between our team as any additional modification will not affect the model.

2.2 Assumptions

The following assumptions were made while designing the system. These assumptions were made in addition to the existing assumptions stated in the assignment brief.

1. Staff can create more than 1 camp each.
2. An enquiry can only be replied to once.
3. All filters implemented are not bounded by case sensitivity.
4. All camp names are in sentence case. (the first letter is uppercase, rest is lowercase)
5. Points gained by creating suggestions will be deducted if the camp committee deletes their own suggestions.
6. Staff are not able to change their own camp's details if participants have joined the camp.
7. Approved suggestions will not be updated in the camp's details as it is a proof of concept.
8. Camps with no participant slots cannot be created.
9. Camp slots can be filled by camp committees even without any attendees.
10. Rejecting suggestions will not affect the points of the camp committee member who suggested it.
11. When filtering camps or generating filtered reports, the output will be sorted in alphabetical order.

2.3 SOLID Design Principles

2.3.1 Single Responsibility Principle

SRP states that "a class should only have one cause to change". MVC implements SRP by separating responsibilities. The model classes strictly contain the state information while the view classes contains element that will be shown to the user. The controller classes ensure the proper passing of information between the model and the view classes while the user is interacting with the application.

2.3.2 Open-Closed Principle

The Open-Closed Principle (OCP) states that software should be open for extension but closed for modification. In the reports package, we created an abstract class called GenerateReport that can be extended to create different types of formats like GenerateCsv and Generatetxt. Each child class will override two methods which are generateReport and generateCampCommittee Report. Therefore, we have designed CAMs such that each type of report format inherits from a base abstract GenerateReport class such that they can extend their own required functionalities.

2.3.3 Liskov Substitution Principle

LSP states that the child class must not bring trouble to the parent class and the child class must be able to do what the parent class does. For example, the User class has subclasses such as StudentModel and StaffModel. The arguments passed inside the StaffController and the StudentController class are substitutable for the User class objects while ensuring that the methods behave the same.

2.3.4 Interface Segregation Principle

The Interface Segregation Principle (ISP) states that it is better to have multiple specific smaller interfaces than a few general bigger ones. For instance, we realised that ICampController initially had very "fat" interfaces. So we had to divide those interfaces further into smaller chunks so that CampStaffController and CampStudentController can implement the interfaces as needed. As a result, we can ensure that the different controllers do not have to implement methods that are not related which helps to improve code readability and prevent a potential domino effect when we modify the code in our application.

2.3.5 Dependency Inversion Principle

The Dependency Inversion Principle (DIP) states that higher-level modules should not know what the lower-level module does, lower-level modules should depend on abstraction. DIP is widely implemented in our application. For example, in our appview package, when our AppViewController wants to display a certain view to the user, instead of depending on every menu Class, the controller depends on the IAppView interface. This allows us to add different types of views when needed, which improves the ease of scalability of the system.

2.4 MVC Architectural Pattern

Model-view-Controller is used extensively throughout the design of CAMs. It shows a distinct separation between the software's internal logic and its displayed output.
There are 3 parts to MVC:

1. The Model: The model handles the data operations by providing a method to retrieve and modify data.
2. The View: Handles the display of information.
3. The Controller: The controller takes in the user input, manipulates the data in the model and subsequently updates the view to reflect the changes made. This ensures a seamless interaction between the user and the underlying data.

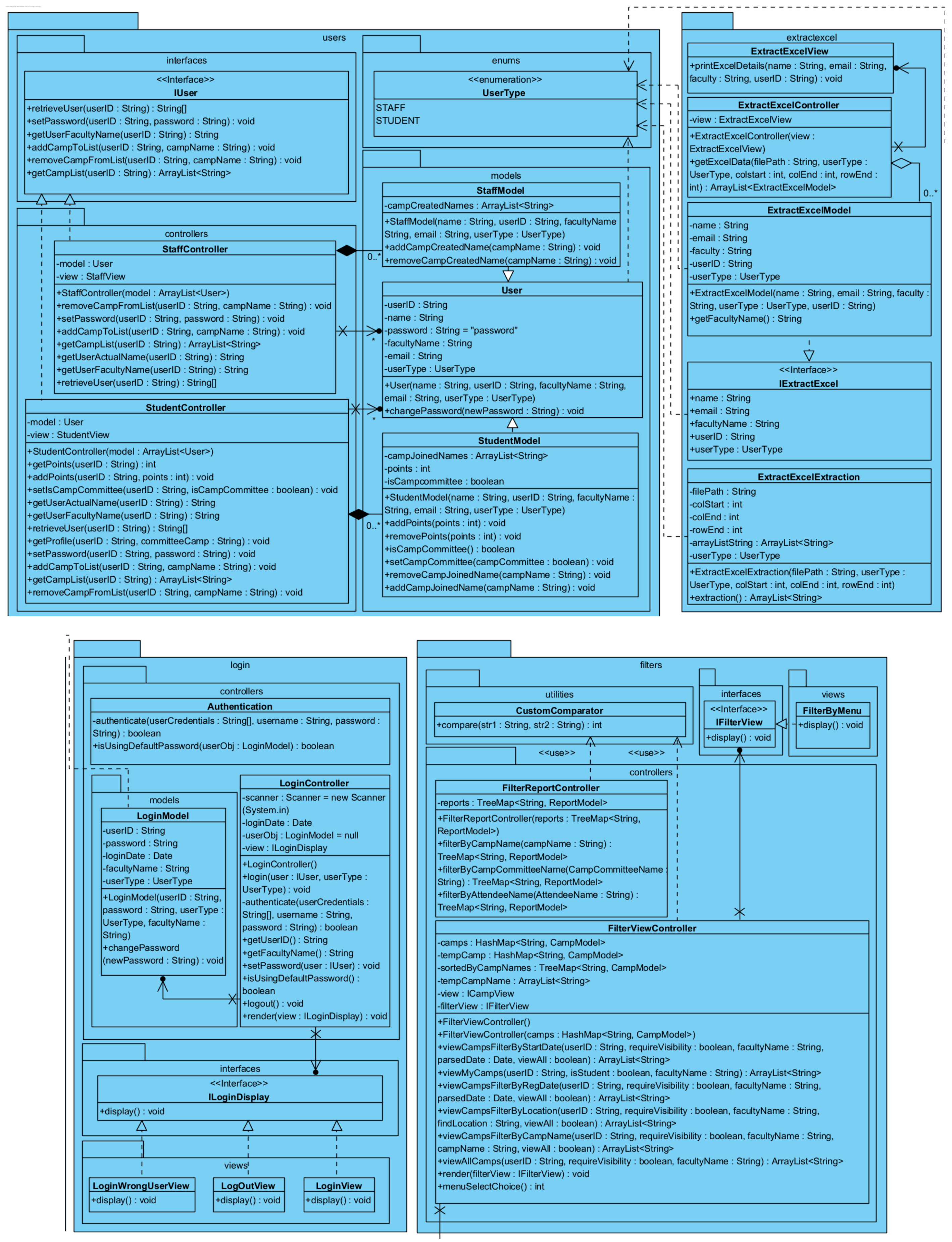
2.5 State Tracking with enumeration

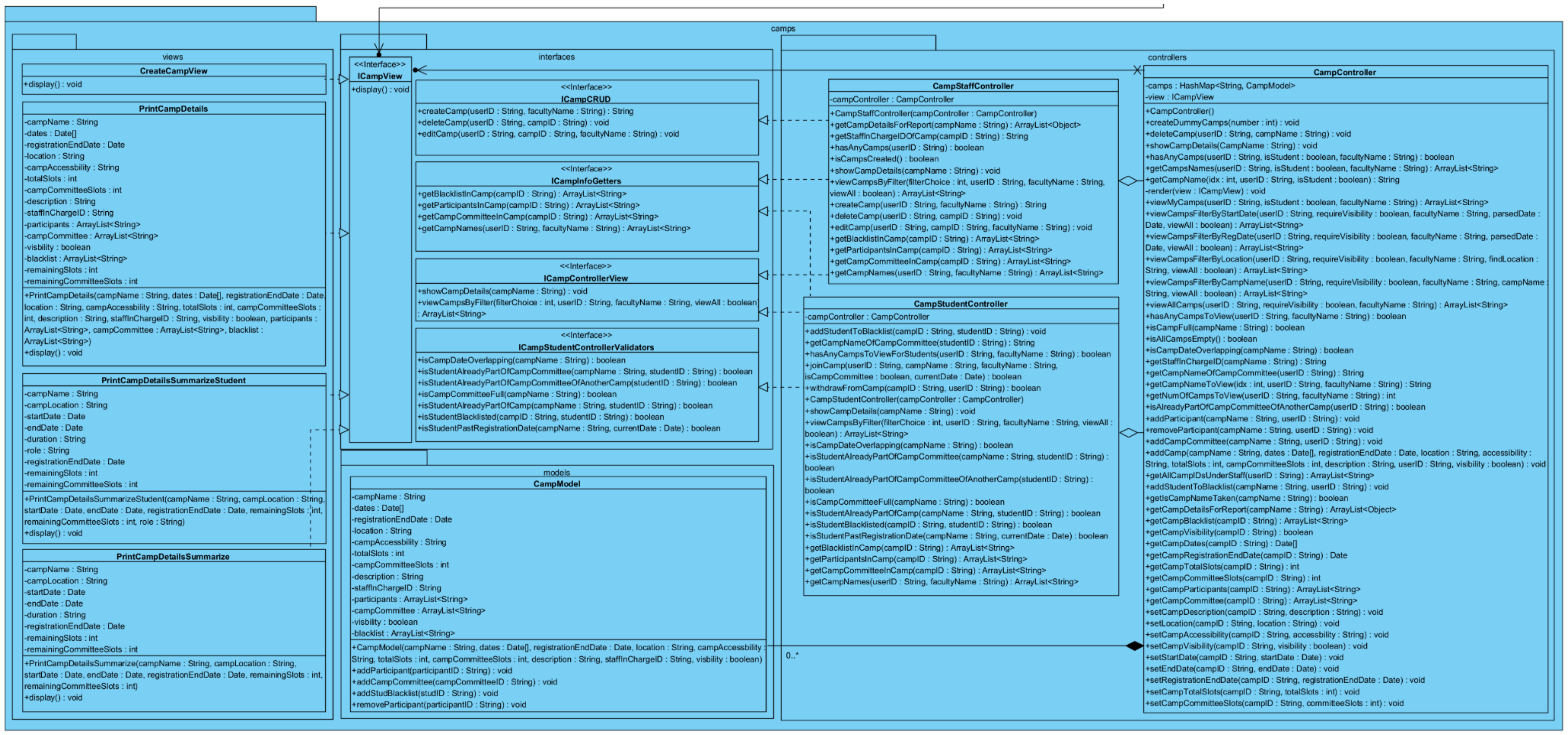
Our application uses an enum to track the states of the enquiries and suggestions, they are:

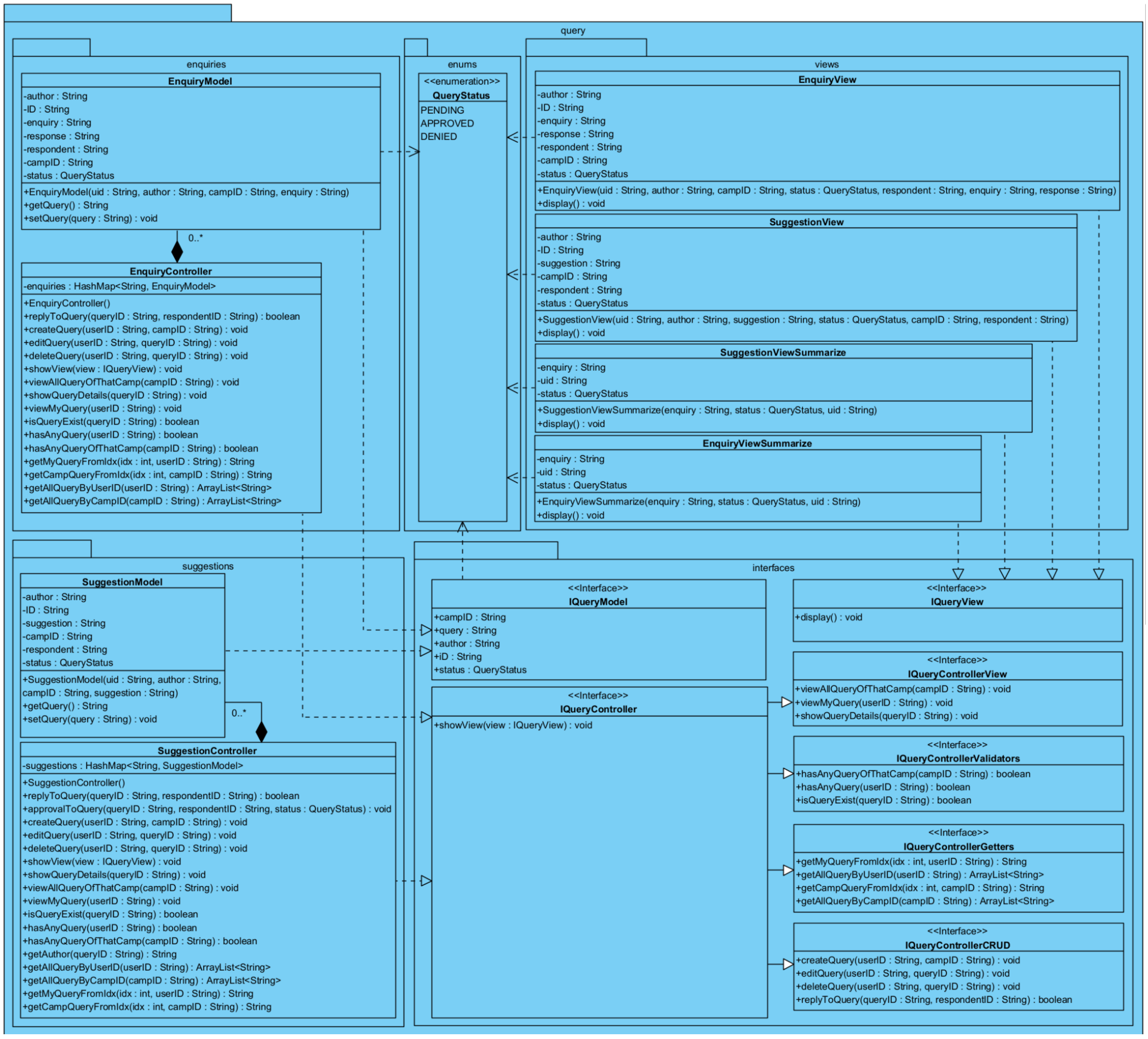
- PENDING
- APPROVED
- DENIED

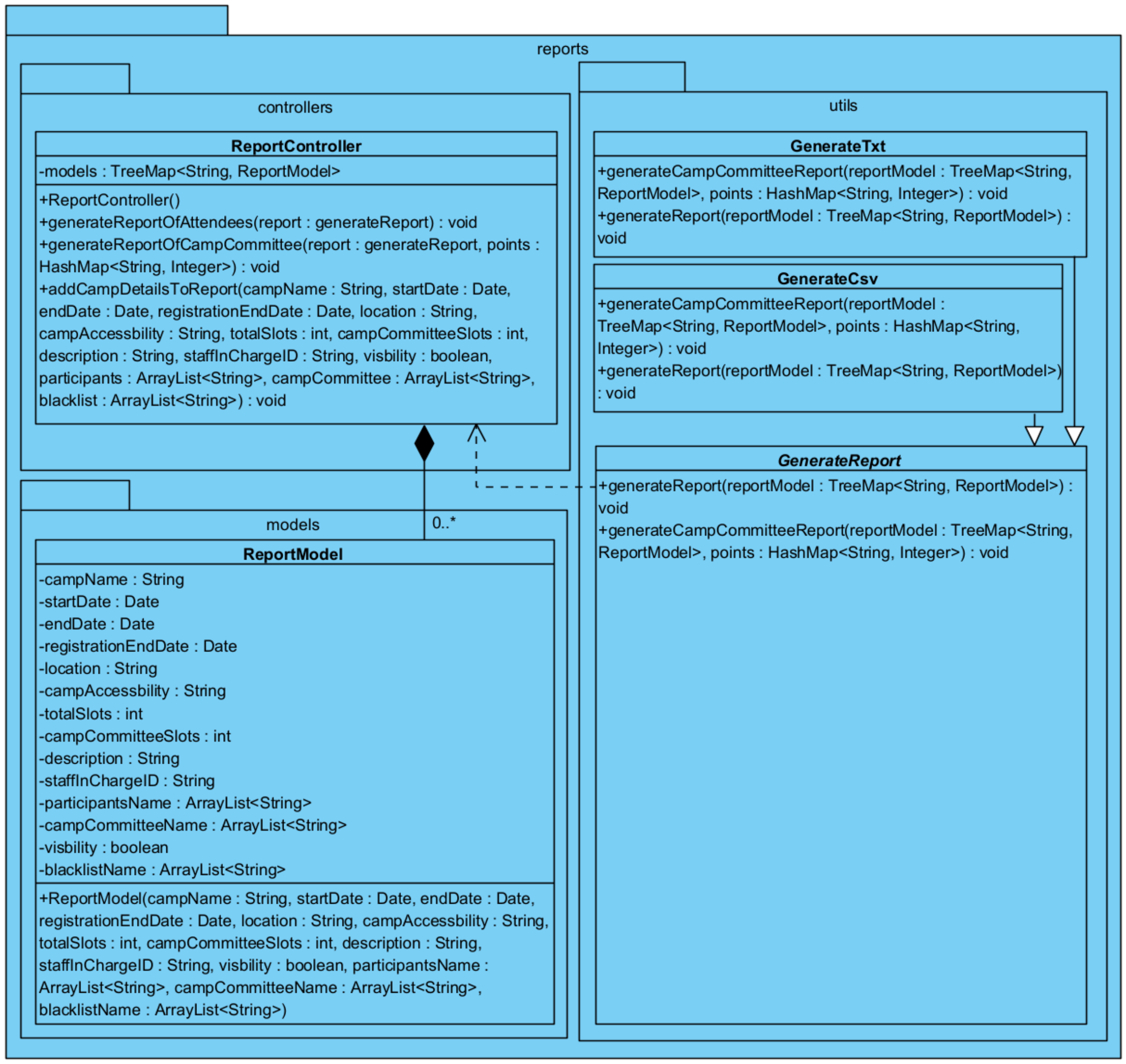
These states are used to constrain the users such that they cannot edit or delete enquiries or suggestions once it has been approved or denied.

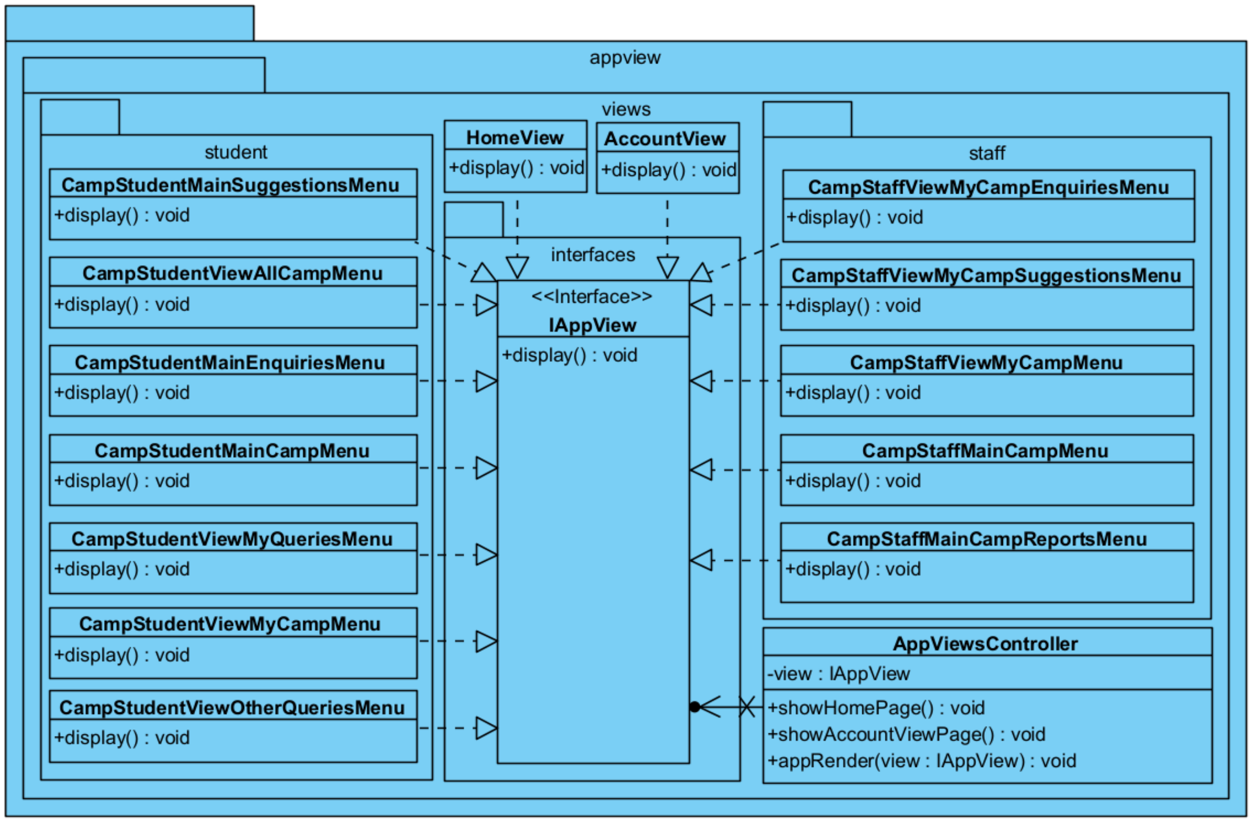
2. Detailed UML Class Diagram


users
interfaces
<<Interface>>
IUser
+retrieveUser(userID : String) : String[]
+setPassword(userID : String, password : String) : void
+getUserFacultyName(userID : String) : String
+addCampToList(userID : String, campName : String) : void
+removeCampFromList(userID : String, campName : String) : void
+getCampList(userID : String) : ArrayList<String>
enums
<<enumeration>>
UserType
STAFF
STUDENT
controllers
StaffController
-model : User
-view : StaffView
+StaffController(model : ArrayList<User>)
+removeCampFromList(userID : String, campName : String) : void
+setPassword(userID : String, password : String) : void
+addCampToList(userID : String, campName : String) : void
+getCampList(userID : String) : ArrayList<String>
+getUserActualName(userID : String) : String
+getUserFacultyName(userID : String) : String
+retrieveUser(userID : String) : String[]
StudentController
-model : User
-view : StudentView
+StudentController(model : ArrayList<User>)
+getPoints(userID : String) : int
+addPoints(userID : String, points : int) : void
+setIsCampCommittee(userID : String, isCampCommittee : boolean) : void
+getUserActualName(userID : String) : String
+getUserFacultyName(userID : String) : String
+retrieveUser(userID : String) : String[]
+getProfile(userID : String, committeeCamp : String) : void
+setPassword(userID : String, password : String) : void
+addCampToList(userID : String, campName : String) : void
+getCampList(userID : String) : ArrayList<String>
+removeCampFromList(userID : String, campName : String) : void
models
StaffModel
-campCreatedNames : ArrayList<String>
+StaffModel(name : String, userID : String, facultyName : String, email : String, userType : UserType)
+addCampCreatedName(campName : String) : void
+removeCampCreatedName(campName : String) : void
0..*
User
-userID : String
-name : String
-password : String = "password"
-facultyName : String
-email : String
-userType : UserType
+User(name : String, userID : String, facultyName : String, email : String, userType : UserType)
+changePassword(newPassword : String) : void
*
*
StudentModel
-campJoinedNames : ArrayList<String>
-points : int
-isCampcommittee : boolean
+StudentModel(name : String, userID : String, facultyName : String, email : String, userType : UserType)
+addPoints(points : int) : void
+removePoints(points : int) : void
+isCampCommittee() : boolean
+setCampCommittee(campCommittee : boolean) : void
+removeCampJoinedName(campName : String) : void
+addCampJoinedName(campName : String) : void
0..*
extractexcel
ExtractExcelView
+printExcelDetails(name : String, email : String, faculty : String, userID : String) : void
ExtractExcelController
-view : ExtractExcelView
+ExtractExcelController(view : ExtractExcelView)
+getExcelData(filePath : String, userType : UserType, colstart : int, colEnd : int, rowEnd : int) : ArrayList<ExtractExcelModel>
0..*
ExtractExcelModel
-name : String
-email : String
-faculty : String
-userID : String
-userType : UserType
+ExtractExcelModel(name : String, email : String, faculty : String, userType : UserType, userID : String)
+getFacultyName() : String
<<Interface>>
IExtractExcel
+name : String
+email : String
+facultyName : String
+userID : String
+userType : UserType
ExtractExcelExtraction
-filePath : String
-colStart : int
-colEnd : int
-rowEnd : int
-arrayListString : ArrayList<String>
-userType : UserType
+ExtractExcelExtraction(filePath : String, userType : UserType, colStart : int, colEnd : int, rowEnd : int)
+extraction() : ArrayList<String>
login
controllers
Authentication
-authenticate(userCredentials : String[], username : String, password : String) : boolean
+isUsingDefaultPassword(userObj : LoginModel) : boolean
models
LoginModel
-userID : String
-password : String
-loginDate : Date
-facultyName : String
-userType : UserType
+LoginModel(userID : String, password : String, userType : UserType, facultyName : String)
+changePassword (newPassword : String) : void
LoginController
-scanner : Scanner = new Scanner (System.in)
-loginDate : Date
-userObj : LoginModel = null
-view : ILoginDisplay
+LoginController()
+login(user : IUser, userType : UserType) : void
-authenticate(userCredentials : String[], username : String, password : String) : boolean
+getUserID() : String
+getFacultyName() : String
+setPassword(user : IUser) : void
+isUsingDefaultPassword() : boolean
+logout() : void
+render(view : ILoginDisplay) : void
interfaces
<<Interface>>
ILoginDisplay
+display() : void
views
LoginWrongUserView
+display() : void
LogOutView
+display() : void
LoginView
+display() : void
filters
utilities
CustomComparator
+compare(str1 : String, str2 : String) : int
interfaces
<<Interface>>
IFilterView
+display() : void
views
FilterByMenu
+display() : void
<<use>>
<<use>>
controllers
FilterReportController
-reports : TreeMap<String, ReportModel>
+FilterReportController(reports : TreeMap<String, ReportModel>)
+filterByCampName(campName : String) : TreeMap<String, ReportModel>
+filterByCampCommitteeName(CampCommitteeName : String) : TreeMap<String, ReportModel>
+filterByAttendeeName(AttendeeName : String) : TreeMap<String, ReportModel>
FilterViewController
-camps : HashMap<String, CampModel>
-tempCamp : HashMap<String, CampModel>
-sortedByCampNames : TreeMap<String, CampModel>
-tempCampName : ArrayList<String>
-view : ICampView
-filterView : IFilterView
+FilterViewController()
+FilterViewController(camps : HashMap<String, CampModel>)
+viewCampsFilterByStartDate(userID : String, requireVisibility : boolean, facultyName : String, parsedDate : Date, viewAll : boolean) : ArrayList<String>
+viewMyCamps(userID : String, isStudent : boolean, facultyName : String) : ArrayList<String>
+viewCampsFilterByRegDate(userID : String, requireVisibility : boolean, facultyName : String, parsedDate : Date, viewAll : boolean) : ArrayList<String>
+viewCampsFilterByLocation(userID : String, requireVisibility : boolean, facultyName : String, findLocation : String, viewAll : boolean) : ArrayList<String>
+viewCampsFilterByCampName(userID : String, requireVisibility : boolean, facultyName : String, campName : String, viewAll : boolean) : ArrayList<String>
+viewAllCamps(userID : String, requireVisibility : boolean, facultyName : String) : ArrayList<String>
+render(filterView : IFilterView) : void
+menuSelectChoice() : int

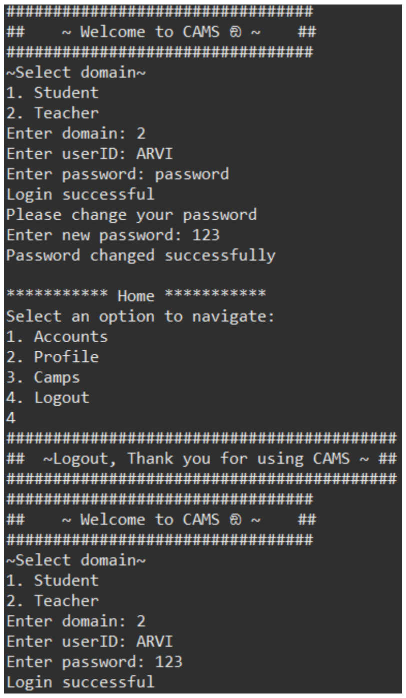
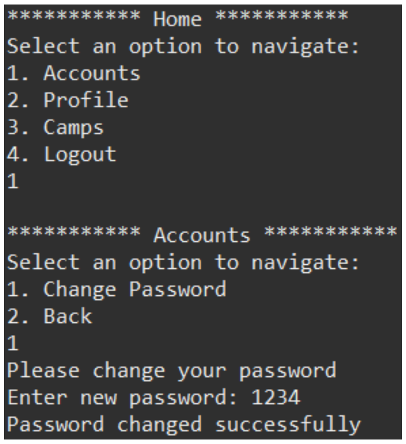
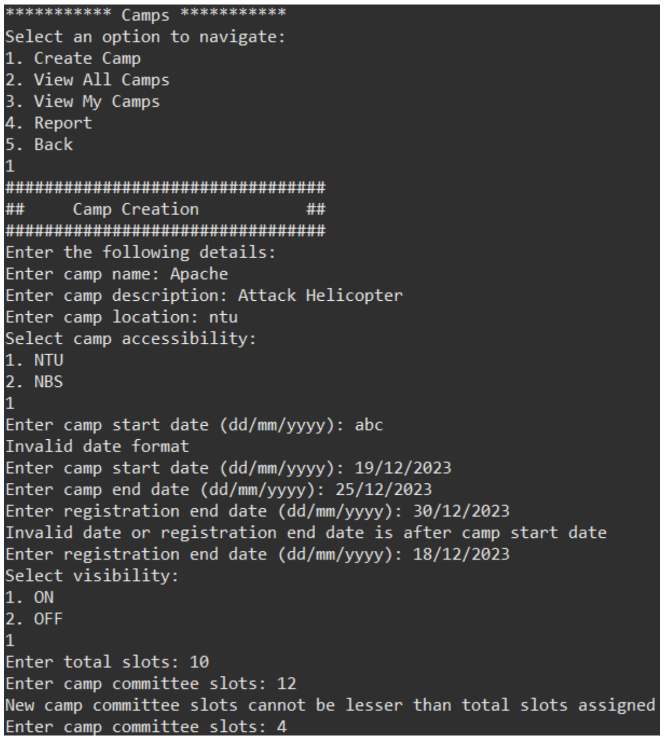

camps
views
CreateCampView
+display() : void
PrintCampDetails
-campName : String
-dates : Date[]
-registrationEndDate : Date
-location : String
-campAccessbility : String
-totalSlots : int
-campCommitteeSlots : int
-description : String
-staffInChargeID : String
-participants : ArrayList<String>
-campCommittee : ArrayList<String>
-visbility : boolean
-blacklist : ArrayList<String>
-remainingSlots : int
-remainingCommitteeSlots : int
+PrintCampDetails(campName : String, dates : Date[], registrationEndDate : Date, location : String, campAccessbility : String, totalSlots : int, campCommitteeSlots : int, description : String, staffInChargeID : String, visbility : boolean, participants : ArrayList<String>, campCommittee : ArrayList<String>, blacklist : ArrayList<String>)
+display() : void
PrintCampDetailsSummarizeStudent
-campName : String
-campLocation : String
-startDate : Date
-endDate : Date
-duration : String
-role : String
-registrationEndDate : Date
-remainingSlots : int
-remainingCommitteeSlots : int
+PrintCampDetailsSummarizeStudent(campName : String, campLocation : String, startDate : Date, endDate : Date, registrationEndDate : Date, remainingSlots : int, remainingCommitteeSlots : int, role : String)
+display() : void
PrintCampDetailsSummarize
-campName : String
-campLocation : String
-startDate : Date
-endDate : Date
-duration : String
-registrationEndDate : Date
-remainingSlots : int
-remainingCommitteeSlots : int
+PrintCampDetailsSummarize(campName : String, campLocation : String, startDate : Date, endDate : Date, registrationEndDate : Date, remainingSlots : int, remainingCommitteeSlots : int)
+display() : void
interfaces
<<Interface>>
ICampView
+display() : void
<<Interface>>
ICampCRUD
+createCamp(userID : String, facultyName : String) : String
+deleteCamp(userID : String, campID : String) : void
+editCamp(userID : String, campID : String, facultyName : String) : void
<<Interface>>
ICampInfoGetters
+getBlacklistInCamp(campID : String) : ArrayList<String>
+getParticipantsInCamp(campID : String) : ArrayList<String>
+getCampCommitteeInCamp(campID : String) : ArrayList<String>
+getCampNames(userID : String, facultyName : String) : ArrayList<String>
<<Interface>>
ICampControllerView
+showCampDetails(campName : String) : void
+viewCampsByFilter(filterChoice : int, userID : String, facultyName : String, viewAll : boolean) : ArrayList<String>
<<Interface>>
ICampStudentControllerValidators
+isCampDateOverlapping(campName : String) : boolean
+isStudentAlreadyPartOfCampCommittee(campName : String, studentID : String) : boolean
+isStudentAlreadyPartOfCampCommitteeOfAnotherCamp(studentID : String) : boolean
+isCampCommitteeFull(campName : String) : boolean
+isStudentAlreadyPartOfCamp(campName : String, studentID : String) : boolean
+isStudentBlacklisted(campID : String, studentID : String) : boolean
+isStudentPastRegistrationDate(campName : String, currentDate : Date) : boolean
models
CampModel
-campName : String
-dates : Date[]
-registrationEndDate : Date
-location : String
-campAccessbility : String
-totalSlots : int
-campCommitteeSlots : int
-description : String
-staffInChargeID : String
-participants : ArrayList<String>
-campCommittee : ArrayList<String>
-visbility : boolean
-blacklist : ArrayList<String>
+CampModel(campName : String, dates : Date[], registrationEndDate : Date, location : String, campAccessbility : String, totalSlots : int, campCommitteeSlots : int, description : String, staffInChargeID : String, visbility : boolean)
+addParticipant(participantID : String) : void
+addCampCommittee(campCommitteeID : String) : void
+addStudBlacklist(studID : String) : void
+removeParticipant(participantID : String) : void
0..*
controllers
CampStaffController
-campController : CampController
+CampStaffController(campController : CampController)
+getCampDetailsForReport(campName : String) : ArrayList<Object>
+getStaffInChargeIDOfCamp(campID : String) : String
+hasAnyCamps(userID : String) : boolean
+isCampsCreated() : boolean
+showCampDetails(campName : String) : void
+viewCampsByFilter(filterChoice : int, userID : String, facultyName : String, viewAll : boolean) : ArrayList<String>
+createCamp(userID : String, facultyName : String) : String
+deleteCamp(userID : String, campID : String) : void
+editCamp(userID : String, campID : String, facultyName : String) : void
+getBlacklistInCamp(campID : String) : ArrayList<String>
+getParticipantsInCamp(campID : String) : ArrayList<String>
+getCampCommitteeInCamp(campID : String) : ArrayList<String>
+getCampNames(userID : String, facultyName : String) : ArrayList<String>
CampStudentController
-campController : CampController
+addStudentToBlacklist(campID : String, studentID : String) : void
+getCampNameOfCampCommittee(studentID : String) : String
+hasAnyCampsToViewForStudents(userID : String, facultyName : String) : boolean
+joinCamp(userID : String, campName : String, facultyName : String, isCampCommittee : boolean, currentDate : Date) : boolean
+withdrawFromCamp(campID : String, userID : String) : boolean
+CampStudentController(campController : CampController)
+showCampDetails(campName : String) : void
+viewCampsByFilter(filterChoice : int, userID : String, facultyName : String, viewAll : boolean) : ArrayList<String>
+isCampDateOverlapping(campName : String) : boolean
+isStudentAlreadyPartOfCampCommittee(campName : String, studentID : String) : boolean
+isStudentAlreadyPartOfCampCommitteeOfAnotherCamp(studentID : String) : boolean
+isCampCommitteeFull(campName : String) : boolean
+isStudentAlreadyPartOfCamp(campName : String, studentID : String) : boolean
+isStudentBlacklisted(campID : String, studentID : String) : boolean
+isStudentPastRegistrationDate(campName : String, currentDate : Date) : boolean
+getBlacklistInCamp(campID : String) : ArrayList<String>
+getParticipantsInCamp(campID : String) : ArrayList<String>
+getCampCommitteeInCamp(campID : String) : ArrayList<String>
+getCampNames(userID : String, facultyName : String) : ArrayList<String>
CampController
-camps : HashMap<String, CampModel>
-view : ICampView
+CampController()
+createDummyCamps(number : int) : void
+deleteCamp(userID : String, campName : String) : void
+showCampDetails(CampName : String) : void
+hasAnyCamps(userID : String, isStudent : boolean, facultyName : String) : boolean
+getCampsNames(userID : String, isStudent : boolean, facultyName : String) : ArrayList<String>
+getCampName(idx : int, userID : String, isStudent : boolean) : String
-render(view : ICampView) : void
+viewMyCamps(userID : String, isStudent : boolean, facultyName : String) : ArrayList<String>
+viewCampsFilterByStartDate(userID : String, requireVisibility : boolean, facultyName : String, parsedDate : Date, viewAll : boolean) : ArrayList<String>
+viewCampsFilterByRegDate(userID : String, requireVisibility : boolean, facultyName : String, parsedDate : Date, viewAll : boolean) : ArrayList<String>
+viewCampsFilterByLocation(userID : String, requireVisibility : boolean, facultyName : String, findLocation : String, viewAll : boolean) : ArrayList<String>
+viewCampsFilterByCampName(userID : String, requireVisibility : boolean, facultyName : String, campName : String, viewAll : boolean) : ArrayList<String>
+viewAllCamps(userID : String, requireVisibility : boolean, facultyName : String) : ArrayList<String>
+hasAnyCampsToView(userID : String, facultyName : String) : boolean
+isCampFull(campName : String) : boolean
+isAllCampsEmpty() : boolean
+isCampDateOverlapping(campName : String) : boolean
+getStaffInChargeID(campName : String) : String
+getCampNameOfCampCommittee(userID : String) : String
+getCampNameToView(idx : int, userID : String, facultyName : String) : String
+getNumOfCampsToView(userID : String, facultyName : String) : int
+isAlreadyPartOfCampCommitteeOfAnotherCamp(userID : String) : boolean
+addParticipant(campName : String, userID : String) : void
+removeParticipant(campName : String, userID : String) : void
+addCampCommittee(campName : String, userID : String) : void
+addCamp(campName : String, dates : Date[], registrationEndDate : Date, location : String, accessibility : String, totalSlots : int, campCommitteeSlots : int, description : String, userID : String, visibility : boolean) : void
+getAllCampIDsUnderStaff(userID : String) : ArrayList<String>
+addStudentToBlacklist(campName : String, userID : String) : void
+getIsCampNameTaken(campName : String) : boolean
+getCampDetailsForReport(campName : String) : ArrayList<Object>
+getCampBlacklist(campID : String) : ArrayList<String>
+getCampVisibility(campID : String) : boolean
+getCampDates(campID : String) : Date[]
+getCampRegistrationEndDate(campID : String) : Date
+getCampTotalSlots(campID : String) : int
+getCampCommitteeSlots(campID : String) : int
+getCampParticipants(campID : String) : ArrayList<String>
+getCampCommittee(campID : String) : ArrayList<String>
+setCampDescription(campID : String, description : String) : void
+setLocation(campID : String, location : String) : void
+setCampAccessibility(campID : String, accessibility : String) : void
+setCampVisibility(campID : String, visibility : boolean) : void
+setStartDate(campID : String, startDate : Date) : void
+setEndDate(campID : String, endDate : Date) : void
+setRegistrationEndDate(campID : String, registrationEndDate : Date) : void
+setCampTotalSlots(campID : String, totalSlots : int) : void
+setCampCommitteeSlots(campID : String, commiteeSlots : int) : void


query
enquiries
EnquiryModel
-author : String
-ID : String
-enquiry : String
-response : String
-respondent : String
-campID : String
-status : QueryStatus
+EnquiryModel(uid : String, author : String, campID : String, enquiry : String)
+getQuery() : String
+setQuery(query : String) : void
0..*
EnquiryController
-enquiries : HashMap<String, EnquiryModel>
+EnquiryController()
+replyToQuery(queryID : String, respondentID : String) : boolean
+createQuery(userID : String, campID : String) : void
+editQuery(userID : String, queryID : String) : void
+deleteQuery(userID : String, queryID : String) : void
+showView(view : IQueryView) : void
+viewAllQueryOfThatCamp(campID : String) : void
+showQueryDetails(queryID : String) : void
+viewMyQuery(userID : String) : void
+isQueryExist(queryID : String) : boolean
+hasAnyQuery(userID : String) : boolean
+hasAnyQueryOfThatCamp(campID : String) : boolean
+getMyQueryFromIdx(idx : int, userID : String) : String
+getCampQueryFromIdx(idx : int, campID : String) : String
+getAllQueryByUserID(userID : String) : ArrayList<String>
+getAllQueryByCampID(campID : String) : ArrayList<String>
enums
<<enumeration>>
QueryStatus
PENDING
APPROVED
DENIED
views
EnquiryView
-author : String
-ID : String
-enquiry : String
-response : String
-respondent : String
-campID : String
-status : QueryStatus
+EnquiryView(uid : String, author : String, campID : String, status : QueryStatus, respondent : String, enquiry : String, response : String)
+display() : void
SuggestionView
-author : String
-ID : String
-suggestion : String
-campID : String
-respondent : String
-status : QueryStatus
+SuggestionView(uid : String, author : String, suggestion : String, status : QueryStatus, campID : String, respondent : String)
+display() : void
SuggestionViewSummarize
-enquiry : String
-uid : String
-status : QueryStatus
+SuggestionViewSummarize(enquiry : String, status : QueryStatus, uid : String)
+display() : void
EnquiryViewSummarize
-enquiry : String
-uid : String
-status : QueryStatus
+EnquiryViewSummarize(enquiry : String, status : QueryStatus, uid : String)
+display() : void
suggestions
SuggestionModel
-author : String
-ID : String
-suggestion : String
-campID : String
-respondent : String
-status : QueryStatus
+SuggestionModel(uid : String, author : String, campID : String, suggestion : String)
+getQuery() : String
+setQuery(query : String) : void
0..*
SuggestionController
-suggestions : HashMap<String, SuggestionModel>
+SuggestionController()
+replyToQuery(queryID : String, respondentID : String) : boolean
+approvalToQuery(queryID : String, respondentID : String, status : QueryStatus) : void
+createQuery(userID : String, campID : String) : void
+editQuery(userID : String, queryID : String) : void
+deleteQuery(userID : String, queryID : String) : void
+showView(view : IQueryView) : void
+showQueryDetails(queryID : String) : void
+viewAllQueryOfThatCamp(campID : String) : void
+viewMyQuery(userID : String) : void
+isQueryExist(queryID : String) : boolean
+hasAnyQuery(userID : String) : boolean
+hasAnyQueryOfThatCamp(campID : String) : boolean
+getAuthor(queryID : String) : String
+getAllQueryByUserID(userID : String) : ArrayList<String>
+getAllQueryByCampID(campID : String) : ArrayList<String>
+getMyQueryFromIdx(idx : int, userID : String) : String
+getCampQueryFromIdx(idx : int, campID : String) : String
interfaces
<<Interface>>
IQueryModel
+campID : String
+query : String
+author : String
+iD : String
+status : QueryStatus
<<Interface>>
IQueryController
+showView(view : IQueryView) : void
<<Interface>>
IQueryView
+display() : void
<<Interface>>
IQueryControllerView
+viewAllQueryOfThatCamp(campID : String) : void
+viewMyQuery(userID : String) : void
+showQueryDetails(queryID : String) : void
<<Interface>>
IQueryControllerValidators
+hasAnyQueryOfThatCamp(campID : String) : boolean
+hasAnyQuery(userID : String) : boolean
+isQueryExist(queryID : String) : boolean
<<Interface>>
IQueryControllerGetters
+getMyQueryFromIdx(idx : int, userID : String) : String
+getAllQueryByUserID(userID : String) : ArrayList<String>
+getCampQueryFromIdx(idx : int, campID : String) : String
+getAllQueryByCampID(campID : String) : ArrayList<String>
<<Interface>>
IQueryControllerCRUD
+createQuery(userID : String, campID : String) : void
+editQuery(userID : String, queryID : String) : void
+deleteQuery(userID : String, queryID : String) : void
+replyToQuery(queryID : String, respondentID : String) : boolean


reports
controllers
ReportController
-models : TreeMap<String, ReportModel>
+ReportController()
+generateReportOfAttendees(report : generateReport) : void
+generateReportOfCampCommittee(report : generateReport, points : HashMap<String, Integer>) : void
+addCampDetailsToReport(campName : String, startDate : Date, endDate : Date, registrationEndDate : Date, location : String, campAccessbility : String, totalSlots : int, campCommitteeSlots : int, description : String, staffInChargeID : String, visbility : boolean, participants : ArrayList<String>, campCommittee : ArrayList<String>, blacklist : ArrayList<String>) : void
models
0..*
ReportModel
-campName : String
-startDate : Date
-endDate : Date
-registrationEndDate : Date
-location : String
-campAccessbility : String
-totalSlots : int
-campCommitteeSlots : int
-description : String
-staffInChargeID : String
-participantsName : ArrayList<String>
-campCommitteeName : ArrayList<String>
-visbility : boolean
-blacklistName : ArrayList<String>
+ReportModel(campName : String, startDate : Date, endDate : Date, registrationEndDate : Date, location : String, campAccessbility : String, totalSlots : int, campCommitteeSlots : int, description : String, staffInChargeID : String, visbility : boolean, participantsName : ArrayList<String>, campCommitteeName : ArrayList<String>, blacklistName : ArrayList<String>)
utils
GenerateTxt
+generateCampCommitteeReport(reportModel : TreeMap<String, ReportModel>, points : HashMap<String, Integer>) : void
+generateReport(reportModel : TreeMap<String, ReportModel>) : void
GenerateCsv
+generateCampCommitteeReport(reportModel : TreeMap<String, ReportModel>, points : HashMap<String, Integer>) : void
+generateReport(reportModel : TreeMap<String, ReportModel>) : void
GenerateReport
+generateReport(reportModel : TreeMap<String, ReportModel>) : void
+generateCampCommitteeReport(reportModel : TreeMap<String, ReportModel>, points : HashMap<String, Integer>) : void


appview
views
student
CampStudentMainSuggestionsMenu
+display() : void
CampStudentViewAllCampMenu
+display() : void
CampStudentMainEnquiriesMenu
+display() : void
CampStudentMainCampMenu
+display() : void
CampStudentViewMyQueriesMenu
+display() : void
CampStudentViewMyCampMenu
+display() : void
CampStudentViewOtherQueriesMenu
+display() : void
HomeView
+display() : void
AccountView
+display() : void
interfaces
<<Interface>>
IAppView
+display() : void
staff
CampStaffViewMyCampEnquiriesMenu
+display() : void
CampStaffViewMyCampSuggestionsMenu
+display() : void
CampStaffViewMyCampMenu
+display() : void
CampStaffMainCampMenu
+display() : void
CampStaffMainCampReportsMenu
+display() : void
AppViewsController
-view : IAppView
+showHomePage() : void
+showAccountViewPage() : void
+appRender(view : IAppView) : void

3. Testing

Test Case	Expected Output	Results/Implementation
Login	Users can log in to their respective domains. Able to prompt change password after login. Able to change password from menu	 
Creating Camps	Staff creates camp with details: 1. Camp Name (Camp name are unique) 2. Description 3. Location 4. Accessibility (School or own Faculty) 5. Start date 6. End date 7. Registration end date 8. Student Visibility 9. Total slots 10. Camp Committee slots Validates, if entered Registration End Date, is before Start Date Validate if camp committee slot ＜total slots	

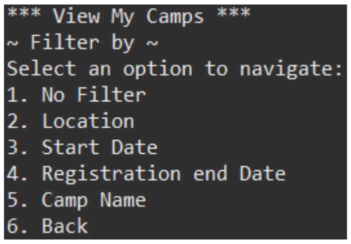
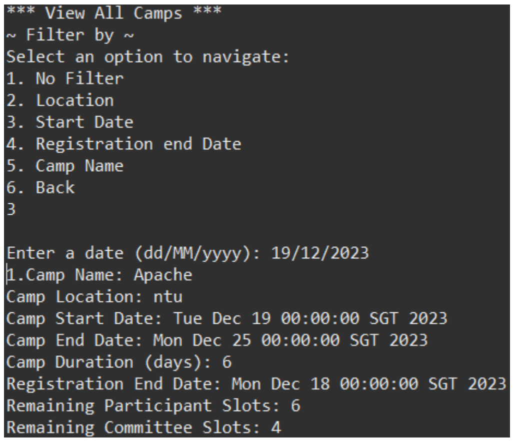
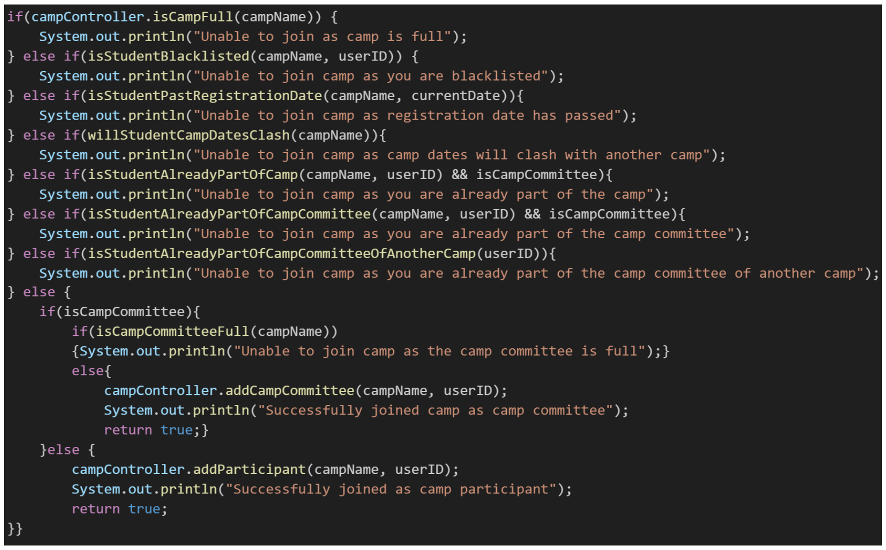
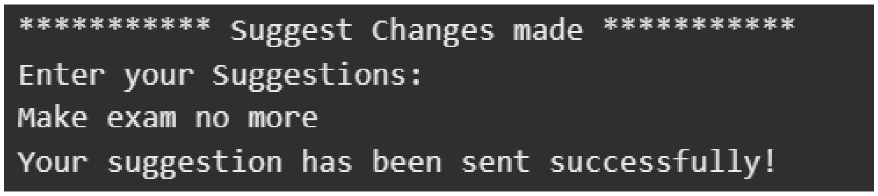
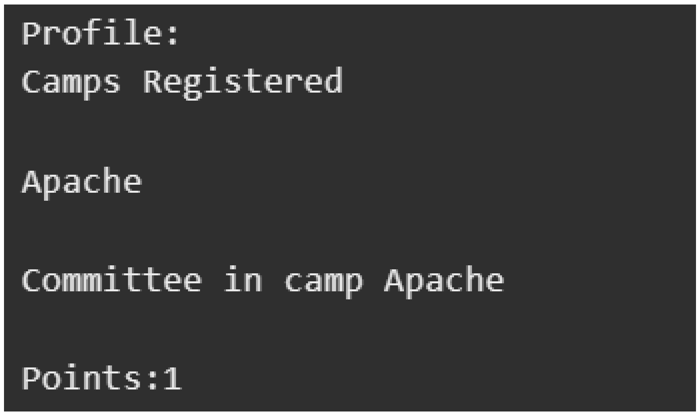
```
#################################
##    ~ Welcome to CAMS ® ~    ##
#################################
~Select domain~
1. Student
2. Teacher
Enter domain: 2
Enter userID: ARVI
Enter password: password
Login successful
Please change your password
Enter new password: 123
Password changed successfully

*********** Home ***********
Select an option to navigate:
1. Accounts
2. Profile
3. Camps
4. Logout
4
#########################################
##  ~Logout, Thank you for using CAMS ~ ##
#########################################
#################################
##    ~ Welcome to CAMS ® ~    ##
#################################
~Select domain~
1. Student
2. Teacher
Enter domain: 2
Enter userID: ARVI
Enter password: 123
Login successful
```

```
*********** Home ***********
Select an option to navigate:
1. Accounts
2. Profile
3. Camps
4. Logout
1

*********** Accounts ***********
Select an option to navigate:
1. Change Password
2. Back
1
Please change your password
Enter new password: 1234
Password changed successfully
```

```
*********** Camps ***********
Select an option to navigate:
1. Create Camp
2. View All Camps
3. View My Camps
4. Report
5. Back
1
#################################
##     Camp Creation           ##
#################################
Enter the following details:
Enter camp name: Apache
Enter camp description: Attack Helicopter
Enter camp location: ntu
Select camp accessibility:
1. NTU
2. NBS
1
Enter camp start date (dd/mm/yyyy): abc
Invalid date format
Enter camp start date (dd/mm/yyyy): 19/12/2023
Enter camp end date (dd/mm/yyyy): 25/12/2023
Enter registration end date (dd/mm/yyyy): 30/12/2023
Invalid date or registration end date is after camp start date
Enter registration end date (dd/mm/yyyy): 18/12/2023
Select visibility:
1. ON
2. OFF
1
Enter total slots: 10
Enter camp committee slots: 12
New camp committee slots cannot be lesser than total slots assigned
Enter camp committee slots: 4
```

Filtering Camps	Able to filter by: 1. No filter (view all) 2. Location 3. Start date 4. Registration end date 5. Camp Name Filter by Location and Camp Name is case-insensitive Filtering dates is by day	 
Joining Camps	For a student to join camp: 1. Slots are available 2. Not Blacklisted 3. Before Registration Date 4. No clash with other camps 5. Not a camp committee of the camp 6. Not an attendee of the camp	
Suggestion	Camp committee can send a suggestion 1 point is gained when a suggestion is sent and approved 1 point is deducted when deleting to prevent unlimited points Suggestion approved/rejected Can no longer be deleted	 

```
*** View My Camps ***
~ Filter by ~
Select an option to navigate:
1. No Filter
2. Location
3. Start Date
4. Registration end Date
5. Camp Name
6. Back
```

```
*** View All Camps ***
~ Filter by ~
Select an option to navigate:
1. No Filter
2. Location
3. Start Date
4. Registration end Date
5. Camp Name
6. Back
3

Enter a date (dd/MM/yyyy): 19/12/2023
1.Camp Name: Apache
Camp Location: ntu
Camp Start Date: Tue Dec 19 00:00:00 SGT 2023
Camp End Date: Mon Dec 25 00:00:00 SGT 2023
Camp Duration (days): 6
Registration End Date: Mon Dec 18 00:00:00 SGT 2023
Remaining Participant Slots: 6
Remaining Committee Slots: 4
```

```java
if(campController.isCampFull(campName)) {
    System.out.println("Unable to join as camp is full");
} else if(isStudentBlacklisted(campName, userID)) {
    System.out.println("Unable to join camp as you are blacklisted");
} else if(isStudentPastRegistrationDate(campName, currentDate)){
    System.out.println("Unable to join camp as registration date has passed");
} else if(willStudentCampDatesClash(campName)){
    System.out.println("Unable to join camp as camp dates will clash with another camp");
} else if(isStudentAlreadyPartOfCamp(campName, userID) && isCampCommittee){
    System.out.println("Unable to join camp as you are already part of the camp");
} else if(isStudentAlreadyPartOfCampCommittee(campName, userID) && isCampCommittee){
    System.out.println("Unable to join camp as you are already part of the camp committee");
} else if(isStudentAlreadyPartOfCampCommitteeOfAnotherCamp(userID)){
    System.out.println("Unable to join camp as you are already part of the camp committee of another camp");
} else {
    if(isCampCommittee){
        if(isCampCommitteeFull(campName))
        {System.out.println("Unable to join camp as the camp committee is full");}
        else{
            campController.addCampCommittee(campName, userID);
            System.out.println("Successfully joined camp as camp committee");
            return true;}
    }else {
        campController.addParticipant(campName, userID);
        System.out.println("Successfully joined as camp participant");
        return true;
}}
```

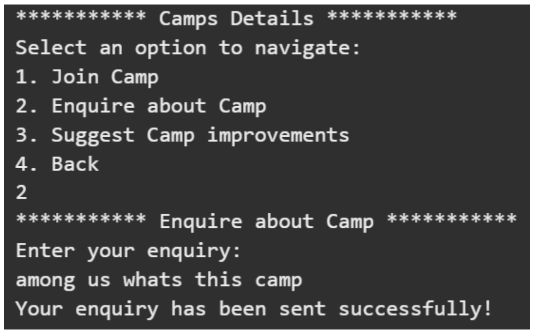
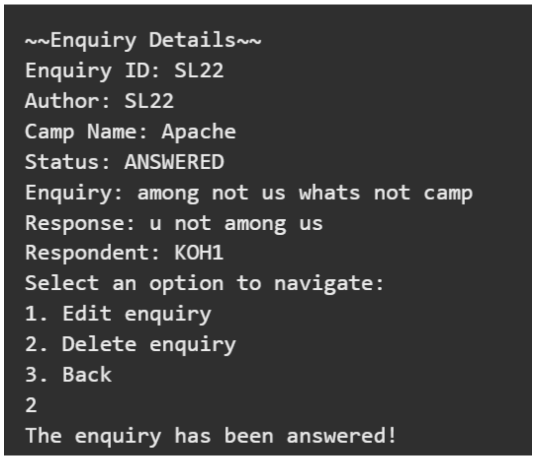
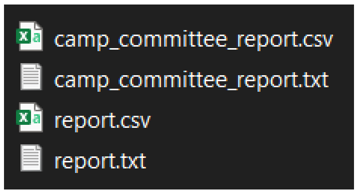
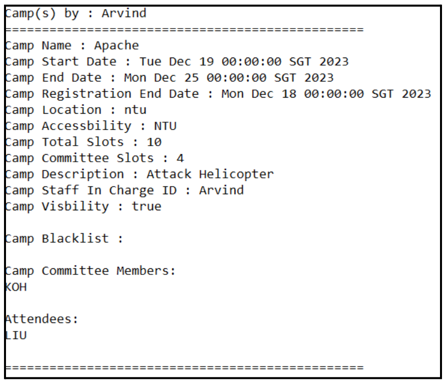
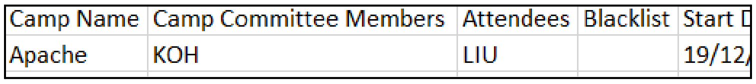
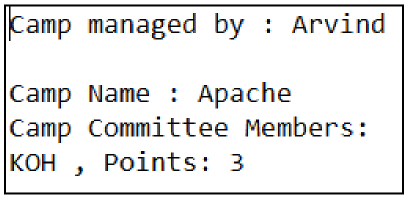
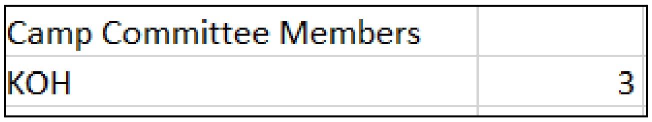
```
*********** Suggest Changes made ***********
Enter your Suggestions:
Make exam no more
Your suggestion has been sent successfully!
```

```
Profile:
Camps Registered

Apache

Committee in camp Apache

Points:1
```

Enquiries	Students can send enquiries about camps Camp committee do not need to enquiry about their own camps Camp committee can reply to enquiries by attendees. Enquiries can only be answered once Enquiries cannot be edited or deleted if it is answered Staff can answer enquiries by students	 
Generating Reports	Both Staff and Camp can generate camp reports in either txt or CSV formats Staff can generate report of Camp Committee performance in points.	    

```
*********** Camps Details ***********
Select an option to navigate:
1. Join Camp
2. Enquire about Camp
3. Suggest Camp improvements
4. Back
2
*********** Enquire about Camp ***********
Enter your enquiry:
among us whats this camp
Your enquiry has been sent successfully!
```

```
~~Enquiry Details~~
Enquiry ID: SL22
Author: SL22
Camp Name: Apache
Status: ANSWERED
Enquiry: among not us whats not camp
Response: u not among us
Respondent: KOH1
Select an option to navigate:
1. Edit enquiry
2. Delete enquiry
3. Back
2
The enquiry has been answered!
```

```
camp_committee_report.csv
camp_committee_report.txt
report.csv
report.txt
```

```
Camp(s) by : Arvind
================================================
Camp Name : Apache
Camp Start Date : Tue Dec 19 00:00:00 SGT 2023
Camp End Date : Mon Dec 25 00:00:00 SGT 2023
Camp Registration End Date : Mon Dec 18 00:00:00 SGT 2023
Camp Location : ntu
Camp Accessbility : NTU
Camp Total Slots : 10
Camp Committee Slots : 4
Camp Description : Attack Helicopter
Camp Staff In Charge ID : Arvind
Camp Visbility : true

Camp Blacklist :

Camp Committee Members:
KOH

Attendees:
LIU

================================================
```

Camp Name	Camp Committee Members	Attendees	Blacklist	Start D
Apache	KOH	LIU		19/12,

```
Camp managed by : Arvind

Camp Name : Apache
Camp Committee Members:
KOH , Points: 3
```

Camp Committee Members	
KOH	3

4. Reflections

From this project, we have understood the importance of writing code that is in compliance with good design principles. In the beginning, all of us had very limited knowledge of how to design a class diagram with well-implemented SOLID principles, so we stagnated in terms of progress when starting out. Our initial diagram was amateurish and did not properly employ SOLID design principles, which set us back even further.

Additionally, working with others was also challenging as integrating our code would result in cascading changes across the entire codebase, which is how we eventually found our design to have tight coupling and low cohesion.

We then looked up more online resources on SOLID principles and similar implementations that could further enhance our understanding of good class design. We came to understand that we could use interfaces to reduce the likelihood of us forgetting to implement methods as interfaces force us to implement methods that other classes inherit. We also learnt that we should not be passing objects into method arguments because it violates the data encapsulation principle, which we were regularly doing throughout the codebase up till then.

Therefore, this project showed us the importance of having software with loose coupling and high cohesion to make the codes reusable, maintainable and scalable, especially when coordinating a project between multiple members.

To further improve our current implementation of CAMs, we thought of implementing a page controller to manage the pages in our application. Every page will have its own controller and display view which complies with the SOLID design principle. The controller can also include additional methods for navigating to the next and previous page. This improves the program flow in the main class as it can help to reduce the need for nested loops and switch statements. Overall, the size of the codebase in the main file could be reduced with the implementation of a page controller, and improve the readability of the code in the main class.