NANYANG
TECHNOLOGICAL
UNIVERSITY
SINGAPORE

CZ2002 Object Oriented Design and Programming

Lab Group SS14

Group 10

Members:

Aloysius Seow Jing Hng (U1920159K)

Jonathan Chan Chew Meng (U1922044L)

Leong Ruo Qing (U1921880A)

Palaniselvam Shyam Sundar (U1923181H)

Soh Qian Yi (U1922306C)

Attached a scanned copy with the report with the filled details and signatures.

Declaration of Original Work for CE/CZ2002 Assignment

We hereby declare that the attached group assignment has been researched, undertaken, completed and submitted as a collective effort by the group members listed below.

We have honored the principles of academic integrity and have upheld Student Code of Academic Conduct in the completion of this work.

We understand that if plagiarism is found in the assignment, then lower marks or no marks will be awarded for the assessed work. In addition, disciplinary actions may be taken.

Name	Course (CE2002 or CZ2002)	Lab Group	Signature /Date
SOH QIAN YI	CZ2002	SS14	/11/23/20
SEOW JING HNG ALOYSIUS	CZ2002	SS14	/11/23/20
PALANISELVAM SHYAM SUNDAR	CZ002	SS14	Shyam /11/23/20
JONATHAN CHAN CHEW MENG	CZ2002	SS14	/11/23/20
LEONG RUO QING	CZ2002	SS14	/11/23/20

Important notes:

1. Name must **EXACTLY MATCH** the one printed on your Matriculation Card.

Table of Contents

1. Introduction

My STudent Automated Registration System, also known as **MyStars**, is a university application meant for each School's academic staff and undergraduate students. The application allows the creation of courses and adding of student records as well as registration of courses and students. Before the start of each semester, students are required to register for their relevant courses. Each course will have its course code, its corresponding index number information, class schedule, venue and the available vacancy.

The report will include our **design considerations, principles, and the use of OO concepts** to implement the MyStars system. A detailed UML Class diagram will be included to show clearly the class relationships, and a detailed UML Sequence diagram will show the flow of "Print student list by course" function to show all participating objects involved with the detailed flow and relevant interaction fragments.

2. Design Considerations

Throughout the course, we were taught that the basis of good design aims to achieve loose coupling and high cohesion between the classes. High reusability, extensibility and maintainability are also important considerations while implementing the application. Therefore, we aim to have our application align to the design principles as much as possible.

a. Single Responsibility Principle (SRP)

The basis of SRP states that each responsibility is an axis of change. If a class assumes more than one responsibility, then there will be more than one reason for it to change. This means that the snowball effect will occur if any of our classes takes on more than one responsibility. Therefore, while designing the application, we strictly adhere to SRP and make any class have one and only one responsibility only. Our manager classes serve one purpose, for example, our *StudentMgr* class and *AdminMgr* class would only manage attributes of a single student and admin respectively.

b. Open-Closed Principle (OCP)

This design principle encourages us to design a code which allows changes with regards to what the module does, without changing the source code of the modules. Therefore, a module should be open for extension but closed for modification. This means that the methods of any class should be kept consistent, but at the same time also allows for more extension, i.e. inheritance. This is shown in our modules such as *StudentMgr* and *AdminMgr*, whereby addition of extensions such as *AllTXT* is allowed but the source code remains unchanged.

c. Interface Segregation Principles (ISP)

ISP conveys the message that classes should not depend on interfaces that they do not use. This means that many client specific interfaces are better than one general purpose interface. This principle states that the user (admin or student) should not be exposed to methods it does not need. To adhere to this principle, our manager classes control each of their entity classes and do not depend on methods it does not use. For example, the user interface is segregated in such a way that there isn’t a situation where the Admin is exposed to functions that it does not need (like the Student functions) and vice versa.

d. Dependency Injection Principle (DIP)

DIP states that high level modules should not depend upon low level modules. Instead, both high and low level modules should depend upon abstractions. Additionally, abstractions should not depend on details, while details should depend on abstractions. Our high level control classes only depend on the methods of our entity classes, but does not depend on the details. Furthermore, implementing the details of these high level modules depend on abstractions, but not abstractions depending on details.

3. Use of Object Oriented Concepts

a. Abstraction

Abstraction denotes the essential characteristics of an object that distinguish it from all other kinds of object, thus providing crisply defined conceptual boundaries. For example, the details of the implementation are hidden from the user and only the necessary information is shown in any interaction between the user and the program.

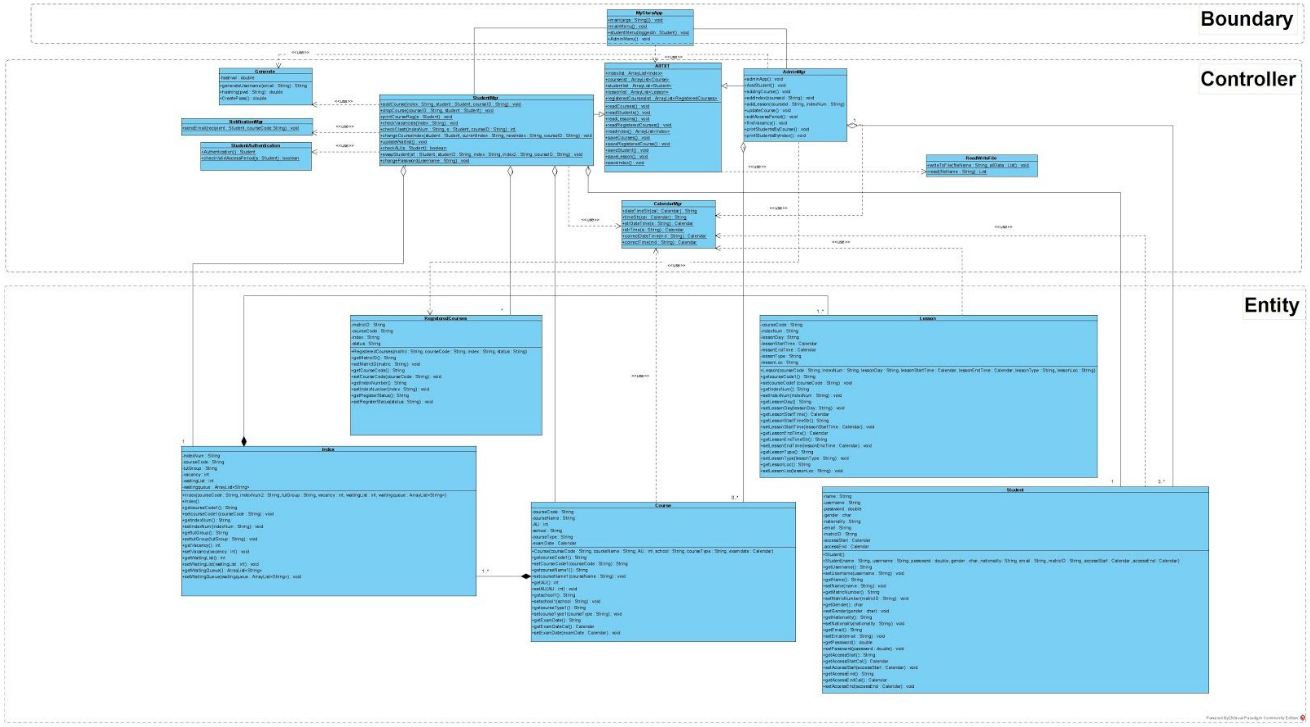
b. Encapsulation and Information Hiding

Encapsulation builds a barrier to protect an object's private data. A class encapsulates objects together. Access to private data can only be done through public methods of the object's class, such as accessors and mutators. Information hiding also hides the implementation of the class from the users. In our entity classes, taking our *Student* class to be an example, attributes such as *name*, *username* and *MatricID* are declared as private so that they can only be accessed using get() and set() methods.

c. Inheritance

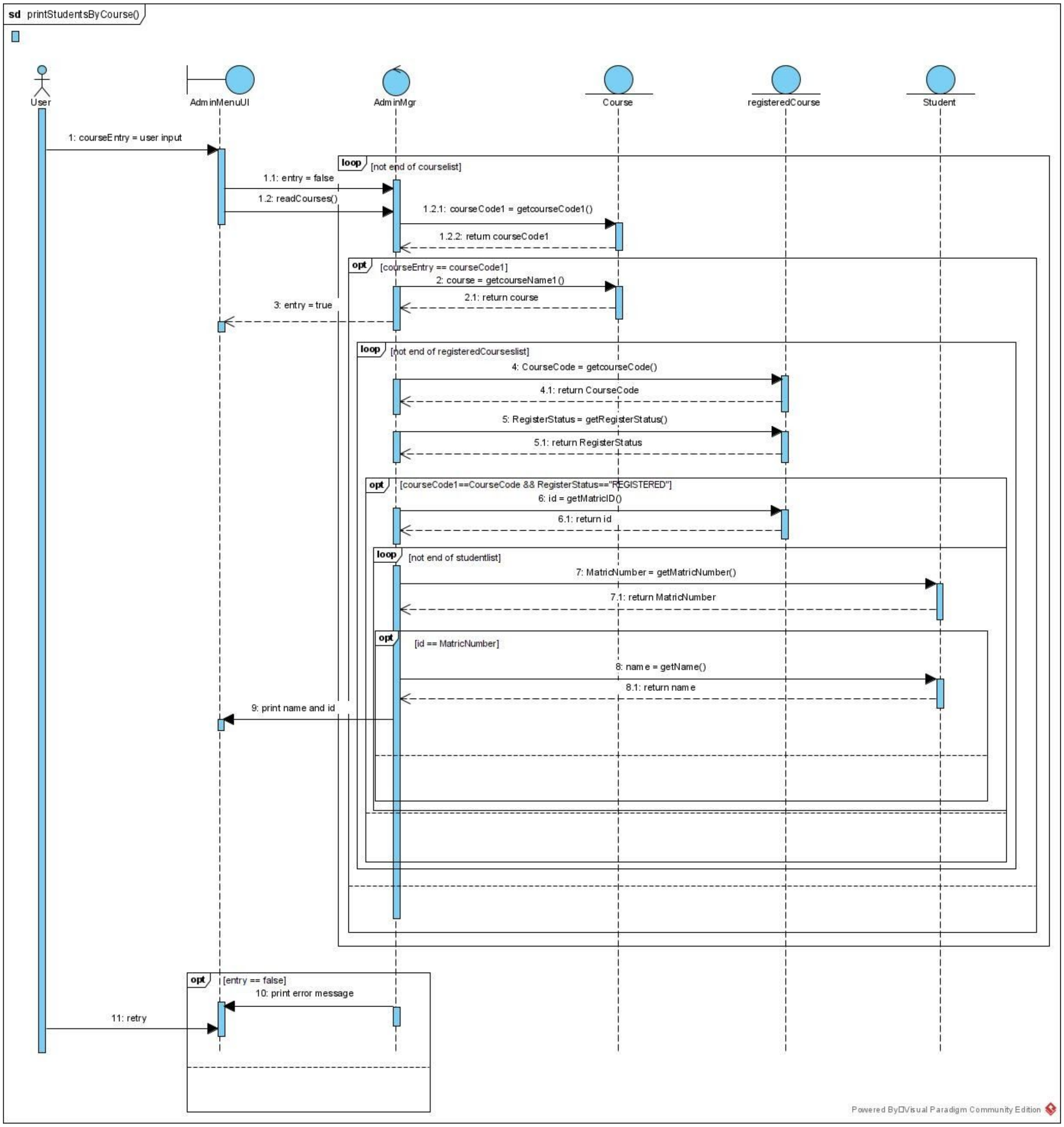
Inheritance is a mechanism in which a class inherits the properties and methods of a parent class. Any inherited behavior can be overridden in the subclass. For example, *AdminMgr* and *StudentMgr* are subclasses of our *AllTXT* class, inheriting all the methods such as *readCourses* and *saveCourses* and arraylists such as *studentlist and courselist.*

4. UML Class Diagram


Boundary
Controller
Entity
MyStarsApp
+main(args : String[]) : void
+mainMenu() : void
+studentMenu(loggedIn : Student) : void
+AdminMenu() : void
<<use>>
Generate
-hashval : double
+generateUsername(email : String) : String
+hashing(pwd : String) : double
+CreatePass() : double
NotificationMgr
+sendEmail(recipient : Student, courseCode String) : void
StudentAuthentication
+Authentication() : Student
+checkValidAccessPeriod(s : Student) : boolean
StudentMgr
+addCourse(index : String, student : Student, courseID : String) : void
+dropCourse(courseID : String, student : Student) : void
+printCourseReg(s : Student) : void
+checkVacancies(index : String) : void
+checkClash(indexNum : String, s : Student, courseID : String) : int
+changeCourseIndex(student : Student, currentIndex : String, newIndex : String, courseID : String) : void
+updateWaitlist() : void
+checkAU(s : Student) : boolean
+swapStudent(s1 : Student, student2 : String, index : String, index2 : String, courseID : String) : void
+changePassword(username : String) : void
AllTXT
+indexlist : ArrayList<Index>
+courselist : ArrayList<Course>
+studentlist : ArrayList<Student>
+lessonlist : ArrayList<Lesson>
+registeredCourseslist : ArrayList<RegisteredCourses>
+readCourses() : void
+readStudents() : void
+readLessons() : void
+readRegisteredCourses() : void
+readIndex() : ArrayList<Index>
+saveCourses() : void
+saveRegisteredCourse() : void
+saveStudent() : void
+saveLesson() : void
+saveIndex() : void
AdminMgr
+adminApp() : void
+AddStudent() : void
+addingCourse() : void
+addIndex(courseid : String) : void
+addLesson(courseid : String, indexNum : String)
+updateCourse() : void
+editAccessPeriod() : void
+findVacancy() : void
+printStudentsByCourse() : void
+printStudentsByIndex() : void
ReadWriteFile
+writeToFile(fileName : String, allData : List) : void
+read(fileName : String) : List
CalendarMgr
+dateTimeStr(cal : Calendar) : String
+timeStr(cal : Calendar) : String
+strDateTime(s : String) : Calendar
+strTime(s : String) : Calendar
+correctDateTime(inp,s : String) : Calendar
+correctTime(ind : String) : Calendar
RegisteredCourses
-matricID : String
-courseCode : String
-index : String
-status : String
+RegisteredCourses(matric : String, courseCode : String, index : String, status : String)
+getMatricID() : String
+setMatricID(matric : String) : void
+getCourseCode() : String
+setCourseCode(courseCode : String) : void
+getIndexNumber() : String
+setIndexNumber(index : String) : void
+getRegisterStatus() : String
+setRegisterStatus(status : String) : void
Lesson
-courseCode : String
-indexNum : String
-lessonDay : String
-lessonStartTime : Calendar
-lessonEndTime : Calendar
-lessonType : String
-lessonLoc : String
+Lesson(courseCode : String, indexNum : String, lessonDay : String, lessonStartTime : Calendar, lessonEndTime : Calendar, lessonType : String, lessonLoc : String)
+getcourseCode1() : String
+setcourseCode1(courseCode : String) : void
+getIndexNum() : String
+setIndexNum(indexNum : String) : void
+getLessonDay() : String
+setLessonDay(lessonDay : String) : void
+getLessonStartTime() : Calendar
+getLessonStartTimeStr() : String
+setLessonStartTime(lessonStartTime : Calendar) : void
+getLessonEndTime() : Calendar
+getLessonEndTimeStr() : String
+setLessonEndTime(lessonEndTime : Calendar) : void
+getLessonType() : String
+setLessonType(lessonType : String) : void
+getLessonLoc() : String
+setLessonLoc(lessonLoc : String) : void
Index
-indexNum : String
-courseCode : String
-tutGroup : String
-vacancy : int
-waitingList : int
-waitingqueue : ArrayList<String>
+Index(courseCode : String, indexNum2 : String, tutGroup : String, vacancy : int, waitingList : int, waitingqueue : ArrayList<String>)
+Index()
+getcourseCode1() : String
+setcourseCode1(courseCode : String) : void
+getindexNum() : String
+setindexNum(indexNum : String) : void
+gettutGroup() : String
+settutGroup(tutGroup : String) : void
+getVacancy() : int
+setVacancy(vacancy : int) : void
+getWaitingList() : int
+setWaitingList(waitingList : int) : void
+getWaitingQueue() : ArrayList<String>
+setWaitingQueue(waitingqueue : ArrayList<String>) : void
Course
-courseCode : String
-courseName : String
-AU : int
-school : String
-courseType : String
-examDate : Calendar
+Course(courseCode : String, courseName : String, AU : int, school : String, courseType : String, examdate : Calendar)
+getcourseCode1() : String
+setCourseCode1(courseCode : String) : String
+getcourseName1() : String
+setcourseName1(courseName : String) : void
+getAU() : int
+setAU(AU : int) : void
+getschool1() : String
+setschool1(school : String) : void
+getcourseType1() : String
+setcourseType1(courseType : String) : void
+getExamDate() : String
+getExamDateCal() : Calendar
+setExamDate(examDate : Calendar) : void
Student
-name : String
-username : String
-password : double
-gender : char
-nationality : String
-email : String
-matricID : String
-accessStart : Calendar
-accessEnd : Calendar
+Student()
+Student(name : String, username : String, password : double, gender : char, nationality : String, email : String, matricID : String, accessStart : Calendar, accessEnd : Calendar)
+getUsername() : String
+setUsername(username : String) : void
+getName() : String
+setName(name : String) : void
+getMatricNumber() : String
+setMatricNumber(matricID : String) : void
+getGender() : char
+setGender(gender : char) : void
+getNationality() : String
+setNationality(nationality : String) : void
+getEmail() : String
+setEmail(email : String) : void
+getPassword() : double
+setPassword(password : double) : void
+getAccessStart() : String
+getAccessStartCal() : Calendar
+setAccessStart(accessStart : Calendar) : void
+getAccessEnd() : String
+getAccessEndCal() : Calendar
+setAccessEnd(accessEnd : Calendar) : void
1
1..*
0..*
*
Powered By Visual Paradigm Community Edition

We have attached a clearer image of the UML Class Diagram in the zip folder.

5. UML Sequence Diagram


sd printStudentsByCourse()
User
AdminMenuUI
AdminMgr
Course
registeredCourse
Student
1: courseEntry = user input
loop [not end of courselist]
1.1: entry = false
1.2: readCourses()
1.2.1: courseCode1 = getcourseCode1()
1.2.2: return courseCode1
opt [courseEntry == courseCode1]
2: course = getcourseName1()
2.1: return course
3: entry = true
loop [not end of registeredCourseslist]
4: CourseCode = getcourseCode()
4.1: return CourseCode
5: RegisterStatus = getRegisterStatus()
5.1: return RegisterStatus
opt [courseCode1==CourseCode && RegisterStatus=="REGISTERED"]
6: id = getMatricID()
6.1: return id
loop [not end of studentlist]
7: MatricNumber = getMatricNumber()
7.1: return MatricNumber
opt [id == MatricNumber]
8: name = getName()
8.1: return name
9: print name and id
opt [entry == false]
10: print error message
11: retry
Powered By Visual Paradigm Community Edition

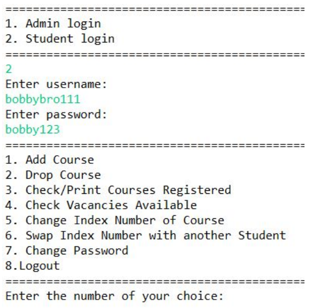
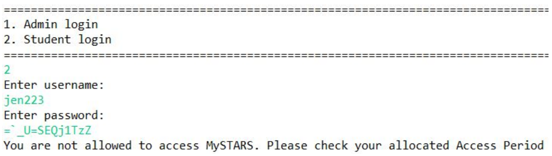
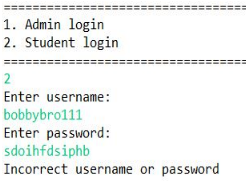
We have attached a clearer image of the UML Sequence Diagram in the zip folder.

6. Demonstration of Test Cases

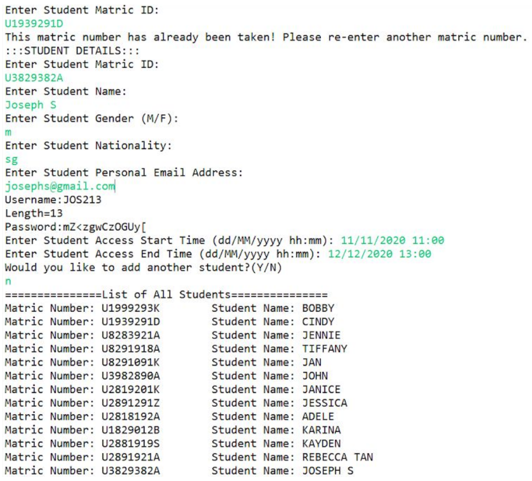
Assumptions Made:

1. Max AUs capped at 18.
2. There is only 1 Admin Account with password = “letmein”.
3. All lessons are for all academic weeks (both odd and even weeks).
4. Students in the waitlist queue will not be removed from the queue even after they hit the maximum AUs registered.
5. The time format used in the CalendarMgr class is in 24-hour format.
6. Upon student creation, the student will know of the generated password and this password can be used by the student to reset later.

a. Student Login

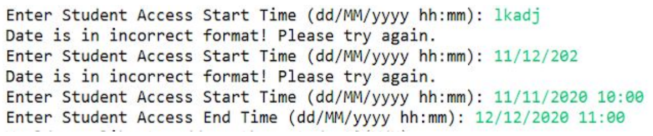
Login Successful	Login before/after allowed period (dates)	Wrong Password
 == 1. Admin login 2. Student login == 2 Enter username: bobbybro111 Enter password: bobby123 == 1. Add Course 2. Drop Course 3. Check/Print Courses Registered 4. Check Vacancies Available 5. Change Index Number of Course 6. Swap Index Number with another Student 7. Change Password 8.Logout == Enter the number of your choice:	 == 1. Admin login 2. Student login == 2 Enter username: jen223 Enter password: =`_U=SEQj1TzZ You are not allowed to access MySTARS. Please check your allocated Access Period	 ==================================== 1. Admin login 2. Student login ==================================== 2 Enter username: bobbybro111 Enter password: sdoihfdsiphb Incorrect username or password

b. Add a Student (ADMIN)

(b) Add Existing Student (a) Add a New Student	 Enter Student Matric ID: U1939291D This matric number has already been taken! Please re-enter another matric number. :::STUDENT DETAILS::: Enter Student Matric ID: U3829382A Enter Student Name: Joseph S Enter Student Gender (M/F): m Enter Student Nationality: sg Enter Student Personal Email Address: josephs@gmail.com Username:JOS213 Length=13 Password:mZ<zgwCzOGUy[Enter Student Access Start Time (dd/MM/yyyy hh:mm): 11/11/2020 11:00 Enter Student Access End Time (dd/MM/yyyy hh:mm): 12/12/2020 13:00 Would you like to add another student?(Y/N) n ===============List of All Students=============== Matric Number: U1999293K Student Name: BOBBY Matric Number: U1939291D Student Name: CINDY Matric Number: U8283921A Student Name: JENNIE Matric Number: U8291918A Student Name: TIFFANY Matric Number: U8291091K Student Name: JAN Matric Number: U3982890A Student Name: JOHN Matric Number: U2819201K Student Name: JANICE Matric Number: U2891291Z Student Name: JESSICA Matric Number: U2818192A Student Name: ADELE Matric Number: U1829012B Student Name: KARINA Matric Number: U2881919S Student Name: KAYDEN Matric Number: U2891921A Student Name: REBECCA TAN Matric Number: U3829382A Student Name: JOSEPH S

c) Invalid Data Entries

1. Invalid Input for Student Access Start Time
2. Invalid Input for Student Access End Time



```
Enter Student Access Start Time (dd/MM/yyyy hh:mm): lkadj
Date is in incorrect format! Please try again.
Enter Student Access Start Time (dd/MM/yyyy hh:mm): 11/12/202
Date is in incorrect format! Please try again.
Enter Student Access Start Time (dd/MM/yyyy hh:mm): 11/11/2020 10:00
Enter Student Access End Time (dd/MM/yyyy hh:mm): 12/12/2020 11:00
```

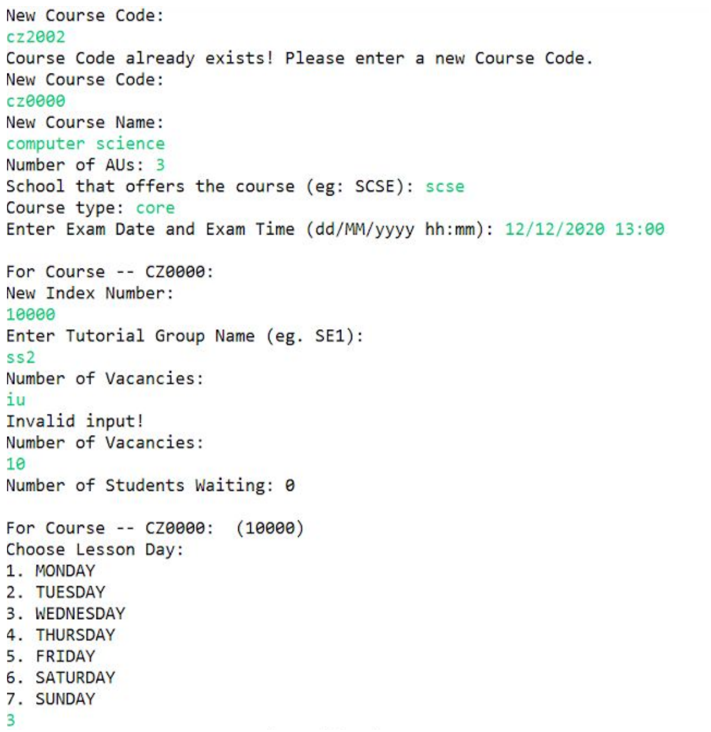
c. Add a Course (ADMIN)

(b) Add Existing Course

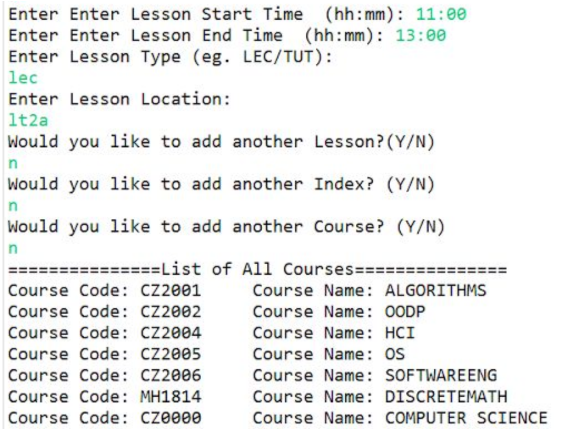
(a) Add a New Course

(c) Invalid Data Entries

1. Invalid Input for Number of Vacancies



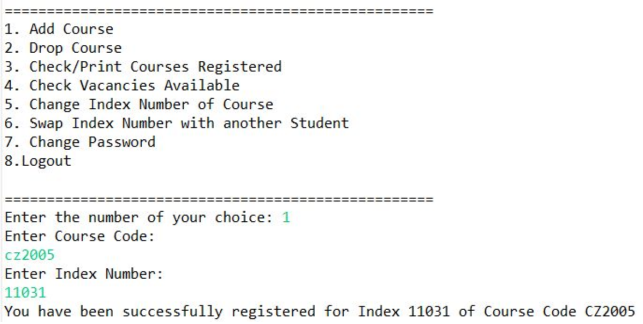
```
New Course Code:
cz2002
Course Code already exists! Please enter a new Course Code.
New Course Code:
cz0000
New Course Name:
computer science
Number of AUs: 3
School that offers the course (eg: SCSE): scse
Course type: core
Enter Exam Date and Exam Time (dd/MM/yyyy hh:mm): 12/12/2020 13:00

For Course -- CZ0000:
New Index Number:
10000
Enter Tutorial Group Name (eg. SE1):
ss2
Number of Vacancies:
iu
Invalid input!
Number of Vacancies:
10
Number of Students Waiting: 0

For Course -- CZ0000:  (10000)
Choose Lesson Day:
1. MONDAY
2. TUESDAY
3. WEDNESDAY
4. THURSDAY
5. FRIDAY
6. SATURDAY
7. SUNDAY
3
```



```
Enter Enter Lesson Start Time  (hh:mm): 11:00
Enter Enter Lesson End Time  (hh:mm): 13:00
Enter Lesson Type (eg. LEC/TUT):
lec
Enter Lesson Location:
lt2a
Would you like to add another Lesson?(Y/N)
n
Would you like to add another Index? (Y/N)
n
Would you like to add another Course? (Y/N)
n
===============List of All Courses===============
Course Code: CZ2001     Course Name: ALGORITHMS
Course Code: CZ2002     Course Name: OODP
Course Code: CZ2004     Course Name: HCI
Course Code: CZ2005     Course Name: OS
Course Code: CZ2006     Course Name: SOFTWAREENG
Course Code: MH1814     Course Name: DISCRETEMATH
Course Code: CZ0000     Course Name: COMPUTER SCIENCE
```

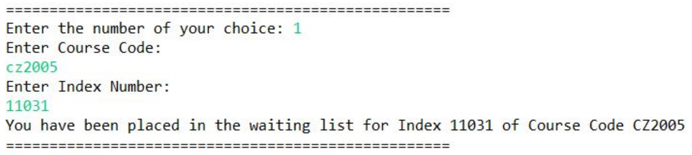
c. Register Student for a Course (STUDENT)

(a) Add a student to a course index with available vacancies



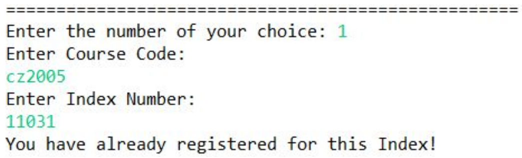
```
==================================================
1. Add Course
2. Drop Course
3. Check/Print Courses Registered
4. Check Vacancies Available
5. Change Index Number of Course
6. Swap Index Number with another Student
7. Change Password
8.Logout

==================================================
Enter the number of your choice: 1
Enter Course Code:
cz2005
Enter Index Number:
11031
You have been successfully registered for Index 11031 of Course Code CZ2005
```

(b) Add a student to a course index with 0 vacancies in Tut / Lab



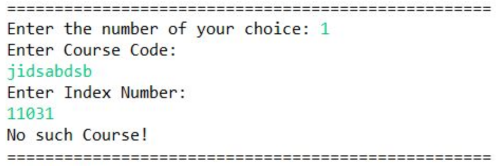
```
==================================================
Enter the number of your choice: 1
Enter Course Code:
cz2005
Enter Index Number:
11031
You have been placed in the waiting list for Index 11031 of Course Code CZ2005
==================================================
```

(c) Register the same course again

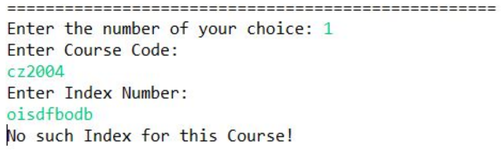


```
==================================================
Enter the number of your choice: 1
Enter Course Code:
cz2005
Enter Index Number:
11031
You have already registered for this Index!
```

(d) Invalid data entries (wrong Index / Course code)

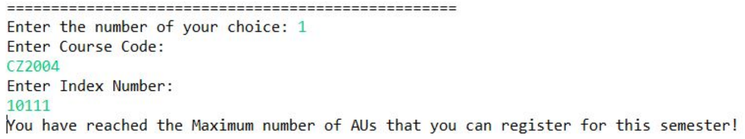


```
==================================================
Enter the number of your choice: 1
Enter Course Code:
jidsabdsb
Enter Index Number:
11031
No such Course!
==================================================
```



```
==================================================
Enter the number of your choice: 1
Enter Course Code:
cz2004
Enter Index Number:
oisdfbodb
No such Index for this Course!
```

(e) Max number of AUs registered (capped at 18)



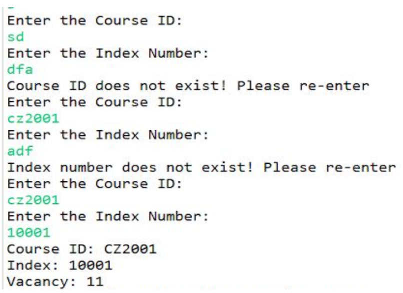
```
==================================================
Enter the number of your choice: 1
Enter Course Code:
CZ2004
Enter Index Number:
10111
You have reached the Maximum number of AUs that you can register for this semester!
```

d. Check Available/Vacancy Slot in a Class (ADMIN/STUDENT)

(a) Check for vacancy in course index

(b) Invalid Data Entries

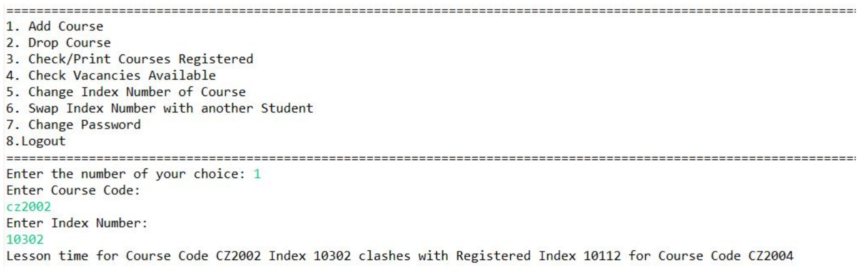
1. Invalid Course ID
2. Invalid Index Number



```
Enter the Course ID:
sd
Enter the Index Number:
dfa
Course ID does not exist! Please re-enter
Enter the Course ID:
cz2001
Enter the Index Number:
adf
Index number does not exist! Please re-enter
Enter the Course ID:
cz2001
Enter the Index Number:
10001
Course ID: CZ2001
Index: 10001
Vacancy: 11
```

e. Day/Time Clash with other Course (STUDENT)

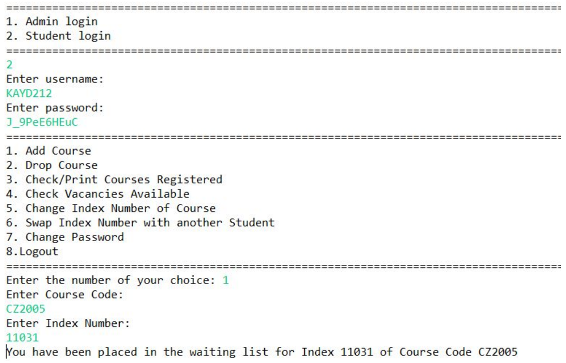
Add a student to a course index with available vacancies.



```
==========================================================================================================================
1. Add Course
2. Drop Course
3. Check/Print Courses Registered
4. Check Vacancies Available
5. Change Index Number of Course
6. Swap Index Number with another Student
7. Change Password
8.Logout
==========================================================================================================================
Enter the number of your choice: 1
Enter Course Code:
cz2002
Enter Index Number:
10302
Lesson time for Course Code CZ2002 Index 10302 clashes with Registered Index 10112 for Course Code CZ2004
```

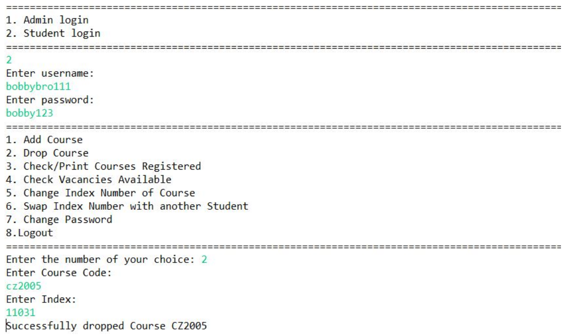
f. Waitlist Notification (STUDENT)

(a) Add studentA to a course index with 0 vacancies student on waitlist

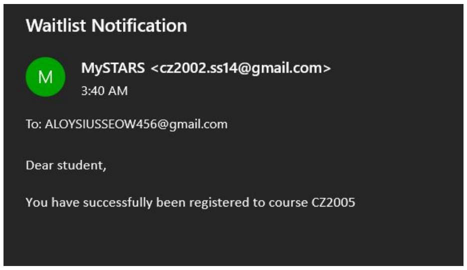
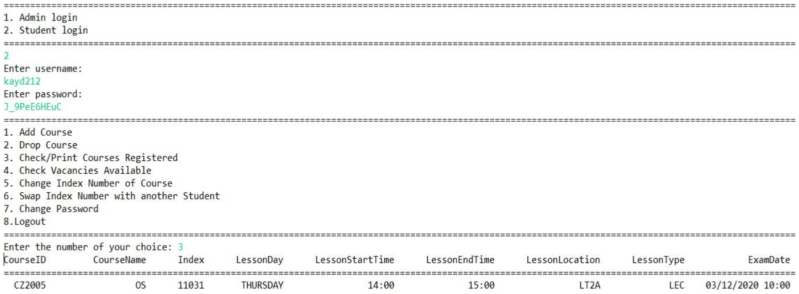


```
==========================================================================================
1. Admin login
2. Student login
==========================================================================================
2
Enter username:
KAYD212
Enter password:
J_9PeE6HEuC
==========================================================================================
1. Add Course
2. Drop Course
3. Check/Print Courses Registered
4. Check Vacancies Available
5. Change Index Number of Course
6. Swap Index Number with another Student
7. Change Password
8.Logout
==========================================================================================
Enter the number of your choice: 1
Enter Course Code:
CZ2005
Enter Index Number:
11031
You have been placed in the waiting list for Index 11031 of Course Code CZ2005
```

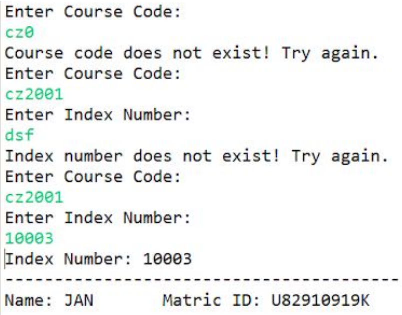
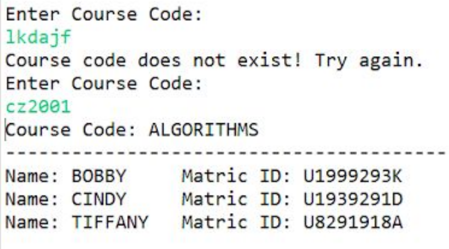
(b) Drop studentB from the same course index StudentB successfully dropped and studentA successfully added.



```
==========================================================================================
1. Admin login
2. Student login
==========================================================================================
2
Enter username:
bobbybro111
Enter password:
bobby123
==========================================================================================
1. Add Course
2. Drop Course
3. Check/Print Courses Registered
4. Check Vacancies Available
5. Change Index Number of Course
6. Swap Index Number with another Student
7. Change Password
8.Logout
==========================================================================================
Enter the number of your choice: 2
Enter Course Code:
cz2005
Enter Index:
11031
Successfully dropped Course CZ2005
```

(c) Waitlist notification sent to studentA	 Waitlist Notification MySTARS <cz2002.ss14@gmail.com> 3:40 AM To: ALOYSIUSSEOW456@gmail.com Dear student, You have successfully been registered to course CZ2005
(d) Print StudentA's Timetable	 == 1. Admin login 2. Student login == 2 Enter username: kayd212 Enter password: J_9PeE6HEuC == 1. Add Course 2. Drop Course 3. Check/Print Courses Registered 4. Check Vacancies Available 5. Change Index Number of Course 6. Swap Index Number with another Student 7. Change Password 8.Logout == Enter the number of your choice: 3 CourseID CourseName Index LessonDay LessonStartTime LessonEndTime LessonLocation LessonType ExamDate == CZ2005 OS 11031 THURSDAY 14:00 15:00 LT2A LEC 03/12/2020 10:00

g. Print Student List by Index Number & Course (ADMIN)

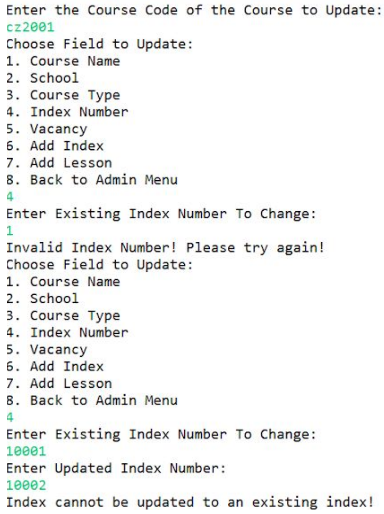
(i) (a) Print List by Index Number (b) Invalid Data Entries 1. Invalid Course Code 2. Invalid Index Number	 Enter Course Code: cz0 Course code does not exist! Try again. Enter Course Code: cz2001 Enter Index Number: dsf Index number does not exist! Try again. Enter Course Code: cz2001 Enter Index Number: 10003 Index Number: 10003 -- Name: JAN Matric ID: U82910919K
(ii) (a) Print List by Course (b) Invalid Data Entry 1. Invalid Course Code	 Enter Course Code: lkdajf Course code does not exist! Try again. Enter Course Code: cz2001 Course Code: ALGORITHMS -- Name: BOBBY Matric ID: U1999293K Name: CINDY Matric ID: U1939291D Name: TIFFANY Matric ID: U8291918A

Additional Test Cases:

h. Update Index Number of a Course (ADMIN)

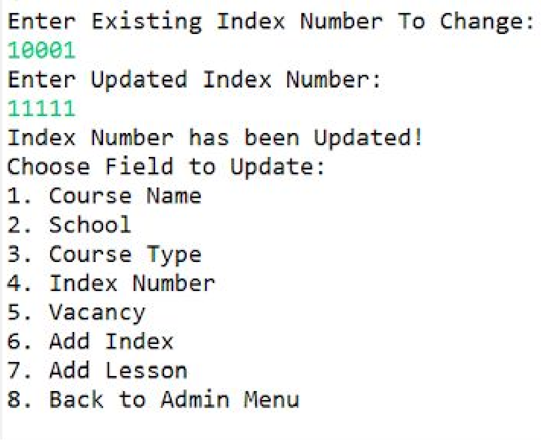
(a) Invalid Data Entries

1. Invalid Index Number
2. User entered Existing Index Number of the same Course



```
Enter the Course Code of the Course to Update:
cz2001
Choose Field to Update:
1. Course Name
2. School
3. Course Type
4. Index Number
5. Vacancy
6. Add Index
7. Add Lesson
8. Back to Admin Menu
4
Enter Existing Index Number To Change:
1
Invalid Index Number! Please try again!
Choose Field to Update:
1. Course Name
2. School
3. Course Type
4. Index Number
5. Vacancy
6. Add Index
7. Add Lesson
8. Back to Admin Menu
4
Enter Existing Index Number To Change:
10001
Enter Updated Index Number:
10002
Index cannot be updated to an existing index!
```

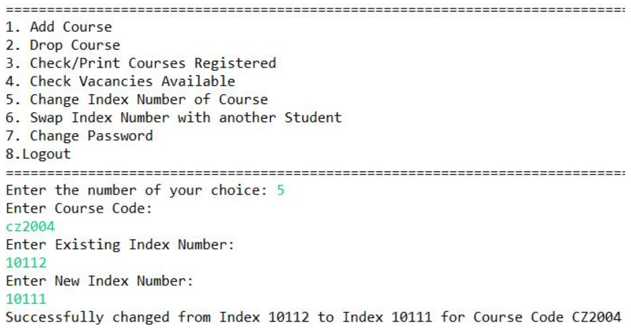
(b) Successfully updated Index Number of CZ2001



```
Enter Existing Index Number To Change:
10001
Enter Updated Index Number:
11111
Index Number has been Updated!
Choose Field to Update:
1. Course Name
2. School
3. Course Type
4. Index Number
5. Vacancy
6. Add Index
7. Add Lesson
8. Back to Admin Menu
```

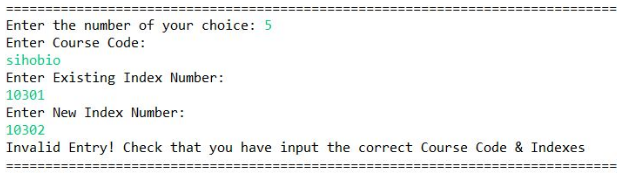
i. Change Existing Index for Student (STUDENT)

(a) Change Existing Registered Index to another Index of the same Course



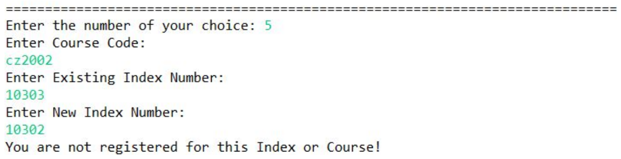
```
=============================================================================
1. Add Course
2. Drop Course
3. Check/Print Courses Registered
4. Check Vacancies Available
5. Change Index Number of Course
6. Swap Index Number with another Student
7. Change Password
8.Logout
=============================================================================
Enter the number of your choice: 5
Enter Course Code:
cz2004
Enter Existing Index Number:
10112
Enter New Index Number:
10111
Successfully changed from Index 10112 to Index 10111 for Course Code CZ2004
```

(b) Invalid entry of Course Code/Index



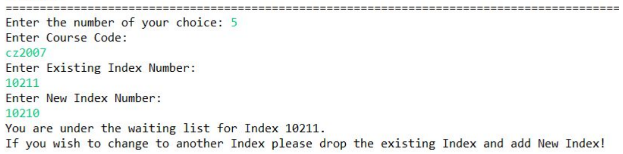
```
=============================================================================
Enter the number of your choice: 5
Enter Course Code:
sihobio
Enter Existing Index Number:
10301
Enter New Index Number:
10302
Invalid Entry! Check that you have input the correct Course Code & Indexes
=============================================================================
```

(c) Index/Course not registered for Student



```
=============================================================================
Enter the number of your choice: 5
Enter Course Code:
cz2002
Enter Existing Index Number:
10303
Enter New Index Number:
10302
You are not registered for this Index or Course!
```

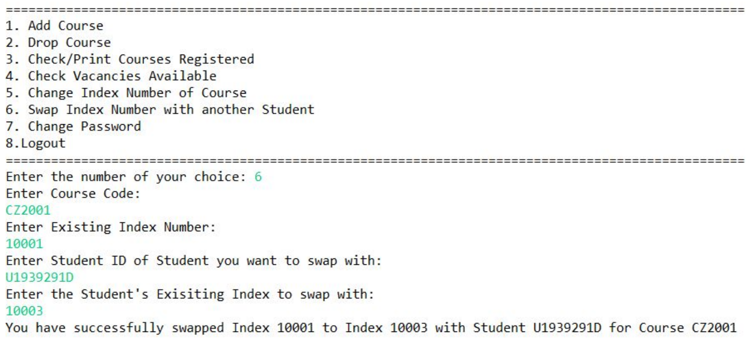
(d) Index for the student under 'Waitlist'



```
==========================================================================================
Enter the number of your choice: 5
Enter Course Code:
cz2007
Enter Existing Index Number:
10211
Enter New Index Number:
10210
You are under the waiting list for Index 10211.
If you wish to change to another Index please drop the existing Index and add New Index!
```

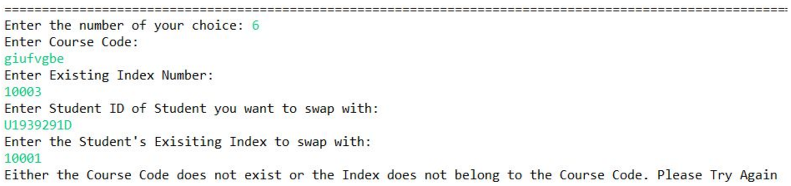
j. Swap Index with Another Student (STUDENT)

(a) Student A swap index of the same course with Student B



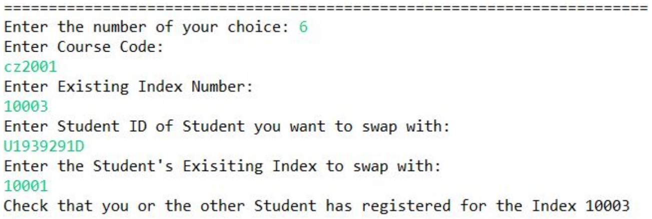
```
================================================================================================
1. Add Course
2. Drop Course
3. Check/Print Courses Registered
4. Check Vacancies Available
5. Change Index Number of Course
6. Swap Index Number with another Student
7. Change Password
8.Logout
================================================================================================
Enter the number of your choice: 6
Enter Course Code:
CZ2001
Enter Existing Index Number:
10001
Enter Student ID of Student you want to swap with:
U1939291D
Enter the Student's Exisiting Index to swap with:
10003
You have successfully swapped Index 10001 to Index 10003 with Student U1939291D for Course CZ2001
```

(b) Invalid Course code/Index number



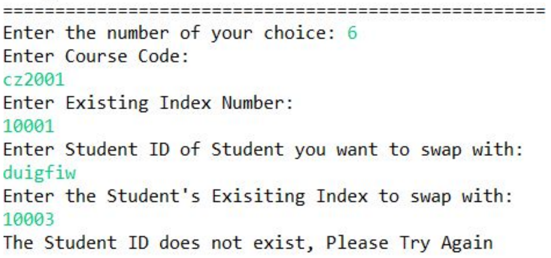
```
====================================================================================================
Enter the number of your choice: 6
Enter Course Code:
giufvgbe
Enter Existing Index Number:
10003
Enter Student ID of Student you want to swap with:
U1939291D
Enter the Student's Exisiting Index to swap with:
10001
Either the Course Code does not exist or the Index does not belong to the Course Code. Please Try Again
```

(c) Either one or both Students have not registered for the index



```
=======================================================================
Enter the number of your choice: 6
Enter Course Code:
cz2001
Enter Existing Index Number:
10003
Enter Student ID of Student you want to swap with:
U1939291D
Enter the Student's Exisiting Index to swap with:
10001
Check that you or the other Student has registered for the Index 10003
```

(d) Invalid Student B Student ID



```
====================================================
Enter the number of your choice: 6
Enter Course Code:
cz2001
Enter Existing Index Number:
10001
Enter Student ID of Student you want to swap with:
duigfiw
Enter the Student's Exisiting Index to swap with:
10003
The Student ID does not exist, Please Try Again
```

7. Demonstration Link

https://youtu.be/OK8SeIsTPBg