CZ2002 Object Oriented Design & Program

Group 4

Team members:

FAVIAN CHAN JUN WEI (U1922015H)

HUANG RUNTAO (U1923490F)

KOTHARI KHUSH MILAN (U1922279J)

TAN CHIN HOE, SAMUEL (U1921228C)

YIN NING (U1923120G)

1. Introduction

The aim of this project is to implement the STARS Planner system of Nanyang Technological University to be exclusively used by both, undergraduate matriculated students and the administration staff of the university. This application created by us is closely related to the actual one used by the university. The students will be able to access various features like adding and dropping courses, swapping indexes of a course with another student, checking registered courses and so on. The administration staff will also be able perform functions like adding students, printing student lists, creating vacancies etc. In this application, we will be focussing on the correct use of design principles and object-oriented concepts for modelling.

2. Design Considerations and use of Object-Oriented Programming Concepts

(i) Approach Taken

The STARS Planner was created in such a way that it is easy to understand and is well-defined. It is divided into many smaller units performing their own respective tasks and access the data through common entity classes, preventing tight coupling and allowing higher cohesion. We have also kept in mind the Model-View-Controller concept while designing our application. Classes such as 'User' and 'Course' act as entity or model classes. Classes such as 'Main' act as the boundary class displaying the menu of actions that the user can perform. Finally, we have controller classes like 'StudentController' and 'AdminController' that are responsible for carrying out all the features and functions provided by our application to students and administration staff respectively and thus, updating the boundary class.

(ii) SOLID Design Principles Used

a. Single Responsibility Principle (SRP)

This principle is to ensure that each class only takes responsibility over a single part of the program, to prevent creating a "God" class that handles multiple functions that are out of its boundary. We incorporated the principle in our program design, so each class only handles what it is intended to do. This is seen by the design of separate controller classes for each entity.

The ‘AdminController’ class performs all the functions available for the administration staff provided in the boundary class. It accesses the data through the ‘Admin’ class which is an entity. The ‘StudentController’ executes all the features provided to the student through the boundary. Each student can perform any function as this controller class uses the data from the ‘Student’ class which also is an entity. The ‘SendMail’ class handles the task of reading the student/administration staff email id and sending the necessary email to them. This design makes the application simple and allows us to avoid tight coupling.

b. Open-Closed Principle (OCP)

The Open-Closed principle states that classes or modules should be open for extension but not for modification. This is visible through the ‘User’ class which is extended by both ‘Student’ and ‘Admin’ classes as they are types of users. Therefore, if a new user, for example ‘Professor’, is to be introduced into the system, then it can be easily done with the help of the ‘User’ class. Hence, we will not be required to change the source code of other classes.

c. Liskov Substitution Principle (LSP)

This principle states that if we replace the object reference of a superclass with the object of any of its subclasses, the program should function correctly. The ‘Course’ and ‘Time’ class replaces the savedata and loaddata function of the superclass ‘Ser_File’. The ‘student’ and ‘admin’ also replace the savedata and loaddata function of the superclass ‘Ser_File’ through the extension of ‘User’ class.

d. Dependency Injection Principle (DIP)

This principle states all the high-level modules should be independent of the low-level modules so that they can be reused easily. It means that the higher-level modules should not be affected by the changes in the low-level modules. Both modules should depend on abstraction. This is seen in class like ‘SendMail’. ‘SendMail’ is a high-level class whose code is used by low-level classes like ‘StudentController’ via an interface ‘EmailNotification’ to send emails. It remains unaffected by any changes made in the ‘StudentController’ class.

(ii) Object-Oriented Programming Concepts Used

a. Encapsulation

Certain information in the class is being classified as 'private', which is only accessible using an accessor function. Classes such as 'Student', 'Admin', 'Course', 'Lesson', 'Index', 'Time' and 'User' all handle their own private information. For example, the variables that contain details of a student's data storage file are private and can only be accessed by that particular class. Due to this, the data in an object will be hidden from the public view and only those data available through accessor methods will be retrieved. Hence, certain attributes of a class are kept as private but the get and set methods are kept as public.

b. Inheritance

Inheritance enables a class to inherit the variable and functions from another class (the parent class). It allows the reusability of codes where the sub-class (child) will reuse the super-class (parent) codes as well as be able to add its own methods. For example, 'Student' and 'Admin' extend (inherit) from the 'User' class, thus, allowing them to use the "hashing" function.

c. Polymorphism

Polymorphism in Java is a concept in which we can perform a single action in multiple ways. Runtime Polymorphism occurs when the call to an overridden method is resolved during runtime rather than compile time. This is seen in the 'Ser_File' class and the 'Student' class where the loadData() and saveData() functions are present in both the classes. However, during runtime, the functions in the 'Student' class will override the functions in the 'Ser_File' class. Thus, polymorphism is enforced.

d. Abstraction

Abstraction is the process of displaying only the essential information to the user and hiding the implementation detail from the user. This is achieved by the usage of the 'User' class. The 'User' class acts as a superclass to all the different types of people who could use the STARS Planner system. Thus, to introduce a new type of user into the system we can just extend the ‘User’ class and then the specific unique attributes for that particular user could be added in the subclass. Hence, any changes made in the modules of any of the users will not affect the code of the ‘User’ class.

3. Further Extensions

We can make changes to the system and add more classes that use the ‘Notification’ interface giving the user the choice to change the format in which he/she receives notifications from the system. For example, if a user prefers to receive notifications through an SMS, we could add a ‘SMS’ class and that would implement the notification interface and this class’s methods could be used by all the controller classes to send notification in SMS format.

4. Assumptions Made

The team has chosen to make the following assumptions regarding our application:

- All the indexes of a course share the same lectures.
- Every course must have at least one lecture per week.
- Admin can log in to the system at any time.
- Courses and indexes will not be deleted once they are successfully saved.
- All indexes are unique.
- Labs and tutorials for each index take place every week.
- There could be at most one lab and one tutorial session every week for each index.

5. Demonstration Video Link

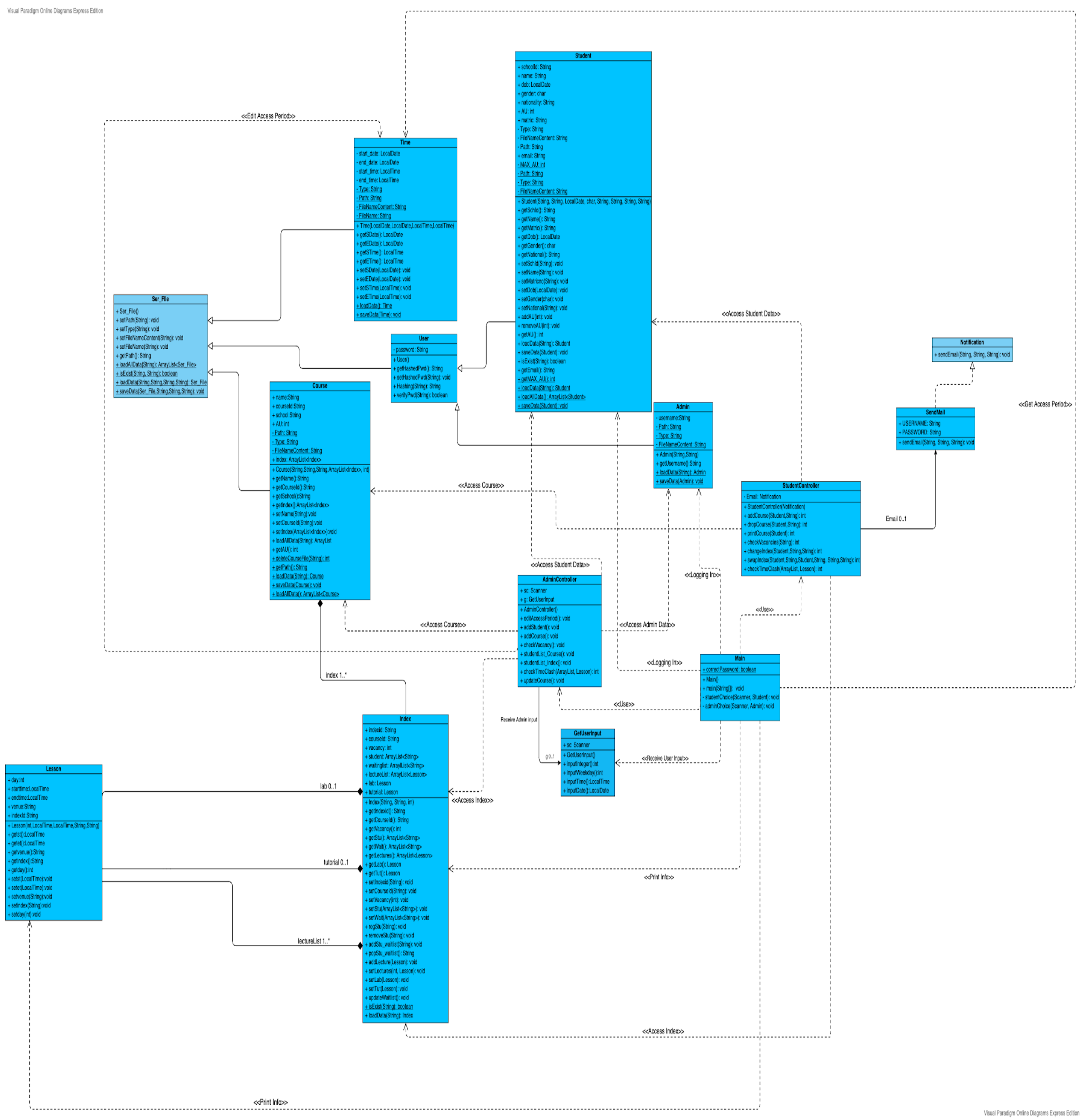
https://youtu.be/M84FPjVRdTE

6. UML Class Diagram

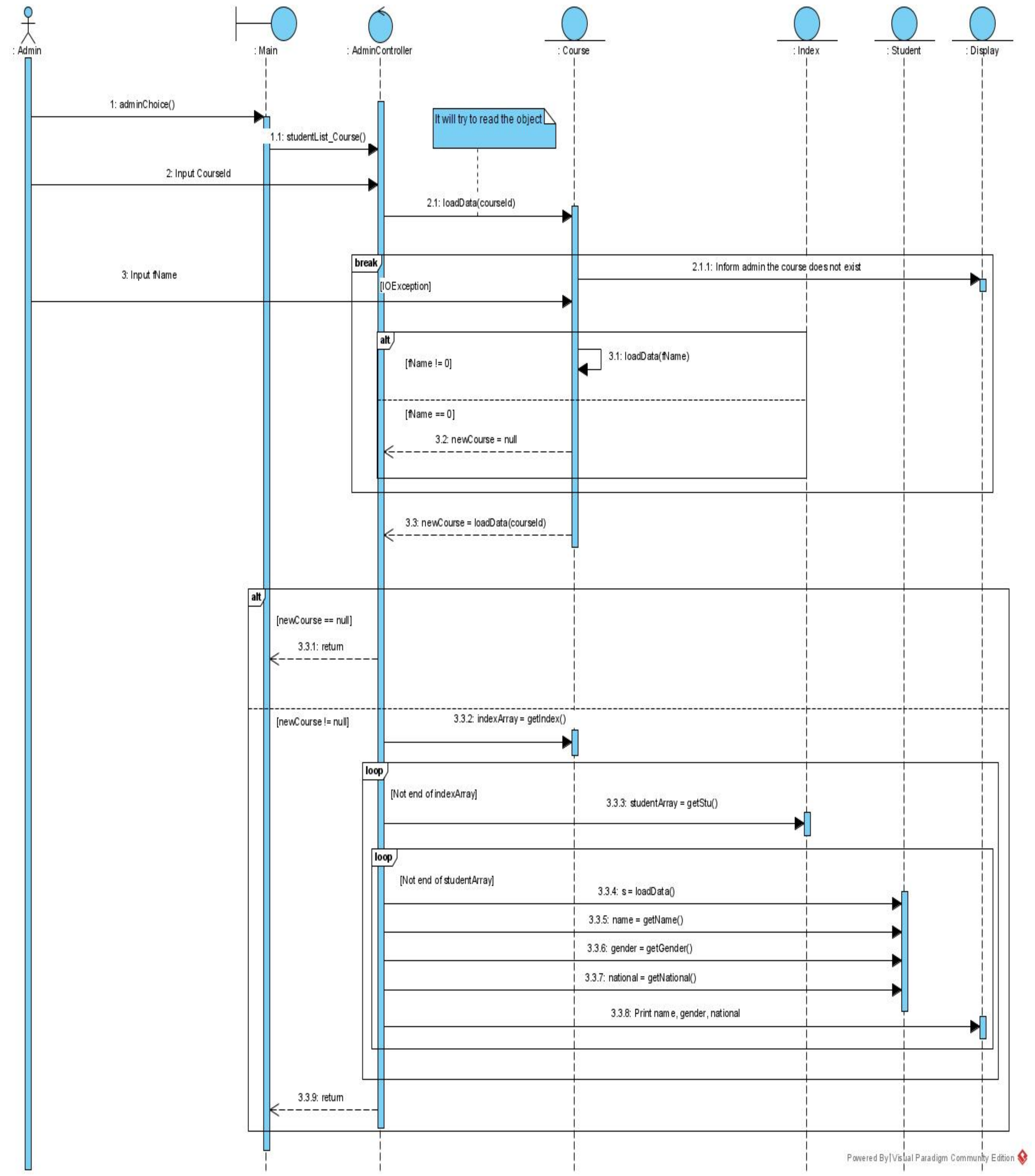
Entities: Student, Admin, Course, Index, Lesson, Time

Boundary: Main

Controllers: StudentController, AdminController



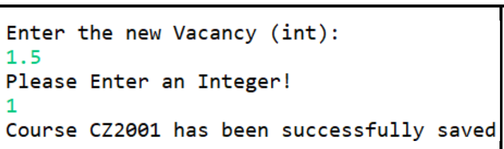
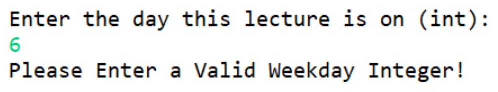
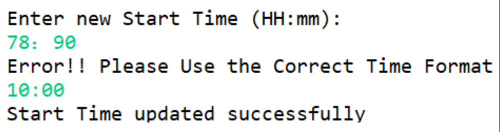
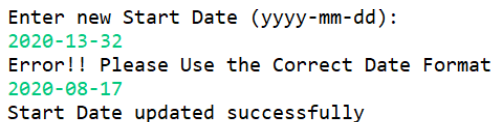
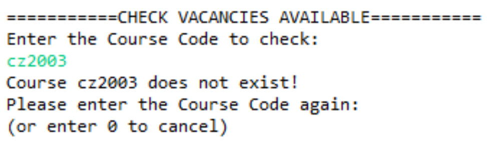
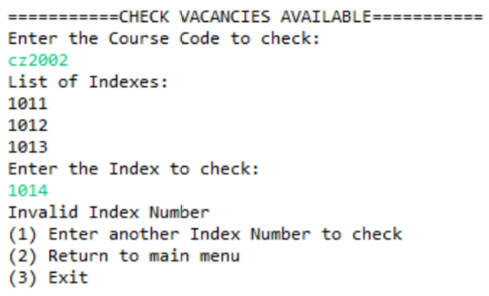
7. **Sequence Diagram**



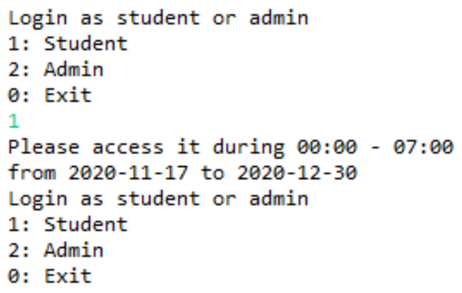
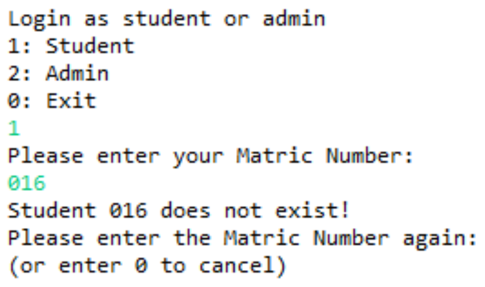
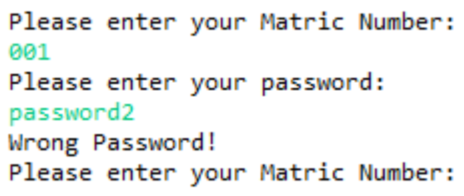
8. Test Cases

The duplicated error messages in different functions will only show once in the form below.

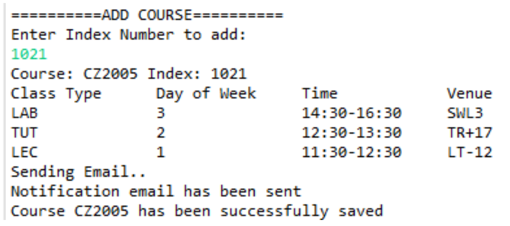
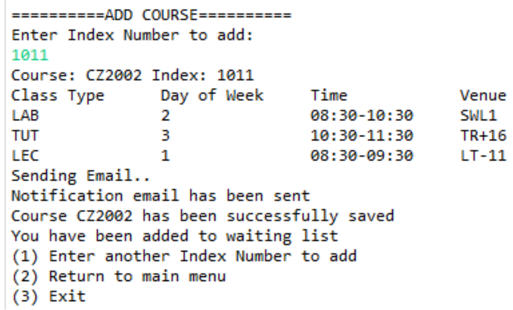
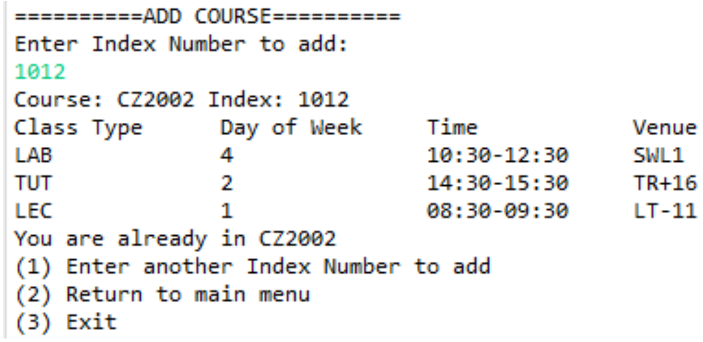
a. General invalid data entries

a	Invalid integer	 Enter the new Vacancy (int): 1.5 Please Enter an Integer! 1 Course CZ2001 has been successfully saved	b	Invalid weekday integer	 Enter the day this lecture is on (int): 6 Please Enter a Valid Weekday Integer!
c	Invalid time	 Enter new Start Time (HH:mm): 78: 90 Error!! Please Use the Correct Time Format 10:00 Start Time updated successfully	d	Invalid date	 Enter new Start Date (yyyy-mm-dd): 2020-13-32 Error!! Please Use the Correct Date Format 2020-08-17 Start Date updated successfully
e	Course code that does not exist	 ===========CHECK VACANCIES AVAILABLE=========== Enter the Course Code to check: cz2003 Course cz2003 does not exist! Please enter the Course Code again: (or enter 0 to cancel)	f	Index number that does not exist	 ===========CHECK VACANCIES AVAILABLE=========== Enter the Course Code to check: cz2002 List of Indexes: 1011 1012 1013 Enter the Index to check: 1014 Invalid Index Number (1) Enter another Index Number to check (2) Return to main menu (3) Exit

b. Student Login

a	Login before allowed period (dates)	 Login as student or admin 1: Student 2: Admin 0: Exit 1 Please access it during 00:00 - 07:00 from 2020-11-17 to 2020-12-30 Login as student or admin 1: Student 2: Admin 0: Exit	c	Invalid Username	 Login as student or admin 1: Student 2: Admin 0: Exit 1 Please enter your Matric Number: 016 Student 016 does not exist! Please enter the Matric Number again: (or enter 0 to cancel)
d	Wrong password	 Please enter your Matric Number: 001 Please enter your password: password2 Wrong Password! Please enter your Matric Number:			

c. Register student for a course

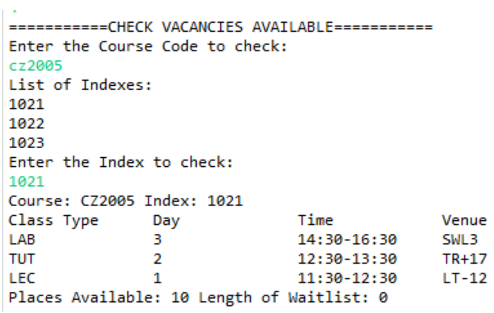
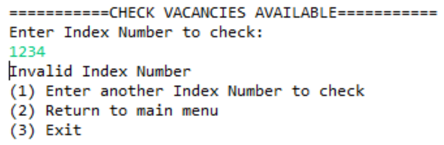
a	Add a student to a course index with available vacancies		b	Add a student to a course index with 0 vacancies in Tut/Lab	
c	Register the same course again				

```
==========ADD COURSE==========
Enter Index Number to add:
1021
Course: CZ2005 Index: 1021
Class Type      Day of Week      Time            Venue
LAB             3                14:30-16:30     SWL3
TUT             2                12:30-13:30     TR+17
LEC             1                11:30-12:30     LT-12
Sending Email..
Notification email has been sent
Course CZ2005 has been successfully saved
```

```
==========ADD COURSE==========
Enter Index Number to add:
1011
Course: CZ2002 Index: 1011
Class Type      Day of Week      Time            Venue
LAB             2                08:30-10:30     SWL1
TUT             3                10:30-11:30     TR+16
LEC             1                08:30-09:30     LT-11
Sending Email..
Notification email has been sent
Course CZ2002 has been successfully saved
You have been added to waiting list
(1) Enter another Index Number to add
(2) Return to main menu
(3) Exit
```

```
==========ADD COURSE==========
Enter Index Number to add:
1012
Course: CZ2002 Index: 1012
Class Type      Day of Week      Time            Venue
LAB             4                10:30-12:30     SWL1
TUT             2                14:30-15:30     TR+16
LEC             1                08:30-09:30     LT-11
You are already in CZ2002
(1) Enter another Index Number to add
(2) Return to main menu
(3) Exit
```

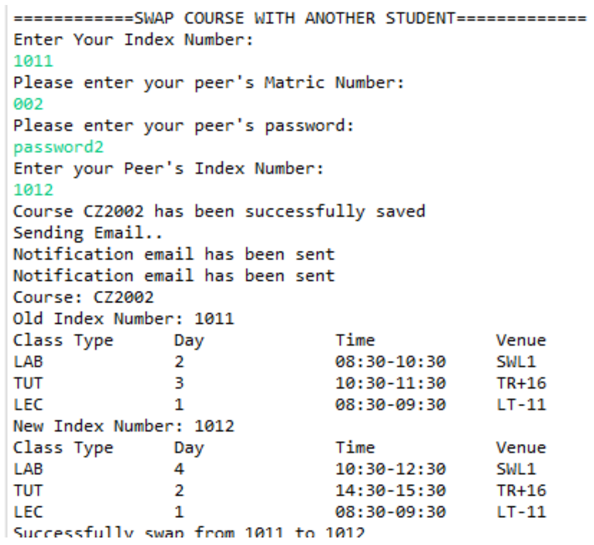
d. Check available slot in a class (vacancy in a class)

a	Check for vacancy in course index		b	Invalid data entries (e.g. course code, class code etc)	

```
===========CHECK VACANCIES AVAILABLE===========
Enter the Course Code to check:
cz2005
List of Indexes:
1021
1022
1023
Enter the Index to check:
1021
Course: CZ2005 Index: 1021
Class Type      Day             Time            Venue
LAB             3               14:30-16:30     SWL3
TUT             2               12:30-13:30     TR+17
LEC             1               11:30-12:30     LT-12
Places Available: 10 Length of Waitlist: 0
```

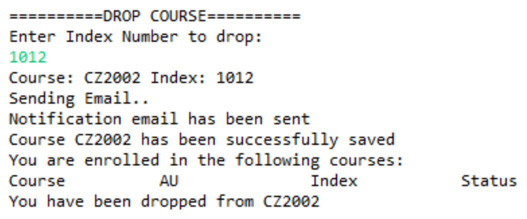
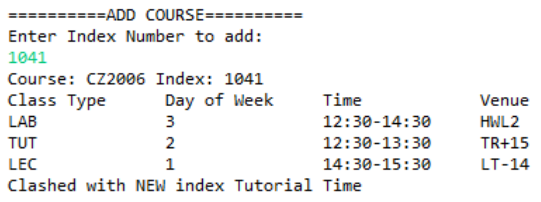
```
===========CHECK VACANCIES AVAILABLE===========
Enter Index Number to check:
1234
Invalid Index Number
(1) Enter another Index Number to check
(2) Return to main menu
(3) Exit
```

e. Swop Index Number with another student

a	Swop index number with another student	

```
============SWAP COURSE WITH ANOTHER STUDENT=============
Enter Your Index Number:
1011
Please enter your peer's Matric Number:
002
Please enter your peer's password:
password2
Enter your Peer's Index Number:
1012
Course CZ2002 has been successfully saved
Sending Email..
Notification email has been sent
Notification email has been sent
Course: CZ2002
Old Index Number: 1011
Class Type      Day             Time            Venue
LAB             2               08:30-10:30     SWL1
TUT             3               10:30-11:30     TR+16
LEC             1               08:30-09:30     LT-11
New Index Number: 1012
Class Type      Day             Time            Venue
LAB             4               10:30-12:30     SWL1
TUT             2               14:30-15:30     TR+16
LEC             1               08:30-09:30     LT-11
Successfully swap from 1011 to 1012
```

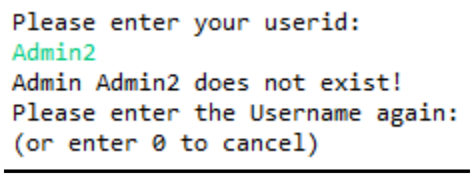
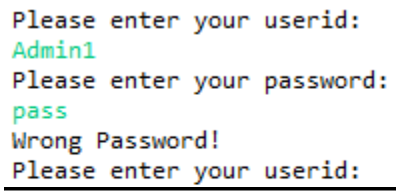
f. Drop student for course **g. Day/Time clash with other course**

Drop a course			Add a student to a course index with available vacancies	

```
==========DROP COURSE==========
Enter Index Number to drop:
1012
Course: CZ2002 Index: 1012
Sending Email..
Notification email has been sent
Course CZ2002 has been successfully saved
You are enrolled in the following courses:
Course          AU              Index           Status
You have been dropped from CZ2002
```

```
==========ADD COURSE==========
Enter Index Number to add:
1041
Course: CZ2006 Index: 1041
Class Type      Day of Week     Time            Venue
LAB             3               12:30-14:30     HWL2
TUT             2               12:30-13:30     TR+15
LEC             1               14:30-15:30     LT-14
Clashed with NEW index Tutorial Time
```

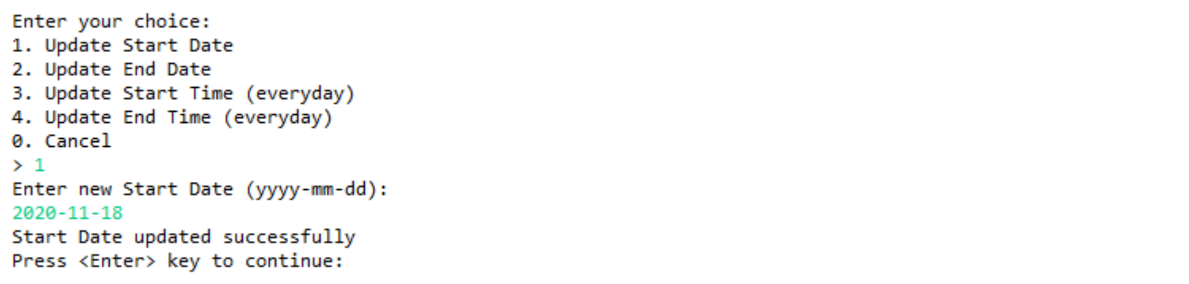
h. Admin login

a	Invalid username		b	Wrong password	

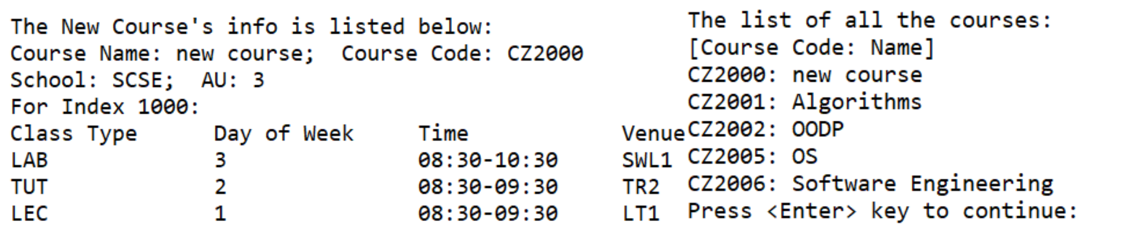
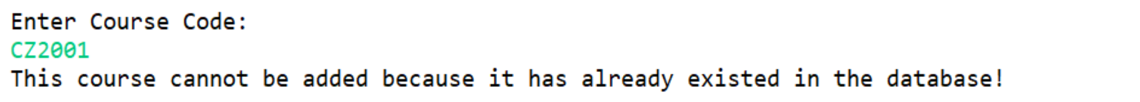
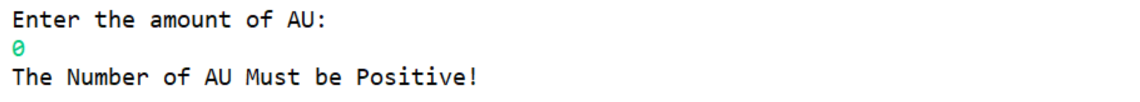
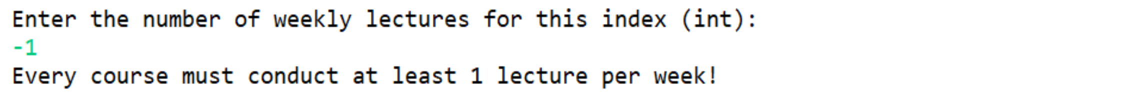
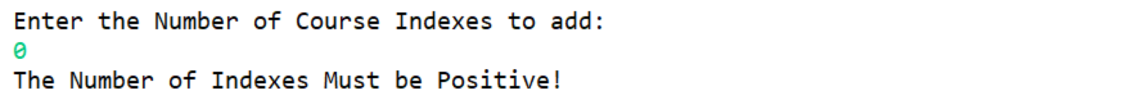
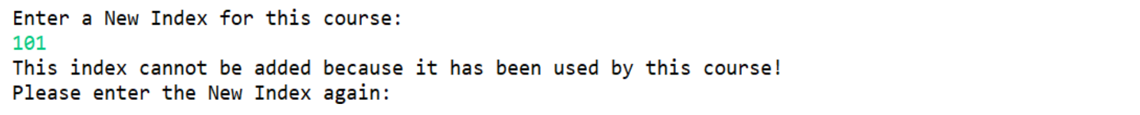
```
Please enter your userid:
Admin2
Admin Admin2 does not exist!
Please enter the Username again:
(or enter 0 to cancel)
```

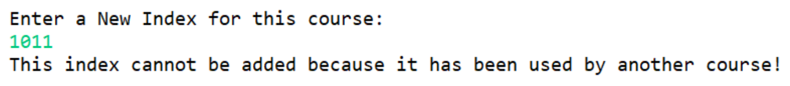
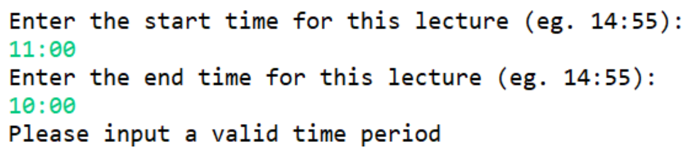
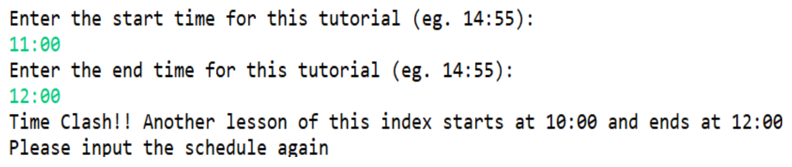
```
Please enter your userid:
Admin1
Please enter your password:
pass
Wrong Password!
Please enter your userid:
```

i. Edit student access period

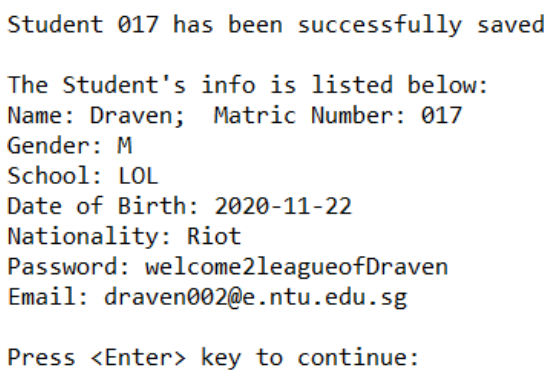
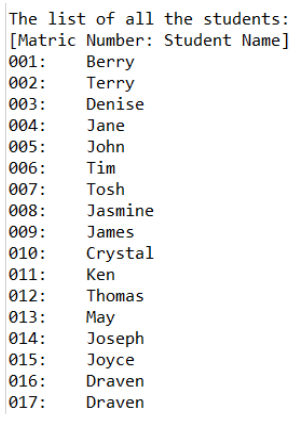
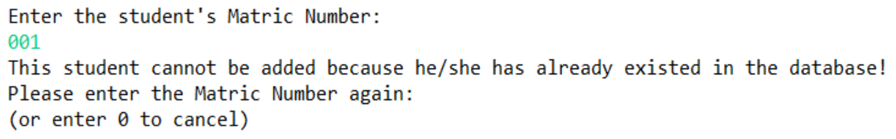
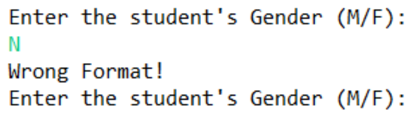
a	Update access period successfully	 Enter your choice: 1. Update Start Date 2. Update End Date 3. Update Start Time (everyday) 4. Update End Time (everyday) 0. Cancel > 1 Enter new Start Date (yyyy-mm-dd): 2020-11-18 Start Date updated successfully Press <Enter> key to continue:

j. Add a course

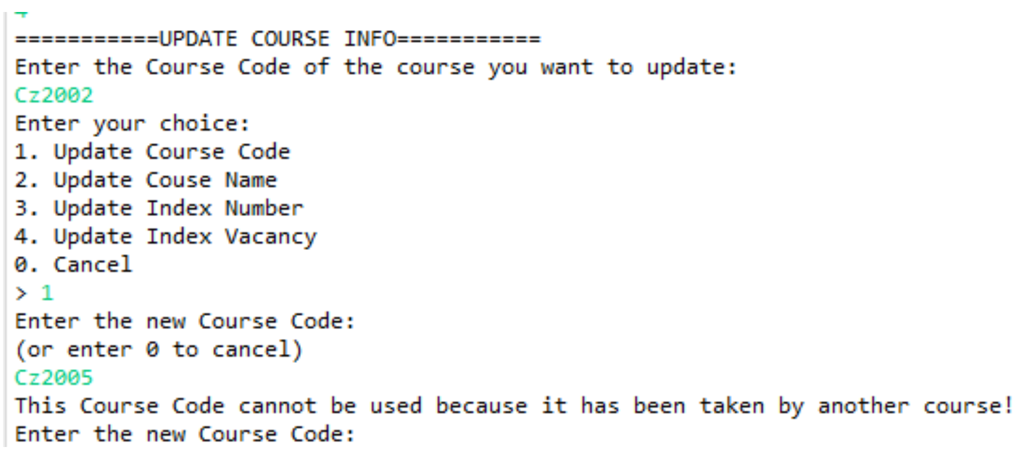
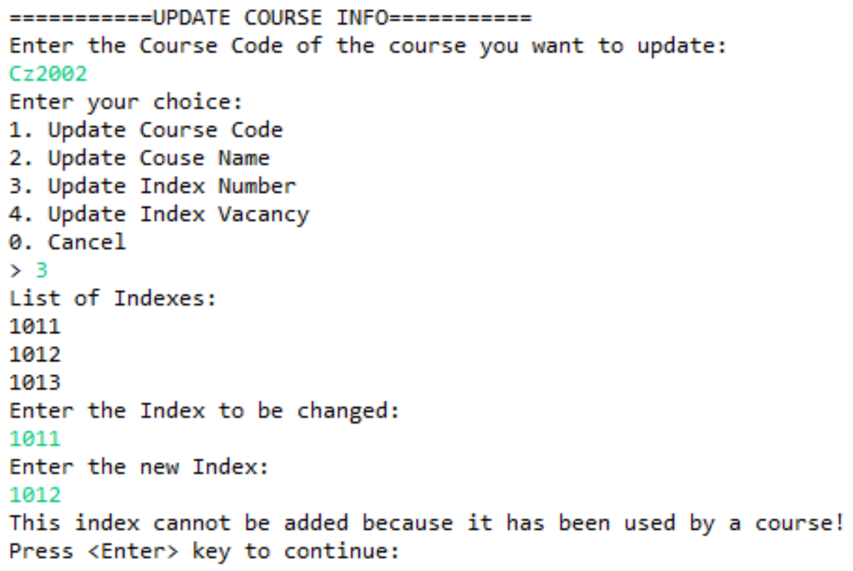
a	Add a new course	 The New Course's info is listed below: Course Name: new course; Course Code: CZ2000 School: SCSE; AU: 3 For Index 1000: Class Type Day of Week Time Venue LAB 3 08:30-10:30 SWL1 TUT 2 08:30-09:30 TR2 LEC 1 08:30-09:30 LT1 The list of all the courses: [Course Code: Name] CZ2000: new course CZ2001: Algorithms CZ2002: OODP CZ2005: OS CZ2006: Software Engineering Press <Enter> key to continue:
b	Add an existing course	 Enter Course Code: CZ2001 This course cannot be added because it has already existed in the database!
	Non-positive number of AU	 Enter the amount of AU: 0 The Number of AU Must be Positive!
	Non-positive number of lectures	 Enter the number of weekly lectures for this index (int): -1 Every course must conduct at least 1 lecture per week!
	Non-positive number of indexes	 Enter the Number of Course Indexes to add: 0 The Number of Indexes Must be Positive!
	New index is already in the new course	 Enter a New Index for this course: 101 This index cannot be added because it has been used by this course! Please enter the New Index again:

	New index is already in another course	 Enter a New Index for this course: 1011 This index cannot be added because it has been used by another course!
	End time is earlier than start time	 Enter the start time for this lecture (eg. 14:55): 11:00 Enter the end time for this lecture (eg. 14:55): 10:00 Please input a valid time period
	Time clash within a course	 Enter the start time for this tutorial (eg. 14:55): 11:00 Enter the end time for this tutorial (eg. 14:55): 12:00 Time Clash!! Another lesson of this index starts at 10:00 and ends at 12:00 Please input the schedule again

k. Add a student

a	Add a new student	 Student 017 has been successfully saved The Student's info is listed below: Name: Draven; Matric Number: 017 Gender: M School: LOL Date of Birth: 2020-11-22 Nationality: Riot Password: welcome2leagueofDraven Email: draven002@e.ntu.edu.sg Press <Enter> key to continue:  The list of all the students: [Matric Number: Student Name] 001: Berry 002: Terry 003: Denise 004: Jane 005: John 006: Tim 007: Tosh 008: Jasmine 009: James 010: Crystal 011: Ken 012: Thomas 013: May 014: Joseph 015: Joyce 016: Draven 017: Draven
b	Add an existing matric number	 Enter the student's Matric Number: 001 This student cannot be added because he/she has already existed in the database! Please enter the Matric Number again: (or enter 0 to cancel)
c	Invalid gender (not “M” or “F”)	 Enter the student's Gender (M/F): N Wrong Format! Enter the student's Gender (M/F):

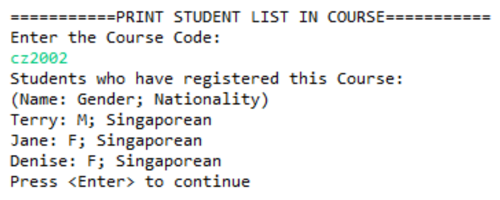
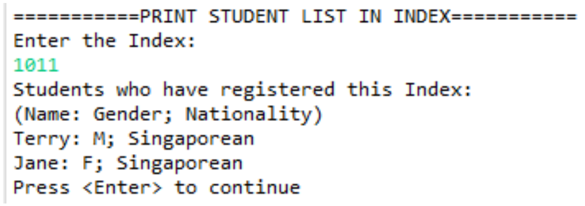
l. Update course

a	Set the new course code to be an existing one	
b	Set the new index number to be an existing one (used by this course / another course)	

```
===========UPDATE COURSE INFO===========
Enter the Course Code of the course you want to update:
Cz2002
Enter your choice:
1. Update Course Code
2. Update Couse Name
3. Update Index Number
4. Update Index Vacancy
0. Cancel
> 1
Enter the new Course Code:
(or enter 0 to cancel)
Cz2005
This Course Code cannot be used because it has been taken by another course!
Enter the new Course Code:
```

```
===========UPDATE COURSE INFO===========
Enter the Course Code of the course you want to update:
Cz2002
Enter your choice:
1. Update Course Code
2. Update Couse Name
3. Update Index Number
4. Update Index Vacancy
0. Cancel
> 3
List of Indexes:
1011
1012
1013
Enter the Index to be changed:
1011
Enter the new Index:
1012
This index cannot be added because it has been used by a course!
Press <Enter> key to continue:
```

m. For Admin, print student list by index number and course:

Print list by course			Print list by index	

```
===========PRINT STUDENT LIST IN COURSE===========
Enter the Course Code:
cz2002
Students who have registered this Course:
(Name: Gender; Nationality)
Terry: M; Singaporean
Jane: F; Singaporean
Denise: F; Singaporean
Press <Enter> to continue
```

```
===========PRINT STUDENT LIST IN INDEX===========
Enter the Index:
1011
Students who have registered this Index:
(Name: Gender; Nationality)
Terry: M; Singaporean
Jane: F; Singaporean
Press <Enter> to continue
```

9. 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 (CE 2002/CZ 2002)	LAB GROUP	SIGNATURE
Favian Chan Jun Wei	CZ2002	SS2 Group 4	[illegible]
Huang Runtao	CZ2002	SS2 Group 4	[illegible]
Kothari Khush Milan	CZ2002	SS2 Group 4	[illegible]
Tan Chin Hoe, Samuel	CZ2002	SS2 Group 4	[illegible]
Yin Ning	CZ2002	SS2 Group 4	尹 宁