



**NANYANG
TECHNOLOGICAL
UNIVERSITY**
SINGAPORE

CE2002 OBJECT ORIENTED DESIGN AND PROGRAMMING 2020/2021 SEMESTER 1

BUILDING AN OO APPLICATION

MySTARS
My Student Automated Registration System

SE1 GROUP 1

TEAM MEMBERS:

ABHIGYAN SINGH

ASURI SIMHAKUTTY KAMAKSHI

CHOCKALINGAM KASI

LIM JUN WEI

MOSHIK SEETLOO

Appendix B:





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 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	Lab Group	Signature/Date
ABHIGYAN SINGH	CE2002	SE1	 22/11/2020
ASURI SIMHAKUTTY KAMAKSHI	CE2002	SE1	 22/11/2020
CHOCKALINGAM KASI	CE2002	SE1	K. Chockalingam 22/11/2020
LIM JUN WEI	CE2002	SE1	 22/11/2020
MOSHIK SEETLOO	CE2002	SE1	 22/11/2020

Important notes:

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

INTRODUCTION

My STudent Automated Registration System(MySTARS) is a university application built for administrators and for undergraduate students. The role of the console is to serve as a medium for admin personnel to create courses being taught in the university and secondly to help the students register them for the courses for their semester. The first role would be administrator mode. As an administrator, they have the access to create and maintain courses that will be taken up by students. As a student, you will be able choose courses, change index and swap index according to their timetable. There will be a dedicated registration period where the students will be required to register the courses. Each course will have its unique course code, course name, index information and class schedules depending on a combination of lectures, tutorial and laboratory sessions. When students who are interested to register in a specific index realize that the index has no more vacant slots they will be placed on a waiting list. Email notifications will be sent to their personal email address at various stages of their registration such as Student creation, Course registration and swapping of indexes.

This report covers the Design Principles and object-oriented programming concepts required to develop this application. The design of the code will be represented with a UML Class Diagram and a UML Sequence Diagram to demonstrate the print function. The diagrams show the interaction and relationships between objects. The test cases and their results are included to prove the smooth functionality of the console. A software demo has been uploaded to YouTube. The link to the video is available in this report.

CONSOLE FUNCTIONAL REQUIREMENTS

STUDENT
<ol style="list-style-type: none">1. Add Course - <i>add course after checking for clash with other courses registered courses</i>2. Drop Course - <i>drop registered courses, allows students on waitlist to be registered automatically</i>3. Check/Print Courses Registered - <i>print courses registered for each unique student</i>4. Check Vacancies Available - <i>check for vacant slots among the unique index</i>5. Change Index Number of Course - <i>drop and change to a new index accordingly</i>6. Swop Index Number with Another Student - <i>flexible peer to peer exchange of index number</i>
ADMIN
<ol style="list-style-type: none">1. Edit Student Access Period - <i>prevents students from access if not within stipulated time frame</i>2. Add a student- <i>store student credentials (name, matric number, gender, nationality and email)</i>3. Add/Update a course - <i>modification of course code, school, index numbers and vacancies</i>4. Check available slot for an index number - <i>check vacancy</i>5. Print student list by index number - <i>print student information by index number</i>6. Print student list by course - <i>print student's name,gender and nationality</i>

DESIGN CONSIDERATIONS AND PRINCIPLES

APPROACH

Our console has to incorporate various features to ensure it is able to serve users efficiently and in a robust manner. We had to take into account various considerations such as Encapsulation, Cohesion and Coupling to ensure the functionality of our console. Encapsulation is used to ensure that values or contents of the console are hidden inside the class, preventing unauthorized personnel from accessing the data. This is important as student information should be protected and only be accessed by the individual student. It is a mechanism to provide various levels of protection for the data ranging from public, private and protected. Coupling is to ensure that the boundary classes, control classes and entity classes are highly cohesive objects that are loosely connected at the same time. It is important to maintain connectivity and also at the same time ensure that a change in a section in the code does not lead to major changes around the entire code. An example of cohesion would be the file manager class and email waiting list which automatically notifies students through emails on registration changes.

Password Masking - passwords have to be masked to prevent users from knowing the content. Eclipse does not support masking of passwords. The actual code needed for password masking is below for reference.

```
Console console =System.console();  
if(console==null){  
System.err.println("No console found");  
System.exit(1);  
}  
char[] pass = console.readPassword("Please enter password:");  
String password =String.valueOf(pass);
```

Note: The first object to be created is commented out as it is only required once.

DESIGN PRINCIPLES (SOLID)

1. Single Responsibility Principle(SRP)

Single Responsibility feature is to ensure that the class has its own unique responsibility and it should not be coupled with other responsibilities to make the functionality of the program straightforward. There should never be more than one reason for a class to change, making the code cohesive. This principle is demonstrated by our entity classes. For example, *course* class maintains information about course name, course code, number of indexes, academic units and *indexNO* class contains information on the number of students enrolled. There is no overlap of responsibilities in our classes and so limiting the information and methods that are used by the class. These ensure that our console is cohesive and each class is able to carry its tasks without having to balance multiple roles at same time.

2. Open-Closed Principle(OCP)

The principle states that the module should be open for extension but closed for modification. The Open-Closed principle is a feature that allows us to build more functions on existing sets of modules without making changes. Hence we can carry out more tasks without making changes to the source code. This is done through abstractions. This feature is found in the *student*, *admin* and *course* interface. In *student* class we can add more students into the system, register them for courses. In *course* class, we have the flexibility to create, modify and remove courses from the system. We can carry out all these functions without having to edit the program code. These interfaces satisfy the open-closed principle which allows us to make changes to the module without making changes to the source code of the modules. This is very important as the code should become “self-sustainable” without having to go through constant alterations.

3. Interface Segregation Principle(ISP)

Interface Segregation Principle means that the many client specific interfaces are better than one general purpose interface. This is to help us streamline our functions and only depend on important functions. We should not be forced to rely on interfaces that we do not need. We can separate them and use the interfaces as per need. Hence, we have decided to create multiple interfaces instead of depending on a single interface with an overarching purpose. Different interfaces such as *admin*, *student* and *course* interface are segregated and have unique methods which can be called upon the requirement that the code. By satisfying the Interface Segregation Principle, we can avoid overpopulating our classes with extra functions that may not be used.

4. Dependency Injection Principle

We have programmed our application in such a way that the higher level modules are not dependent on lower level modules. For example the *course* boundary class does not depend on the *starwars* class in our code. As a result it will allow us to use the higher level modules that are independent and can be used frequently.

USE OF OBJECT ORIENTED CONCEPTS

1. Composition

Composition is a “has a” relationship where the parts are dependent on the whole. The whole creates the parts and the parts are closely linked to the whole. When the whole is deleted the parts are deleted as well. There is only a one to one relationship where one whole can have only one part. We implemented a composition relationship in our code with the *indexNO* class and *coursetype* class. The *indexNO* class contains information on index number, index vacancy and it has a *coursetype* class which contains information on lesson timings and lesson venues. The *coursetype* class can only be created by the *indexNO* class and only exists if the *indexNO* class exists. Hence we are able to demonstrate the Composition principle.

2. Dependency

Dependency is a temporary connection between two classes that are not associated and related. They have a loose temporary relationship. Example would be the dependency relationship between *courseinterface* and *courseclass* and the dependency relationship between *studentinterface* and *student* class. These interfaces are temporarily related to one another, they use this temporary link to share information.

3. Inheritance

Inheritance feature allows us to extend or modify an existing class. It is an inheritance “is-a” relationship between two objects. Using inheritance we are able to use the code in the super class. An example of how inheritance is displayed by the superclass *serializable* that is parent class to entity classes. Example, Entity class “is a” serializable object. We are able to inherit properties from the serializable object and are able to use its function to write the data into binary file format.

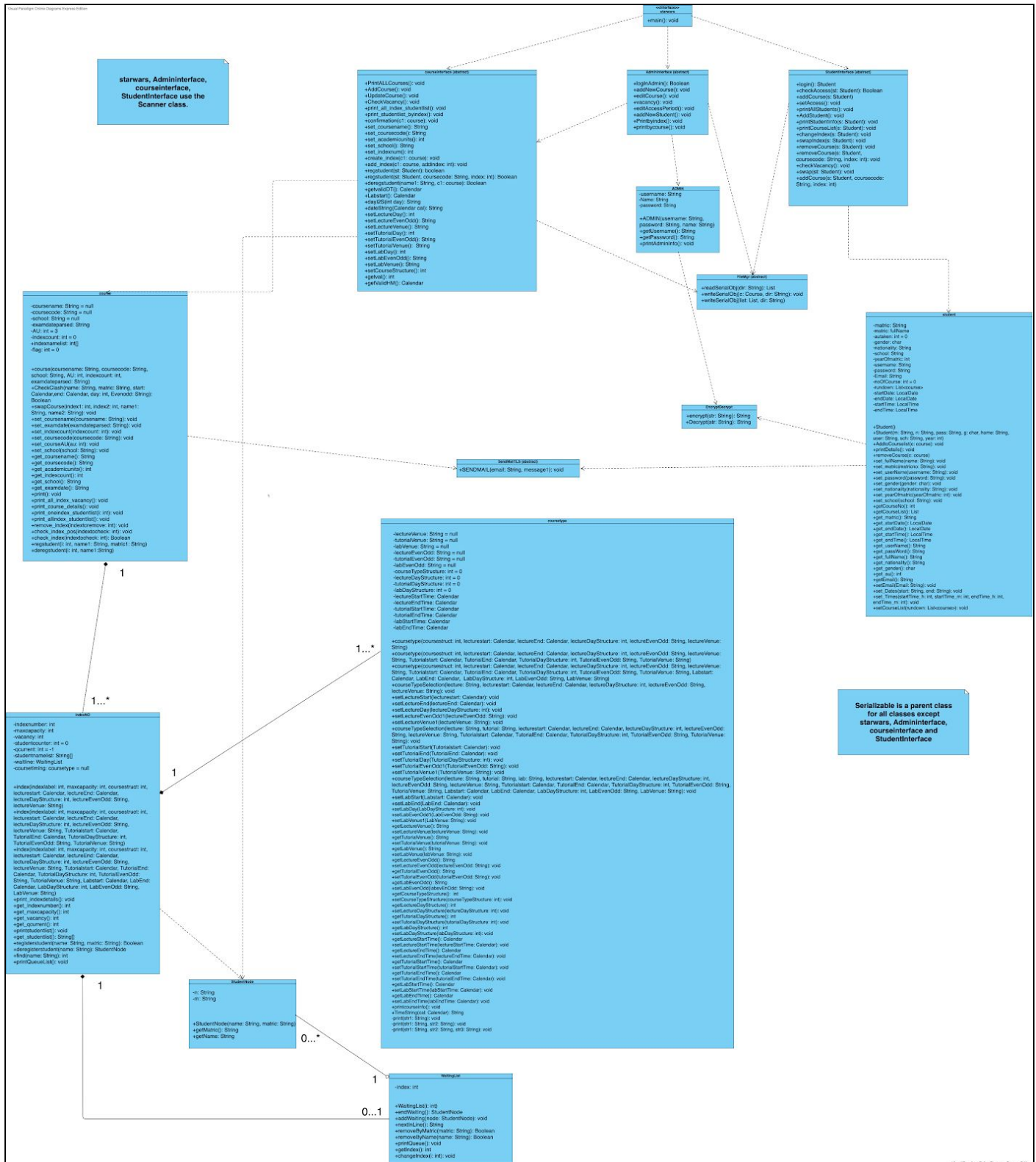
4. Abstraction

Abstraction means that the information is hidden from other objects and only reveals essential characteristics. It only provides a set of methods such as get and set methods for accessing and changing the private values. External objects will not be able to identify the way the data is stored. It is not visible. For example students can make changes or retrieve details from their course using the getter and setter methods. This provides confidentiality on the process that occurs in the background.

5. Polymorphism

Polymorphism is the ability to exist in many forms. The methods that are called are not known at compile time and are dynamically decided at runtime. This is known as dynamic polymorphism. This is observed in the *coursetype* class where we are able to decide the structure of the lessons. If a certain module has a lecture, tutorial and lab. It utilizes function call on lecture and tutorial methods to get the details accordingly.

UML CLASS DIAGRAM



Link to Class Diagram Image File:

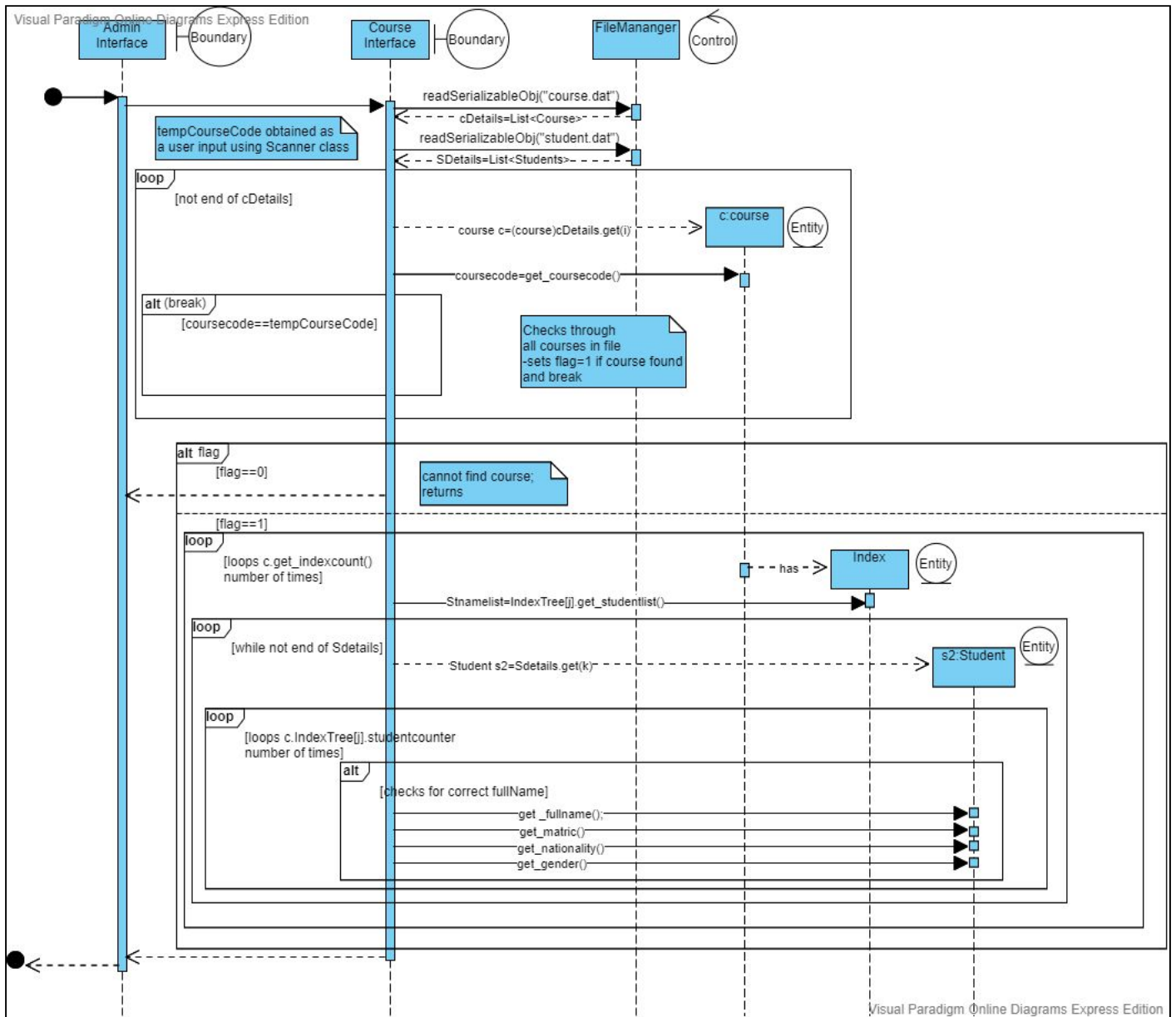
https://drive.google.com/file/d/1pDVVxjKCJxITwXveBO81K_MzGYa0PINo/view?usp=sharing

Link to Class Diagram Source File:

<https://drive.google.com/file/d/14pfDJZegX67IUE7b-jo2kN-6WvJJQElg/view?usp=sharing>

UML SEQUENCE DIAGRAM

[print student list by course function]



Link to Sequence Diagram Image File:

<https://drive.google.com/file/d/1dDoj6TfcTdu3ezMcJ4go34PKXN91TYGs/view?usp=sharing>

Link to Sequence Diagram Source File:

<https://drive.google.com/file/d/12xxK07UviLJSv-ByKJrZySe9lSlFxFfv/view?usp=sharing>

SOFTWARE DEMONSTRATION YOUTUBE LINK

<https://youtu.be/NkJOLzZmAcQ>


TEST CASES

1. Student Login

a	Login before allowed period (dates)	<pre> Welcome to Star Wars Choose your Domain 1) Admin 2) Student 0) Exit 2 ----- Access not allowed You are accessing it early. Try again within Access period LOGIN FAILED TRY AGAIN </pre>
b	Login after allowed period (dates)	<pre> Welcome to Star Wars Choose your Domain 1) Admin 2) Student 0) Exit 2 ----- Access not allowed You are accessing it Late. Try again within Access period LOGIN FAILED TRY AGAIN </pre>
c	Wrong password	<pre> Welcome to Star Wars Choose your Domain 1) Admin 2) Student 0) Exit 2 ----- Username: ABHIGYAN001 Password: ABHIGYAN001 Wrong password LOGIN FAILED TRY AGAIN </pre>
d	Wrong username	<pre> Welcome to Star Wars Choose your Domain 1) Admin 2) Student 0) Exit 2 ----- Username: ABHI001 Password: A001 Wrong username LOGIN FAILED TRY AGAIN </pre>

2. Add a student

a	<p>Add a new student</p> <p>Left Picture: abhiyan student created</p> <p>password encrypted</p> <p>Right: chocka student created after abhigyan</p>	<pre> (ADMIN) Main Menu 1. Change access period 2. Add a Student 3. Add a Course 4. Update a Course 5. Check availability slot for an Index Number 6. Print Student List by Index Number 7. Print Student List by Course 0. Sign Out ----- 2 Student Name: Abhigyan Singh Matriculation Number: U1922629G Gender: M Nationality: Indian Network Username: ABHIGYAN001 Network Password: A001 Enter Email: ABHIGYAN001@e.ntu.edu.sg School: SCSE Year of matriculation: 2019 1 Name : Abhigyan Singh PASS : {zzr Matric number : U1922629G Gender : M Username : ABHIGYAN001 Nationality : Indian School : SCSE year of matric : 2019 </pre>	<pre> 2 Student Name: CHOCKA Matriculation Number: U1920428E Gender: M Nationality: Singaporean Network Username: CHOC001 Network Password: C001 Enter Email: CHOC0010@e.ntu.edu.sg School: SCSE Year of matriculation: 2019 2 Name : Abhigyan Singh PASS : {zzr Matric number : U1922629G Gender : M Username : ABHIGYAN001 Nationality : Indian School : SCSE year of matric : 2019 Name : CHOCKA PASS : #zzr Matric number : U1920428E Gender : M Username : CHOC001 Nationality : Singaporean School : SCSE year of matric : 2019 </pre>
---	---	---	--

b	<p>Add an existing student</p> <p>adding abhiyan again</p>	<pre> Student Name: Abhigyan Singh Matriculation Number: U1922629G Gender: M Nationality: Indian Network Username: ABHIGYAN001 Network Password: A001 Enter Email: ABHIGYAN001@e.ntu.edu.sg School: SCSE Year of matriculation: 2019 Student already exists! try again! </pre>
c	<p>Invalid data entries</p>	<pre> Nationality: INDIAN Network Username: JUN001 Network Password: j001 Enter Email: JUN001@kdsdsj.com School: SCSE Year of matriculation: 2021 You cannot enter a future year please try registering the student again! </pre>
d	<p>Notification on change in Access Period Extra</p>	<div> <pre> (ADMIN) Main Menu 1. Change access period 2. Add a Student 3. Add a Course 4. Update a Course 5. Check availability slot for an Index Number 6. Print Student List by Index Number 7. Print Student List by Course 0. Sign Out ----- 1 Enter start date (Format: YYYY-MM-DD) 2020-11-16 Enter end date (Format: YYYY-MM-DD) 2020-12-20 Enter start time: hours 0 Enter start time: minutes 0 Enter end time: hours 0 Enter end time: minutes 0 </pre> </div> <div> <p>COURSE REGISTRATION UPDATE</p> <p> prayingforbetteryear2020@gmail.com <prayingforbetteryear2020@gmail.com> 15/11/2020 10:50 pm</p> <p>To: #CHOCKALINGAM KASI#</p> <p>PLEASE NOTE THAT THE ACCESS DATE HAS CHANGED TO : FROM 2020-11-16 To 2020-12-20 FROM TIME : 0:0 to 0:0</p> <p>Regards STARS TEAM</p> </div>

3. Add a course

a	<p>Add a new course</p> <p>(with combination of (ii) from above)</p>	<pre> 1.Monday 2.Tuesday 3.Wednesday 4.Thursday 5.Friday 6.Saturday 7.Sunday Choose: 3 Selected Day is Wednesday Is the lesson held: Press 0 for Weekly Press 1 for odd Press 2 for even: 2 Enter lab venue: HW lab 1 Updated lab venue: HW 4 Course code CX2001 Course name Algorithms School name SCSE Au 3 Course code CX1003 Course name Data Structures School name SCSE Au 3 Course code CX2002 Course name OODP School name SCSE Au 3 Course code CX1001 Course name Digital Logic School name SCSE Au 3 </pre>
---	--	--

b	Add an existing course	<pre> 1. Change access period 2. Add a Student 3. Add a Course 4. Update a Course 5. Check availability slot for an Index Number 6. Print Student List by Index Number 7. Print Student List by Course 0. Sign Out ----- 3 Enter Course Name: Algorithms Enter Course Code: CX2001 The course already exists! Try again 1 Course code CX2001 Course name Algorithms School name SCSE Au 3 </pre>
c	Invalid data entries	<pre> (ADMIN) Main Menu 1. Change access period 2. Add a Student 3. Add a Course 4. Update a Course 5. Check availability slot for an Index Number 6. Print Student List by Index Number 7. Print Student List by Course 0. Sign Out ----- 3 Enter Course Name: Algorithms Enter Course Code: CX2001 Enter the school offering the course: SCSE Enter Academic Units: 3 How many indexes do you plan to Add: 1 Enter Exam Date and Time (dd/MM/yyyy HH:mm): 12/12/2020 Wrong format! Enter Exam Date and Time (dd/MM/yyyy HH:mm): 13/13/2020 Wrong format! Enter Exam Date and Time (dd/MM/yyyy HH:mm): </pre>

4. Register student for a course

a	Add a student to a course index with available vacancies.	<pre> (STUDENT) Main Menu 1. ADD a course 2. Drop a Student 3. Print courses registered 4. Check index vacancy 5. Change index for a course 6. Swap index for a course 0. Sign Out ----- 1 Enter Course code : CX2001 Enter Index Number : 10001 1. Course Code: CX2001 2. Course Name: Algorithms 3. Academic Units: 3 4. Exam Date: 13/12/2020 15:00 5. School: SCSE 6. Number of Indexes: 1 Index List: 10001 Press 0 for 10001 with vacancy 1 Confirm the number of the respective Index to register the student0 Course code CX2001 Course name Algorithms School name SCSE Au 3 </pre>
b	Add a student to a course index with 0 vacancies in Tut / Lab.	<pre> Username: CHOC001 Password: C001 ----- (STUDENT) Main Menu 1. ADD a course 2. Drop a Student 3. Print courses registered 4. Check index vacancy 5. Change index for a course 6. Swap index for a course 0. Sign Out ----- 1 Enter Course code : CX2001 Enter Index Number : 10001 1. Course Code: CX2001 2. Course Name: Algorithms 3. Academic Units: 3 4. Exam Date: 13/12/2020 15:00 5. School: SCSE 6. Number of Indexes: 1 Index List: 10001 Press 0 for 10001 with vacancy 0 Confirm the number of the respective Index to register the student0 Added to waiting list! </pre>

c	Register the same course again	<pre> (STUDENT) Main Menu 1. ADD a course 2. Drop a Student 3. Print courses registered 4. Check index vacancy 5. Change index for a course 6. Swap index for a course 0. Sign Out ----- 1 Enter Course code : CX2001 Enter Index Number : 10001 You are already registered to this course! Try again! ----- </pre>
d	Invalid data entries (eg wrong student ID / course code, etc)	<pre> ----- Username: ABHIGYAN001 Password: A001 ----- (STUDENT) Main Menu 1. ADD a course 2. Drop a Student 3. Print courses registered 4. Check index vacancy 5. Change index for a course 6. Swap index for a course 0. Sign Out ----- 1 Enter Course code : CX1000 Enter Index Number : 1001 Wrong course name. ----- </pre>

5. Check available slot in a class

a	Check for vacancy in course index	<pre> ----- (ADMIN) Main Menu 1. Change access period 2. Add a Student 3. Add a Course 4. Update a Course 5. Check availability slot for an Index Number 6. Print Student List by Index Number 7. Print Student List by Course 0. Sign Out ----- 5 Enter the Subject code : CX2001 Enter the index Number : 10001 Vacancy for the index 10001 is 0/1 ----- </pre>	<pre> ----- (ADMIN) Main Menu 1. Change access period 2. Add a Student 3. Add a Course 4. Update a Course 5. Check availability slot for an Index Number 6. Print Student List by Index Number 7. Print Student List by Course 0. Sign Out ----- 5 Enter the Subject code : CX2002 Enter the index Number : 10003 Vacancy for the index 10003 is 1/1 ----- </pre>
b	Invalid data entries (eg course code, class code etc)	<pre> ----- (ADMIN) Main Menu 1. Change access period 2. Add a Student 3. Add a Course 4. Update a Course 5. Check availability slot for an Index Number 6. Print Student List by Index Number 7. Print Student List by Course 0. Sign Out ----- 5 Enter the Subject code : CX1000 Enter the index Number : 10001 No Such course found ----- </pre>	<pre> ----- (ADMIN) Main Menu 1. Change access period 2. Add a Student 3. Add a Course 4. Update a Course 5. Check availability slot for an Index Number 6. Print Student List by Index Number 7. Print Student List by Course 0. Sign Out ----- 5 Enter the Subject code : CX2001 Enter the index Number : 10000 No such index found. ----- </pre>

6. Day/Time clash with other course

a	Add a student to a course index with available vacancies.	<pre> Username: CHOC001 Password: C001 ----- (STUDENT) Main Menu 1. ADD a course 2. Drop a Student 3. Print courses registered 4. Check index vacancy 5. Change index for a course 6. Swap index for a course 0. Sign Out ----- 1 Enter Course code : CX2002 Enter Index Number : 10003 1. Course Code: CX2002 2. Course Name: OODP 3. Academic Units: 3 4. Exam Date: 17/12/2020 15:00 5. School: SCSE 6. Number of Indexes: 1 Index List: 10003 There is a clash on Monday! Do you still want to continue (y/n)?n </pre>
---	---	---

7. Waitlist notification

a(i)	Add studentA to a course index with 0 vacancies	<pre> Username: CHOC001 Password: C001 ----- (STUDENT) Main Menu 1. ADD a course 2. Drop a Student 3. Print courses registered 4. Check index vacancy 5. Change index for a course 6. Swap index for a course 0. Sign Out ----- 1 Enter Course code : CX2001 Enter Index Number : 10001 1. Course Code: CX2001 2. Course Name: Algorithms 3. Academic Units: 3 4. Exam Date: 13/12/2020 15:00 5. School: SCSE 6. Number of Indexes: 1 Index List: 10001 Press 0 for 10001 with vacancy 0 Confirm the number of the respective Index to register the student0 Added to waiting list! </pre>
a(ii)	Drop studentB from the same course index	<pre> starwars [Java Application] /Library/Java/JavaVirtualMachines/adoptopenjdk-11-openj9/jdk/Contents/Home/bin/java (Nov 15, 2020, 11:07:00 PM) 1. ADD a course 2. Drop a Student 3. Print courses registered 4. Check index vacancy 5. Change index for a course 6. Swap index for a course 0. Sign Out ----- 2 Enter the course code of the course you want to remove: CX2001 Printing all students enrolled in this course Algorithms 0 10001 Student Count: 1 Student Vacancy: 0 Abhigyan Singh Enter index number 10001 Confirm Deregistration of Abhigyan Singh Press 0 to Deregister or Press any number to cancel Deregistration 0 Deregistration successful </pre> <p>COURSE REGISTRATION UPDATE</p> <p> prayingforbetteryear2020@gmail.com <prayingforbetteryear2020@gmail.com> 15/11/2020 11:08 pm</p> <p>To: #CHOCKALINGAM KASI#</p> <p>HELLO CHOCKA(U1920428E) ! CONGRATULATIONS YOU HAVE BEEN REGISTERED FOR THE FOLLOWING COURSE! Course name: Algorithms Course code: CX2001 Number of AU: 3 Index number: 10001</p> <p>Regards STARS TEAM</p>

a(iii)	Display studentA timetable	<pre> Username: CHOC001 Password: C001 ----- (STUDENT) Main Menu 1. ADD a course 2. Drop a Student 3. Print courses registered 4. Check index vacancy 5. Change index for a course 6. Swap index for a course 0. Sign Out ----- 3 Course code CX2001 Course name Algorithms School name SCSE Au 3 ----- </pre>
--------	----------------------------	--

8. Print student list by index number, course

a	Print list by (i) Course (Right) (ii) index (left)	<pre> Admin LogIn... Username: CX2002 Password: 123456 Logging In.. ----- (ADMIN) Main Menu 1. Change access period 2. Add a Student 3. Add a Course 4. Update a Course 5. Check availability slot for an Index Number 6. Print Student List by Index Number 7. Print Student List by Course 0. Sign Out ----- 7 Enter the Subject code : BU8101 Student list is : Name : Abhigyan Singh Matric no : U1922629G Nationality: Indian Gender: M Name : CHOCKA Matric no : U1920428E Nationality: Singaporean Gender: M </pre>	<pre> Admin LogIn... Username: CX2002 Password: 123456 Logging In.. ----- (ADMIN) Main Menu 1. Change access period 2. Add a Student 3. Add a Course 4. Update a Course 5. Check availability slot for an Index Number 6. Print Student List by Index Number 7. Print Student List by Course 0. Sign Out ----- 6 Enter the Subject code : BU8101 Enter the index Number : 10002 Student list is : 2 Name : Abhigyan Singh Matric no : U1922629G Nationality: Indian Gender: M Name : CHOCKA Matric no : U1920428E Nationality: Singaporean Gender: M </pre>
b	Invalid data entries (eg course code, index code etc)	<pre> Admin LogIn... Username: CX2002 Password: 123456 Logging In.. ----- (ADMIN) Main Menu 1. Change access period 2. Add a Student 3. Add a Course 4. Update a Course 5. Check availability slot for an Index Number 6. Print Student List by Index Number 7. Print Student List by Course 0. Sign Out ----- 6 Enter the Subject code : CX2001 Enter the index Number : 10005 The index does not exist </pre>	<pre> Admin LogIn... Username: CX2002 Password: 123456 Logging In.. ----- (ADMIN) Main Menu 1. Change access period 2. Add a Student 3. Add a Course 4. Update a Course 5. Check availability slot for an Index Number 6. Print Student List by Index Number 7. Print Student List by Course 0. Sign Out ----- 7 Enter the Subject code : CX1000 The course you entered does not exist try again </pre>