

Final Report
7COM1025-0105-2021
COURSE MANAGEMENT SYSTEM

SHAYA SATHIYAN

of semester B in the year 2022 in
partial fulfillment of the requirements for the award of
Master of Science
in
Advanced Computer Science

Department of Computer Science
University of Hertfordshire
April 2022

Contents

Abstract	4
1.0 Introduction.....	5
1.1 Project Background.....	5
1.2 Project aims / objectives	6
1.2.1 Computerization.....	6
1.2.2 No Redundant Data.....	6
1.2.3 Automation	6
1.2.4 Easy Interaction	6
1.3 Assumptions.....	6
2.0 System Analysis.....	7
2.1 Software Requirement Specification	7
2.1.1 Functional Requirements	7
2.1.2 Non-functional Requirements	7
2.2 Tools	7
2.3 Technologies	7
3.0 System Design and Implementation	8
3.1 UML Diagram.....	8
3.1.1 ER Diagram	8
3.1.2 Class Diagram.....	9
3.2 Basic GUI.....	10
3.3 Overall Structure	13
3.3.1 Object Oriented programming principles	15
3.3.2 Design Patterns	15
3.3.3 SOLID Principles.....	16
3.3.4 Collection Framework	16
3.3.5 Version Control.....	16
4.0 Testing.....	19
4.1 Unit Testing	19
5.0 Conclusion	21

Table of Figures Contents

Figure 1: Entity Relationship (ER) Diagram	3
Figure 2: Class Diagram	9
Figure 3: Login Page.....	10
Figure 4: Dashboard.....	11
Figure 5: Booking Page	11
Figure 6: When using a day to filter the timetable to book a lesson.....	11
Figure 7: Students rate and review the lessons they have attended.	12
Figure 8: Edit or Cancel the Upcoming Booking in Booking list.....	12
Figure 9: Student's Booking History	12
Figure 10: Generate the Monthly Course Report.....	13
Figure 11: Generate the Monthly Champion Course Report	13
Figure 12: Git Bash window	16
Figure 13: Some snapshots of my commit messages.....	18
Figure 14: Test Results	19
Figure 15: Example of a testcase for verifying a user's credentials	19

Abstract

Implementing a system in Java using Swing can be a challenging task that necessitates knowledge of several major libraries as well as advanced Java concepts. A windowing toolkit is generally responsible for providing a framework that allows a graphical user interface (GUI) to display the right features on the screen at the right moment in a graphical system. In this report, I designed and developed a simple course management system project to manage student demands for group exercise courses. The goal of this project is to create a system that allows you to check the timetable and find an open slot for a given day, time, and lesson. Examining the cost of each lesson as well as the rating of each lesson. The Java programming language was used to build this project, which included components from the swing library. As a result of our project, students will be able to use FreeHand to book a class, rate the lesson, and perform any other activities in a simple manner.

Keywords: NetBeans IDE 8.2, Swing, GUI, Object-Oriented Programming, Java 8, Stream, SOLID Principles, Data Transfer Object (DTO), Data Access Object (DAO), ArrayList, Map, Set, Collection Framework.

Project Repository: <https://github.com/skshaya/course-management-system>

1.0 Introduction

This chapter gives an overview about the aim, objectives and background of the system. It also mentioned the assumptions that I made when developing this system.

1.1 Project Background

Online systems form the backbone of every nation, and hence it is important to provide a strong technology foundation to younger generations to ensure the development of open-minded global citizens securing the future of everyone. Advanced technology available today can play a crucial role in streamlining all processes to promote solidarity among people. This application is designed to reduce the inconveniences & disadvantages of the manual based booking in course management.

Online Course Management System is a system which University Sports Centre (USC) needs software for managing the bookings of group exercise lessons made by the students. It must maintain track of the price of each lesson, the rating of each lesson, the timetable, and the available slot for a specific day, time, and lesson. This is very difficult to manage manually. Maintenance of all this information manually is a very complex task. Owing to the advancement of technology, organization of an Online Course Management becomes much simple. The Online Course Management has been designed to computerize and automate the operations performed over the information about the student's booking issues and rating and all other operations. This computerization of course helps in many instances of its maintenances. It minimizes management's workload by reducing the amount of manual work required.

Course management system consists of task such as login to the student account, checking the timetable, booking a lesson, keeping controlling the booking, producing the monthly reports, and providing rating and reviews.

1.2 Project aims / objectives

The project aims and objectives that will be achieved after completion of this project are discussed in this subchapter. The aims and objectives are as follows:

- a. Check the course timetable.
- b. Book a course.
- c. Change a booking.
- d. Cancel the future booking.
- e. Provide a numerical rating.
- f. Write a review.
- g. Search availability of course and time.
- h. Facility to generate the monthly reports. (Monthly lesson report & champion exercise report)

1.2.1 Computerization

- a. To save the stationary wastage.
- b. To provide Prior information about lesson's availability.
- c. To provide a computerized management of reviews, rating and attendance.

1.2.2 No Redundant Data

- a. To provide Reliable update on student's attendance and monthly reports.
- b. To provide the review and rating for the service.

1.2.3 Automation

- a. To reduce the time that spends to store/ update / retrieve the records.
- b. To provide an automated attendance, monthly lesson report and champion exercise report.

1.2.4 Easy Interaction

- a. To designed for better interaction between student and management.
- b. To provide easy platform to store day today lesson activities.

1.3 Assumptions

In order to construct this project, I made some assumptions.

- a. When the user wants to filter the timetable by date, he or she cannot combine day, time, or course with date. They can only filter by date.
- b. When the user provides the numerical rating only, his/her status changed to "Attended".

- c. I maintained the records for booking availability until June 5, 2022

2.0 System Analysis

2.1 Software Requirement Specification

This section explains what the system will do and how it will be expected to perform. It also defines the features that the product must have to meet the needs of end-users.

2.1.1 Functional Requirements

- a. Course Management System shall be Online based.
- b. Student shall login by his/her account.
- c. Student shall check the timetable.
- d. Student shall book new lesson.
- e. Student shall change the lesson and time according to the availability.
- f. Student shall cancel the future booking.
- g. Student shall view future booking of their lessons.
- h. Student shall view history of their lessons.
- i. Student shall view their attendance.
- j. Student shall generate the monthly reports.
- k. Student shall provide the review and rating for their lessons.

2.1.2 Non-functional Requirements

- a. The System should be easy to use.
- b. The System should be available 24 hour.
- c. The System should response on time.
- d. The System should provide specific information to specific user.
- e. The System should not fail.
- f. Right information is available to right student at right time.

2.2 Tools

- a. NetBeans 8.2
- b. GitHub

2.3 Technologies

- a. Java 8
 - i. Collection Framework (ArrayList, Map and Set)
 - ii. Swing

iii. Stream API

3.0 System Design and Implementation

The entity relationships, UI designs, and how the system works all are discussed in this section. I further highlighted the best practices I followed to implement this system in place.

3.1 UML Diagram

3.1.1 ER Diagram

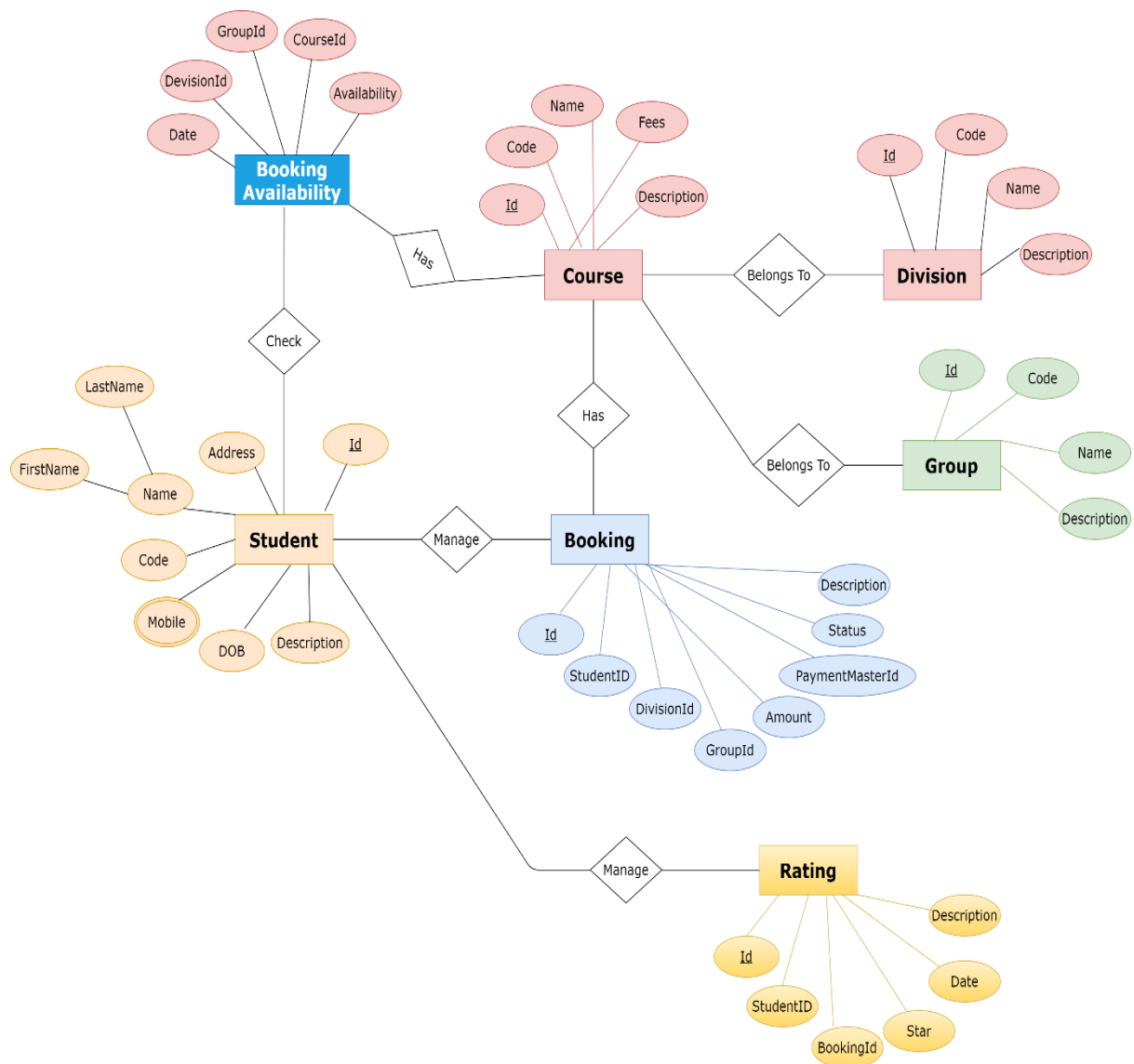


Figure 1: Entity Relationship (ER) Diagram

3.1.2 Class Diagram

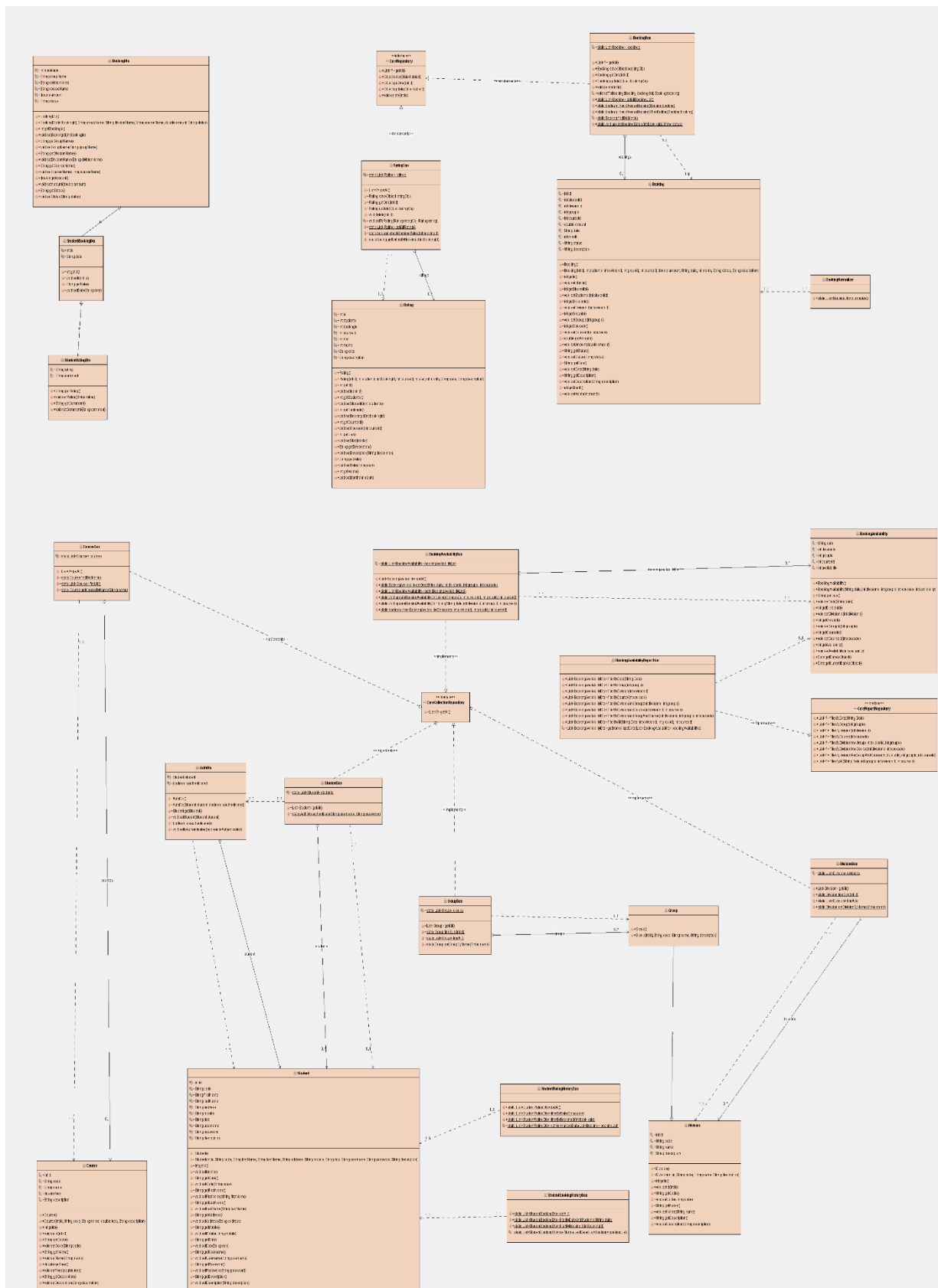


Figure 2: Class Diagram

3.2 Basic GUI

A GUI program provides a much more advanced user interface, in which the user interacts with GUI components such as windows, buttons, text input boxes, and so on, using a mouse and keyboard. In this case, the GUI was implemented using Swing. In Swing,

- a. **JFrame and JPanel:** Java has a built-in class to represent windows. Although there are various distinct types of windows, the `JFrame` class represents the most prevalent (which is included in the package `javax.swing`). A `JFrame` is an independent window that can be the main window of an application. `JPanel` is a content, display panel, which is used as a drawing surface and holds other components.
- b. **Events and Listeners:** The physical appearance of a GUI is determined by the structure of containers and components, but it reveals nothing about how the GUI behaves. The majority of graphical user interfaces are event-driven, which means that the software waits for actions that are triggered by the user's activities. The application responds to an event by calling an event-handling method that developers build the methods to respond to the events that the user is involved in. The most common technique for handling events in Java is to use event listeners. A listener is an object that includes one or more event-handling methods.

Here, I have included some screenshots of my system's user interface.

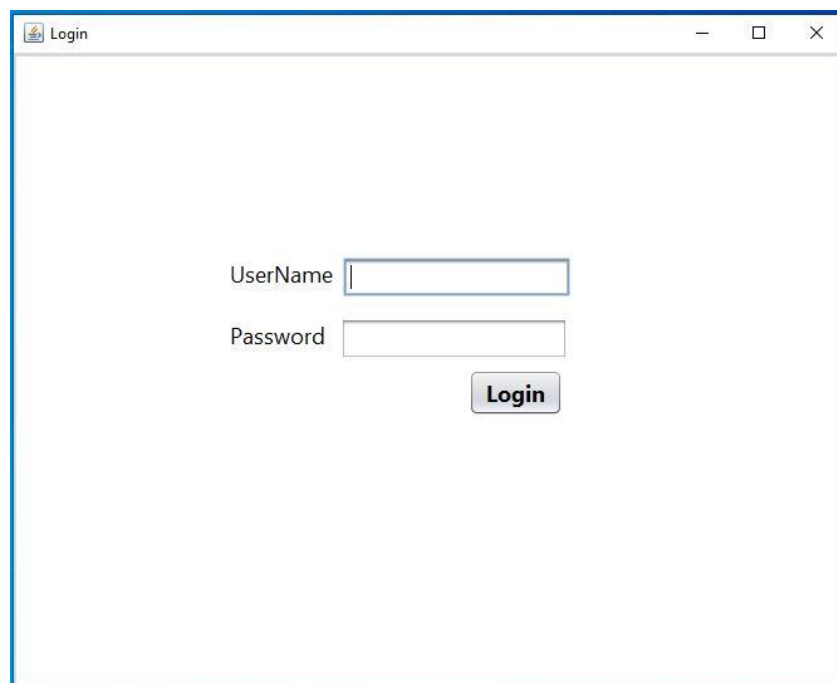


Figure 3: Login Page

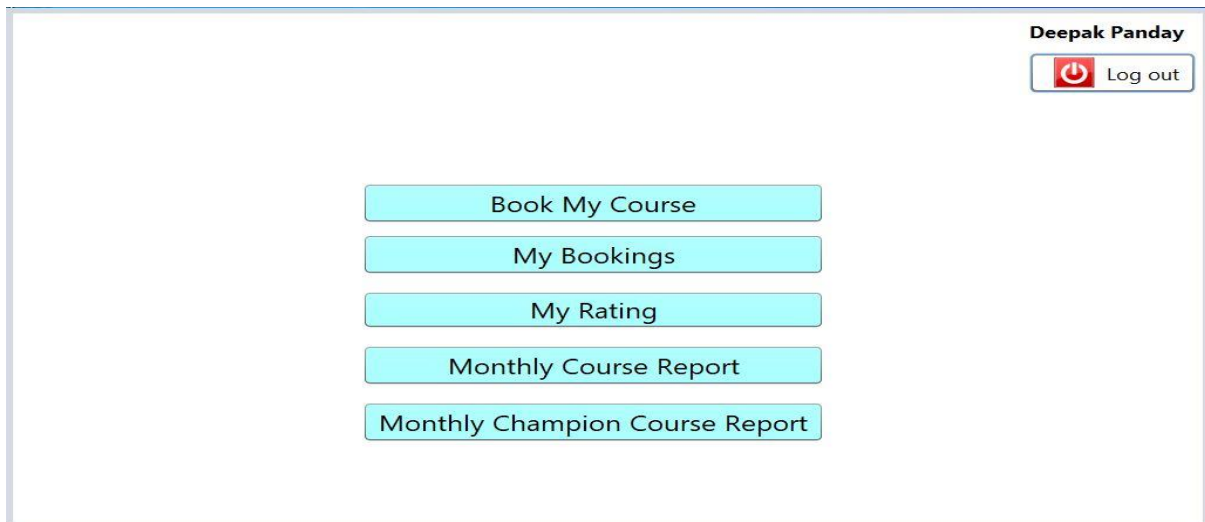


Figure 4: Dashboard

Book Reservation

Day: Select, Time: Select, Course: Select, Date: [Calendar Icon]

Search, Clear

Date	Division	Group	Course	Availability	Amount
------	----------	-------	--------	--------------	--------

Booking ID, Student ID, Division, Group, Course, Amount

Description

Book Now

Figure 5: Booking Page

Book Reservation

Day: Saturday, Time: Select, Course: Select, Date: Apr 16, 2022

Search, Clear

Date	Division	Group	Course	Availability	Amount
2022-04-16	Saturday	Afternoon	Yoha	4	1000.0
2022-04-16	Saturday	Afternoon	Zumba	4	2000.0
2022-04-16	Saturday	Afternoon	Aquacise	4	3000.0
2022-04-16	Saturday	Afternoon	Box Fit	4	4000.0
2022-04-16	Saturday	Afternoon	Body Bliz	4	5000.0
2022-04-16	Saturday	Evening	Zumba	4	2000.0

Booking ID: 72, Student ID: 1, Division: Saturday, Group: Afternoon, Course: Box Fit, Amount: 4000.0

Description

Book Now

Figure 6: When using a day to filter the timetable to book a lesson

My Booking List

Booking ID

Booking Date

Search
Clear

Id	Date	Division	Group	Course	Amount	Status	Description
1	2022-01-01	Saturday	Morning	Yoha	1000.0	Attended	Desc
11	2022-01-02	Sunday	Afternoon	Box Fit	4000.0	Attended	Desc
19	2022-01-08	Saturday	Evening	Body Bliz	5000.0	Attended	Desc
26	2022-01-09	Sunday	Morning	Box Fit	4000.0	Attended	Desc
28	2022-01-15	Saturday	Morning	Body Bliz	5000.0	Attended	Desc
39	2022-01-16	Sunday	Afternoon	Box Fit	4000.0	Attended	Desc

Booking ID

Student ID

Division

Group

Course

Amount

Select

Select

Description

Figure 8: Student's Booking History

My Booking List

Booking ID

Booking Date

Search
Clear

Id	Date	Division	Group	Course	Amount	Status	Description
39	2022-01-16	Sunday	Afternoon	Box Fit	4000.0	Attended	Desc
40	2022-01-22	Saturday	Morning	Body Bliz	5000.0	Attended	Desc
48	2022-01-23	Sunday	Evening	Zumba	2000.0	Attended	Desc
60	2022-02-05	Saturday	Morning	Box Fit	4000.0	Attended	Desc
68	2022-02-06	Sunday	Morning	Aquacise	3000.0	Attended	Desc
72	2022-04-16	Saturday	Afternoon	Box Fit	4000.0	Booked	

Booking ID

Student ID

Division

Group

Course

Amount

Afternoon

Box Fit

Description

Cancel
Edit

Figure 9: Edit or Cancel the Upcoming Booking in Booking list

My Rating List

Booking ID

Booking Date

Search
Clear

BookingId	Date	Division	Group	Course	Fee	Rating	Comments
39	2022-01-16	Sunday	Afternoon	Box Fit	4000.0	4	Desc
40	2022-01-22	Saturday	Morning	Body Bliz	5000.0	4	Desc
48	2022-01-23	Sunday	Evening	Zumba	2000.0	4	Desc
60	2022-02-05	Saturday	Morning	Box Fit	4000.0	4	Desc
68	2022-02-06	Sunday	Morning	Aquacise	3000.0	5	Desc
72	2022-04-16	Saturday	Afternoon	Box Fit	4000.0	Review Needed	

Booking ID

Student ID

Rating

Review

Select

Add

Figure 7: Students rate and review the lessons they have attended.

Monthly Course Report

Select Month

January ▼

Generate

Clear

Course	Average Rating	Round	Total Enroll
Body Bliz	4.1904761904...	4.2	21
Yoha	4.0	4	12
Zumba	3.9090909090...	3.9	11
Aquacise	3.6111111111...	3.6	18
Box Fit	3.3333333333...	3.3	9

Figure 10: Generate the Monthly Course Report

Monthly Champion Course Report

Select Month

February ▼

Generate

Clear

Course	Total Revenue	Total Enroll
Body Bliz	105000.0	21
Aquacise	57000.0	19
Box Fit	44000.0	11

Figure 11: Generate the Monthly Champion Course Report

3.3 Overall Structure

This is a stand-alone application for booking a group fitness lesson. I used Java 8 to do this project. This project does not use any external databases. To develop this system, I used a variety of best practices, including design patterns, SOLID principles, Collection framework, Stream API, and version control.

The system's key feature is the ability to schedule a lesson. The student must first check the lesson's availability, price, and timing. After successful login, the user is redirected to a dashboard (Figure 4). There are five features in the Dashboard, including book my course, student booking list, student rating, and two types of monthly reports. If a student needs to register a lesson, he or she should select "Book My Course" from the dashboard menu. It takes

you to the page where you may make a reservation (Figure 5). There, students can check the timetable in a variety of ways:

- a. one is by specifying the day (Saturday or Sunday).
- b. the other one is by specifying the course name (Yoga, Zumba...).
- c. the other one is by specifying the time (Morning, Afternoon or Evening).
- d. the other one is by specifying the date.
- e. the other one is by specifying the time and day.
- f. the other one is by specifying the time, course name and day.
- g. the other one is by specifying the Course name and day.
- h. the other one is by specifying the time, date, course name and day

After screening the timetable, the student chooses a course and books the lesson by clicking the Book button. The page is navigated to the Booking List after a successful booking. Students can see their past and future lessons that they have already registered on that list. Students can verify their attended, cancelled, changed and booked lesson from that list. Also, if he or she needs to change or cancel the future booking, pick that row and click the edit button to alter or click the cancel button to cancel the booking according to availability and preference. User cannot edit or cancel the past events. After their action, the status of the lesson is updated as changed or cancelled, depending on what they did.

If a student takes a class, he or she is required to rate that class. A button labelled "My Rating" can be found on the Dashboard. When you click that button, you'll be taken to My Rating List (Figure 9). In such situation, Student will be unable to leave any feedback on future bookings. Only students have the ability to rate previous lessons that they attended. If the student gives a rating only, the status changes to attended; otherwise, the status does not change to attended. Providing a numerical rating to a course is a must-do task for students. But if they want to give review then they can send review as well.

In every month, the system generates and print the two monthly reports. Those are,

- a. Monthly Course Report (Figure 10) – It containing the number of students per group exercise lesson on each day, along with the average rating.
- b. Monthly Champion Course Report (Figure 11) - It containing the group exercise which has generated the highest income, counting all the same exercise lessons together.

When the student has completed his or her tasks, he or she can log out of the system at any moment. I assumed the students were already registered in the system based on the coursework guidelines.

For implementing the project, Design Patterns, SOLID Principles, Collection Framework, and Version Control were some of the best practices I followed.

3.3.1 Object Oriented programming principles

Java supports object-oriented programming techniques that are based on a hierarchy of classes and well-defined and cooperating objects. One object-oriented concept that helps objects work together is inheritance. The relationship is one of parent to child, with the child or extending class inheriting all of the parent class's attributes. All classes in Java derived from **java.lang.Object** and shared its methods. The **java.lang.Object** methods are also shown because they are inherited and implemented by all of its subclasses, which are all of the Java API libraries' classes. Encapsulation was also used to achieve loose coupling and protects an object from unwanted access by users. I used private variables in classes to accomplish this. Not only these two concepts used, but also abstraction and polymorphism as well. In Java, I used interface to achieve abstraction. So, in order to develop this system, I used all four object-oriented concepts.

3.3.2 Design Patterns

I can make my code more flexible, reusable, and maintained by adopting design patterns. Because java follows design patterns internally, this is the most significant aspect. Here I used Data Transfer Object (DTO) and Data Access Object (DAO) patterns to build this project.

DAO pattern is a structural pattern that uses an abstract API to separate the application/business layer from the persistence layer. I want the domain model of the application to be fully independent of the other models. As a result, I'll design a simple DAO class to keep these components neatly isolated from one another. To build this system, I used 11 DAO classes such as BookingDao, CourseDao, StudentDao, DivisionDao and so on. As a result, it helps with the separation of a data resource's client interface from its data access mechanisms, as well as adapts a specific data resource's access API to a generic client interface.

The DTO Design Pattern calls for the use of objects that aggregate and encapsulate data for transfer. It is, essentially, like a data structure and should not contain any business logic but should contain serialization and deserialization mechanisms. In my project I implemented 7 DTO classes such as BookingDto, BookingAvailabilityDto, AuthDto, StudentBookingDto and so on. It helps in the transfer of data between processes, reducing the number of methods calls and ensuring data security.

3.3.3 SOLID Principles

SOLID refers to five design principles in object-oriented programming, designed to reduce code bad designs and improve the value, function, and maintainability of software. The SOLID principles assist the user in writing code that is less coupled. I applied the SOLID principles here for good practice and code that everyone can comprehend easily.

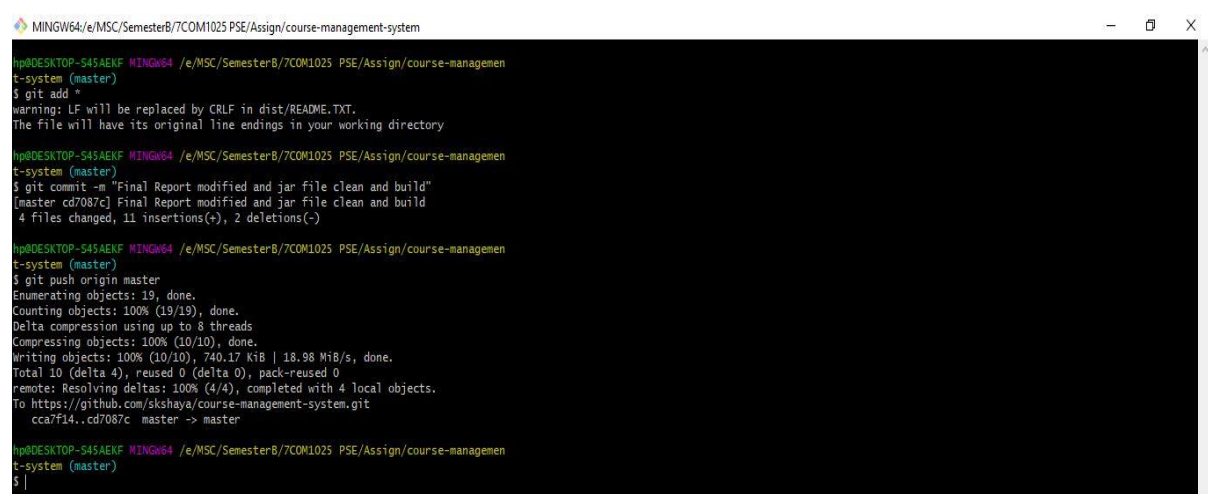
3.3.4 Collection Framework

According to the guidelines, I am not permitted to save the data in an external database. So, I went with ArrayList, Map, and Set as my collection frameworks. ArrayList is a data structure for storing dynamically scaled collections of data such as student information, lesson information, booking information, and so on. I need to associate a key with a value in order to generate a report. It means that the user will be prompted to enter a month number (e.g., 12 for December). As a result, I used Map for this purpose. I used Set to avoid the duplication.

3.3.5 Version Control

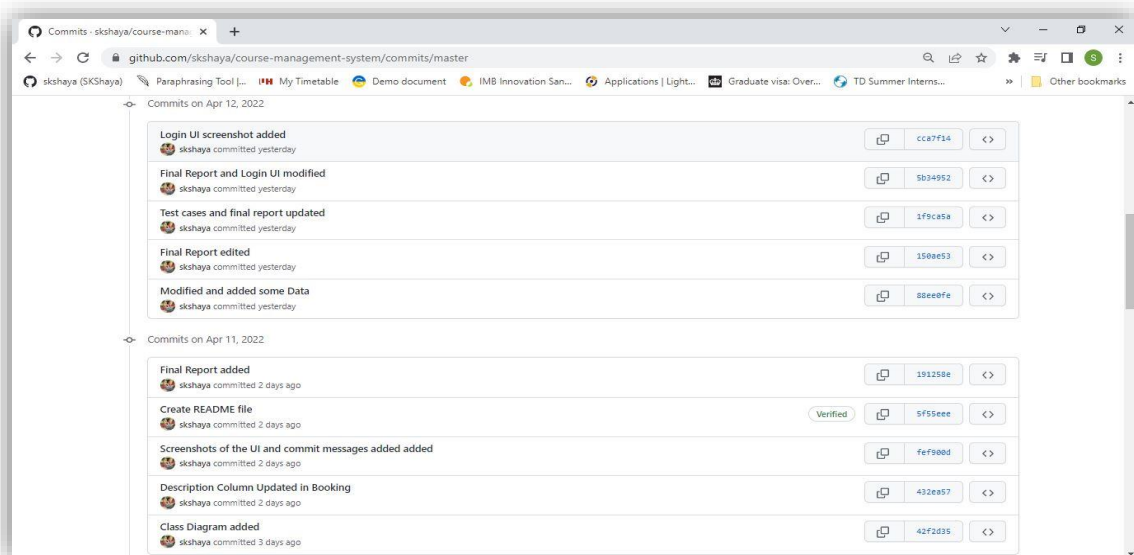
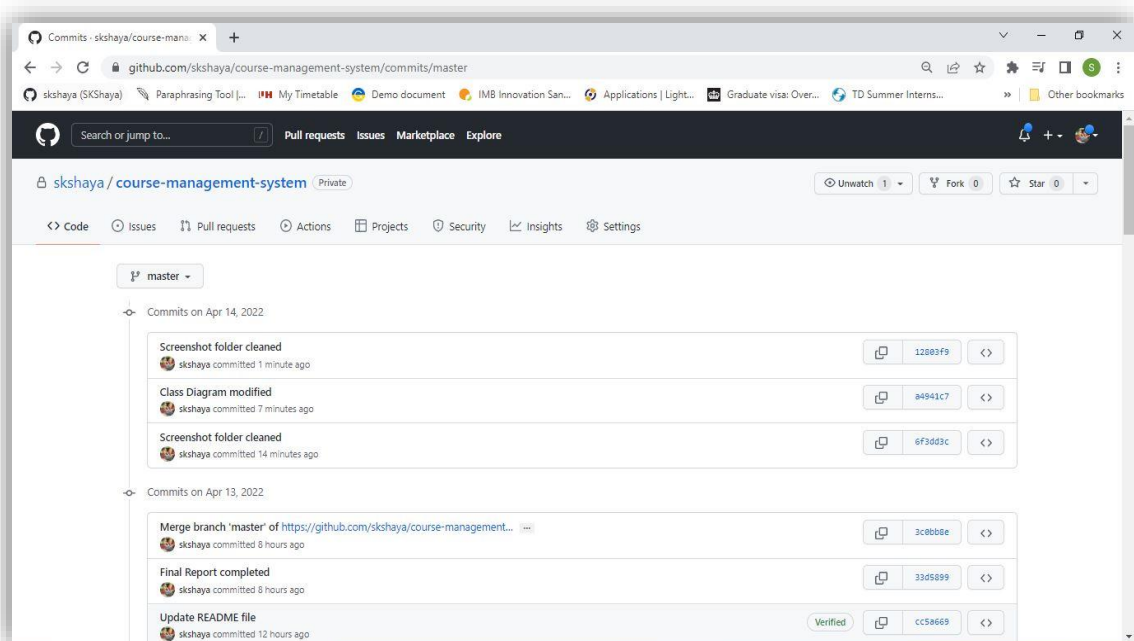
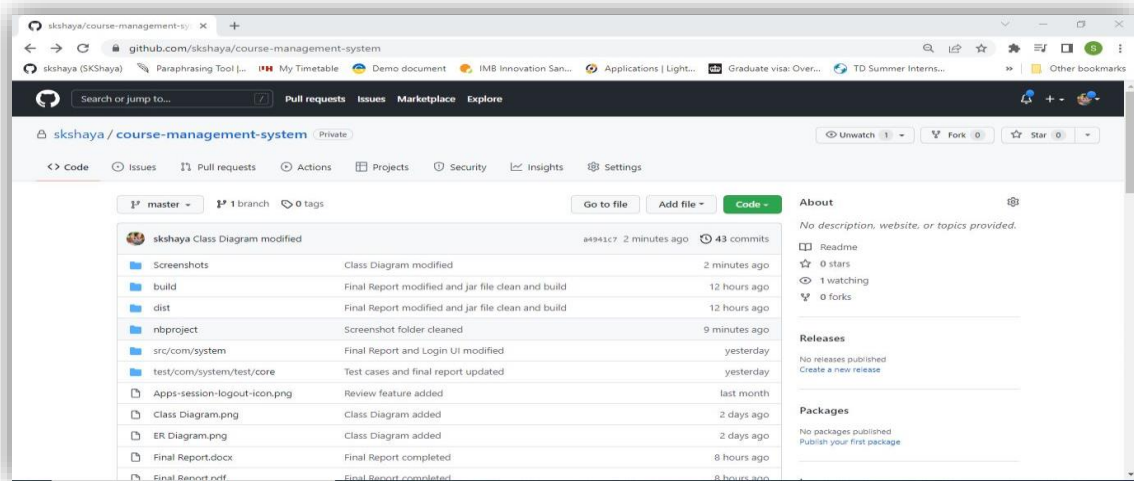
Version control is tracking and managing changes to software code and help developer to manage changes to source code over time. It keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake easily. Here, I used GitHub as my version control. Always I pushed the changes that I made in my IDE through Git Bash. Also I can clone the project and take latest of the project through Git Bash.

Repository Link: <https://github.com/skshaya/course-management-system>



```
MINGW64/e/MSC/SemesterB/7COM1025 PSE/Assign/course-management-system
hp8DESKTOP-S4S4EKF MINGW64 /e/MSC/SemesterB/7COM1025 PSE/Assign/course-managemen
t-system (master)
$ git add *
warning: LF will be replaced by CRLF in dist/README.TXT.
The file will have its original line endings in your working directory
hp8DESKTOP-S4S4EKF MINGW64 /e/MSC/SemesterB/7COM1025 PSE/Assign/course-managemen
t-system (master)
$ git commit -m "Final Report modified and jar file clean and build"
[master cd7087c] Final Report modified and jar file clean and build
4 files changed, 11 insertions(+), 2 deletions(-)
hp8DESKTOP-S4S4EKF MINGW64 /e/MSC/SemesterB/7COM1025 PSE/Assign/course-managemen
t-system (master)
$ git push origin master
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 8 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (10/10), 740.17 KiB | 18.98 MiB/s, done.
Total 10 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/skshaya/course-management-system.git
cca7f14..cd7087c master -> master
hp8DESKTOP-S4S4EKF MINGW64 /e/MSC/SemesterB/7COM1025 PSE/Assign/course-managemen
t-system (master)
$
```

Figure 12: Git Bash window



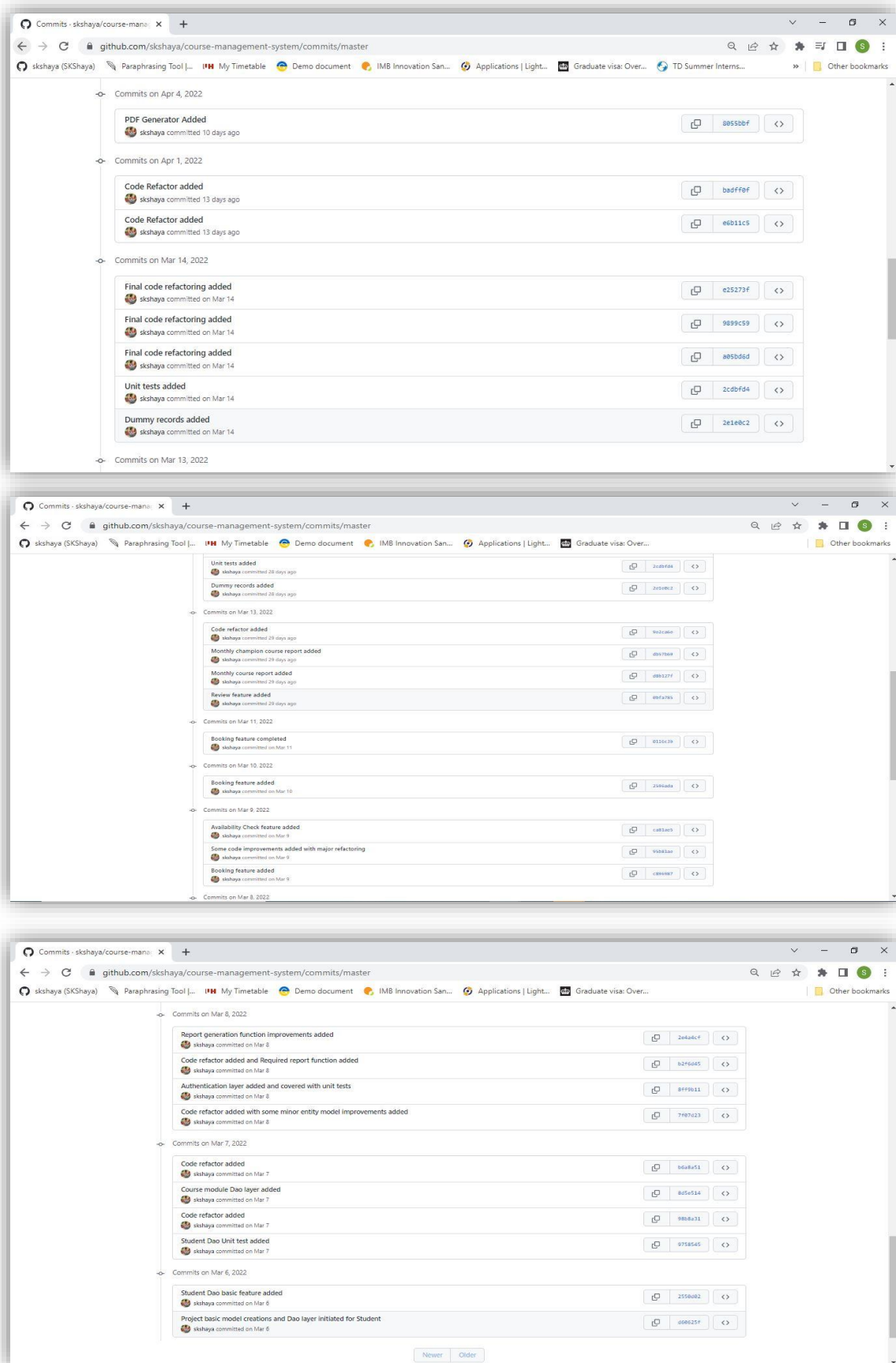


Figure 13: Some snapshots of my commit messages

4.0 Testing

The purpose of the testing procedure was to identify defects in our project. The program was subjected to a set of test inputs and various observations were made and based on these observations it will be decided whether the program behaves as expected or not. My Project went through the unit testing.

4.1 Unit Testing

Unit testing is undertaken when a module has been created and successfully reviewed. In order to test a single module, I need to provide a complete environment is besides the module I would require,

- a. The procedures belonging to other modules that the module under test calls.
- b. A procedure to call the functions of the module under test with appropriate parameters.

JUnit was used to test the components in this application. I developed 10 testcases to see if the predicted result matches the actual output.

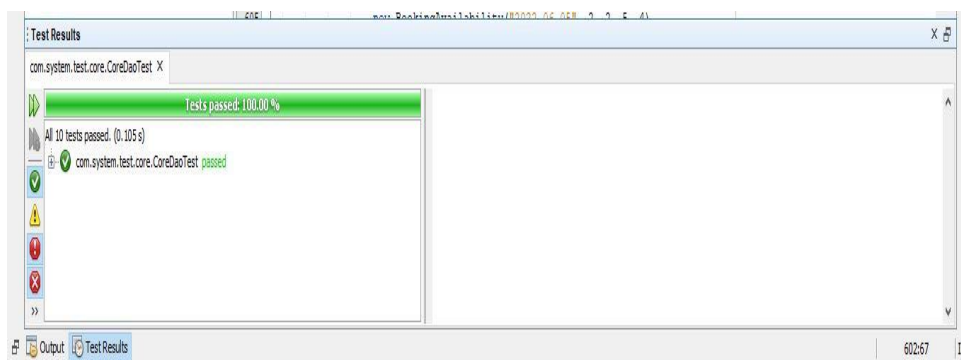


Figure 14: Test Results

For example, I wrote 10 testcases for verifying some of the main functionalities such as login, filtering the timetable, getting all booking history and so on.

```
@Test
public void testAuth() {
    AuthDto authDto = studentDao.authenticate("Ramesh", "1234");
    assertTrue(authDto.isIsAuthenticated());
}

@Test
public void testInvalidAuth() {
    AuthDto authDto = StudentDao.authenticate("user", "1234");
    assertFalse(authDto.isIsAuthenticated());
}
```

Figure 15: Example of a testcase for verifying a user's credentials

```

71
72 @Test
73 public void testGetAllBookingAvailabilityRecords() {
74     List<BookingAvailability> bookingAvailabilities = bookingAvailabilityDao.getAllBookingAvailabilityList();
75     assertEquals(446, bookingAvailabilities.size());
76     assertTrue(bookingAvailabilities.get(0) instanceof BookingAvailability);
77 }
78
79 @Test
80 public void testGetAllBookingAvailabilityRecordByIdentifier() {
81     BookingAvailability bookingAvailability = BookingAvailabilityDao.getOne("2022-04-02", 1, 1, 1);
82     assertTrue(bookingAvailability instanceof BookingAvailability);
83     assertEquals(bookingAvailability.getDate(), "2022-04-02");
84     assertEquals(DivisionDao.findById(bookingAvailability.getDivisionId()).getName(), "Saturday");
85     assertEquals(GroupDao.findById(bookingAvailability.getGroupId()).getName(), "Morning");
86     assertEquals(CourseDao.findById(bookingAvailability.getCourseId()).getName(), "Yoha");
87 }
88
89 @Test
90 public void testGetAllBooking() {
91     List<Booking> booking = BookingDao.getAllBookingList();
92     assertEquals(83, booking.size());
93     assertTrue(booking.get(0) instanceof Booking);
94 }
95

```

```

95
96 @Test
97 public void testBooking() {
98     Booking booking = new Booking();
99     booking.setId(BookingDao.getAllBookingList().size() + 1);
100     booking.setStudentId(1);
101     booking.setDivisionId(2);
102     booking.setGroupId(3);
103     booking.setCourseId(1);
104     booking.setAmount(1000.00);
105     booking.setDate("2022-02-06");
106     booking.setMonth(2);
107     booking.setStatus("Booked");
108     booking.setDescription("Test");
109     Booking result = bookingDao.save(booking);
110     assertTrue(result instanceof Booking);
111     assertEquals(84, BookingDao.getAllBookingList().size());
112 }
113
114 @Test
115 public void testGetbookingById() {
116     Booking booking = bookingDao.getOne(1);
117     assertTrue(booking instanceof Booking);
118     assertEquals("2022-01-01", booking.getDate());
119     assertEquals(1000.00, booking.getAmount(), 0);
120     assertEquals(DivisionDao.findById(booking.getDivisionId()).getName(), "Saturday");
121     assertEquals(GroupDao.findById(booking.getGroupId()).getName(), "Morning");
122     assertEquals(CourseDao.findById(booking.getCourseId()).getName(), "Yoha");
123 }
124

```

```

125
126 @Test
127 public void testOverlapBooking() {
128     Booking booking = bookingDao.getOne(1);
129     booking.setCourseId(2);
130     boolean result = BookingDao.checkOverlapBooking(booking);
131     assertTrue(result);
132 }
133
134 @Test
135 public void testGetAllRating() {
136     List<Rating> ratingList = RatingDao.getAllRating();
137     assertEquals(71, ratingList.size());
138     assertTrue(ratingList.get(0) instanceof Rating);
139 }

```

Figure 16: Example testcases for checking the booking related functionalities

5.0 Conclusion

This system provides a computerized version of the course management system which will benefit the students as well as the staff of the University Sports Centre. It brings the entire booking process online, allowing students to look up timetables, course pricing, and lesson availability. It also features a student login system, which allows students to login into the system and check the status and history of their bookings and also requests to cancel or change future bookings, as well as provides a review and numerical rating for a lesson they attended. Not only that, but the User can generate and print monthly course reports and monthly champion course reports.

My project source code, UML Diagrams, and Project report are available on my git repository.