**PROGRAMMING FOR SOFTWARE ENGINEERING**

**7COM1025**

**ASHOK KUMAR RAJU THATIGOTLA**

**22102346**

# INTRODUCTION

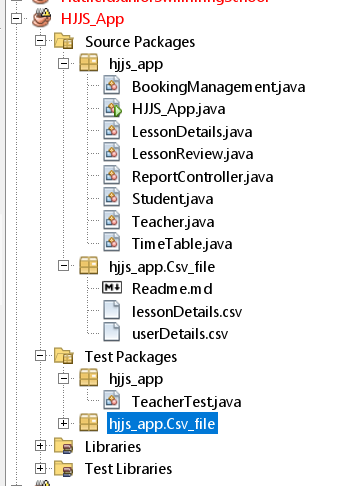
The system will be designed to manage swimming lesson bookings for HJSS, allowing learners to view and book lessons based on grade levels, coaches, and days. It will also track learner records, handle booking changes and cancellations, and generate reports on learner and coach information after four weeks. The system will be self-contained and not require an external database or security protocol.

# ASSUMPTIONS

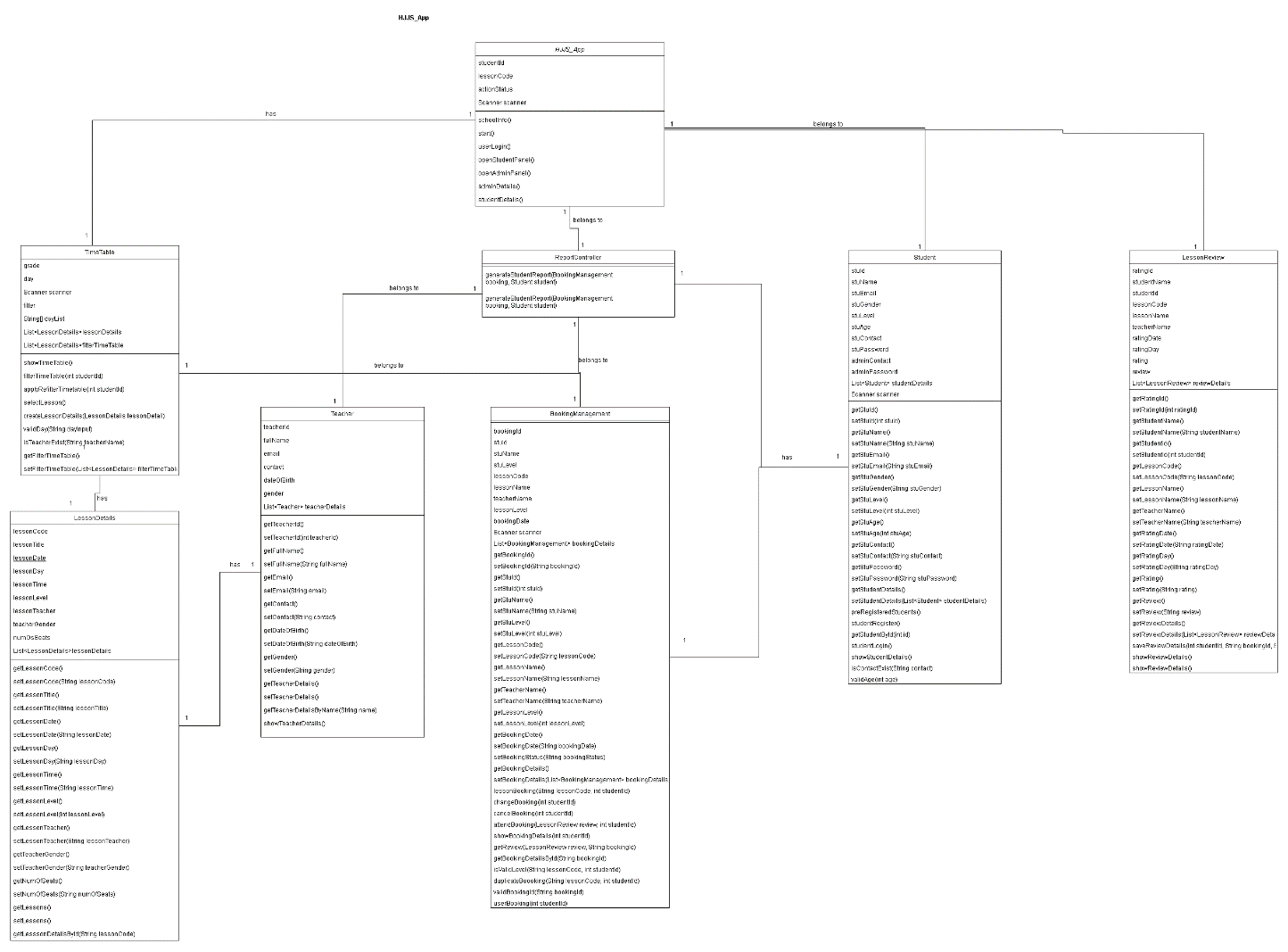
* The assumption in the HJJS\_App project is that if a student cancels a class, they are not allowed to attend that specific class.
* The user needs to insert correct ***username*** at the time of ***login.***

# Class diagram

The “class diagram" refers to a visual representation of the relationships and structure between different classes in a program. It helps in understanding the design and organization of the program. The program mentioned follows a hierarchical structure, that consist of main class (or parent class) and multiple subclasses (child classes) that extend from the main class. This is known as inheritance, a fundamental concept in object-oriented programming.



Inheritance allows the subclasses to inherit properties, methods, and characteristics from the main class. This creates a relationship where the subclasses are considered as specialized versions of the main class. The term "user encapsulation in lesson details class" refer to a specific class that encapsulates user-related functionalities or data within the program. The image displayed below is the class diagram; the structure of the system includes the main class, attributes and methods.

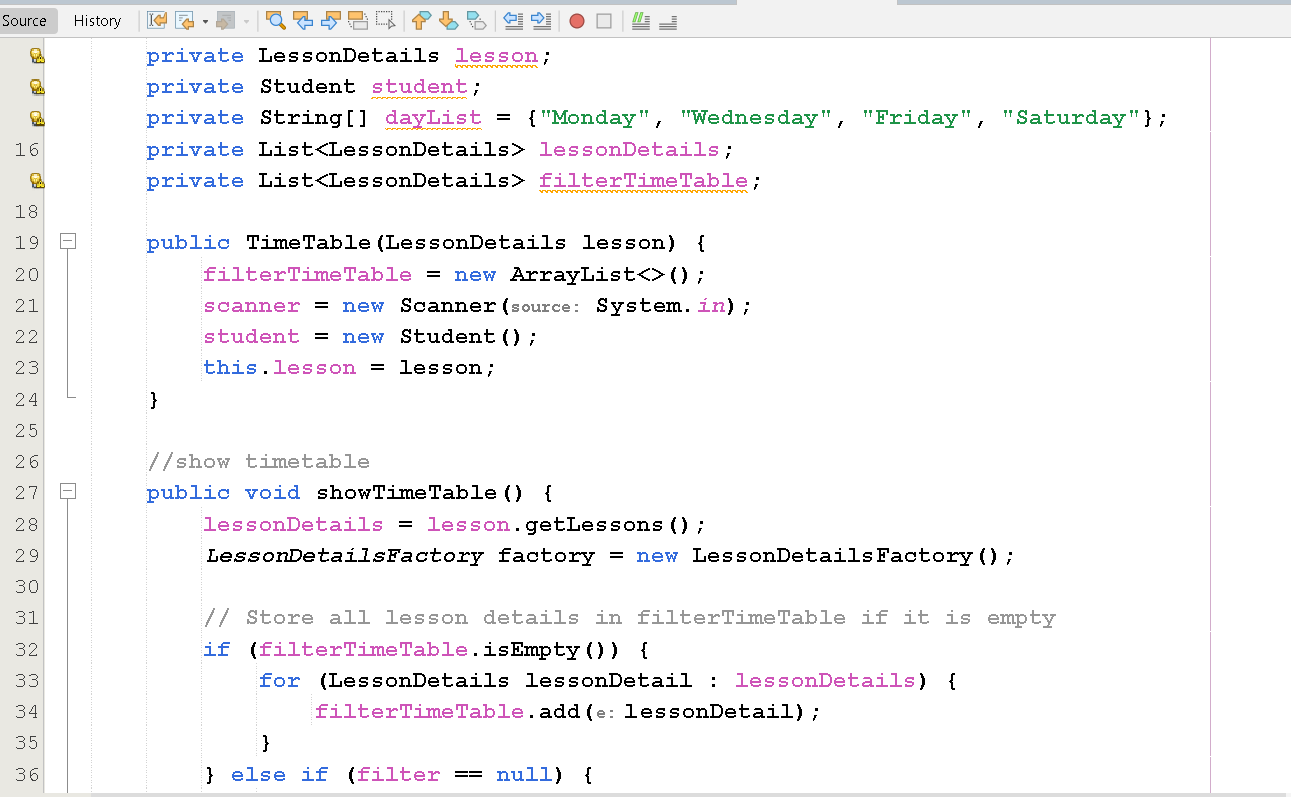


The image displayed above is the class diagram for the Hatfield Junior Swimming School (HJSS).

|  |  |
| --- | --- |
| Main class | Attributes and details |
| ***Teacher*** | ***Teacher*** class contains the personal details of teachers such as teacher ID, full name, email, contact, DOB, and gender. |
| **BookingManagement** | **BookingManagement** class contains the details of booking. |
| **HJJS\_App** | **HJJS\_App** class include the details of students, such as studentID, lessonCode, actionStatus, and Scanner. |
| **Student** | **Student** class contains the detail of students including stuID, stuName, stuEmail, stuLevel, etc. |
| **LessonReview** | **LessonReview** class contains attributes such as ratingID, studentName, studentID, lessonCode, lessonName, rating, review and few more. |
| **TimeTable** | **TimeTable** contains grade, day, filter, string[] dayList. |

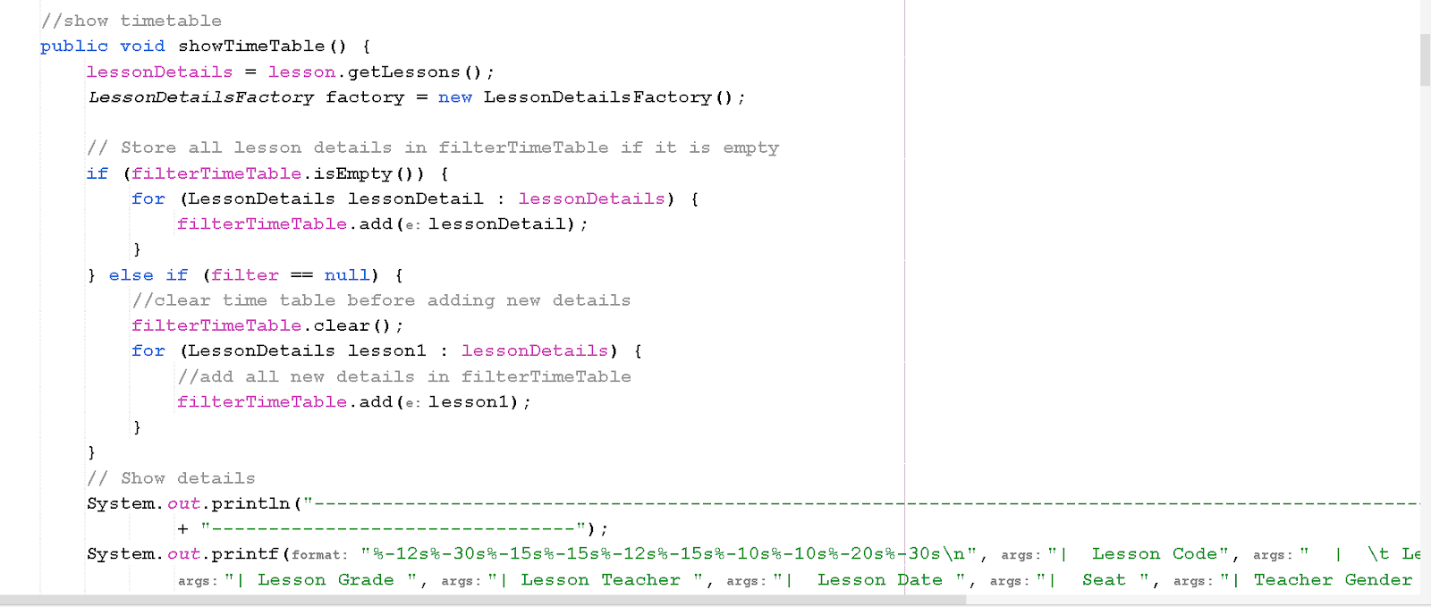
# DESIGN PATTERN

The Factory Method design pattern is implemented in the timetable class of the HJSS\_App Java project. This design pattern is a creational pattern that provides an interface for creating instances of a class, but allows subclasses to alter the type of objects that will be created. The Factory Method design pattern likely allows the timetable class to create instances of different types of lessons or bookings based on specific criteria, without the need to specify the exact class of the object being created. This promotes flexibility and extensibility in the project's design, as new types of lessons or bookings can be added without modifying the existing code. Other than this, the project includes various classes such as LessonDetails, BookingManagement, student, teacher, and lesson review, which are likely involved in the implementation of the Factory Method design pattern and the overall functionality of the HJJS\_App project.



# REFACTORING

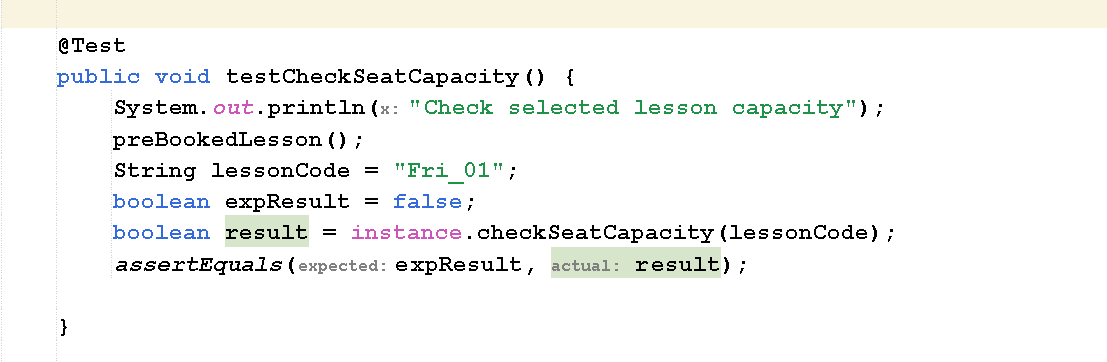
The refactoring in the TimeTable class involved the implementation of organization, encapsulation, and readability improvements. Also, the showTimeTable method was extracted into a separate method. This refactoring aimed to enhance the overall structure and maintainability of the TimeTable class by organizing the code more effectively, encapsulating data and methods to improve modularity, and improving the code's readability for easier comprehension and maintenance.



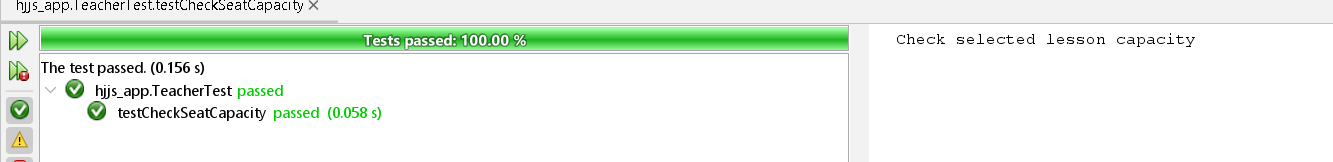
# Junit Testing

JUNIT TESTING has been performed to ensure the functionality and reliability of the software system. It allows for the systematic testing of individual units or components of the program to verify that each part performs as intended. By using JUnit testing, the engineers can validate the correctness of the code and identify and fix any defects, also ensure that the system meets the specified requirements.

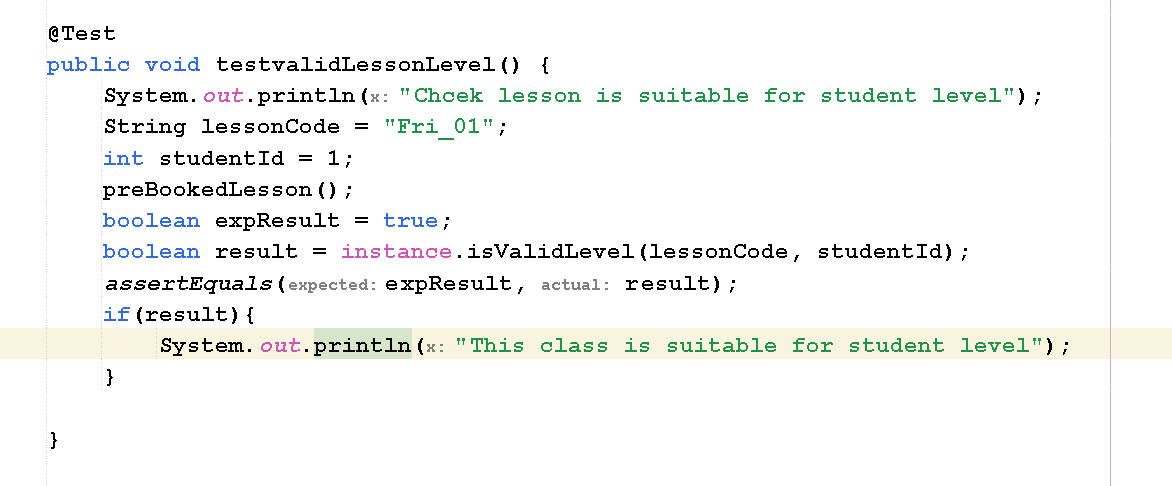
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | description | input | expResult | Actual Result |
| 1 | Test selected lesson seat not full | lessonCode | Functionality instance.checkSeatCapacity(lessonCode) return false | Functionality instance.checkSeatCapacity(lessonCode) return false |



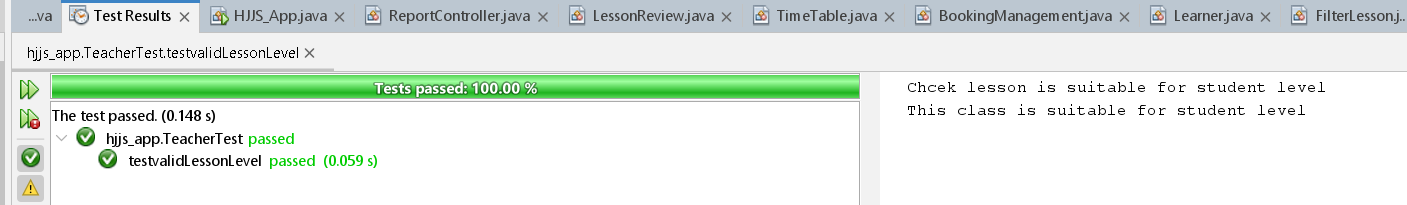
Output :



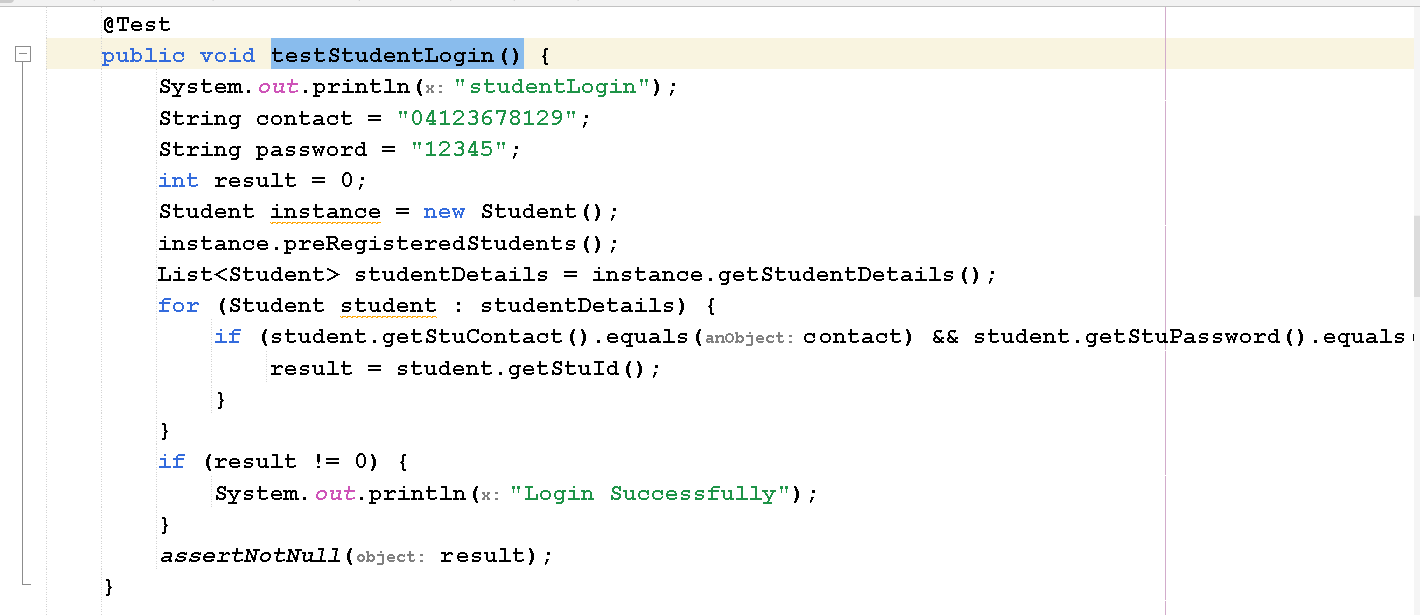
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | description | input | expResult | Actual Result |
| 2 | Test selected lesson level is suitable for student level | lessonCode  studentId | This lesson is suitable for student level | This lesson is suitable for student level |

****

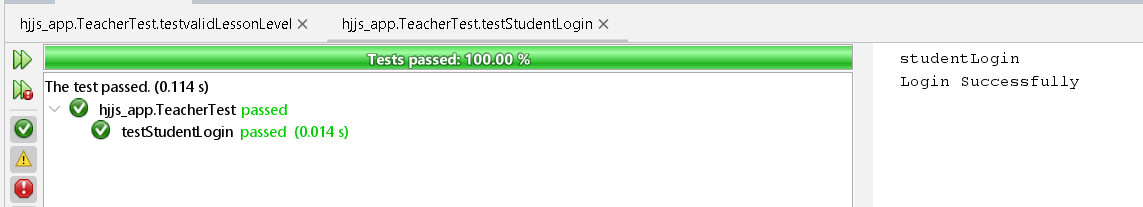
**Output :**

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | description | input | expResult | Actual Result |
| 3 | Test student login | Contact  password | Return student if after successfully login | Return student if after successfully login |

****

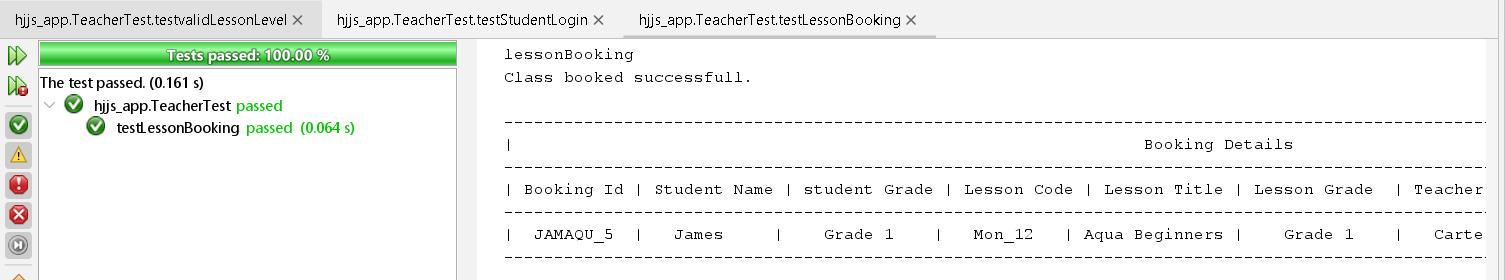
**Output :**

****

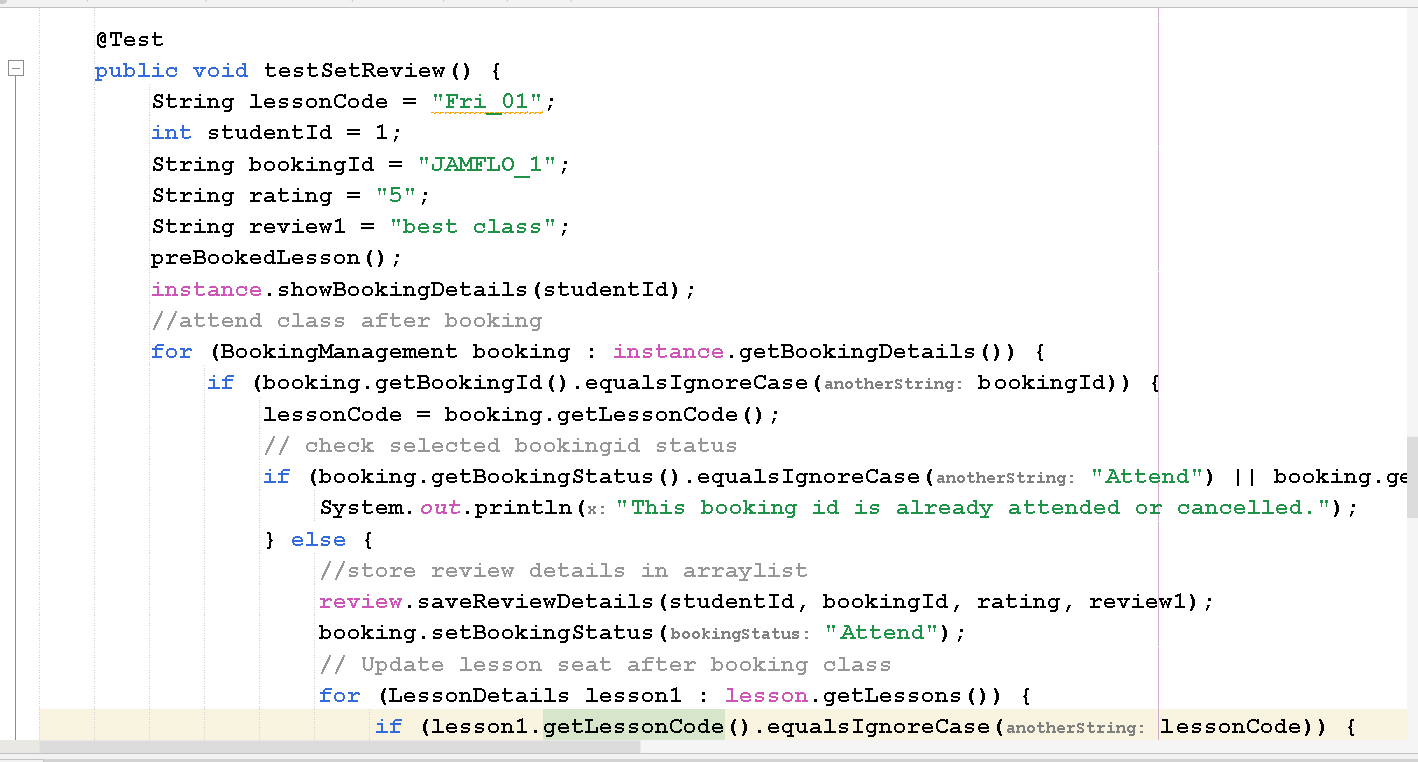
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | description | input | expResult | Actual Result |
| 4 | Test lesson booking by student | lessonCode  studentId | Book Lesson if valid lessonCode or not duplicate booking and store in arrayList | Book Lesson if valid lessonCode or not duplicate booking and store in arrayList |

****

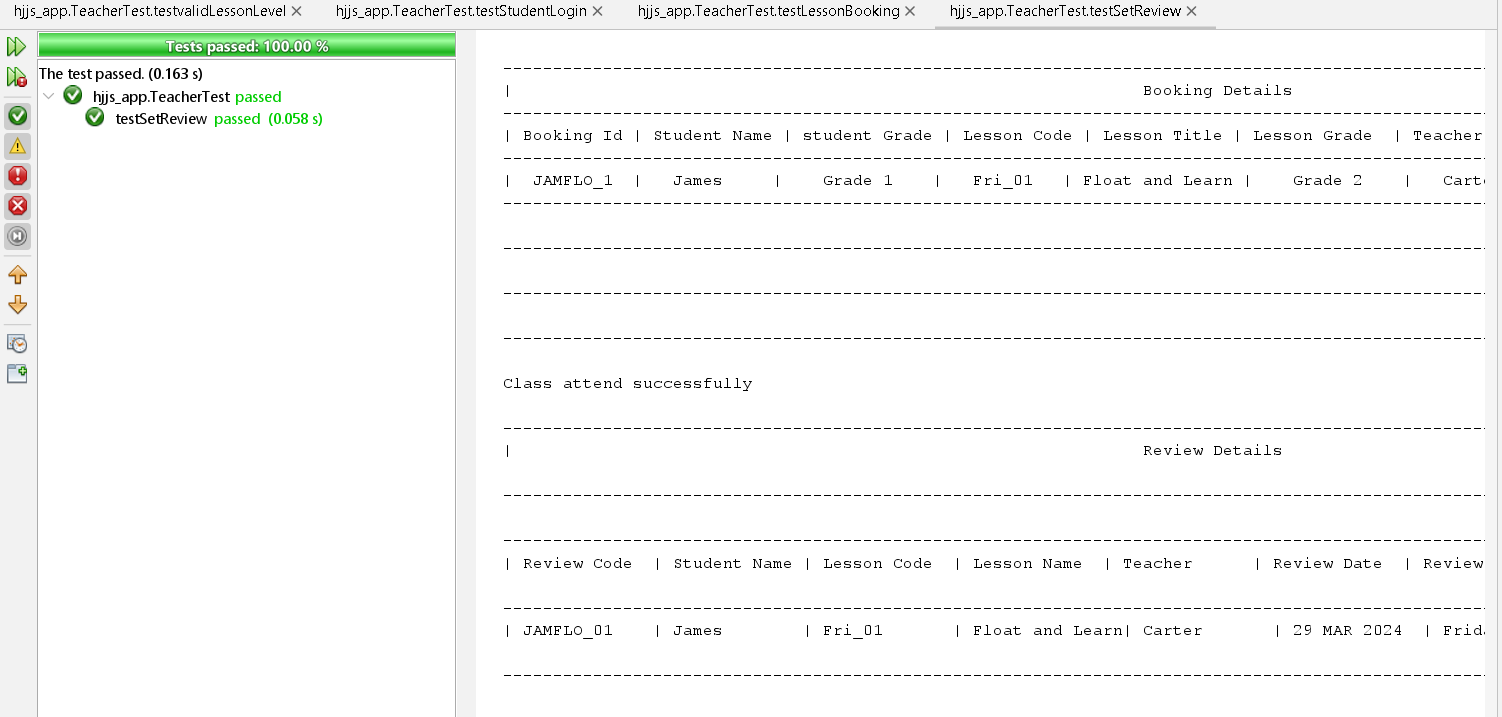
**Output :**

****

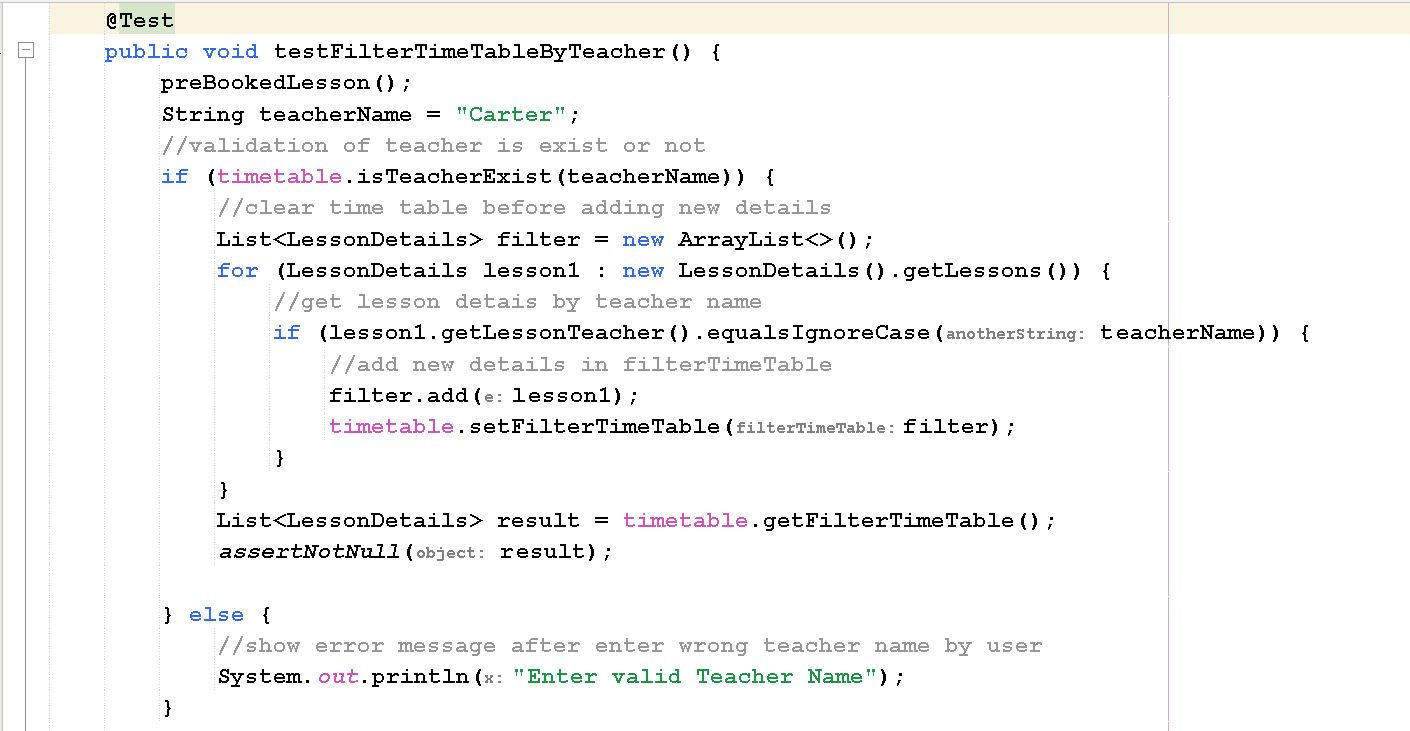
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | description | input | expResult | Actual Result |
| 5 | Test get rating from student after attend class | lessonCode  studentId  bookingId  Rating  review1 | Get rating from student if valid rating and review store in arraylist | Get rating from student if valid rating and review store in arraylist |

****

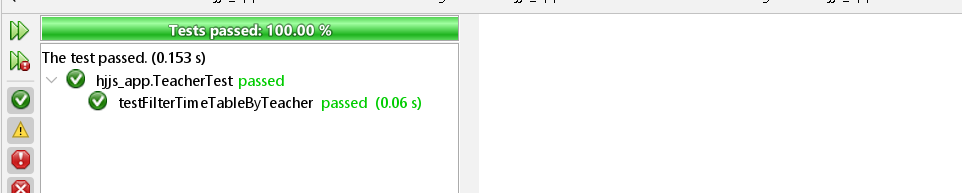
**Output :**

****

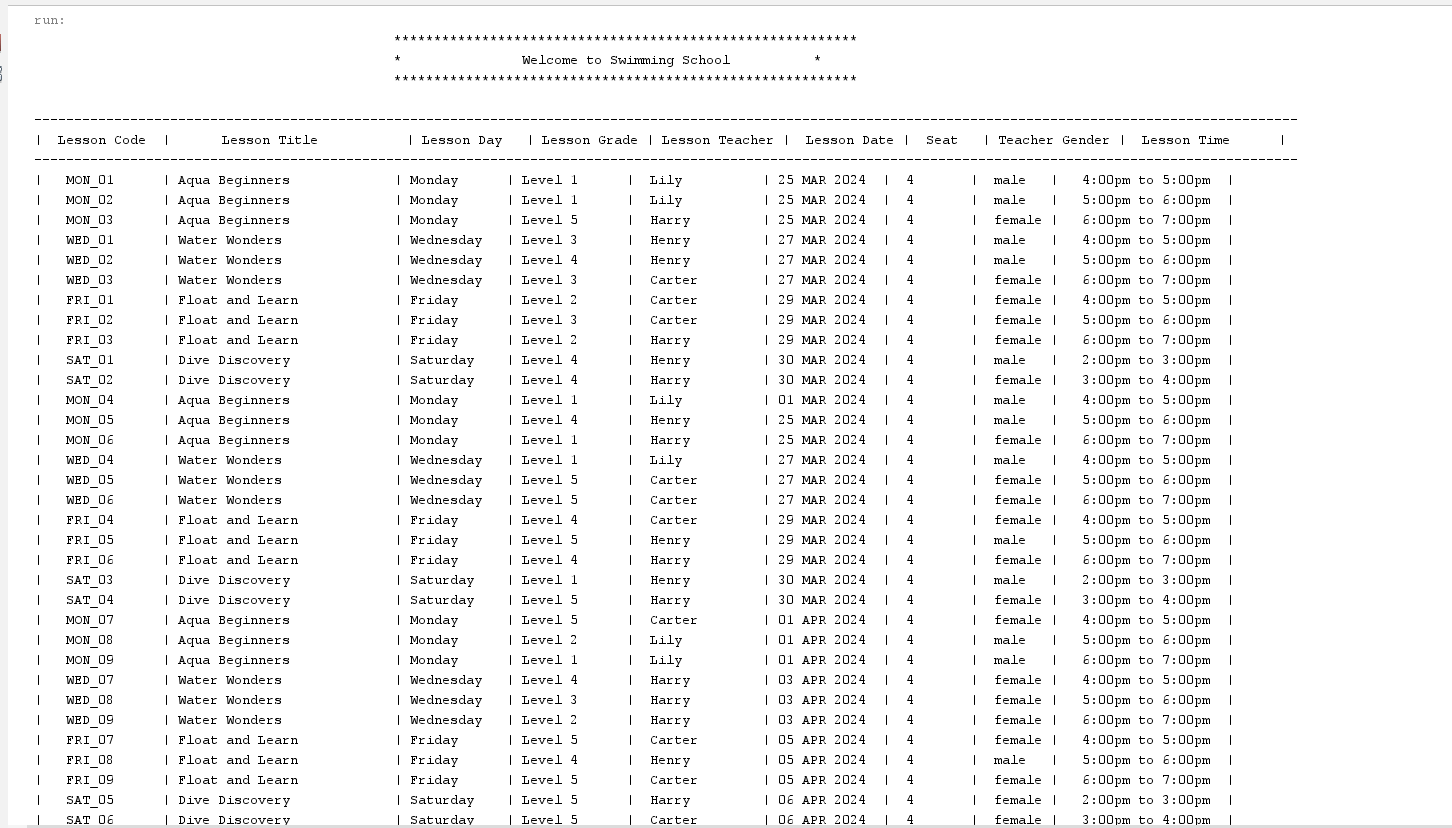
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | description | input | expResult | Actual Result |
| 6 | Test filter timetable by teacherName | teacherName | Fiter timetable by given teacherName and store in filterTimeTable list | Fiter timetable by given teacherName and store in filterTimeTable list |



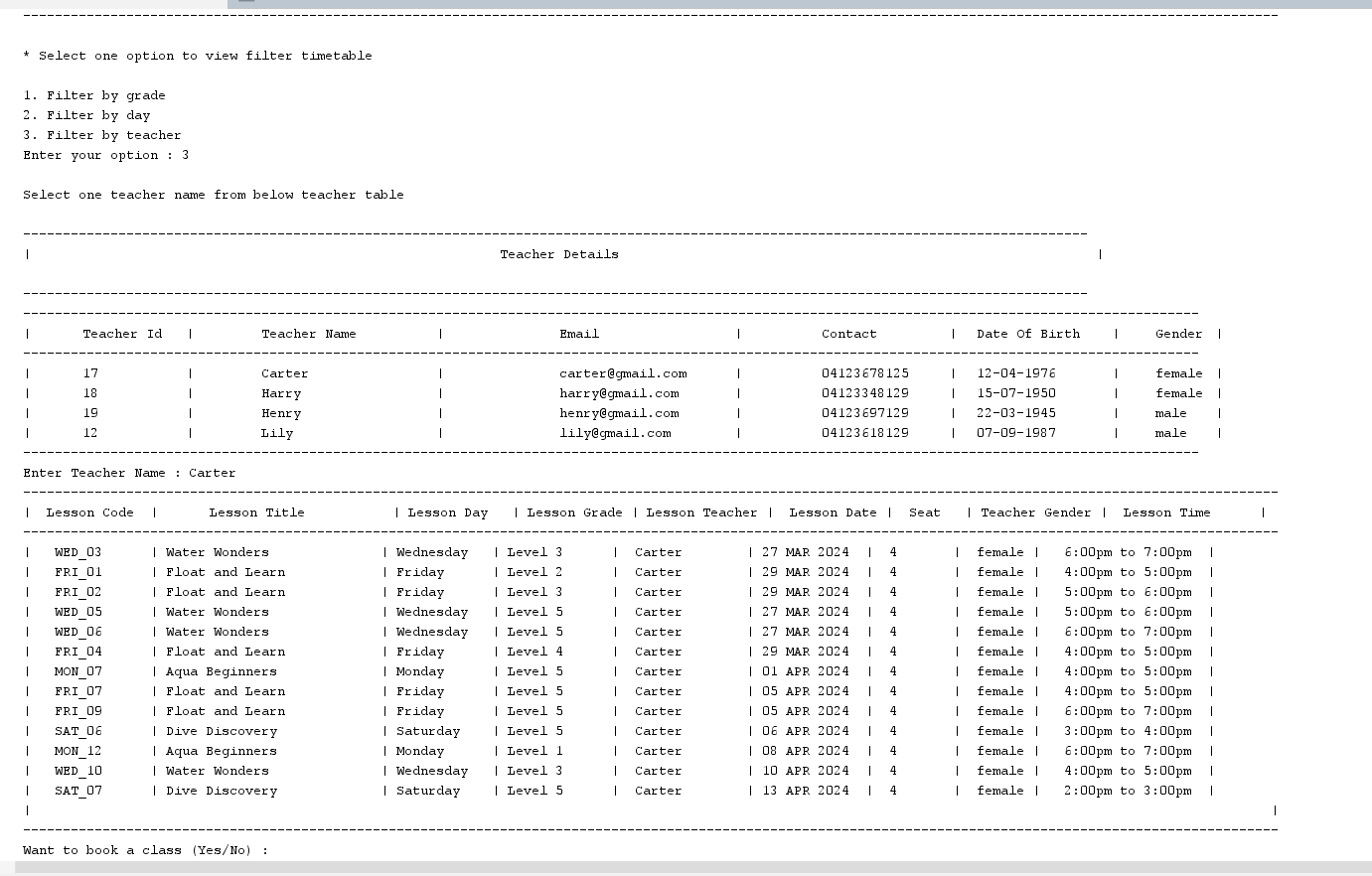
Output :



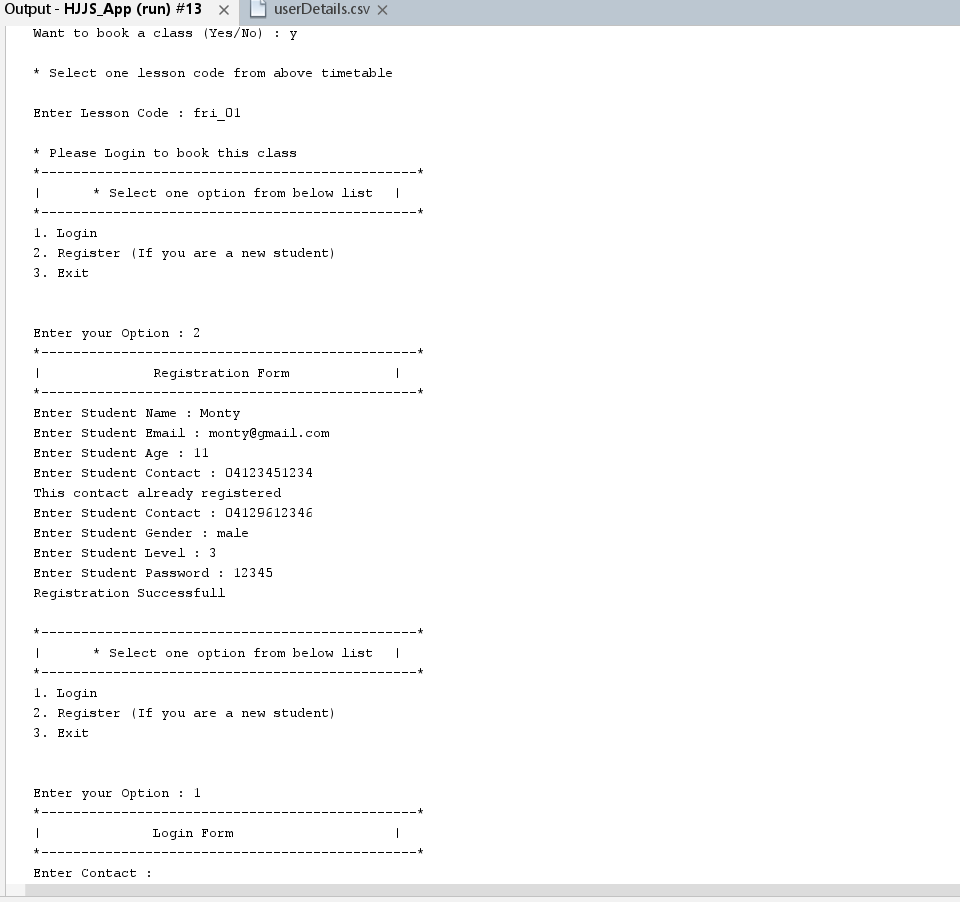
## Output screen- shot of this project

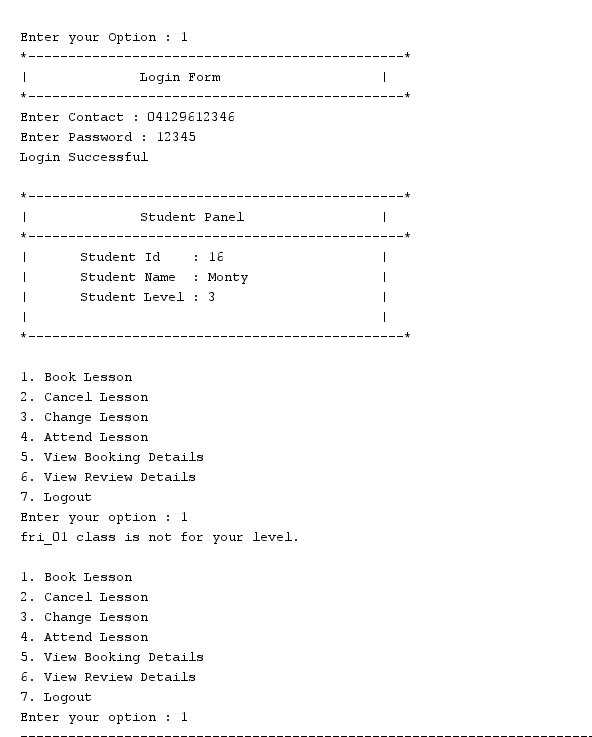
****

### Take input from student to filter time table

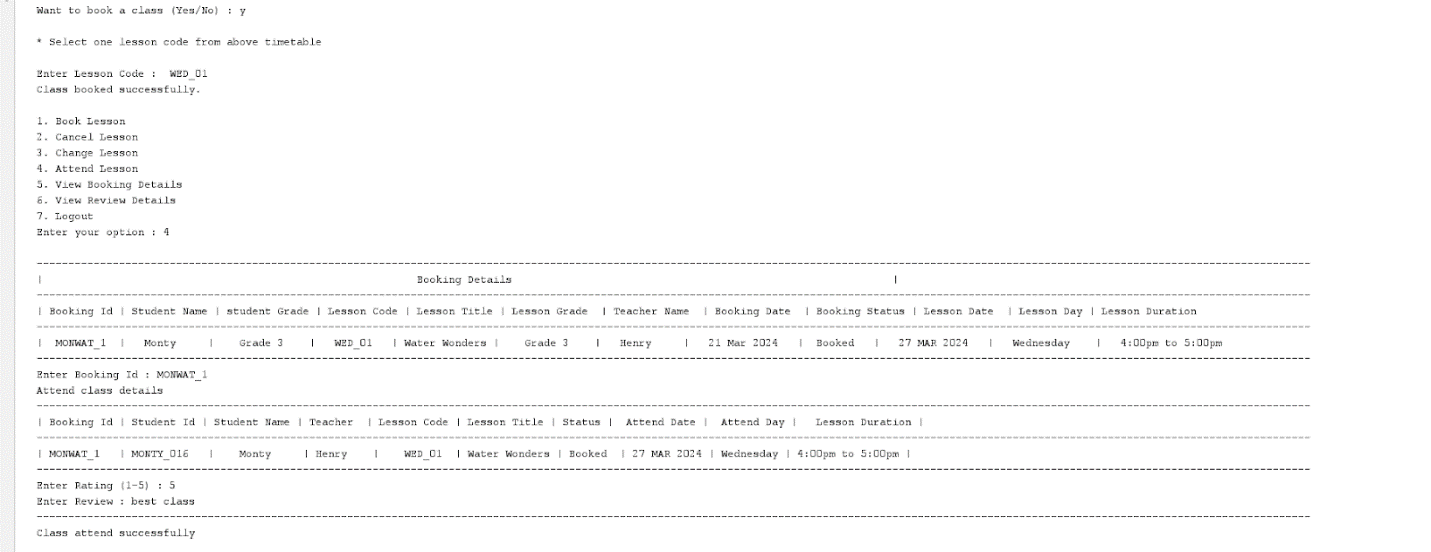
****

### Student register and login

****

****

### Book and attend class

****