LONDON
metropolitan
university

islington college
(इस्लिङ्टन कलेज)

CS5002NI Software Engineering

Sound Strong Music Institute

35% Individual Coursework

2020-21 Spring

Student Name: MD OSHAMA
London met ID:19031128
College ID:NP01CP4A190177

Coursework Given: week 20
Submission Date: week 26(2nd May 2021)

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Table of Content

Contents

List of Figure

List of Table

1. GANTT Chart

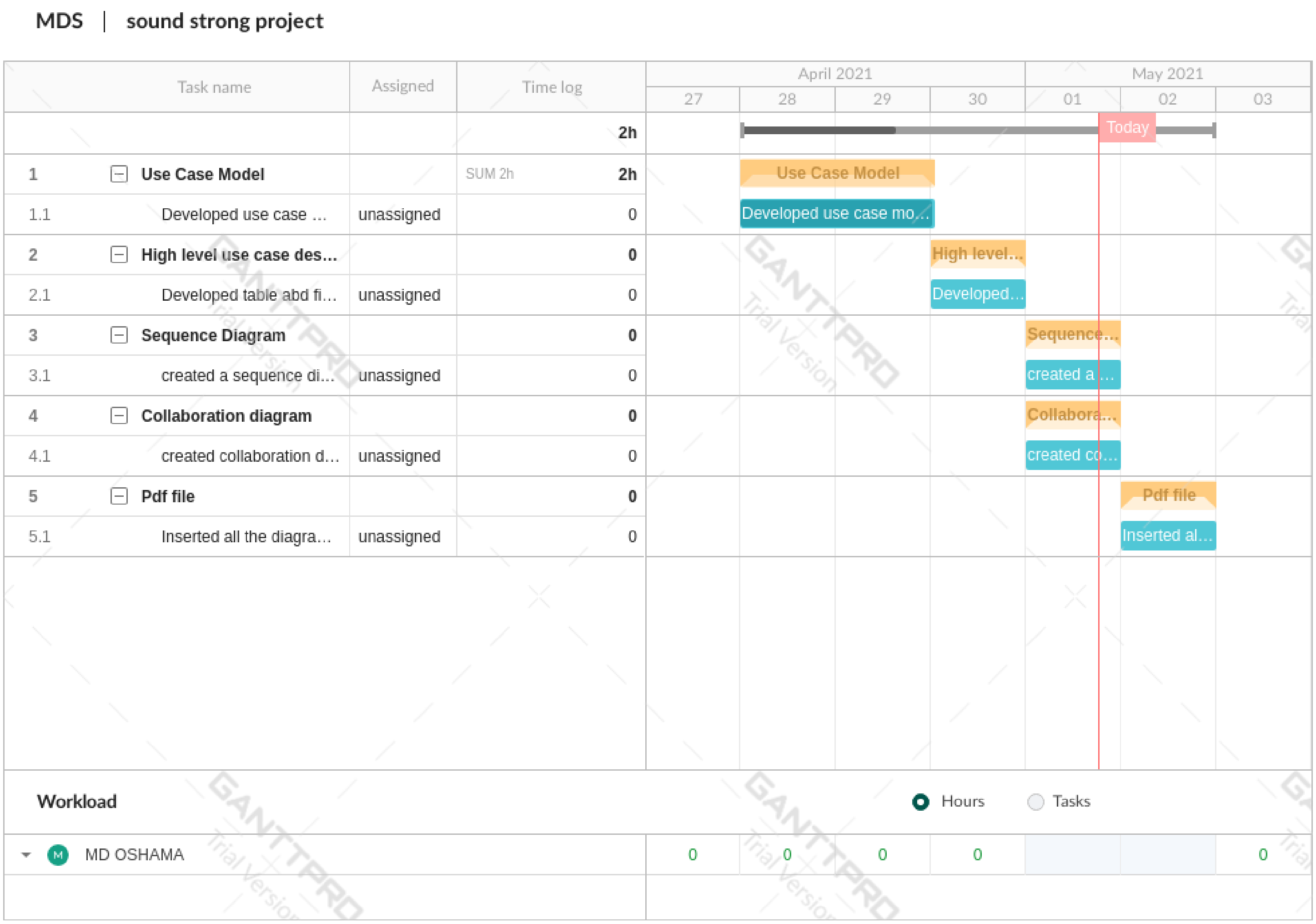
A Gantt chart is a project management tool assisting in the planning and scheduling of projects of all sizes, although they are particularly useful for simplifying complex projects. Project management timelines and tasks are converted into a horizontal bar chart, showing start and end dates, as well as dependencies, scheduling and deadlines, including how much of the task is completed per stage and who is the task owner. This is useful to keep tasks on track when there is a large team and multiple stakeholders when the scope changes.

As it's in a bar chart format it is possible to check on progress with a quick glance. We can easily see:

1. a visual display of the whole project,
2. timelines and deadlines of all tasks,
3. relationships and dependencies between the various activities,
4. project phases
5. Project management solutions that integrate Gantt charts give managers visibility into team workloads, as well as current and future availability, which allows for more accurate scheduling.

Gantt charts have been around for nearly a century, having been invented by Henry Gantt, an American mechanical engineer, around 1910. *(APM, 2020)*

For the Given Coursework I have also developed a GANTT chart so that it will be very easy to carry out my works on time.



MDS | sound strong project

	Task name	Assigned	Time log
			2h
1	⊟ **Use Case Model**		SUM 2h **2h**
1.1	Developed use case ...	unassigned	0
2	⊟ **High level use case des...**		**0**
2.1	Developed table abd fi...	unassigned	0
3	⊟ **Sequence Diagram**		**0**
3.1	created a sequence di...	unassigned	0
4	⊟ **Collaboration diagram**		**0**
4.1	created collaboration d...	unassigned	0
5	⊟ **Pdf file**		**0**
5.1	Inserted all the diagra...	unassigned	0

Workload ● Hours ○ Tasks

	27	28	29	30	01	02	03
M MD OSHAMA	0	0	0	0			0

Figure 1: GANTT chart

2. Use Case Model

A use-case model is a model of how different types of users interact with the system to solve a problem. As such, it describes the goals of the users, the interactions between the users and the system, and the required behavior of the system in satisfying these goals.

A use-case model consists of a number of model elements. The most important model elements are: use cases, actors and the relationships between them.

A use-case diagram is used to graphically depict a subset of the model to simplify communications. There will typically be several use-case diagrams associated with a given model, each showing a subset of the model elements relevant for a particular purpose. The same model element may be shown on several use-case diagrams, but each instance must be consistent. If tools are used to maintain the use-case model, this consistency constraint is automated so that any changes to the model element (changing the name for example) will be automatically reflected on every use-case diagram that shows that element.

The use-case model may contain packages that are used to structure the model to simplify analysis, communications, navigation, development, maintenance and planning.

Much of the use-case model is in fact textual, with the text captured in the Use-Case Specifications that are associated with each use-case model element. These specifications describe the flow of events of the use case.

The use-case model serves as a unifying thread throughout system development. It is used as the primary specification of the functional requirements for the system, as the basis for analysis and design, as an input to iteration planning, as the basis of defining test cases and as the basis for user documentation. (utm.mx, 2020)

Development of use case Diagram for Sound Strong Company

a. Identification of Actor
b. Listing out Use cases
c. Linking them with the actor
d. Generalization
e. making a Frame

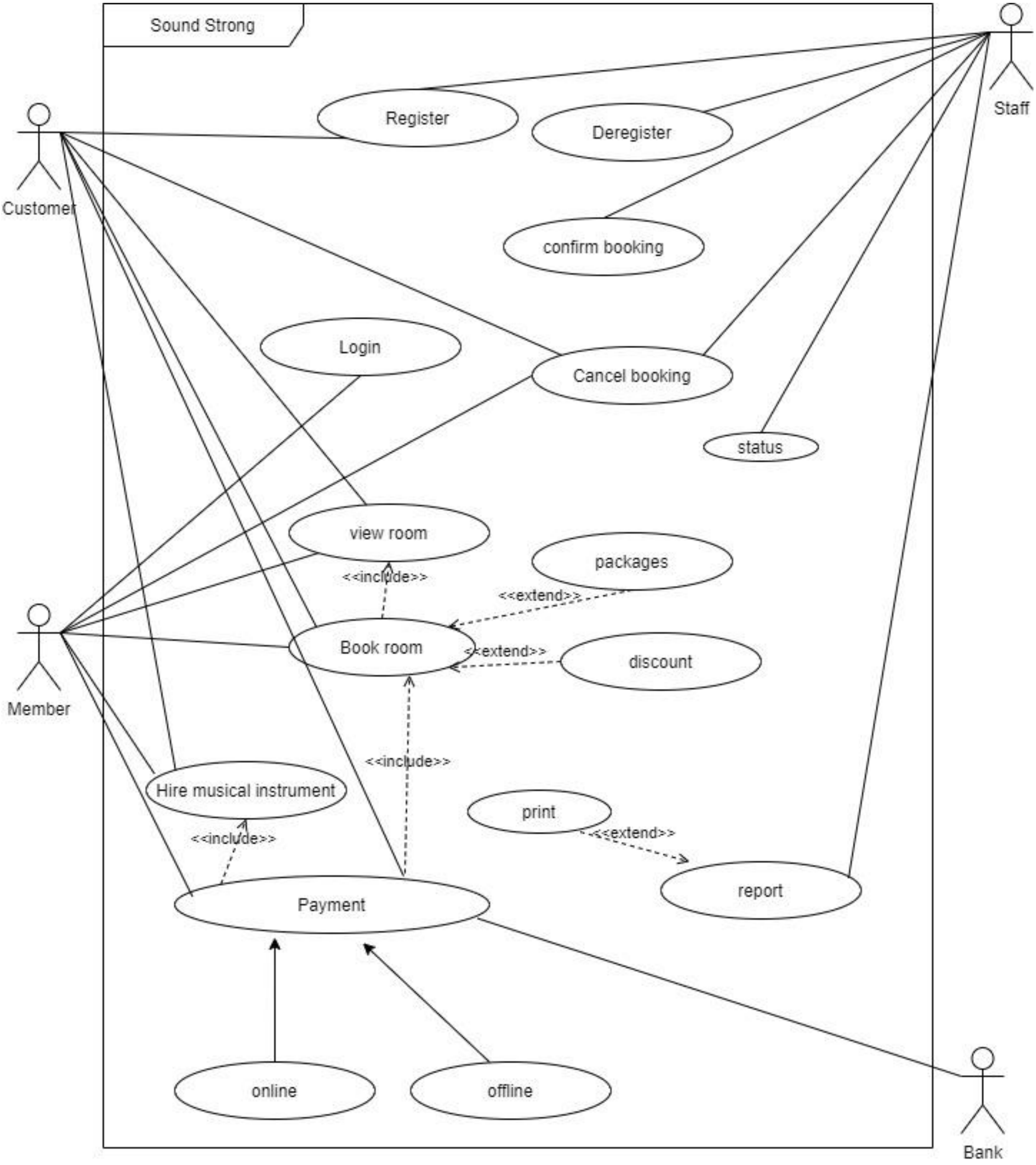


Figure 2 : Use Case Diagram

Actors
a. Customer
b. Member
c. Staff
d. Bank

Use Cases
a. Login
b. View Room
c. Book Room
d. Hire Musical instrument
e. Payment
f. Online
g. Offline
h. Register
I. Deregister
j. Confirm booking
k. cancel booking
l. Status
m. Report
n. Print
o. Packages
p. Discount

High level use Case Description

Use Case	Login
Actor	Member, staff
Description	A member should login to the server by entering login details. Staff also need to login to the server.

Table 1: Login

Use Case	View room
Actor	Member, customer
Description	When a customer register by filling out the details he/she become able to view the rooms similarly when member login to the server he/she can view the rooms.

Table 2: View room

Use Case	Book room
Actor	Member, customer
Description	Both the member and customer can book the room if the are inside the software.

Table 3: Book room

Use Case	Hire musical instrument
Actor	Member, customer
Description	Both the member and customer can hire musical instruments.

Table 4: hire musical instruments

Use Case	payment
Actor	Member, customer, bank
Description	After the booking of room member or customer have to do the payment either offline or online.

Table 5: Payment

Use Case	online
Actor	Member, customer, bank
Description	If the customer or member select to pay online bank validates there card and confirms the payment.

Table 6:online

Use Case	offline
Actor	Member, customer
Description	If the member or customer does the payment offline simply a slip is generated by server and given to them

Table 7:offline

Use Case	Register
Actor	Customer, staff
Description	Customer and staff must fill up the details to register them in the software.

Table 8: Register

Use Case	Deregister
Actor	staff
Description	A staff can easily de register any customer or staff itself can be deregistered if he/she left the job.

Table 9: Deregister

Use Case	Confirm booking
Actor	Staff
Description	A staff can confirm the booking of the customer or member by checking the status

Table 10:confirm booking

Use Case	Cancel booking
Actor	Staff, customer, member
Description	A customer or member can cancel their bookings. A staff can also cancel the customer or member booking by checking the status

Table 11: cancel booking

Use Case	status
Actor	staff
Description	A staff can check the status of rooms and musical instrument.

Table 12:status

Use Case	Report
Actor	staff
Description	A staff need to make the report of the bookings, cancellation etc.

Table 13:Report

Use Case	Print
Actor	staff
Description	After making the report staff may print that report or may not print just save it.

Table 14: print

Expanded Use Case Description

Use Case	Book room
Actor	Customer, Member,
Description	A customer or member can book room by entering to the server.
Member	System
1. Enter Login details	2. confirm login
3.Book room	4. Check the status
	5. Send confirmation
6.Select payment method	7. Provides option offline/online
6.Pay online	7. Enter card details
8. Enters Details	8. Confirms the payment
	9.Provides Payment receipt
10.Gets the payment receipt/bill	
11. Log out from server	

Table 15:Expanded use case diagram

3. Sequence Diagram

Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

Purpose of Sequence Diagram

a. Model high-level interaction between active objects in a system
b. Model the interaction between object instances within a collaboration that realizes a use case
c. Model the interaction between objects within a collaboration that realizes an operation
d. Either model generic interactions (showing all possible paths through the interaction) or specific instances of
e. interaction (showing just one path through the interaction) (paradigm, 2020)

For the sound Strong Company, a Sequence Diagram is Drawn below

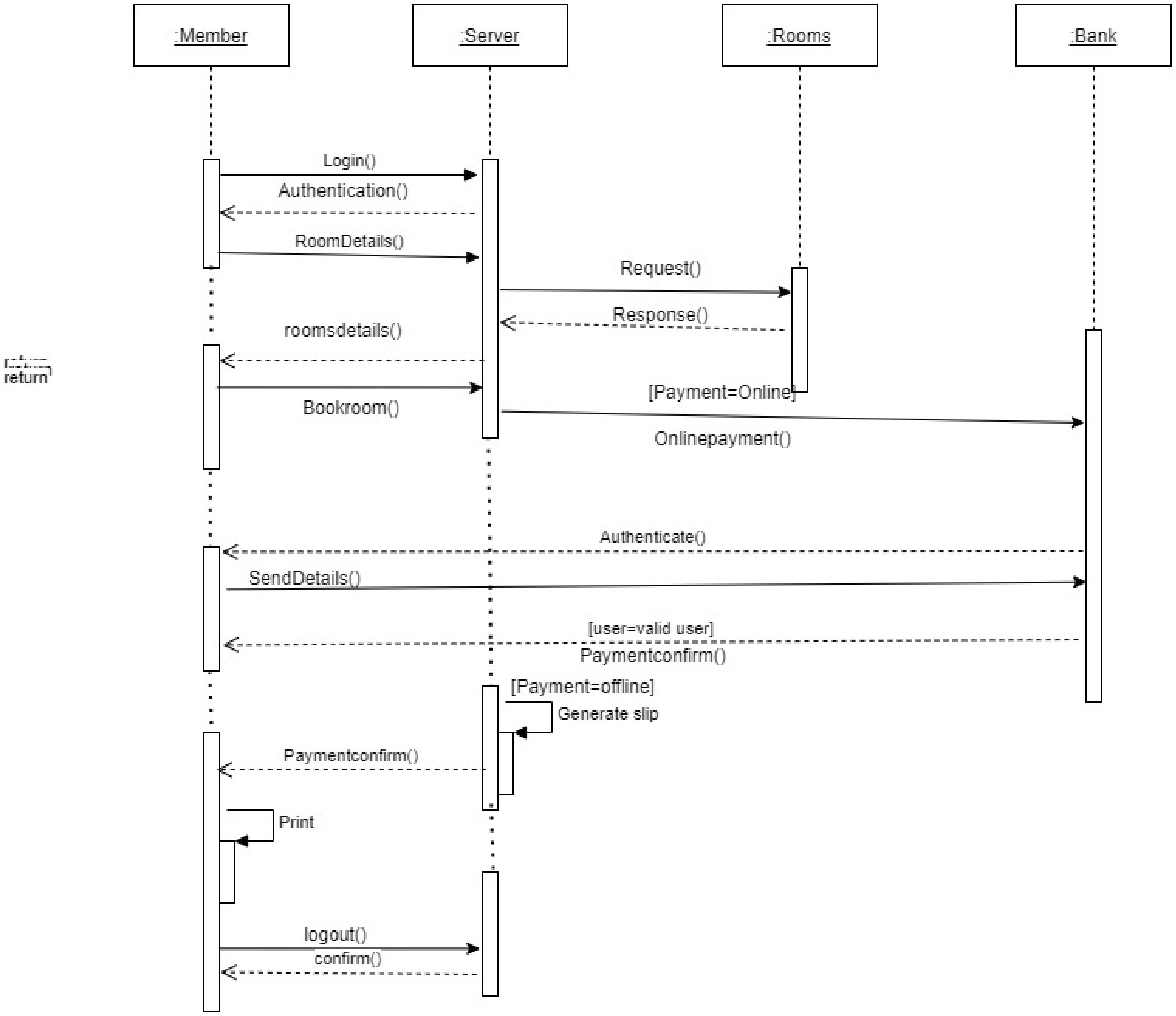


Figure 3:Sequence Diagram

In the above sequence diagram, I have made altogether four objects named as
a. Member
b. Server
c. Rooms
d. Bank

I have represented step by step process.

Step 1: Member's login
step 2: Authentication comes from server
step 3: Member ask for room details
step 4: server requests room details from object rooms
step 5: object rooms Response to the server
step 6: server sends the room details to member
step 7: Member Books the room
step 8: if payment is online then server takes to the bank
step 9: Bank authenticate
step 10: member sends card details
step 11: if user is valid payment is confirmed
step 12: if payment is offline server generates slip
step 13: server confirms the payment
step 14: customer can print the slip
step 15: member ask to logout
step 16: server confirms logout

4. Collaboration/communication diagram

Collaboration diagrams are used to show how objects interact to perform the behavior of a particular use case, or a part of a use case. Along with sequence diagrams, collaborations are used by designers to define and clarify the roles of the objects that perform a particular flow of events of a use case. They are the primary source of information used to determining class responsibilities and interfaces.

Unlike a sequence diagram, a collaboration diagram shows the relationships among the objects. Sequence diagrams and collaboration diagrams express similar information, but show it in different ways. Collaboration diagrams show the relationships among objects and are better for understanding all the effects on a given object and for procedural design.

Because of the format of the collaboration diagram, they tend to better suited for analysis activities (see Activity: Use-Case Analysis). Specifically, they tend to be better suited to depicting simpler interactions of smaller numbers of objects. As the number of objects and messages grows, the diagram becomes increasingly hard to read. In addition, it is difficult to show additional descriptive information such as timing, decision points, or other unstructured information that can be easily added to the notes in a sequence diagram. (Artifacts, 2020)

For the Sound Strong Company, I have created a Collaboration Diagram

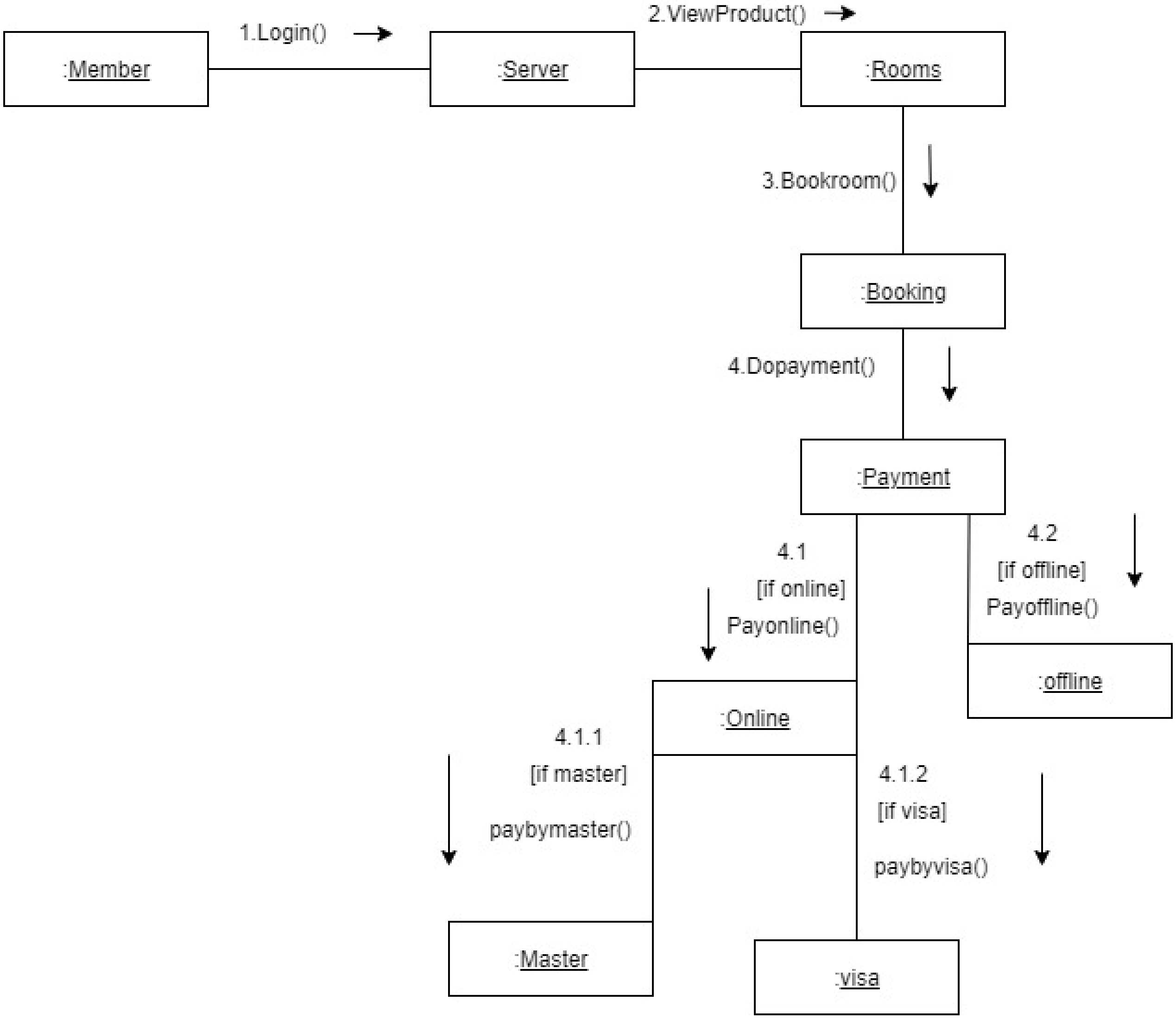


Figure 4: Collaboration Diagram

Explanation of Collaboration model step by step

Step 1: Member login to the server
step 2: Server takes to the Rooms where customer can view rooms
step 3: Member books room
step 4: the information is stored in booking
step 5: Member have to do the payment
step 6: Select online or offline payment Method
step 7: If Payment online pay through Either master card or Visa Card
Step 8: If Payment is offline Program is aborted by printing slip

5. Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

Purpose of Class Diagrams

Shows static structure of classifiers in a system

Diagram provides a basic notation for other structure diagrams prescribed by UML

Helpful for developers and other team members too

Business Analysts can use class diagrams to model systems from a business perspective

A UML class diagram is made up of:

A set of classes and

A set of relationships between classes

A class notation consists of three parts:

1. Class Name

    The name of the class appears in the first partition.

2. Class Attributes

    Attributes are shown in the second partition.

    The attribute type is shown after the colon.

    Attributes map onto member variables (data members) in code.

3. Class Operations (Methods)

    Operations are shown in the third partition. They are services the class provides.

    The return type of a method is shown after the colon at the end of the method signature.

    The return type of method parameters is shown after the colon following the parameter name.

    Operations map onto class methods in code (paradigm, 2020)

For the Sound Strong Company, a Class Diagram is developed by me

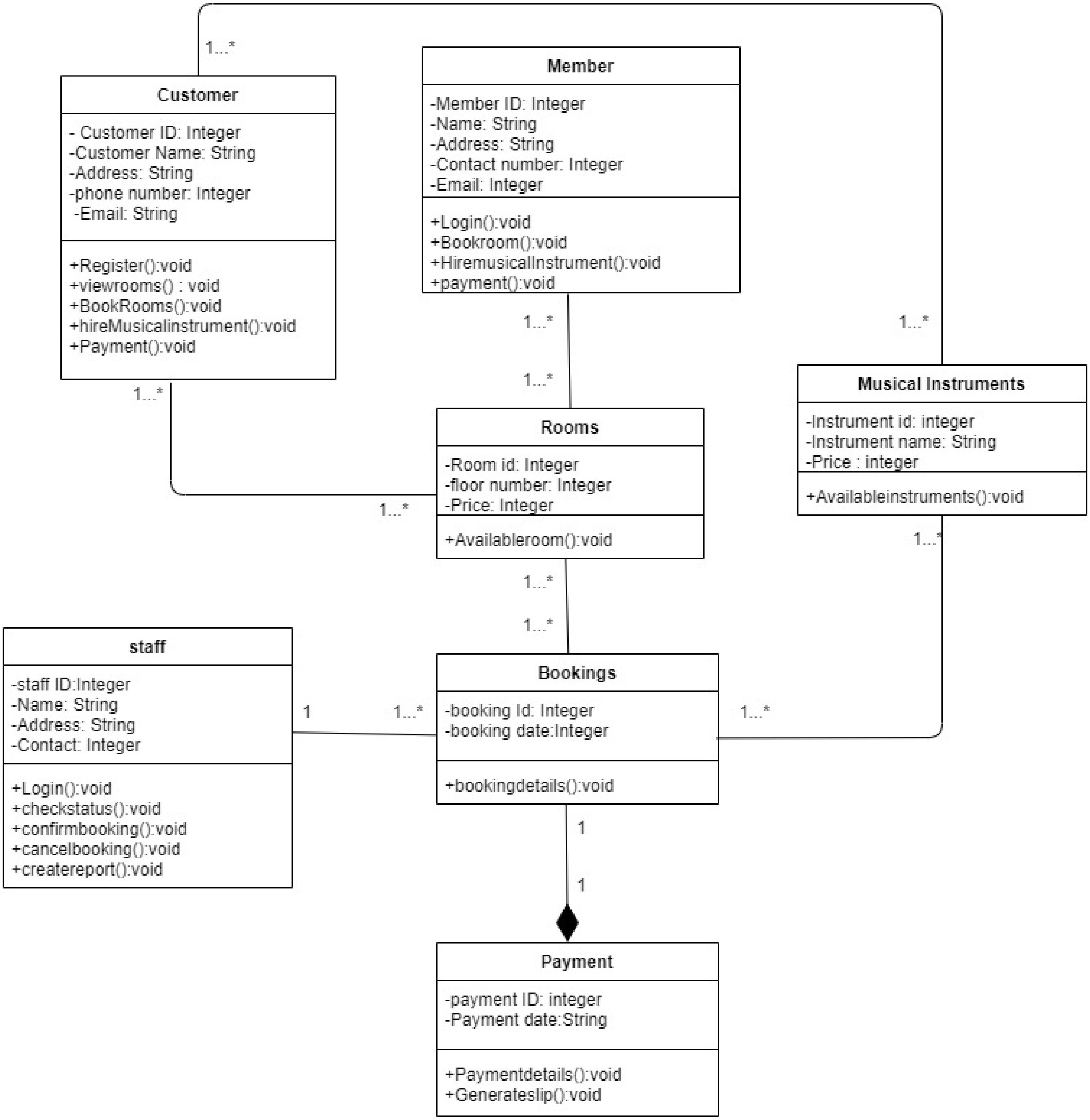


Figure 5:Class Diagram

In the above Class Diagram
There are altogether seven classes made named as

1. Member
2. Customer
3. Rooms
4. Musical Instrument
5. Booking
6. Staff
7. Payments

All the classes have their own attributes and methods and every classes are inter-related with each other
1.Relationship between Customer and Rooms

One customer can view one or many rooms and in other side one room can be viewed by one or many customers. This kind of relation is denoted by 1…* ---------- 1…*

2. Relationship between Member and rooms

Member have the same thing as the customer One member can view one or many rooms and in other side one room can be viewed by one or many members. This kind of relation is denoted by 1…* ---------- 1…*

3. Relationship between staff and booking

One staff can take one or many bookings but in other side one booking can be taken by only one staff. This kind of relation is denoted by 1 ------------- 1…*

4. Relationship between booking and Payment

Payment is totally dependent on the bookings. Only one payment is accepted for one booking similarly one booking is paid one time. If someone does not pay then booking will not take place. This kind of Relation is denoted by Filled diamond arrow and 1 ----------------- 1.

Similarly other classes are also related with each other.

Design

Techniques to Design

a. Structure Charts
The Structure Chart is a basic tool of Structured System Design. It is a graphic representation of a hierarchy of modules and the relationships between them. In Section 2 two types of Structure Charts were discussed; they were referred to as Structure Charts and Program Structure Charts. The first type represents the detailed functional breakdown of the software without taking packaging into account. The second type illustrates the actual programs and procedures to be developed.

A Structure Chart represents system structure. It can also be used to illustrate flow of information between modules. It does not illustrate flow of control; it does not provide any information about sequencing of events or about condition tests.

b. Data Flow Oriented Structure

Data flow-oriented decomposition is a method of deriving the modular design based on the data flow diagram. It is the recommended method for processing-oriented applications such as Accounting, Payroll and Inventory.

c. Data Structure Oriented Design

Data Structure Oriented Design is a technique of modular decomposition which transforms a representation of a data structure into a representation of software.
This method can be used for systems with a well-defined, hierarchical structure of information, heavy input/output flow and little processing logic; e.g., a library catalogue system

D. Object-Oriented Design

Object-Oriented Design is an approach to modular decomposition which views a problem in terms of objects and operations and defines the solution in the same terms. It is applicable to applications such as simulation systems, real time systems, and to systems with complex or dynamic entities which are difficult to represent using traditional techniques.

This approach requires a language in which real world objects can be represented. These languages typically build classes of an object in the application.

For each class, the attributes and properties that each object in the class shares are defined. The actions which each object can perform are described. These are written in the form of messages to which each object in the class can respond. As a result, all communication between any two objects, whether they are from the same class or not, is performed by sending messages. Thus, object-oriented languages have a very high degree of encapsulation, in that each object is responsible for its own operation, and no other object may access its structure directly. The net result is that each message is like a separate black-box, a property which makes system design more manageable.

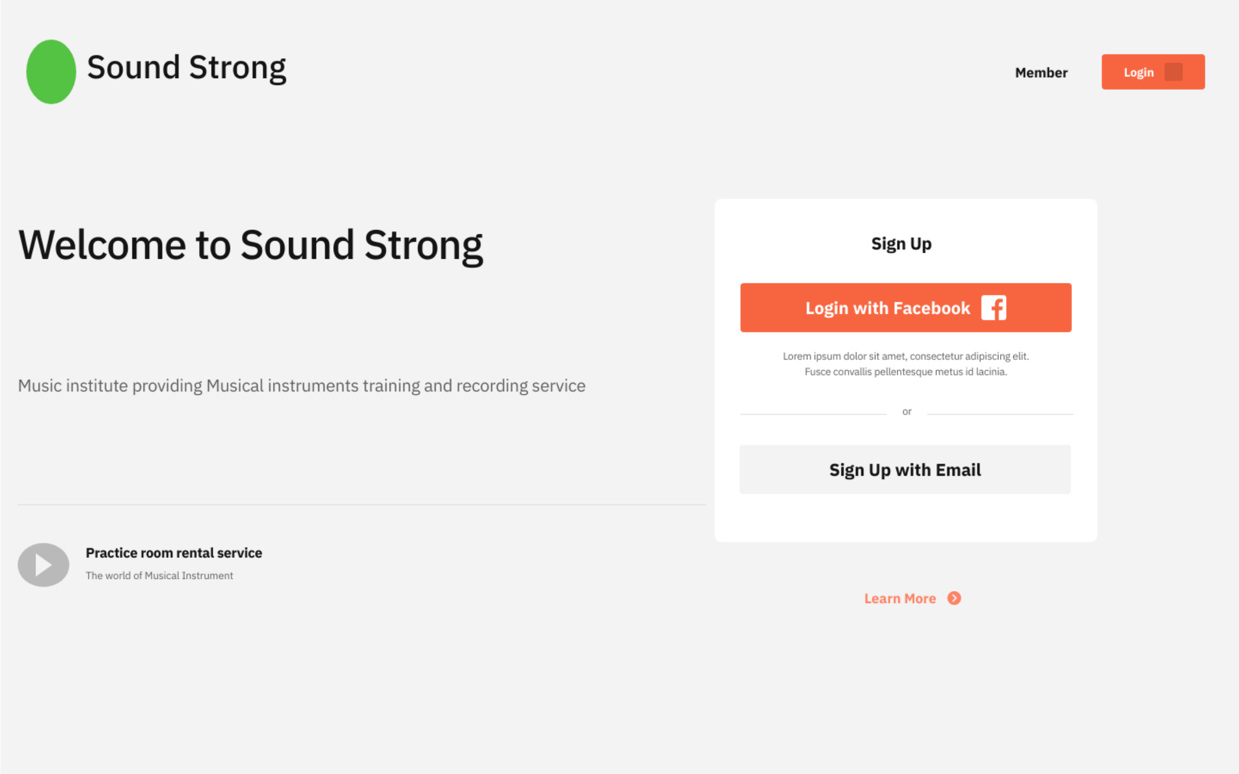
New classes are introduced as specializations or generalizations of existing classes. For example, in a naval simulation system, a class SHIP could be introduced, and the attributes and properties of a ship defined. Specializations of this class, such as FRIGATE and DESTROYER, could then be introduced. These two new classes could inherit the properties of the class SHIP, while defining those properties which makes them unique.

Many object-oriented languages are now becoming available, such as SMALLTALK, LISP FLAVERS, Objective C, C++, ADA, and SIMULA.

The following is an example of a temperature monitoring system which will be used to illustrate Object Oriented Design. The system is described as follows:
‘There are ten independent sensors which continually monitor temperature. Initially all of the sensors are disabled. If any of the enabled sensors register an out of limits value the system must immediately post an alarm condition. It must also record the status of all sensors every fifteen minutes. Asynchronously it must be able to get a user command to enable or disable a sensor, or set the temperature limits.” (ToolBox, 2020)

Prototype

1. Sign Up Page



Welcome to Sound Strong

Music institute providing Musical instruments training and recording service

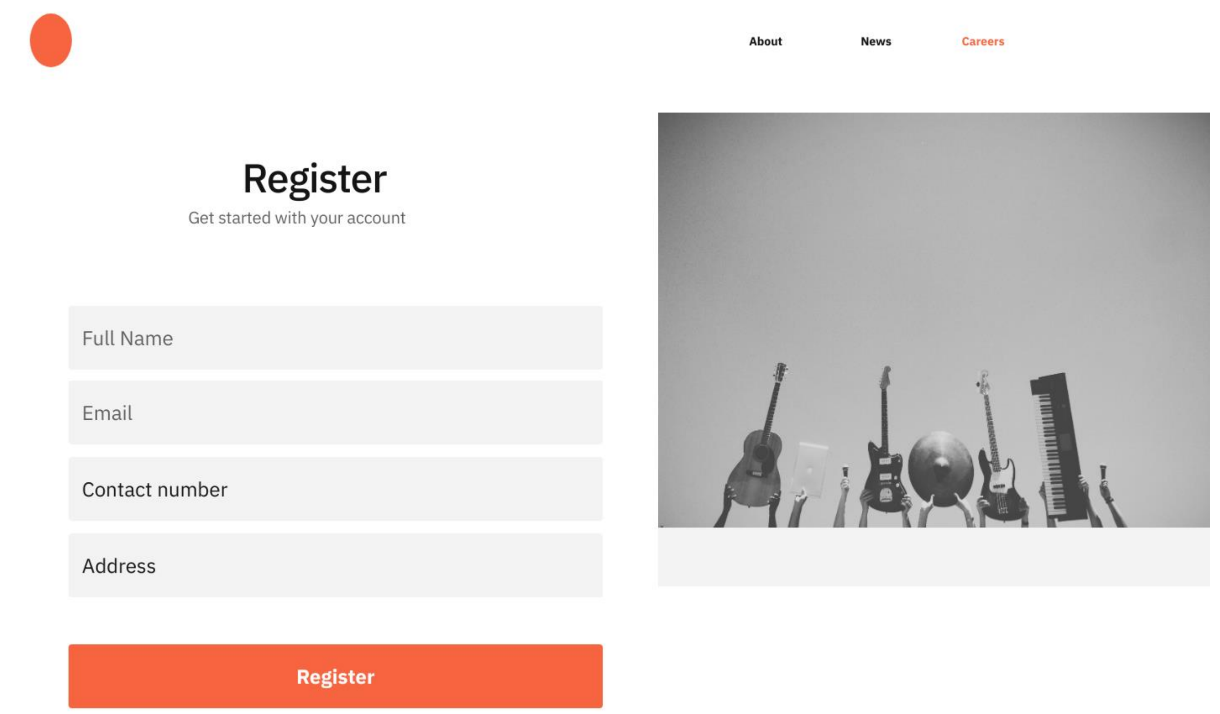
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce convallis pellentesque metus id lacinia.

Practice room rental service

The world of Musical Instrument

Figure 6: Sign up Page

2.Register Page



Register

Get started with your account

Figure 7: Registration form page

3. Login Page

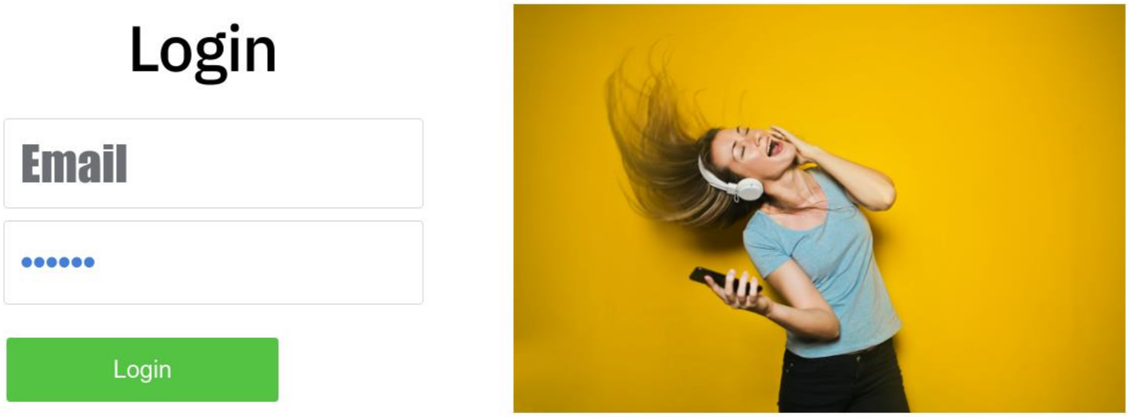


Figure 8:Login Page

4. Book Room Section

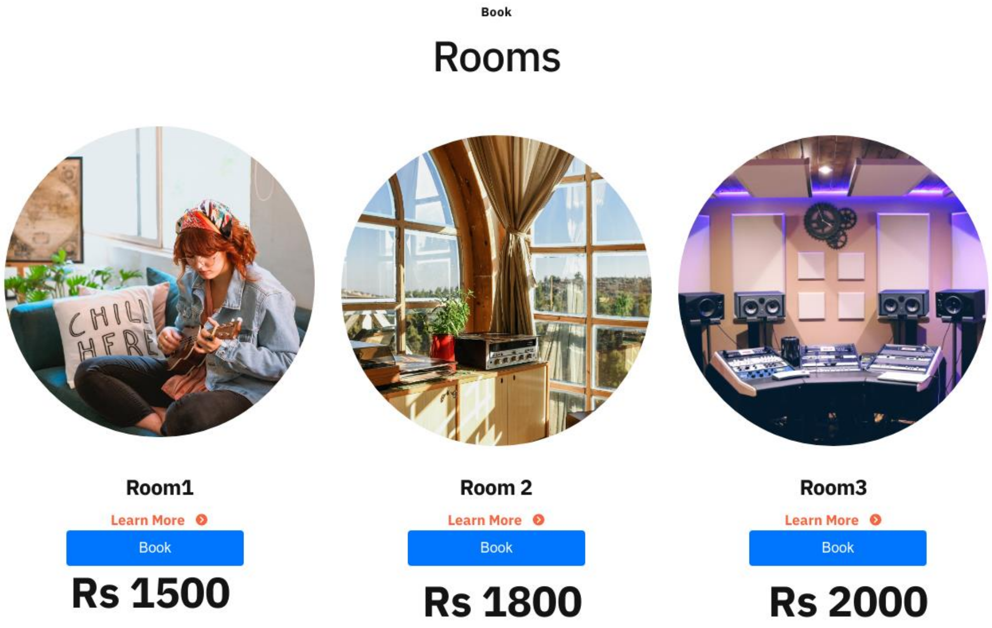


Figure 9: Book room section

5. Hire Musical instruments

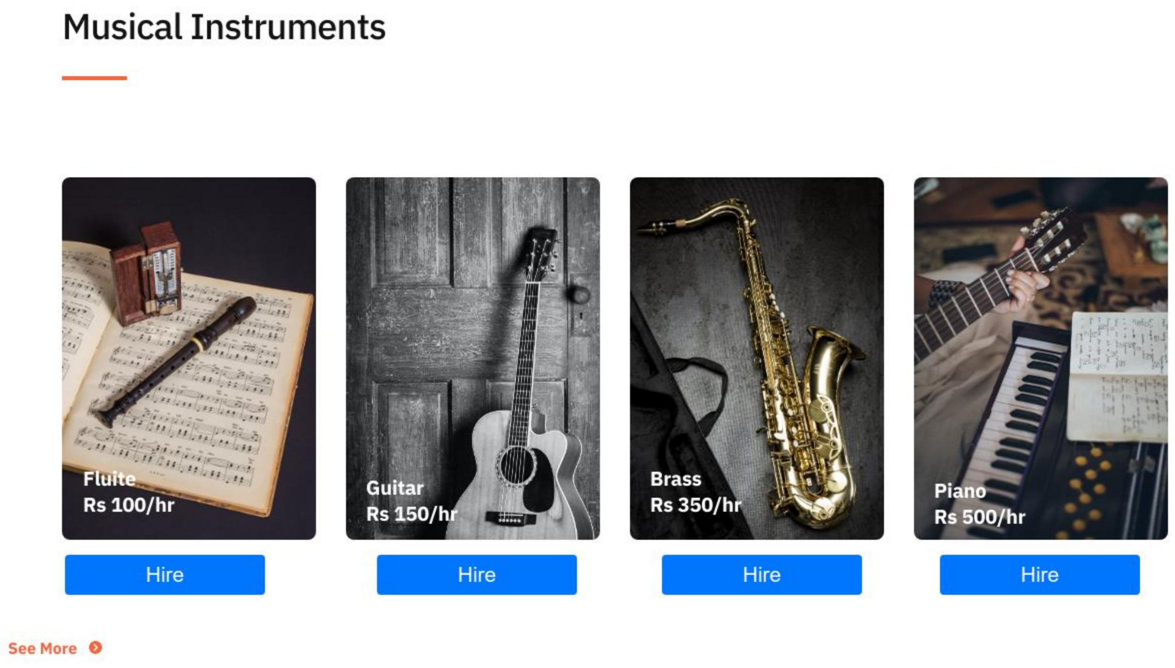
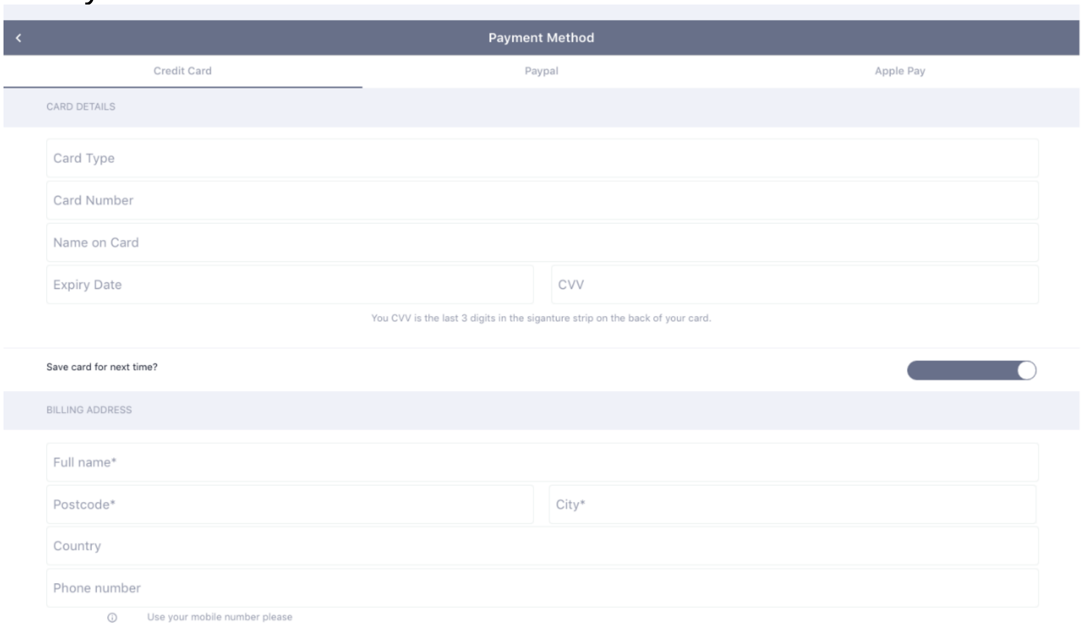


Figure 10 Hire musical instrument

6. Payment



Payment Method

Credit Card | Paypal | Apple Pay

CARD DETAILS

Card Type

Card Number

Name on Card

Expiry Date

CVV

You CVV is the last 3 digits in the siganture strip on the back of your card.

Save card for next time?

BILLING ADDRESS

Full name*

Postcode*

City*

Country

Phone number

Use your mobile number please

Figure 11Payment Section

Conclusion

Talking about the main goal of the project that was before building a software or a system what things must be done so that the system will be easily build. There were many diagrams created and many processes shown. As for the Sound Strong Company Simply the customer can book the practice rooms and hire the musical instrument while practicing so the system simply register the new customer and allows the customer to view the practice rooms and book the practice rooms. A customer can become a member also and get discounts and packages so simply system allows the member to login into the server and they can also book and hire and also do the payment. Payment is accepted offline and online too. If a customer selects to do the payment online then the system simply asks for the card details and if the customer select offline then simply system generates a slip/bill and give it to the customer after the payment the members can log out from the system.

The system for the sound strong company was developed. Creation of Class table, Use case diagram, Sequence Diagram and collaboration diagram took place. High level description was also given for the use cases.

Compiling these all things then we come to a conclusion that I learned a lot from this coursework, the previous group coursework which was assigned to us helped a lot in doing this coursework and now i have confidence to develop any kinds of system for any organization if provided to me.
Finally, I am very thankful to my module teachers for this coursework and there tutorial video present on YouTube also helped me a lot to complete this coursework.

Thank You