



Table of Contents

Abstract.....	3
Introduction	3
Database design.....	4
ER Modeling	4
Normalization.....	5
UNF.....	5
1NF	5
2NF	6
3NF	7
Software Design	8
Use Case Diagram	8
Functional Requirements.....	9
Activity Diagrams	9
List Activity Diagram	9
Add Activity Diagram	10
UPDATE Activity Diagram.....	11
DELETE Activity Diagram	12
Use Case Specification Diagrams	13
Class Diagram.....	15
Sequence Diagram	16
Graphical User Interface (GUI).....	17
Testing.....	18
Evidences	18
Conclusion.....	20
Limitations/Critical thinking.....	20
Future approach.....	20
References	21

Abstract

After analysing the coursework specifications, a step by step procedure was laid out to work on same. At first, a database design was created thereby identifying major entities for the coursework. Subsequently, lots of effort were put into coding the gym management system. At the same time, testing was carried out side by side with coding to reduce errors. A final testing was applied once the project has been successfully implemented. We have come to the conclusion that the system is a fully functioning user friendly one.

Introduction

The goal of this assignment was to create a gym management system to principally alleviate difficulties faced by the gym staffs when clients make several gym bookings and amendments at the same time.

A connection had to be established between the system and a database which stores trainers and client details so that any changes made to the system is simultaneously reflected in the database.

Two specific programming languages, Java and MySQL, were targeted throughout the implementation even though Java was widely used. This system also makes use of a client-server architecture which is easily accessible to the staff members.

A proper database design had to be implemented to accomplish the project. An ER diagram was used to understand the basic requirements of the project scope followed by normalisation to remove abnormalities. Unified Modeling Language was also applied to the scenario to visualise the system design.

Lastly, a Graphical User Interface(GUI) was used to assist the staffs into manipulating elements on the screen to achieve desired functionalities. This report also gives an insight how the system was developed and consists of 4 chapters namely database design, software design, software testing and conclusion.

Database design

A proper database design is crucial for the realisation of any project and it assists in the creation of a logical and a physical design model of a proposed database.

- Logical model: Developing a database based on functional requirements.
- Physical model: Implementing the logical model with Database Management System (DBMS).

The main database design techniques implemented in this project are as follows:

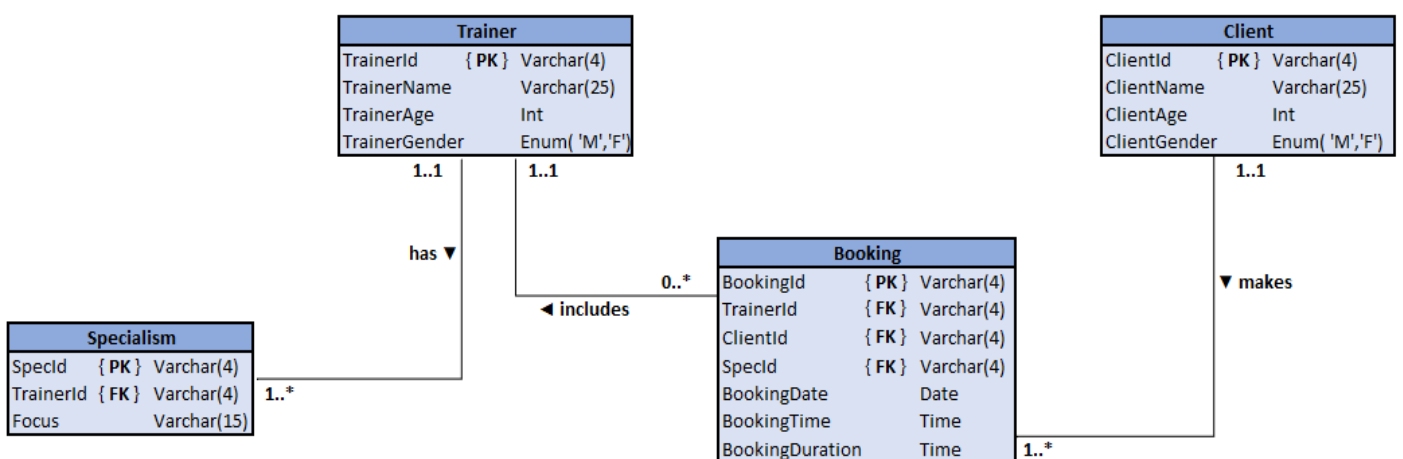
1. ER Modeling
2. Normalization

ER Modeling

ER Modeling is used to model entities, attributes and the relationship between entities. In order to produce the ER model, the core functionalities had to be identified. After analysing the coursework specifications, below core functionalities were identified:

- List bookings
- Add bookings
- Update bookings
- Delete bookings

Once the above functionalities were finalised, a final ERD was then constructed as shown below:



Assumptions made during the project:

1. In order to be a client, the client must have at least one booking and each booking session consists of only one client.
2. A trainer can have no booking at all, but each booking session is supervised by only one personal trainer.
3. A trainer can have many specialisms but a specialism is taught by only one personal trainer.

Normalization

Relations containing redundant information are sometimes misleading, and they tend to suffer from Insertion, Deletion and Modification abnormalities. This is where normalization comes in handy. It reduces redundancy and data dependency by dividing larger tables into smaller one and links them together in terms of relationships.

UNF

BookingId	TrainerId	TrainerName	TrainerAge	TrainerGender	ClientId	ClientName	ClientAge	ClientGender	SpecId	Focus	BookingDate	BookingTime	BookingDuration
B001	PT01	David	35	M	C001	Anil	25	M	S001	Weight Loss	2020-10-01	09:00	01:00
B002	PT02	Arnold	37	M	C002	Karim	34	M	S003	Flexibility	2020-10-02	10:00	02:00
B003	PT03	Alexa	30	F	C003	Mary	22	F	S005	Box Jump	2020-10-03	11:00	03:00
B004	PT04	Sam	40	M	C004	Anita	30	F	S007	Bench Press	2020-10-04	09:00	04:00
B005									S008	Basu Squats	2020-10-05	10:00	01:00
B006	PT05	Robet	38	M	C005	Joee	38	M	S009	Foam Roller	2020-10-06	09:00	02:00
B007									S010	Zumba Dance	2020-10-07	10:00	03:00

Table1: Booking Table (UNF)

The Booking table is in Unnormalized Form (UNF), since records are redundant. Therefore, it has to be flattened out to be in First Normal Form(1NF) as shown below:

1NF

BookingId	TrainerId	TrainerName	TrainerAge	TrainerGender	ClientId	ClientName	ClientAge	ClientGender	SpecId	Focus	BookingDate	BookingTime	BookingDuration
B001	PT01	David	35	M	C001	Anil	25	M	S001	Weight Loss	2020-10-01	09:00	01:00
B002	PT02	Arnold	37	M	C002	Karim	34	M	S003	Flexibility	2020-10-02	10:00	02:00
B003	PT03	Alexa	30	F	C003	Mary	22	F	S005	Box Jump	2020-10-03	11:00	03:00
B004	PT04	Sam	40	M	C004	Anita	30	F	S007	Bench Press	2020-10-04	09:00	04:00
B005	PT04	Sam	40	M	C004	Anita	30	F	S008	Basu Squats	2020-10-05	10:00	01:00
B006	PT05	Robet	38	M	C005	Joee	38	M	S009	Foam Roller	2020-10-06	09:00	02:00
B007	PT05	Robet	38	M	C005	Joee	38	M	S010	Zumba Dance	2020-10-07	10:00	03:00

Table2: Booking Table (1NF)

2NF

Once, the full functional dependencies and partial dependencies have been identified, the 1NF Table can be further broken down into a Second Normal Form(2NF) table as shown below:

Trainer			
TrainerId	TrainerName	TrainerAge	TrainerGender
PT01	David	35	M
PT02	Arnod	37	M
PT03	Alexa	30	F
PT04	Sam	40	M
PT05	Robet	38	M

Table3: Trainer Table (2NF)

Client			
ClientId	ClientName	ClientAge	ClientGender
CO01	Anil	25	M
CO02	Karim	34	M
CO03	Mary	22	F
CO04	Anita	30	F
CO05	Joee	38	M

Table4: Client Table (2NF)

Booking							
BookingId	TrainerId	ClientId	SpecId	Focus	BookingDate	BookingTime	BookingDuration
BO01	PT01	CO01	SO01	Weight Loss	2020-10-01	09:00	01:00
BO02	PT02	CO02	SO03	Flexibility	2020-10-02	10:00	02:00
BO03	PT03	CO03	SO05	Box Jump	2020-10-03	11:00	03:00
BO04	PT04	CO04	SO07	Bench Press	2020-10-04	09:00	04:00
BO05	PT04	CO04	SO08	Basu Squats	2020-10-05	10:00	01:00
B006	PT05	CO05	SO09	Foam Roller	2020-10-06	09:00	02:00
B007	PT05	CO05	SO10	Zumba Dance	2020-10-07	10:00	03:00

Table5: Booking Table (2NF)

3NF

Transitive dependencies need to be identified for a database to be in Third Normal Form(3NF). Same can be done by determining which attribute's value relies upon another attribute through a second attribute column.

Booking						
BookingId	TrainerId	ClientId	SpecId	BookingDate	BookingTime	BookingDuration
BO01	PT01	CO01	SO01	2020-10-01	09:00	01:00
BO02	PT02	CO02	SO03	2020-10-02	10:00	02:00
BO03	PT03	CO03	SO05	2020-10-03	11:00	03:00
BO04	PT04	CO04	SO07	2020-10-04	09:00	04:00
BO05	PT04	CO04	SO08	2020-10-05	10:00	01:00
BO06	PT05	CO05	SO09	2020-10-06	09:00	02:00
BO07	PT05	CO05	SO10	2020-10-07	10:00	03:00

Table6: Booking Table (3NF)

Trainer			
TrainerId	TrainerName	TrainerAge	TrainerGender
PT01	David	35	M
PT02	Arnold	37	M
PT03	Alexa	30	F
PT04	Sam	40	M
PT05	Robert	38	M

Table7: Trainer Table (3NF)

Client			
ClientId	ClientName	ClientAge	ClientGender
CO01	Anil	25	M
CO02	Karim	34	M
CO03	Mary	22	F
CO04	Anita	30	F
CO05	Jojo	38	M

Table8: Client Table (3NF)

Specialism		
SpecId	TrainerId	Focus
SO01	PT01	Weight Loss
SO02	PT01	Muscle Gain
SO03	PT02	Flexibility
SO04	PT02	Squat
SO05	PT03	Box Jump
SO06	PT03	Plank
SO07	PT04	Bench Press
SO08	PT04	Barbell Squats
SO09	PT05	Foam Roller
SO10	PT05	Zumba Dance

Table9: Specialism Table (3NF)

Software Design

Software design is one of the most crucial phases of a software development cycle. It assists the programmer to understand user requirements before coding and implementation. The software implemented make use of a client-server architecture and a JDBC driver to access data from the database as shown below:

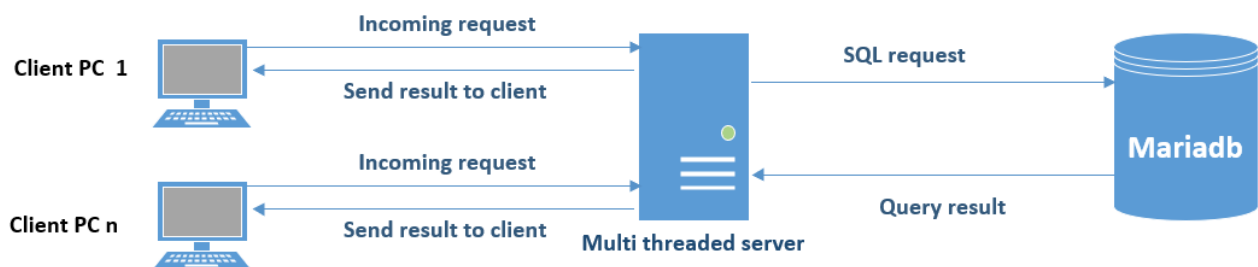


Fig1: Client server architecture

Whenever a client pc connects to the network, a separate thread is allocated to that client and each client receives the same priority level from the server. Any request made by client pcs are sent to the multi-threaded server via Object Streams for assistance. The Mariadb will then process the SQL query and will send the result back to the client via the multi-threaded server.

Use Case Diagram

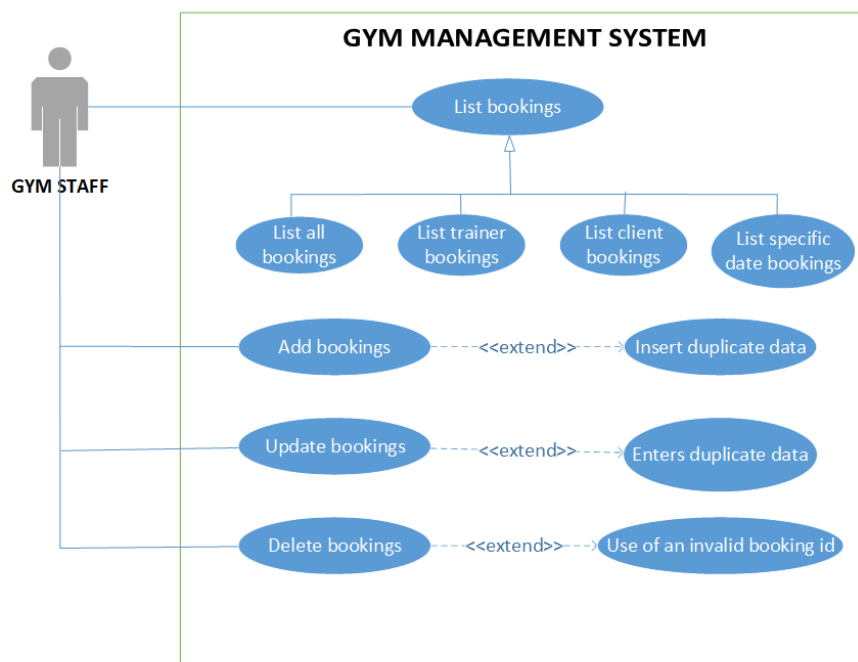


Fig2: Use case diagram

The use case above describes the functionality of the software and the main actor(s) who interact with it. “extends” has been used to indicate that there are exceptions in the use cases.

Functional Requirements

1. Implement a software to list bookings in the gym.
2. Implement a software to add new bookings.
3. Implement a software to update existing booking details.
4. Implement a software to delete a booking record.

Activity Diagrams

Activity diagrams are used to illustrate the steps required for the execution of each use cases from the initial mode to the end node.

List Activity Diagram

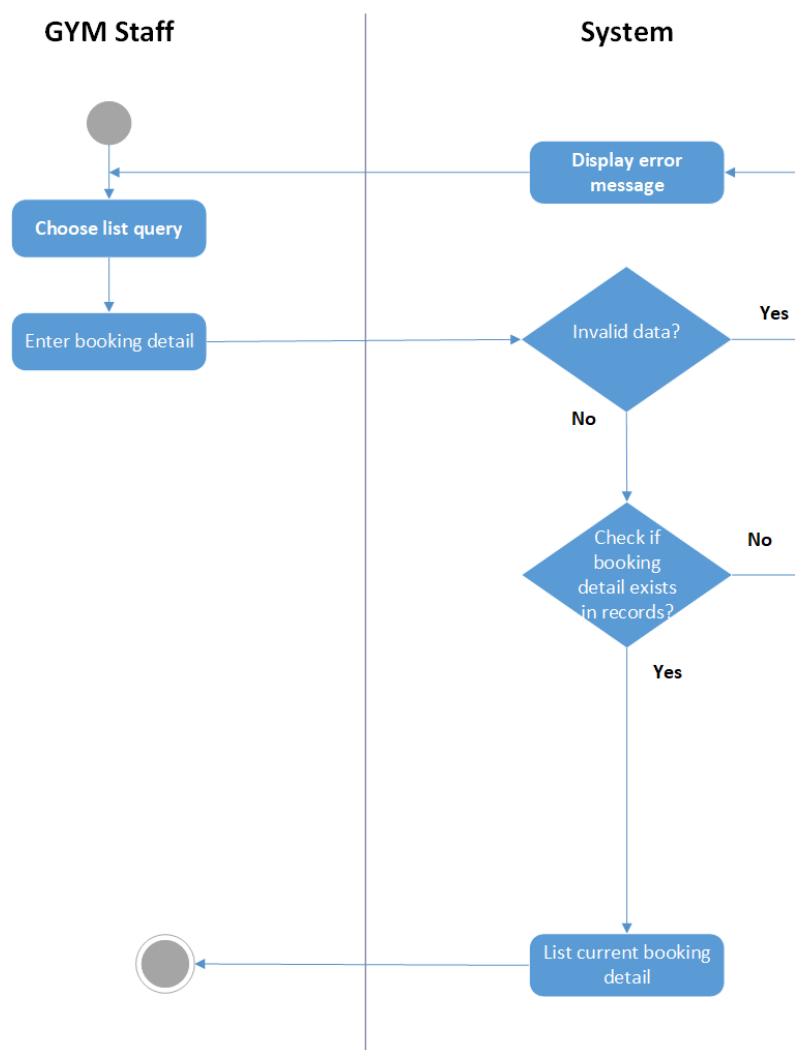


Fig3: LIST activity diagram

Add Activity Diagram

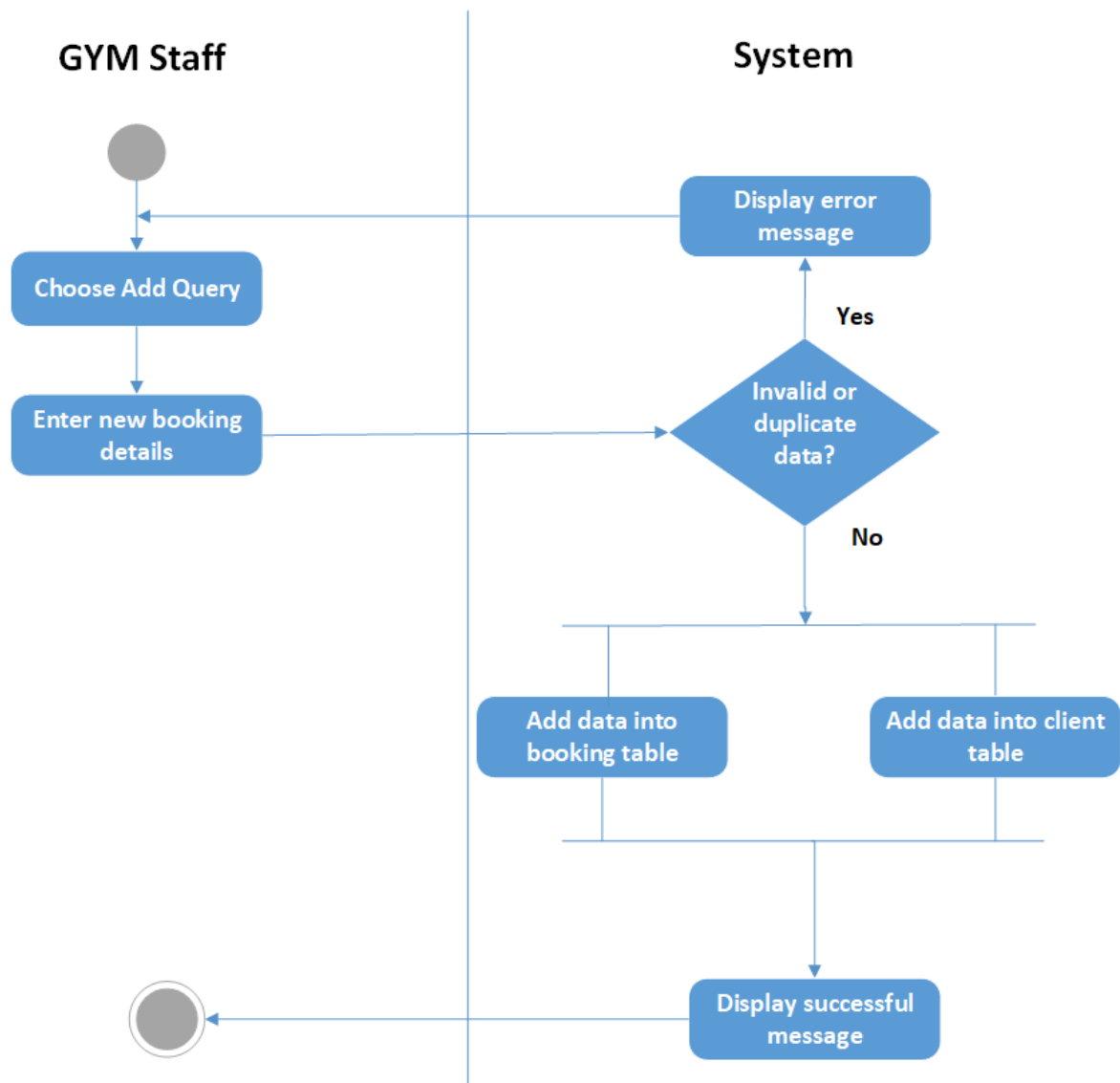


Fig4: ADD activity diagram

UPDATE Activity Diagram

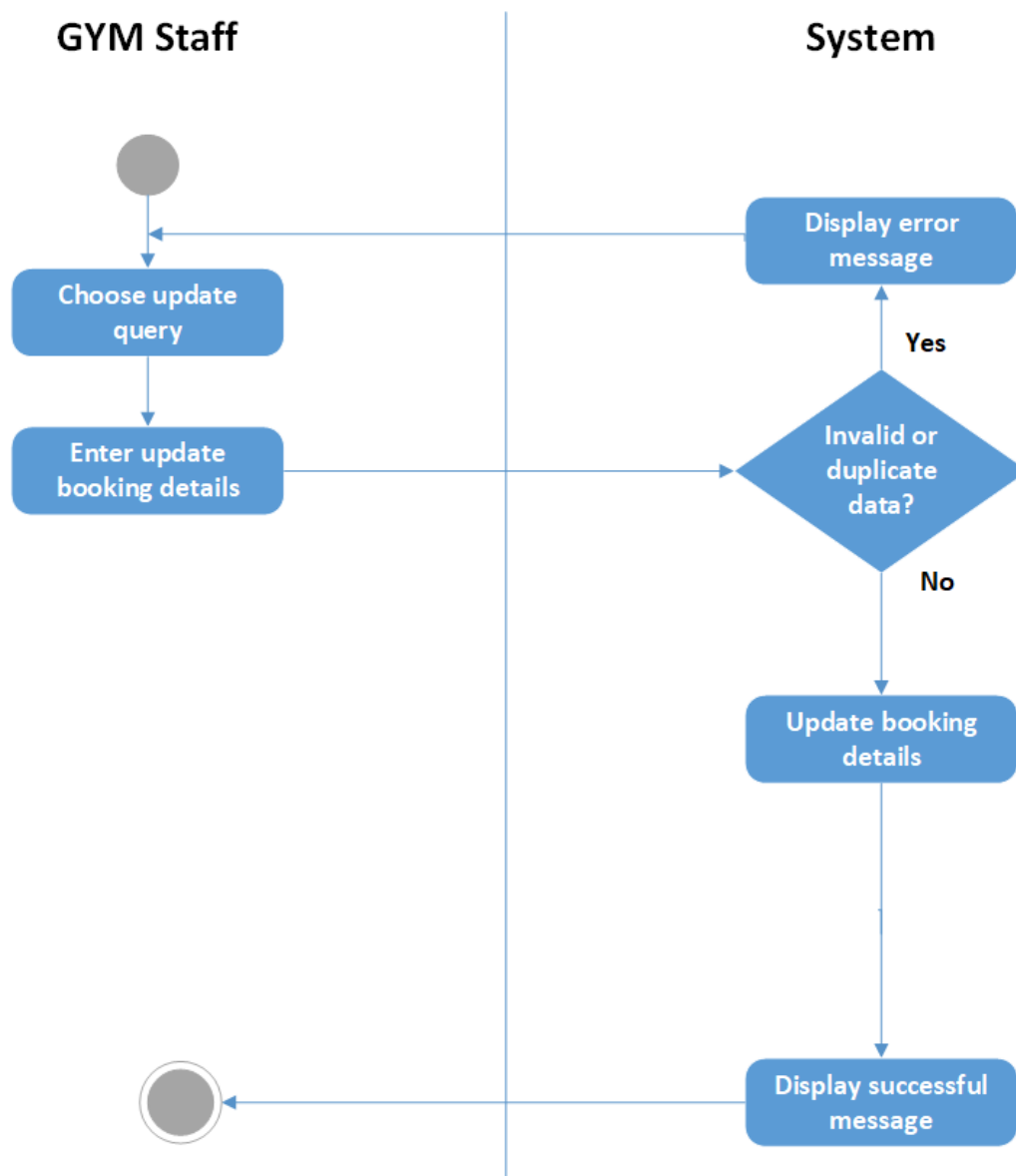


Fig5: UPDATE activity diagram

DELETE Activity Diagram

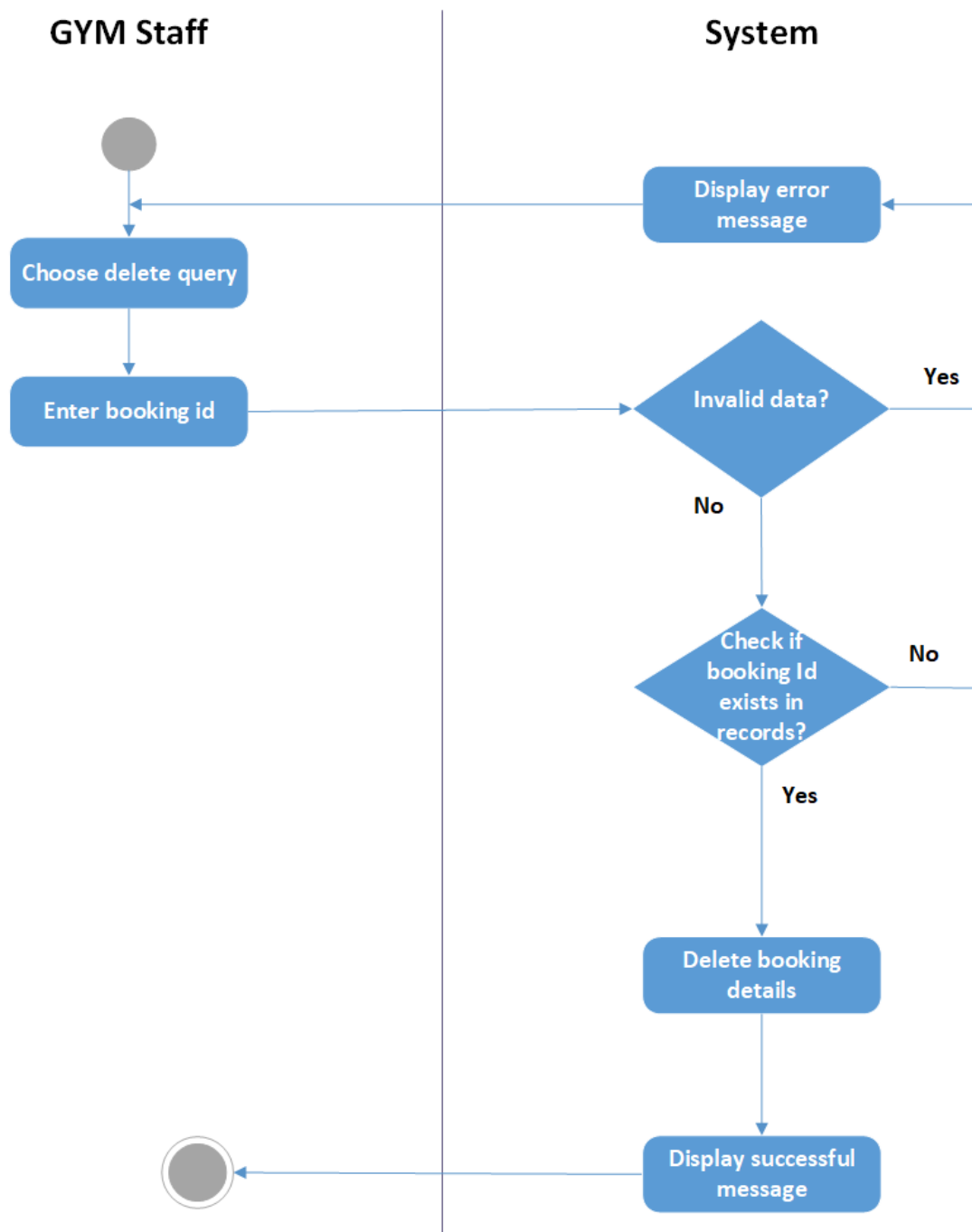


Fig6: DELETE activity diagram

Use Case Specification Diagrams

Use case : List bookings
ID: 1
Brief description: List booking records to gym staff(client)
Primary actors: Gym staff
Secondary actors: None
Preconditions:
1. Gym staff is already logged into the system
2. Connection has been established between server and database
3. A valid command has been used
Main flow:
1. Staff selects type of LIST command
2. Staff enters appropriate booking detail when prompted
3. System executes command
Postconditions:
1. Staff receives the booking records
Alternate flows:
None

Table10: LIST specification diagram

Use case : Add bookings
ID: 2
Brief description: Add new booking records
Primary actors: Gym staff
Secondary actors: None
Preconditions:
1. Gym staff is already logged into the system
2. Connection has been established between server and database
3. A valid command has been used
Main flow:
1. Staff selects ADD command
2. Staff enters valid and unique booking details
3. System executes command
Postconditions:
1. Staff receives a confirmation message
Alternate flows:
None

Table11: ADD specification diagram

Use case : Update bookings
ID: 3
Brief description: Update existing booking records
Primary actors: Gym staff
Secondary actors: None
Preconditions:
1. Gym staff is already logged into the system
2. Connection has been established between server and database
3. A valid command has been used
Main flow:
1. Staff selects UPDATE command
2. Staff enters valid and unique booking details
3. System executes command
Postconditions:
1. Staff receives a confirmation message
Alternate flows:
None

Table10: UPDATE specification diagram

Use case : Delete bookings
ID: 4
Brief description: Delete existing booking record
Primary actors: Gym staff
Secondary actors: None
Preconditions:
1. Gym staff is already logged into the system
2. Connection has been established between server and database
3. A valid command has been used
Main flow:
1. Staff selects DELETE command
2. Staff enters a valid booking Id
3. System executes command
Postconditions:
1. Staff receives a confirmation message
Alternate flows:
None

Table11: DELETE specification diagram

Class Diagram

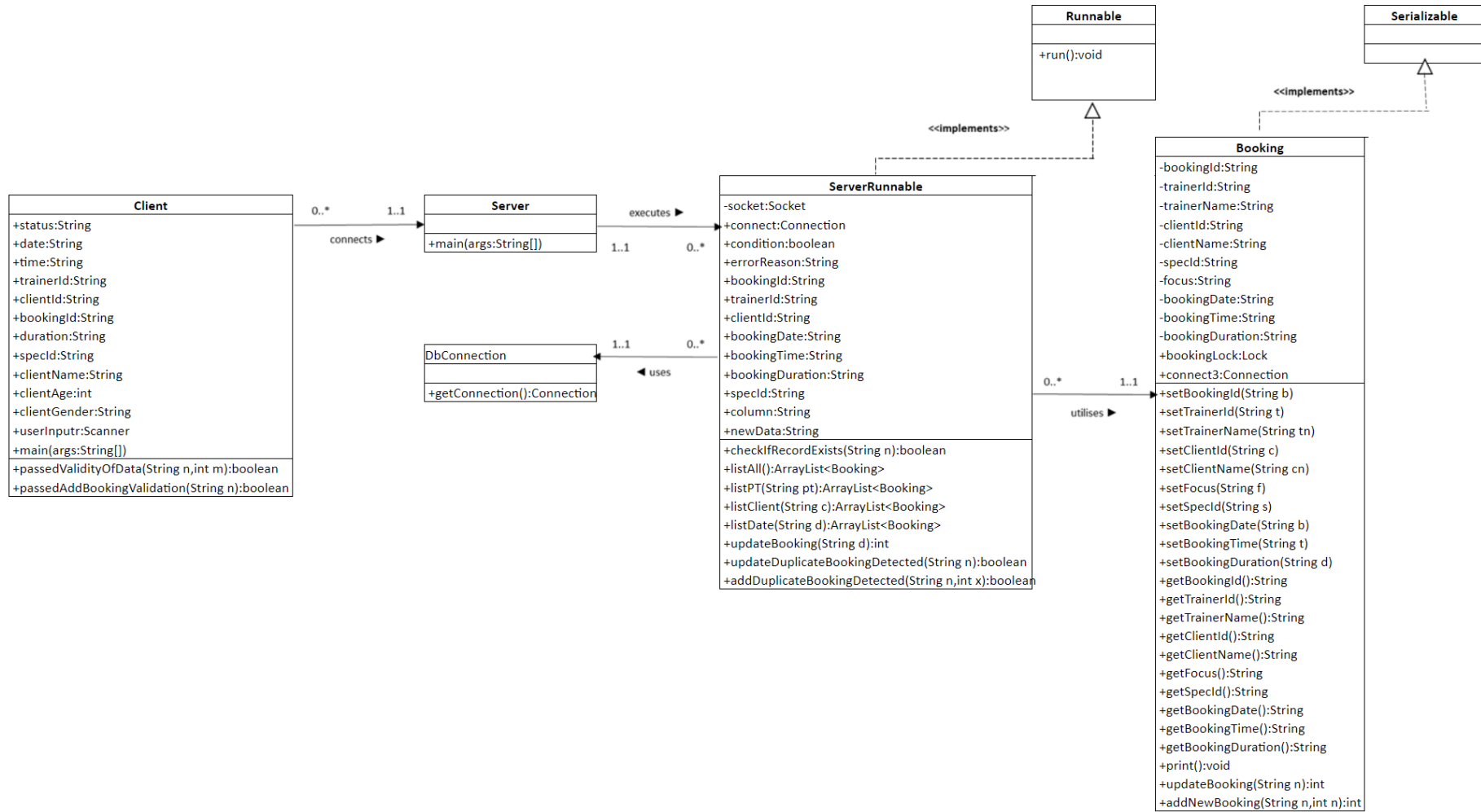


Fig7: Class diagram

Sequence Diagram

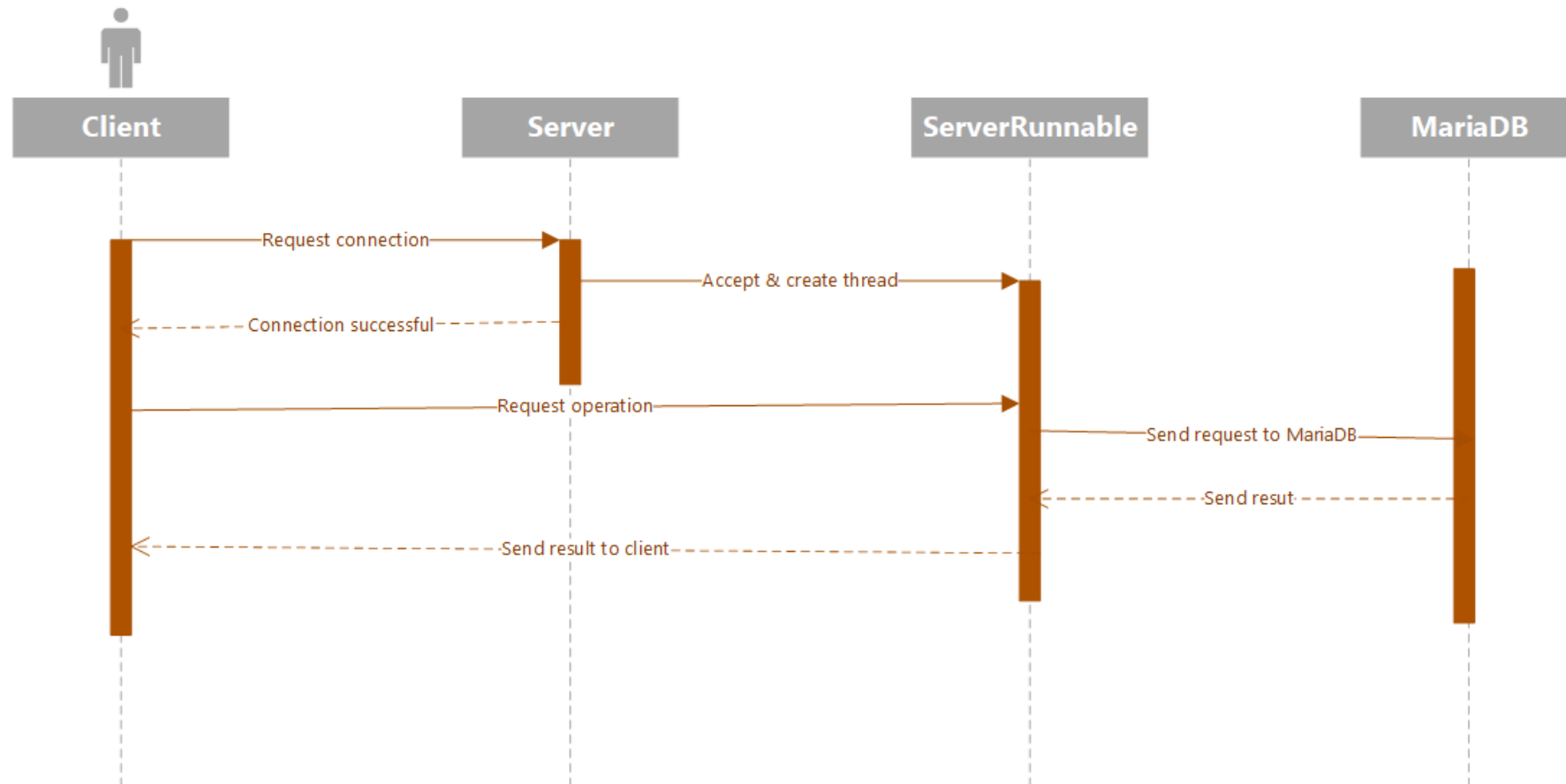


Fig8: Sequence Diagram

Graphical User Interface (GUI)

Initially, a GUI was proposed to make the system more user friendly using JavaFX as illustrated below:

GYM MANAGEMENT SYSTEM

LISTALL

LISTPT

LISTCLIENT

LISTDAY

ADD

UPDATE

DELETE

EXIT

Fig9: Main interface

GYM MANAGEMENT SYSTEM

BookingId: BookingDate:

TrainerId: BookingTime:

ClientId: BookingDuration:

FocusId:



 

Fig10: Add/Update interface

GYM MANAGEMENT SYSTEM

Add new booking Update booking Delete booking

BookingId	TrainerId	TrainerName	ClientId	ClientName	SpecId	Focus	BookingDate	BookingTime	BookingDuration

Fig11: List interface

Testing

Testing is an important phase of a software development cycle, with the aim of finding errors in a software. If proper testing are not carried out, the likelihood of the whole system to crash, have just rocketed. Testing carried out on the software are listed below:

Test No	Description	Command	User Input	Expected result	Actual result	Evidence
1	Invalid command entered	Mdx	-	Invalid command!	Invalid command!	E1
2	List a valid booking date	LISTDAY date	2020-10-01	List booking	Booking details listed	E2
3	Enter an invalid date	LISTDAY date	4999-784-1	Invalid date format!	Invalid date format!	E3
4	Enter an invalid bookingId	DELETE	Blgd	Invalid Booking Id entered!	Invalid Booking Id entered!	E4
5	Enter a non existing bookingId	DELETE	B046	DELETION FAILURE! No record exists with this BookingId	DELETION FAILURE! No record exists with this BookingId	E5
6	Enter an invalid clientId	LISTCLIENT	CfkW	Invalid data entered for ClientId	Invalid data entered for ClientId	E6
7	Enter a non existing clientId	LISTCLIENT	C009	ClientId does not exist in our records	ClientId does not exist in our records	E7
8	Enter an existing clientId	LISTCLIENT	C003	List booking	List booking	E8
9	Delete an existing booking detail	DELETE	B007	Deletion completed	Deletion completed	E9
10	Double booking for Trainer	UPDATE	B006	Current Personal trainer is already booked on that date and time	Current Personal trainer is already booked on that date and time	E10

Table12: Testing table

Evidences

```

|-----|
|  QUERIES  |
|-----|
| LISTALL   |
| LISTPT    |
| LISTCLIENT|
| LISTDAY   |
| ADD       |
| UPDATE    |
| DELETE    |
| QUIT      |
|-----|
Query : djfjf
Invalid command!
  
```

E1

```

Query : LISTDAY
Enter date yyyy-mm-dd : 4999-784-1

Invalid date format!
  
```

E3

```

Query : DELETE
Enter Booking id: Blgd

Invalid Booking id entered!
  
```

E4

```

|-----|
|  QUERIES  |
|-----|
| LISTALL   |
| LISTPT    |
| LISTCLIENT|
| LISTDAY   |
| ADD       |
| UPDATE    |
| DELETE    |
| QUIT      |
|-----|
Query : DELETE
Enter Booking id: B007

Deletion completed
  
```

E9

```

Query : LISTDAY
Enter date yyyy-mm-dd : 2020-10-01

|-----|
| BookingId | TrainerId | TrainerName | ClientId | ClientName | SpecId | Focus | BookingDate | BookingTime | BookingDuration |
|-----|
| B001      | PT01      | David       | C001     | Anil       | S001   | Weight Loss | 2020-10-01 | 09:00:00 | 01:00:00 |
|-----|
  
```

E2

```
Query : DELETE
Enter Booking id: B046

DELETION FAILURE! No record exists with this BookingId
```

E5

```
Query : LISTCLIENT
Enter ClientId: CfkW
Invalid data entered for ClientId
```

E6

```
Query : LISTCLIENT
Enter ClientId: C009
ClientId does not exist in our records
```

E7

```
Field: BookingDate

Enter date yyyy-mm-dd : 2020-10-01

Current Personal trainer is already booked on that date and time
```

E10

```
Query : LISTCLIENT
Enter ClientId: C003
```

BookingId	TrainerId	TrainerName	ClientId	ClientName	SpecId	Focus	BookingDate	BookingTime	BookingDuration
B003	PT03	Alexa	C003	Mary	S005	Box Jump	2020-10-03	11:00:00	03:00:00

E8

Conclusion

During the implementation phase, core specifications has been targeted and same were successfully implemented. A database design was created, proper testing were carried out and subsequently a GUI was implemented to facilitate human interactions with the system. Different changes were made to the original design since unwanted constraints popped up during the development stages.

Limitations/Critical thinking

- The current system does not posses a User Login Interface, thereby anyone can log into the system and easily retrieve client's personal data through unauthorised access.
- The GUI does not offer many functionalities to the user.
- Due to time constraints, all types of testing could not be attained.

Future approach

- Better planning of resources available will be a priority.
- Feedbacks from gym users will be sought and taken into consideration.
- Beta testers will be included to limit future abnormalities.

References

- *#1 Database Design & Modeling Tool Online | DB Designer*. [online] DB Designer. Available at: <https://www.dbdesigner.net/> [Accessed 31 Jan. 2020].
- Docs.oracle.com. (2020). *Lesson: JDBC Basics (The Java™ Tutorials > JDBC(TM) Database Access)*. [online] Available at: <https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html> [Accessed 31 Jan. 2020].
- Essential SQL. (2020). *Database Third Normal Form Explained in Simple English - Essential SQL*. [online] Available at: <https://www.essentialsql.com/get-ready-to-learn-sql-11-database-third-normal-form-explained-in-simple-english/> [Accessed 31 Jan. 2020].
- GeeksforGeeks. (2020). *Multithreading in Java - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/multithreading-in-java/> [Accessed 31 Jan. 2020].
- GeeksforGeeks. (2020). *Types of Software Testing - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/types-software-testing/> [Accessed 31 Jan. 2020].
- Guru99.com. (2020). *Database Design Tutorial: Learn Data Modeling*. [online] Available at: <https://www.guru99.com/database-design.html> [Accessed 31 Jan. 2020].
- Guru99.com. (2020). *Database Design Tutorial: Learn Data Modeling*. [online] Available at: <https://www.guru99.com/database-design.html> [Accessed 31 Jan. 2020].
- Guru99.com. (2020). *What is Normalization? 1NF, 2NF, 3NF & BCNF with Examples*. [online] Available at: <https://www.guru99.com/database-normalization.html> [Accessed 31 Jan. 2020].
- Software Testing Fundamentals. (2020). *White Box Testing - Software Testing Fundamentals*. [online] Available at: <http://softwaretestingfundamentals.com/white-box-testing/> [Accessed 31 Jan. 2020].

- Yang, A. and Bachar, Y. (1999). Using Java and the socket interface in teaching client/server programming. *ACM SIGCSE Bulletin*, 31(3), p.206.