REPORT

Date: 31/01/2020

CST2550
Software engineering management and development

Table of Contents

Abstract

A database was designed by first designing the ERD. After this, UML diagrams were designed to be able to develop the software and implement the database. The database was fully implemented and is perfectly running. The console part of the software is working smoothly with the server. Multi-threading has been applied to the server and well as re-entrant locks. All required functionalities are working with the console as well as validations. However, the GUI part is still in development. But part of it has been completed. In this report, you will find more about the database designed and the software (console) implementation as well as the GUI Wireframe.

Introduction

As MyGym is receiving more and more customers nowadays, to take bookings on paper and verify double booking or check if a certain trainer does a certain specialism is becoming more and more difficult. Even to update a booking, it has to be searched from line to line manually as bookingID are not sorted. There this database and software is being introduced to help solve the above-mentioned problems.

An ERD diagram is designed and separated into other smaller tables. This process called normalisation helps to reduce many different problems which could be faced using paper. After the normalisation, a use case diagram will be drawn. Based on this, use case specifications, activity diagram, class diagrams and sequence diagrams can be deduced. A GUI software will also be implemented to allow easier use of the database.

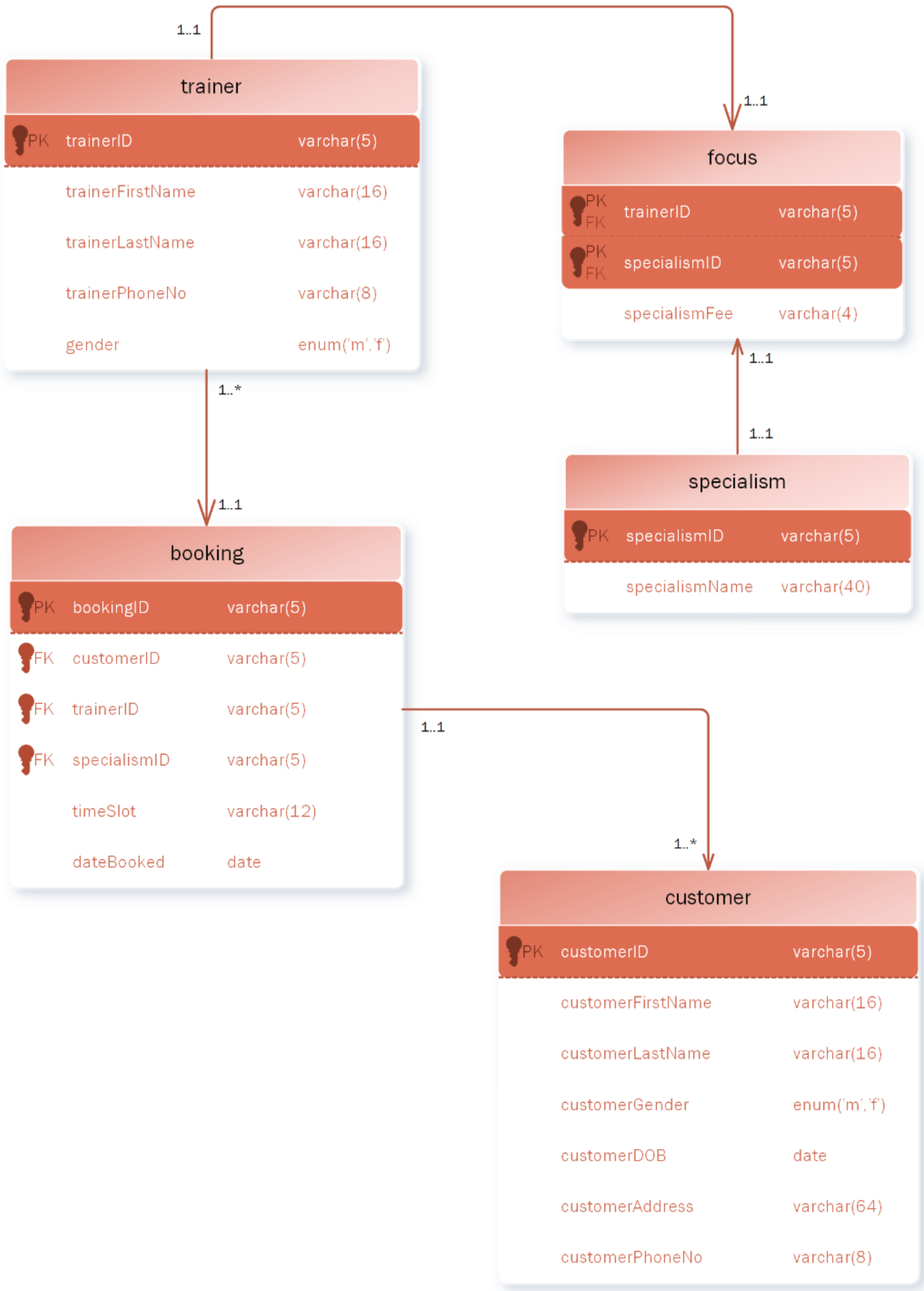
After building the program, maximum testing will be done to ensure reliability and stability of the program.

Database Design

Written Description

A database is a way of storing data securely. A database alleviates the need for a human to search through loads of files to find a specific record. It is a standardised and performant way to store and retrieve data. Normalisation is crucial to design and realise a database. Normalisation helps remove redundancy, save storage and improve access time. Databases are secure as only users with permission has access to them. In this project, a database is being designed and used to store client data, trainer data and booking data. Normalisation is applied to improve performance when accessing data, to prevent data leaks and theft, prevent duplication of data and bookings.

ER-Diagram


1..1
trainer
PK trainerID varchar(5)
trainerFirstName varchar(16)
trainerLastName varchar(16)
trainerPhoneNo varchar(8)
gender enum('m','f')
1..1
focus
PK FK trainerID varchar(5)
PK FK specialismID varchar(5)
specialismFee varchar(4)
1..*
1..1
1..1
1..1
specialism
PK specialismID varchar(5)
specialismName varchar(40)
booking
PK bookingID varchar(5)
FK customerID varchar(5)
FK trainerID varchar(5)
FK specialismID varchar(5)
timeSlot varchar(12)
dateBooked date
1..1
1..*
customer
PK customerID varchar(5)
customerFirstName varchar(16)
customerLastName varchar(16)
customerGender enum('m','f')
customerDOB date
customerAddress varchar(64)
customerPhoneNo varchar(8)

Normalisation

UNF - Booking Table

BookingID	DBGJ2	DG2VF	ADSFG	GEAGE	JNVDG	7DG7A
CustomerID	FLW35	FAB68	FDR48	MBR01	MOD12	MFM12
CustomerFirstName	Lacie	Alfie-Jay	Darla	Bernard	Olivier	Filip
CustomerLastName	Watkins	Burt	Ray	Rocha	Dixon	Mitchell
CustomerGender	F	F	F	M	M	M
CustomerDOB	2001-08-01	2002-10-30	1999-04-12	2002-06-10	1998-04-22	1999-05-11
CustomerAddress	Curepipe	Flic en Flac	Port-Louis	Vacoas	Phoenix	Mahebourg
CustomerPhoneNo	54840435	57818268	54185148	54188401	58401812	54840112
TrainerID	FPJ17	FSE89	MOM75	MOM75	MLS07	MVS67
TrainerFirstName	Priscilla	Silvia	Otto	Otto	Leon	Vasil
TrainerLastName	Johnsen	Esteves	MacBay	MacBay	Starosta	Shaw
TrainerGender	F	F	M	M	M	M
TrainerPhoneNo	54246317	57479689	54864575	54864575	53450507	55122767
SpecialismID	ATTRA	WEILO	MUSEF	MUSEF	ENSTE	CARFI
SpecialismName	Attractiveness	Weight loss	Muscle strength, endurance & flexibility	Muscle strength, endurance & flexibility	Energy, stamina & endurance	Cardiov-ascular fitness
SpecialismFee	300	400	200	200	400	700
DateBooked	2020-03-10	2020-02-24	2020-04-30	2020-06-12	2020-05-15	2020-10-01
TimeSlot	18:00-19:30	19:00-20:00	20:00-21:00	18:30-19:30	15:15-16:30	19:00-20:45

1NF

The UNF booking table has been split into 3 tables to flatten the tables.

Booking Table

BookingID	TrainerID	CustomerID	SpecialismID	TimeSlot	DateBooked
DBGJ2	FLW35	FPJ17	ATTRA	18:00-19:30	2020-03-10
DG2VF	FAB68	FSE89	WEILO	19:00-20:00	2020-02-24
ADSFG	FDR48	MOM75	MUSEF	20:00-21:00	2020-04-30
GEAGE	MBR01	MOM75	MUSEF	18:30-19:30	2020-06-12
JNVDG	MOD12	MLS07	ENSTE	15:15-16:30	2020-05-15
7DG7A	MFM12	MVS67	CARFI	19:00-20:45	2020-10-01

Customer Table

CustomerID	Customer FirstName	Customer LastName	Customer Gender	Customer DOB	Customer Address	Customer PhoneNo
FLW35	Lacie	Watkins	F	2001-08-01	Curepipe	54840435
FAB68	Alfie-Jay	Burt	F	2002-10-30	Flic en Flac	57818268
FLP55	Layla-Mae	Pitt	F	2000-11-25	Rose-Hill	57818355
MFM12	Filip	Mitchell	M	1999-05-11	Mahebourg	54840112
MBR01	Bernard	Rocha	M	2002-06-10	Vacoas	54188401
MOD12	Olivier	Dixon	M	1998-04-22	Phoenix	58401812
FDR48	Darla	Ray	F	1999-04-12	Port-Louis	54185148

Trainer Table

TrainerID	Trainer FirstName	Trainer LastName	Trainer PhoneNo	Trainer Gender	SpecialismID	Specialism Name	Specialism Fee
FSE89	Silvia	Esteves	57479689	F	MUSGA	Muscle gain	800
MLS07	Leon	Starosta	53450507	M	ENSTE	Energy, stamina & End.	400
MOM75	Otto	MacBay	54864575	M	MUSEF	Muscle strength, end. & flex.	200
FSE89	Silvia	Esteves	57479689	F	WEILO	Weight loss	400
FPB22	Pelagiya	Bergmann	54629922	F	WEILO	Weight loss	500
MVS67	Vasil	Shaw	55122767	M	COORD	Coordination	600
MLS07	Leon	Starosta	53450507	M	MUSGA	Muscle gain	200
FPJ17	Priscilla	Johnsen	54246317	F	ATTRA	Attractiveness	300
MVS67	Vasil	Shaw	55122767	M	CARFI	Cardiovascular fitness	700

2NF

The 1NF tables are converted to 2NF tables to remove partial dependencies.

Booking Table

BookingID	TrainerID	CustomerID	SpecialismID	TimeSlot	DateBooked
DBGJ2	FLW35	FPJ17	ATTRA	18:00-19:30	2020-03-10
DG2VF	FAB68	FSE89	WEILO	19:00-20:00	2020-02-24
ADSFG	FDR48	MOM75	MUSEF	20:00-21:00	2020-04-30
GEAGE	MBR01	MOM75	MUSEF	18:30-19:30	2020-06-12
JNVDG	MOD12	MLS07	ENSTE	15:15-16:30	2020-05-15
7DG7A	MFM12	MVS67	CARFI	19:00-20:45	2020-10-01

Customer Table

CustomerID	Customer FirstName	Customer LastName	Customer Gender	Customer DOB	Customer Address	Customer PhoneNo
FLW35	Lacie	Watkins	F	2001-08-01	Curepipe	54840435
FAB68	Alfie-Jay	Burt	F	2002-10-30	Flic en Flac	57818268
FLP55	Layla-Mae	Pitt	F	2000-11-25	Rose-Hill	57818355
MFM12	Filip	Mitchell	M	1999-05-11	Mahebourg	54840112
MBR01	Bernard	Rocha	M	2002-06-10	Vacoas	54188401
MOD12	Olivier	Dixon	M	1998-04-22	Phoenix	58401812
FDR48	Darla	Ray	F	1999-04-12	Port-Louis	54185148

Trainer Table

TrainerID	TrainerFirstName	TrainerLastName	TrainerPhoneNo	TrainerGender
FPB22	Pelagiya	Bergmann	54629922	F
FSE89	Silvia	Esteves	57479689	F
FPJ17	Priscilla	Johnsen	54246317	F
MLS07	Leon	Starosta	53450507	M
MOM75	Otto	MacBay	54864575	M
MVS67	Vasil	Shaw	55122767	M

Specialism Table

SpecialismID	SpecialismName	TrainerID	SpecialismFee
MUSGA	Muscle gain	FSE89	800
ENSTE	Energy, stamina & endurance	MLS07	400
MUSEF	Muscle strength, endurance & flexibility	MOM75	200
WEILO	Weight loss	FSE89	400
WEILO	Weight loss	FPB22	500
COORD	Coordination	MVS67	600
MUSGA	Muscle gain	MLS07	200
ATTRA	Attractiveness	FPJ17	300
CARFI	Cardiovascular fitness	MVS67	700

3NF

The 2NF tables are converted to 3NF tables to remove transitive dependencies.

Booking Table

BookingID	TrainerID	CustomerID	SpecialismID	TimeSlot	DateBooked
DBGJ2	FLW35	FPJ17	ATTRA	18:00-19:30	2020-03-10
DG2VF	FAB68	FSE89	WEILO	19:00-20:00	2020-02-24
ADSFG	FDR48	MOM75	MUSEF	20:00-21:00	2020-04-30
GEAGE	MBR01	MOM75	MUSEF	18:30-19:30	2020-06-12
JNVDG	MOD12	MLS07	ENSTE	15:15-16:30	2020-05-15
7DG7A	MFM12	MVS67	CARFI	19:00-20:45	2020-10-01

Customer Table

CustomerID	Customer FirstName	Customer LastName	Customer Gender	Customer DOB	Customer Address	Customer PhoneNo
FLW35	Lacie	Watkins	F	2001-08-01	Curepipe	54840435
FAB68	Alfie-Jay	Burt	F	2002-10-30	Flic en Flac	57818268
FLP55	Layla-Mae	Pitt	F	2000-11-25	Rose-Hill	57818355
MFM12	Filip	Mitchell	M	1999-05-11	Mahebourg	54840112
MBR01	Bernard	Rocha	M	2002-06-10	Vacoas	54188401
MOD12	Olivier	Dixon	M	1998-04-22	Phoenix	58401812
FDR48	Darla	Ray	F	1999-04-12	Port-Louis	54185148

Trainer Table

TrainerID	TrainerFirstName	TrainerLastName	TrainerPhoneNo	TrainerGender
FPB22	Pelagiya	Bergmann	54629922	F
FSE89	Silvia	Esteves	57479689	F
FPJ17	Priscilla	Johnsen	54246317	F
MLS07	Leon	Starosta	53450507	M
MOM75	Otto	MacBay	54864575	M
MVS67	Vasil	Shaw	55122767	M

Specialism Table

SpecialismID	SpecialismName
WEILO	Weight loss
MUSGA	Muscle gain
CARFI	Cardiovascular fitness
ENSTE	Energy, stamina & endurance
SPOPE	Sports performance
MUSEF	Muscle strength, endurance & flexibility
COORD	Coordination
IMMUN	Immunity
STRAN	Stress & anxiety
LIBID	Libido
ATTRA	Attractiveness

Focus Table

TrainerID	SpecialismID	FocusFee
FSE89	MUSGA	800
MLS07	ENSTE	400
MOM75	MUSEF	200
FSE89	WEILO	400
FPB22	WEILO	500
MVS67	COORD	600
MLS07	MUSGA	200
FPJ17	ATTRA	300
MVS67	CARFI	700

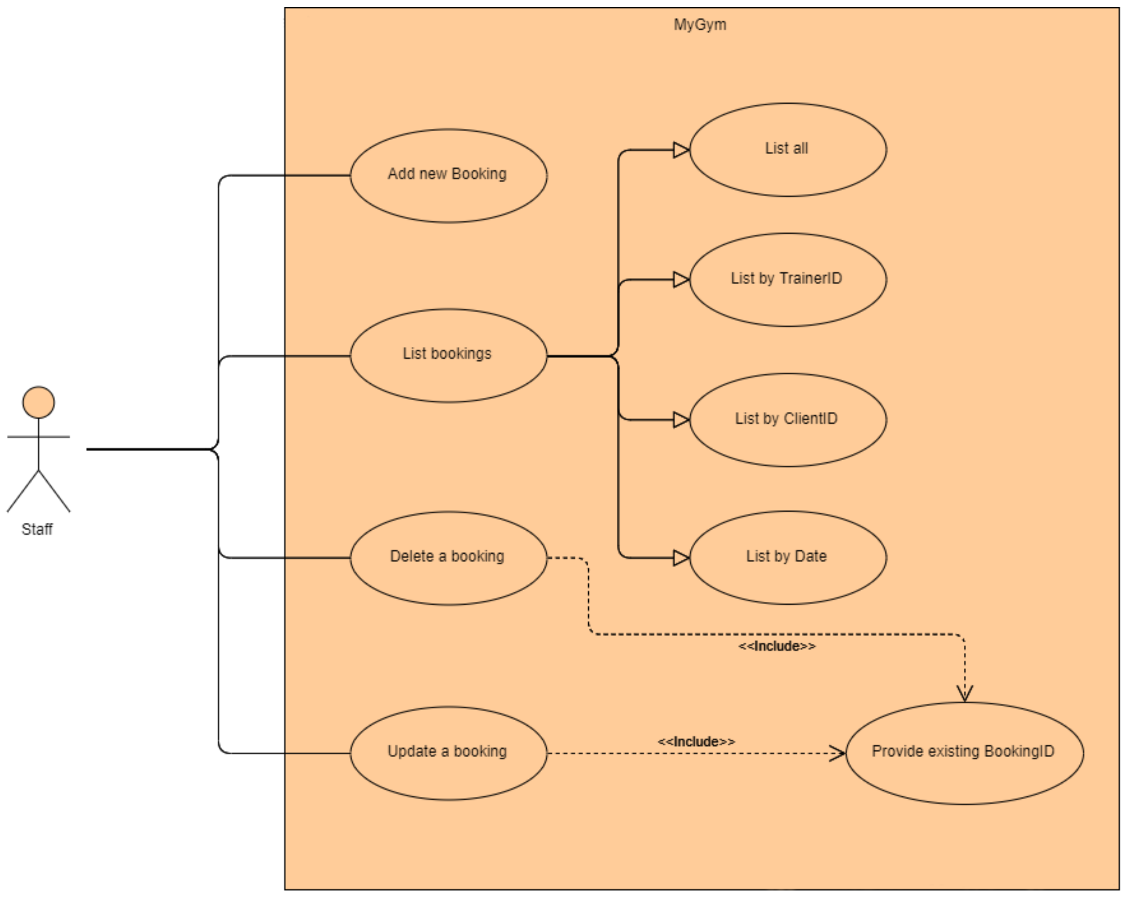
Software Design

Written Description

Software design is the most important phase of a software development cycle. Before starting to write codes, the structure of the program is important. Use case diagrams, activity diagrams, class diagrams and sequence diagrams are designed to be able to have a better view of the program. These UML diagrams help the developer to easily manage their codes. The program is divided into 2 parts; the server and the client. The server program is the only one which has access to the SQL Database. The server is run on a port (in this case, PORT:8888). The server is multi-threaded; several clients can use the server simultaneously. However, there are two functions in the server which are single-threaded. The add function and the update function has the Re-entrant Lock applied to them so that only one thread can access them at once. SQL queries are written inside the server program. A client is connected to the server providing the hostname and the port number. The user at the client end writes command lines/use the GUI version of the program to request and send data to the server which eventually read and write to the database accordingly. Data are sent and received using Object Input and Output Streams.

UML Diagrams

Use Case Diagram



Use Case Specification

Use Case: Add new booking

ID: 1
Brief description: Add a new booking with a personal trainer for a client
Primary actors: Staff
Secondary actors: Client
Preconditions: 1. Staff logged into the system
Main flow: 1. Client provides ClientID 2. Client chooses trainer and specialism 3. Client chooses date and time for booking
Postconditions: 1. Booking is added into the database
Alternate flows: None

Use Case: List bookings

ID: 2
Brief description: Displays a booking table to the staff
Primary actors: Staff
Secondary actors: None
Preconditions: 1. Staff decides how to list the booking
Main flow: 1. Staff chooses to display either all bookings/by trainerID/by ClientID/by date
Postconditions: 1. The table is not empty
Alternate flows: None

Use Case: Delete a booking

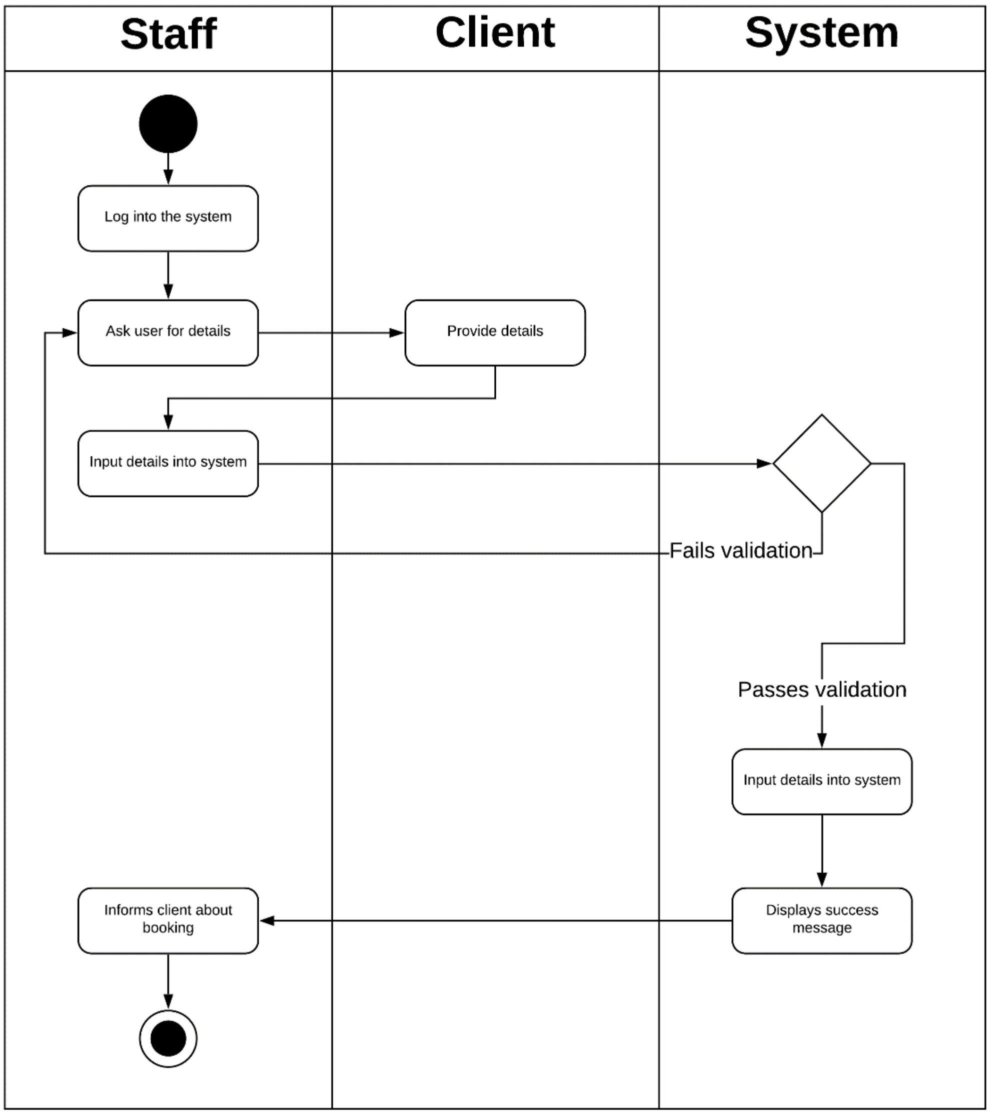
ID: 3
Brief description: Delete an existing booking from the database
Primary actors: Staff
Secondary actors: None
Preconditions: 1. Staff logged into the system 2. Staff confirms that the booking should be deleted
Main flow: 1. Staff inserts bookingID to be deleted 2. Staff confirms deletion of selected booking
Postconditions: 1. Selected booking exists
Alternate flows: None

Use Case: Update a booking

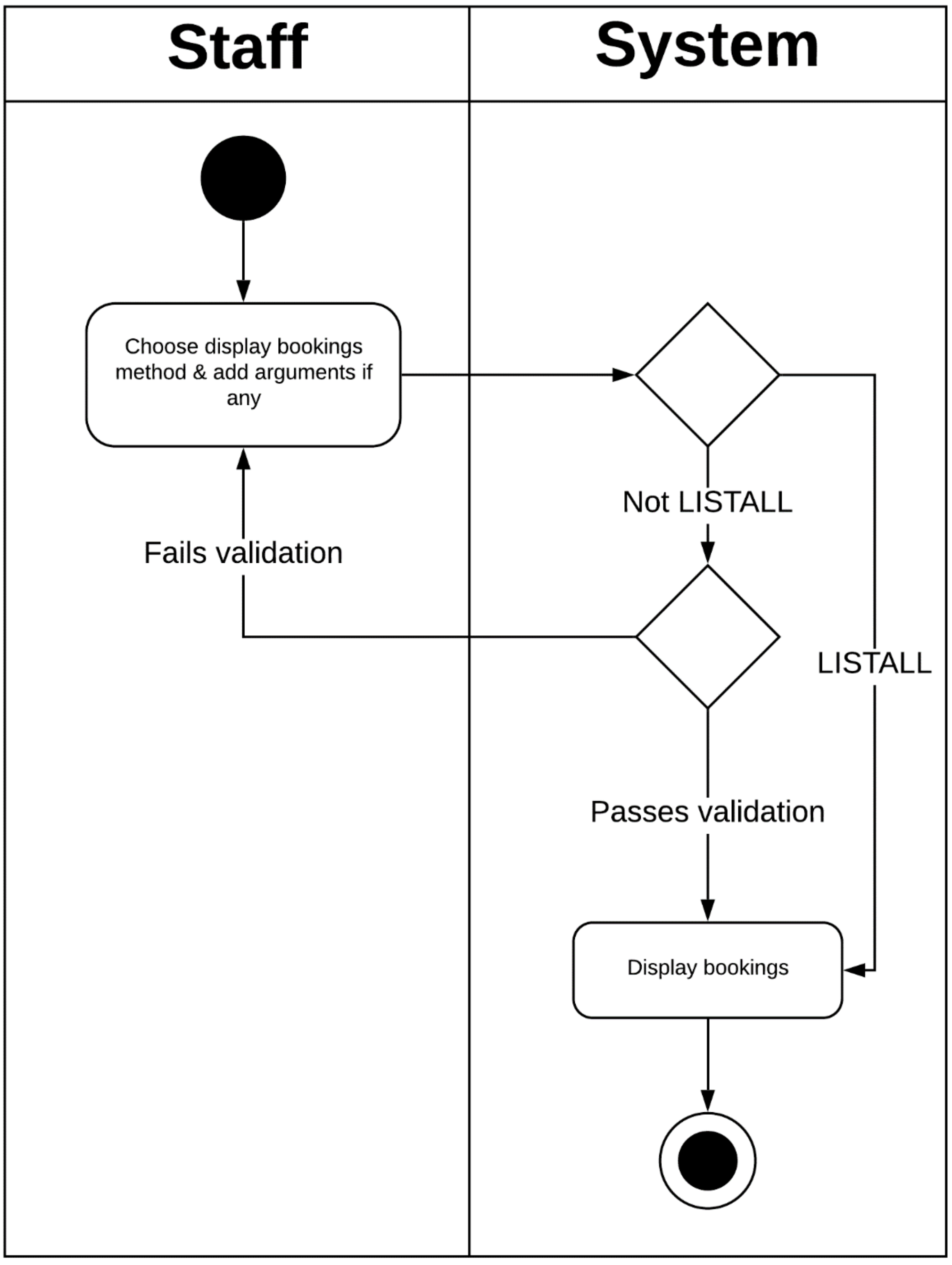
ID: 4
Brief description: Client gets to update his/her existing booking
Primary actors: Staff
Secondary actors: Client
Preconditions: 1. Staff logged into the system
Main flow: 1. Client provides bookingID 2. Client chooses trainer and specialism 3. Client chooses date and time for booking
Postconditions: 1. Booking already exists in the database
Alternate flows: None

Activity Diagram

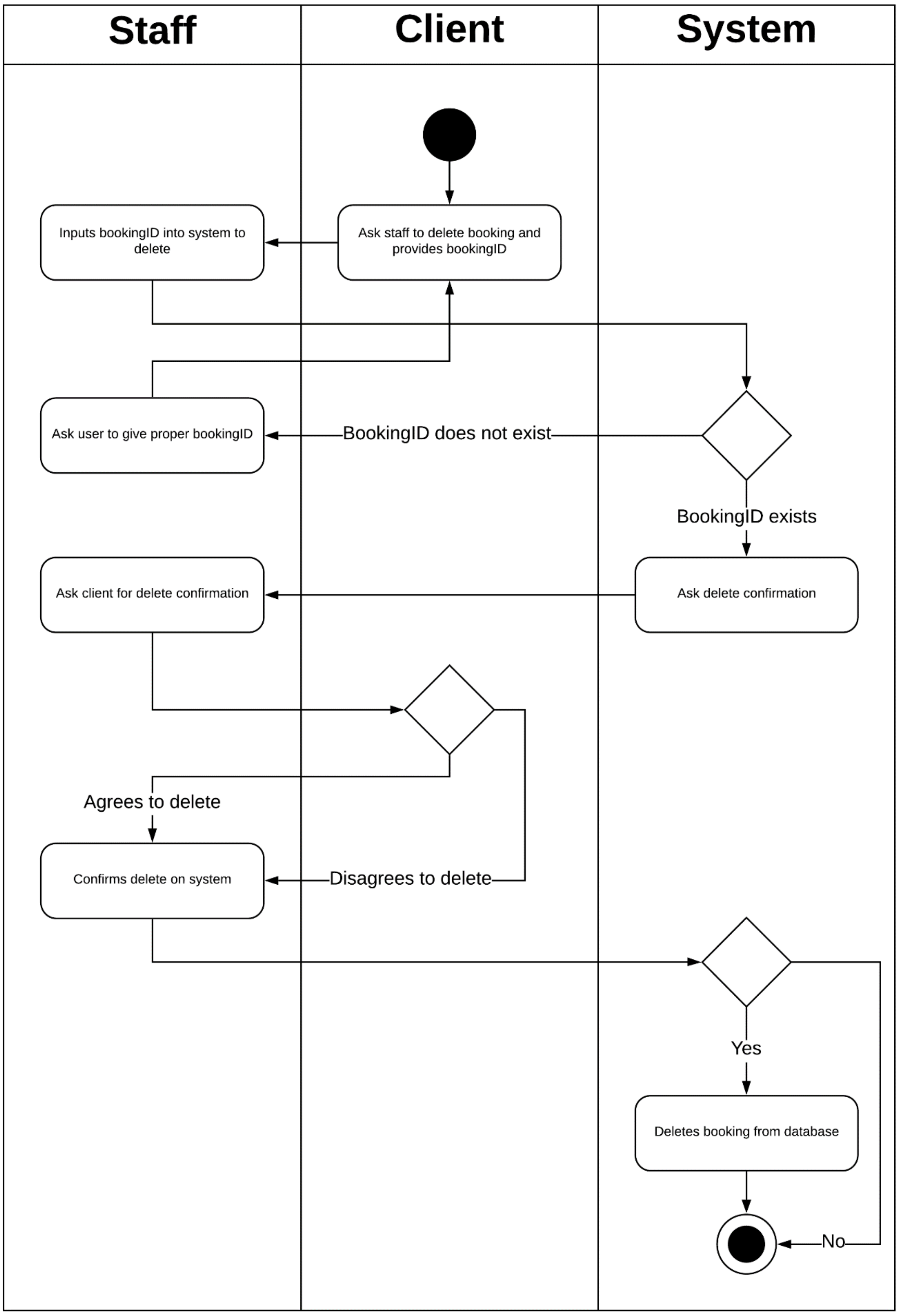
Use Case 1: Add new booking



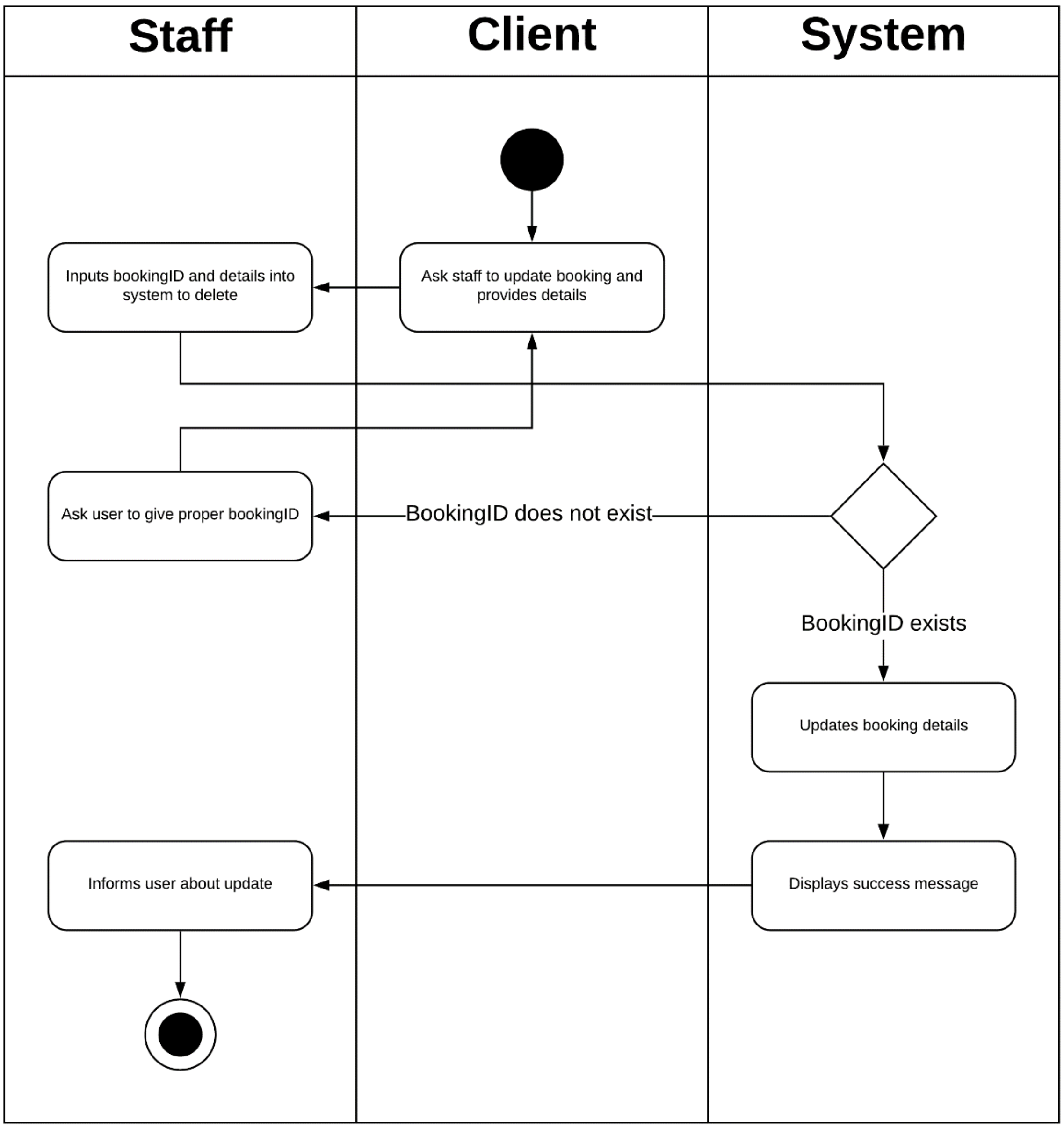
Use Case 2: List bookings



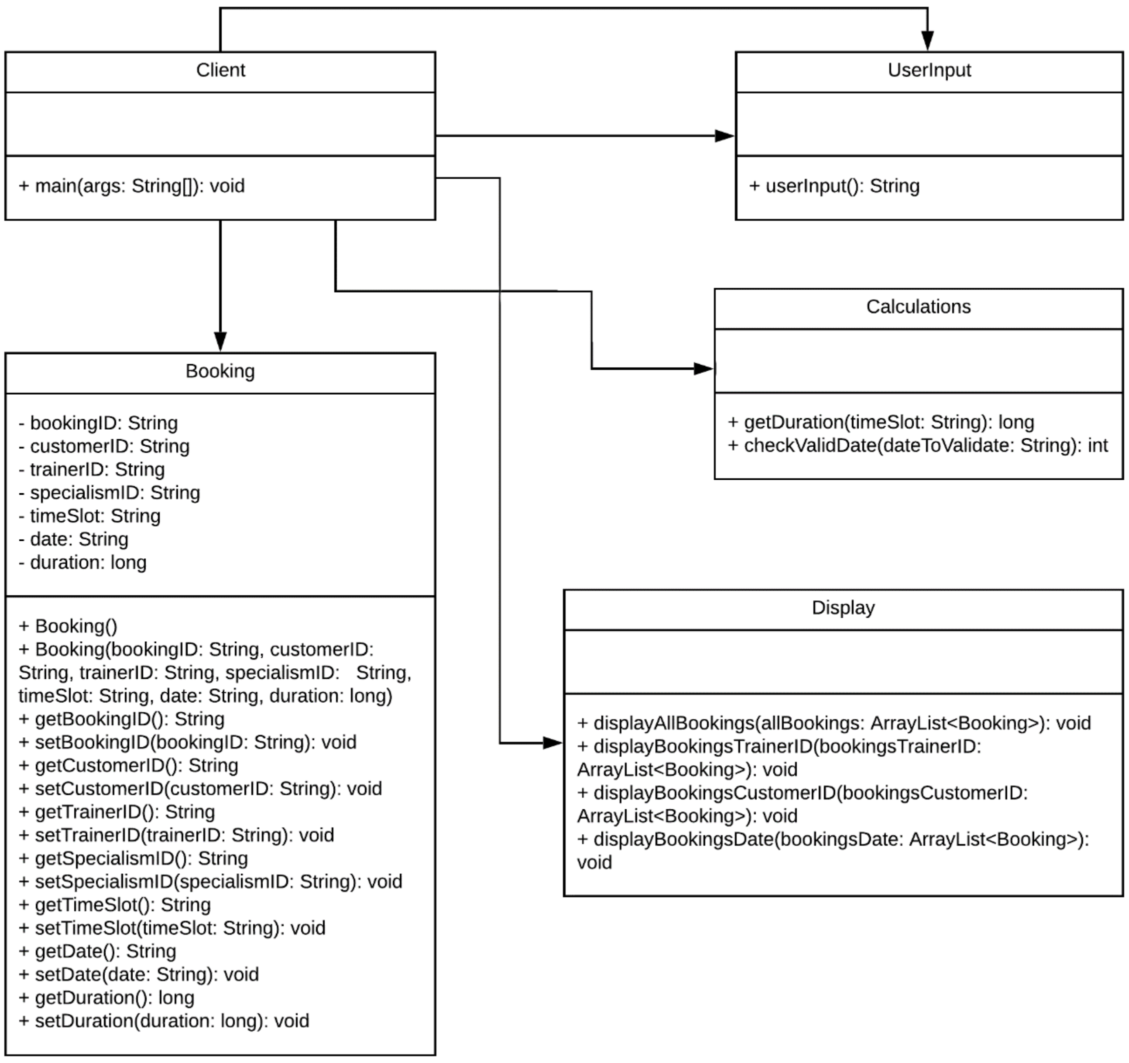
Use Case 3: Delete a booking


Staff
Client
System
Inputs bookingID into system to delete
Ask staff to delete booking and provides bookingID
Ask user to give proper bookingID
BookingID does not exist
BookingID exists
Ask client for delete confirmation
Ask delete confirmation
Agrees to delete
Confirms delete on system
Disagrees to delete
Yes
Deletes booking from database
No

Use Case 4: Update a booking

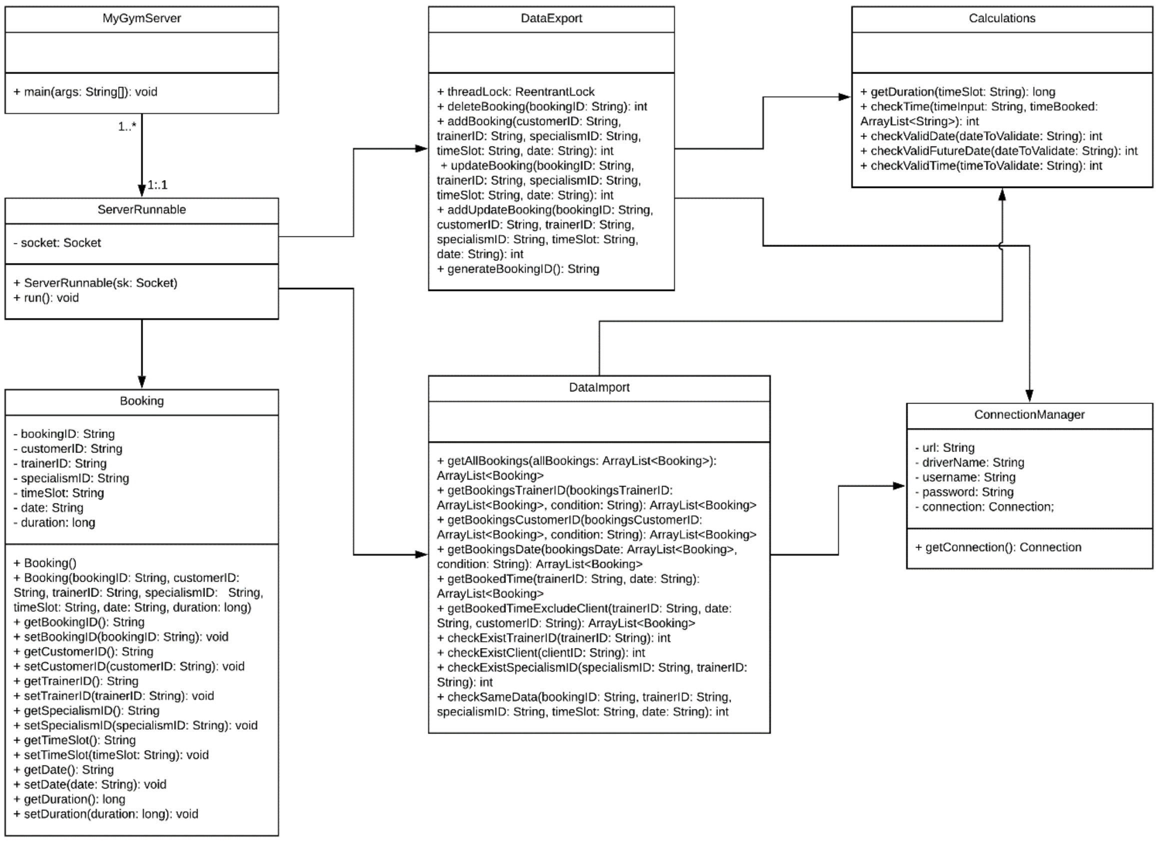

Staff
Client
System
Inputs bookingID and details into system to delete
Ask staff to update booking and provides details
Ask user to give proper bookingID
BookingID does not exist
BookingID exists
Updates booking details
Informs user about update
Displays success message

Class Diagram – Client



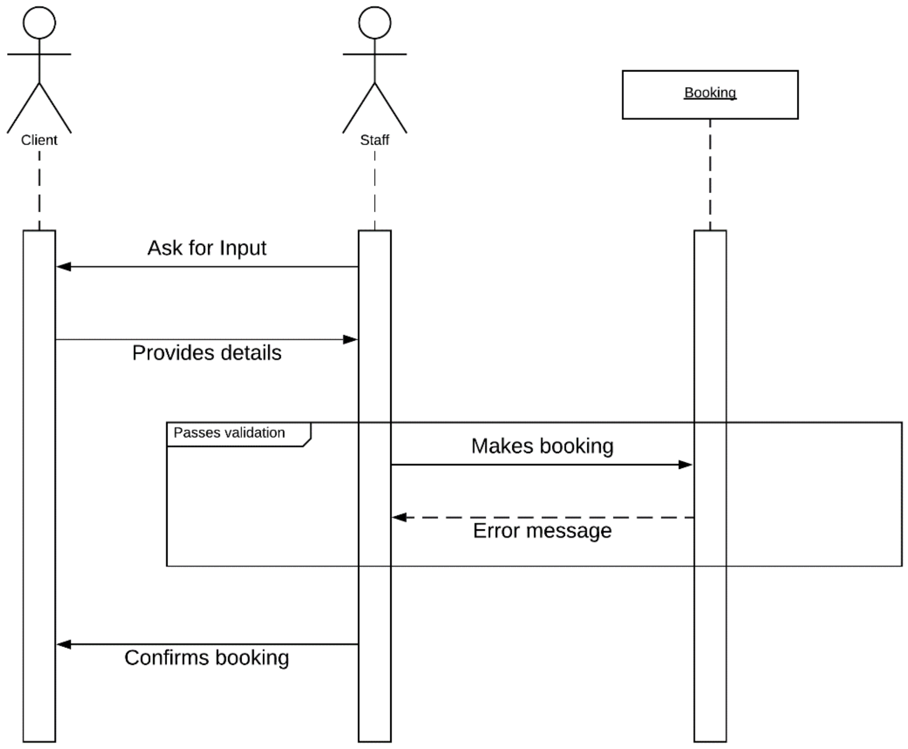
Client
+ main(args: String[]): void
UserInput
+ userInput(): String
Calculations
+ getDuration(timeSlot: String): long
+ checkValidDate(dateToValidate: String): int
Booking
- bookingID: String
- customerID: String
- trainerID: String
- specialismID: String
- timeSlot: String
- date: String
- duration: long
+ Booking()
+ Booking(bookingID: String, customerID: String, trainerID: String, specialismID: String, timeSlot: String, date: String, duration: long)
+ getBookingID(): String
+ setBookingID(bookingID: String): void
+ getCustomerID(): String
+ setCustomerID(customerID: String): void
+ getTrainerID(): String
+ setTrainerID(trainerID: String): void
+ getSpecialismID(): String
+ setSpecialismID(specialismID: String): void
+ getTimeSlot(): String
+ setTimeSlot(timeSlot: String): void
+ getDate(): String
+ setDate(date: String): void
+ getDuration(): long
+ setDuration(duration: long): void
Display
+ displayAllBookings(allBookings: ArrayList<Booking>): void
+ displayBookingsTrainerID(bookingsTrainerID: ArrayList<Booking>): void
+ displayBookingsCustomerID(bookingsCustomerID: ArrayList<Booking>): void
+ displayBookingsDate(bookingsDate: ArrayList<Booking>): void

Class Diagram – Server


MyGymServer
+ main(args: String[]): void
1..*
1:.1
ServerRunnable
- socket: Socket
+ ServerRunnable(sk: Socket)
+ run(): void
Booking
- bookingID: String
- customerID: String
- trainerID: String
- specialismID: String
- timeSlot: String
- date: String
- duration: long
+ Booking()
+ Booking(bookingID: String, customerID: String, trainerID: String, specialismID: String, timeSlot: String, date: String, duration: long)
+ getBookingID(): String
+ setBookingID(bookingID: String): void
+ getCustomerID(): String
+ setCustomerID(customerID: String): void
+ getTrainerID(): String
+ setTrainerID(trainerID: String): void
+ getSpecialismID(): String
+ setSpecialismID(specialismID: String): void
+ getTimeSlot(): String
+ setTimeSlot(timeSlot: String): void
+ getDate(): String
+ setDate(date: String): void
+ getDuration(): long
+ setDuration(duration: long): void
DataExport
+ threadLock: ReentrantLock
+ deleteBooking(bookingID: String): int
+ addBooking(customerID: String, trainerID: String, specialismID: String, timeSlot: String, date: String): int
+ updateBooking(bookingID: String, trainerID: String, specialismID: String, timeSlot: String, date: String): int
+ addUpdateBooking(bookingID: String, customerID: String, trainerID: String, specialismID: String, timeSlot: String, date: String): int
+ generateBookingID(): String
DataImport
+ getAllBookings(allBookings: ArrayList<Booking>): ArrayList<Booking>
+ getBookingsTrainerID(bookingsTrainerID: ArrayList<Booking>, condition: String): ArrayList<Booking>
+ getBookingsCustomerID(bookingsCustomerID: ArrayList<Booking>, condition: String): ArrayList<Booking>
+ getBookingsDate(bookingsDate: ArrayList<Booking>, condition: String): ArrayList<Booking>
+ getBookedTime(trainerID: String, date: String): ArrayList<Booking>
+ getBookedTimeExcludeClient(trainerID: String, date: String, customerID: String): ArrayList<Booking>
+ checkExistTrainerID(trainerID: String): int
+ checkExistClient(clientID: String): int
+ checkExistSpecialismID(specialismID: String, trainerID: String): int
+ checkSameData(bookingID: String, trainerID: String, specialismID: String, timeSlot: String, date: String): int
Calculations
+ getDuration(timeSlot: String): long
+ checkTime(timeInput: String, timeBooked: ArrayList<String>): int
+ checkValidDate(dateToValidate: String): int
+ checkValidFutureDate(dateToValidate: String): int
+ checkValidTime(timeToValidate: String): int
ConnectionManager
- url: String
- driverName: String
- username: String
- password: String
- connection: Connection;
+ getConnection(): Connection

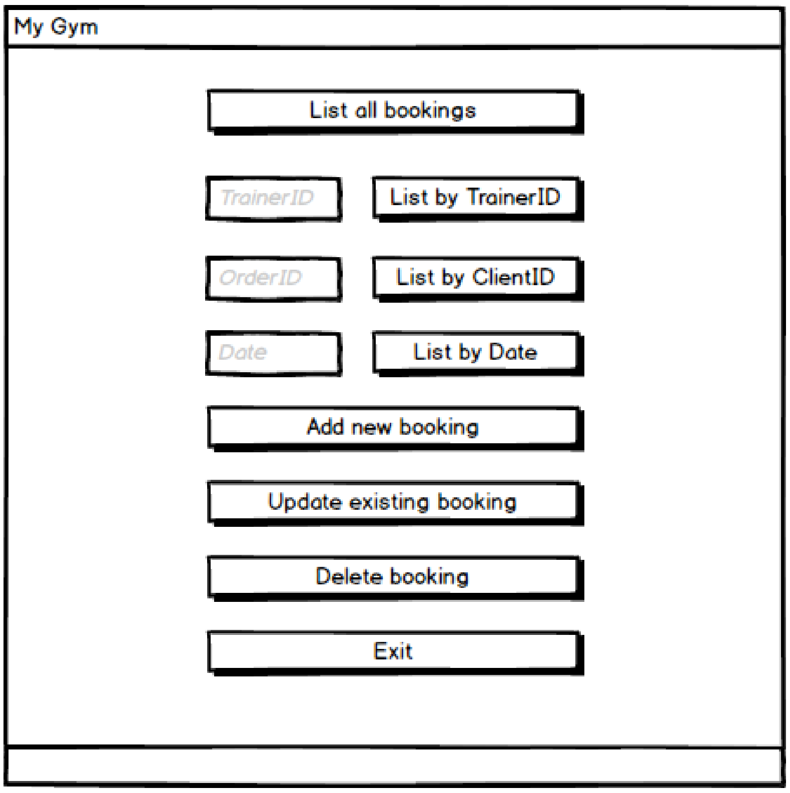
Sequence Diagram

Use Case 1: Add new booking

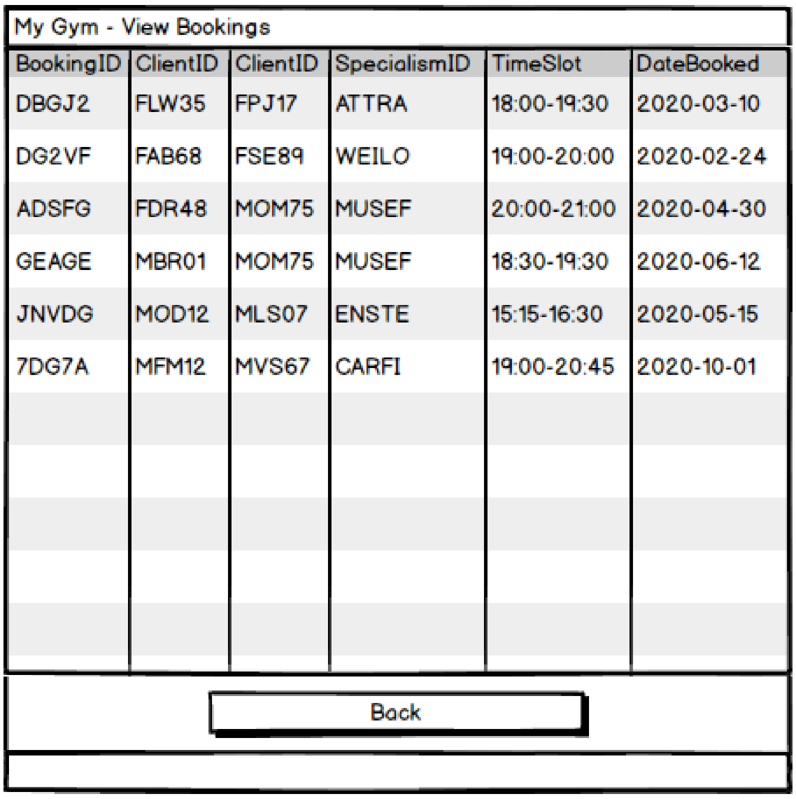


GUI Wireframe

Main menu



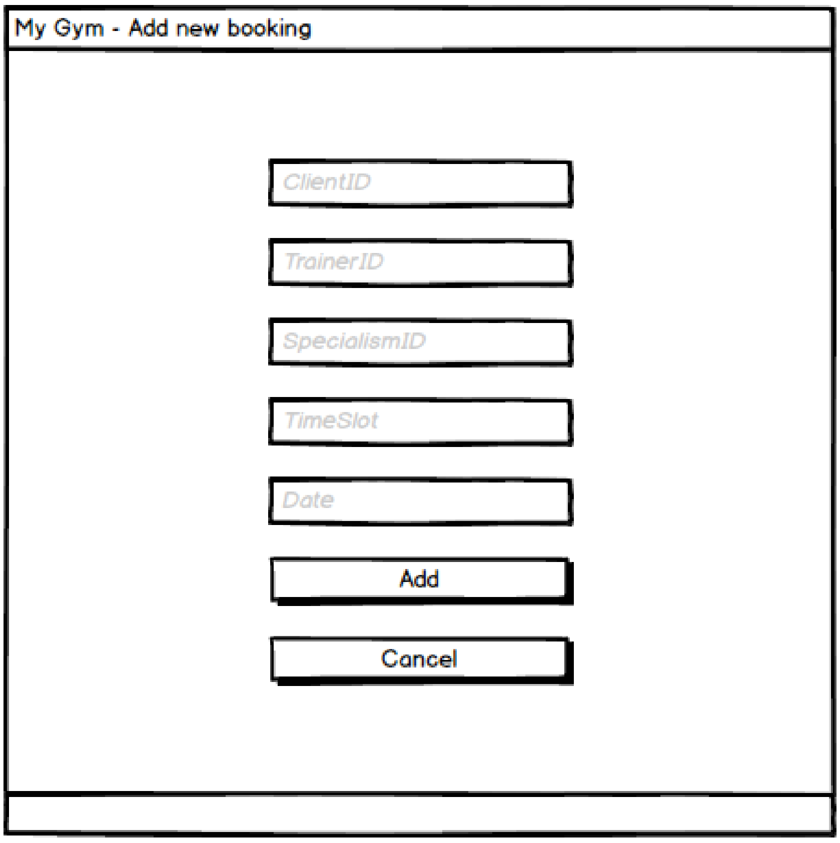
List table



My Gym - View Bookings

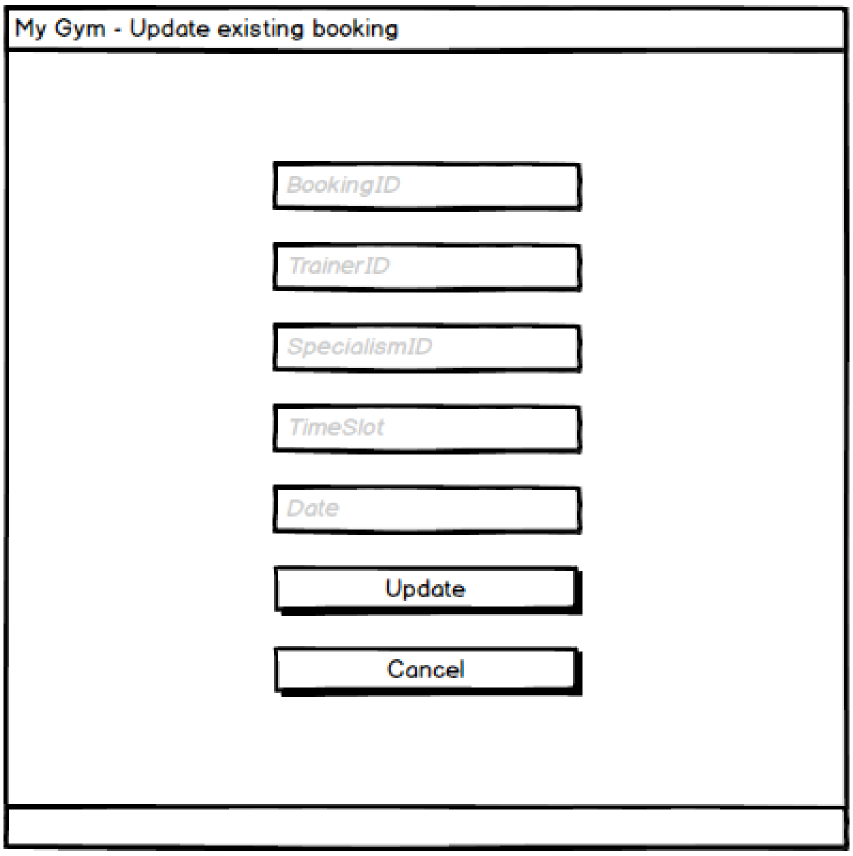
BookingID	ClientID	ClientID	SpecialismID	TimeSlot	DateBooked
DBGJ2	FLW35	FPJ17	ATTRA	18:00-19:30	2020-03-10
DG2VF	FAB68	FSE89	WEILO	19:00-20:00	2020-02-24
ADSFG	FDR48	MOM75	MUSEF	20:00-21:00	2020-04-30
GEAGE	MBR01	MOM75	MUSEF	18:30-19:30	2020-06-12
JNVDG	MOD12	MLS07	ENSTE	15:15-16:30	2020-05-15
7DG7A	MFM12	MVS67	CARFI	19:00-20:45	2020-10-01

Add new booking

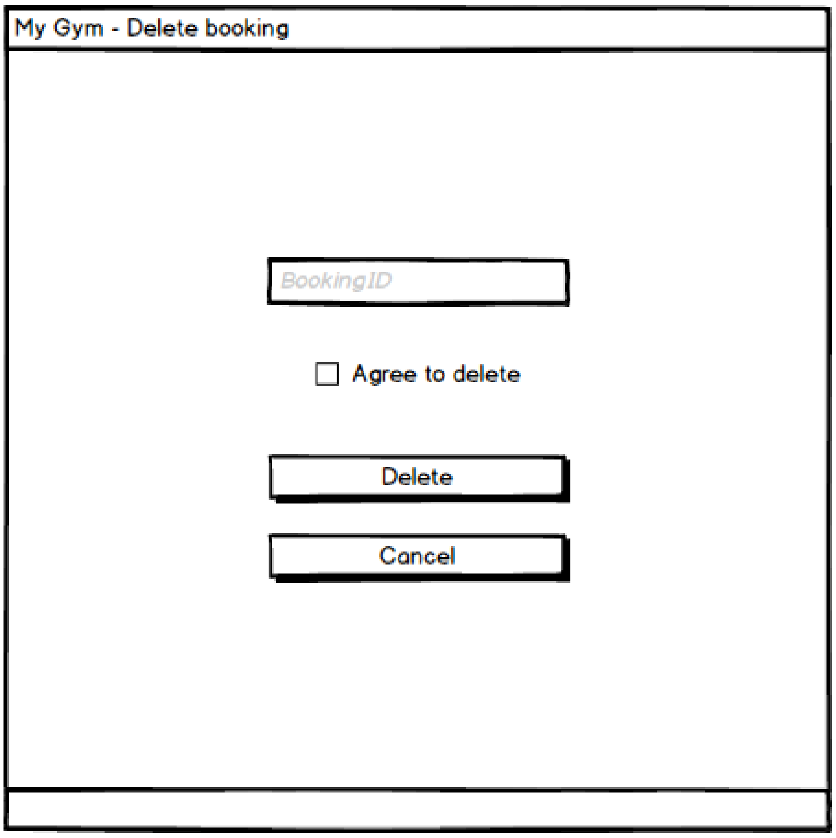


My Gym - Add new booking

Update existing booking



Delete booking



Testing

Testing was carried out manually by trying different possible inputs and verifying the outputs.

Some results and tests carried out are in the table below.

To ensure that the program is reliable, maximum number of tests was carried out.

Test	Expectation	Result	Pass or Fail ?
Type invalid command	Invalid command	Invalid command	Pass
Type "quit" or "exit"	Socket close and program exits	Socket close and program exits	Pass
LISTALL with arguments	Too many arguments	Too many arguments	Pass
LISTPT with zero argument	Not enough arguments	Not enough arguments	Pass
LISTPT with more than one argument	Too many arguments	Too many arguments	Pass
LISTPT with one argument of incorrect length	TrainerID contains 5 characters only	TrainerID contains 5 characters only	Pass
LISTPT with incorrect characters	TrainerID is alphanumeric	TrainerID is alphanumeric	Pass
LISTPT with trainerID not available in table	TrainerID does not exist	TrainerID does not exist	Pass
LISTPT with trainerID who does not have booking	No booking available	No booking available	Pass
LISTPT with correct syntax and available booking	Displays booking table	Displays booking table	Pass
LISTCLIENT with zero argument	Not enough arguments	Not enough arguments	Pass
LISTCLIENT with more than one argument	Too many arguments	Too many arguments	Pass

LISTCLIENT with one argument of incorrect length	ClientID contains 5 characters only	ClientID contains 5 characters only	Pass
LISTCLIENT with incorrect characters	ClientID is alphanumeric	ClientID is alphanumeric	Pass
LISTCLIENT with clientID not available in table	ClientID does not exist	ClientID does not exist	Pass
LISTCLIENT with clientID who does not have booking	No booking available	No booking available	Pass
LISTCLIENT with correct syntax and available booking	Displays booking table	Displays booking table	Pass
LISTDAY with zero argument	Not enough arguments	Not enough arguments	Pass
LISTDAY with more than one argument	Too many arguments	Too many arguments	Pass
LISTDAY with incorrect characters	Incorrect date format	Incorrect date format	Pass
LISTDAY with date who does not have booking	No booking available	No booking available	Pass
LISTDAY with correct syntax and available booking	Displays booking table	Displays booking table	Pass
DELETE with no argument	Not enough argument	Not enough argument	Pass
DELETE with more than one argument	Too many argument	Too many argument	Pass
DELETE with bookingID of invalid length	bookingId contains 5 characters only	bookingId contains 5 characters only	Pass
DELETE with bookingID which does not exist	bookingID does not exist	bookingID does not exist	Pass

DELETE with correct syntax	Delete successful	Delete successful	Pass
ADD with less than 5 arguments	Not enough arguments	Not enough arguments	Pass
ADD with more than 5 arguments	Too many arguments	Too many arguments	Pass
ADD with customerID having more than 5 characters	customerID should have 5 characters	customerID should have 5 characters	Pass
ADD with customerID having invalid characters	customerID is alphanumeric	customerID is alphanumeric	Pass
ADD with customerID which does not exist	customerID does not exist	customerID does not exist	Pass
ADD with trainerID having more than 5 characters	trainerID should be 5 characters	trainerID should be 5 characters	Pass
ADD with trainerID having invalid characters	trainerID is alphanumeric	trainerID is alphanumeric	Pass
ADD with trainerID which does not exist	trainerID does not exist	trainerID does not exist	Pass
ADD with specialismID having more than 5 characters	SpecialismID should be 5 characters	SpecialismID should be 5 characters	Pass
ADD with specialismID having invalid characters	SpecialismID is alphanumeric	SpecialismID is alphanumeric	Pass
ADD with specialismID which does not exist for mentioned trainer	Trainer does not do this specialism	Trainer does not do this specialism	Pass

ADD with wrong time format	Wrong time format	Wrong time format	Pass
ADD with wrong start time	Wrong start time	Wrong start time	Pass
ADD with wrong end time	Wrong end time	Wrong end time	Pass
ADD with time before 06:00	Gym opens at 06:00	Gym opens at 06:00	Pass
ADD with time after 23:00	Gym closes at 23:00	Gym closes at 23:00	Pass
ADD with start time after end time	Start time cannot be after end time	Start time cannot be after end time	Pass
ADD with start time equals to end time	Session cannot be 0 minutes	Session cannot be 0 minutes	Pass
ADD with duration less than 30 minutes	Minimum session is 30 minutes	Minimum session is 30 minutes	Pass
ADD with wrong date format	Wrong date format	Wrong date format	Pass
ADD with date in the past	Date cannot be in the past	Date cannot be in the past	Pass
ADD with trainer already booked during mentioned time and date	Trainer already booked	Trainer already booked	Pass
ADD with correct syntax	Booking successful	Booking successful	Pass
UPDATE with less than 5 arguments	Not enough arguments	Not enough arguments	Pass
UPDATE with more than 5 arguments	Too many arguments	Too many arguments	Pass
UPDATE with bookingID having more than 5 characters	bookingID should be 5 characters	bookingID should be 5 characters	Pass

UPDATE with bookingID having invalid characters	bookingID should be alphanumeric	bookingID should be alphanumeric	Pass
UPDATE with bookingID which does not exist	bookingID does not exist	bookingID does not exist	Pass
UPDATE with trainerID having more than 5 characters	trainerID should be 5 characters	trainerID should be 5 characters	Pass
UPDATE with trainerID having invalid characters	trainerID should be alphanumeric	trainerID should be alphanumeric	Pass
UPDATE with trainerID which does not exist	trainerID does not exist	trainerID does not exist	Pass
UPDATE with specialismID having more than 5 characters	specialismID should be 5 characters	specialismID should be 5 characters	Pass
UPDATE with specialismID having invalid characters	SpecialismID is alphanumeric	SpecialismID is alphanumeric	Pass
UPDATE with specialismID which does not exist for mentioned trainer	Trainer does not do this specialism	Trainer does not do this specialism	Pass
UPDATE with wrong time format	Wrong time format	Wrong time format	Pass
UPDATE with wrong start time	Wrong start time	Wrong start time	Pass
UPDATE with wrong end time	Wrong end time	Wrong end time	Pass
UPDATE with time before 06:00	Gym opens at 06:00	Gym opens at 06:00	Pass

UPDATE with time after 23:00	Gym closes at 23:00	Gym closes at 23:00	Pass
UPDATE with start time after end time	Start time cannot be after end time	Start time cannot be after end time	Pass
UPDATE with start time equals to end time	Session time cannot be zero minutes	Session time cannot be zero minutes	Pass
UPDATE with duration less than 30 minutes	Minimum session time is 30 minutes	Minimum session time is 30 minutes	Pass
UPDATE with wrong date format	Wrong date format	Wrong date format	Pass
UPDATE with date in the past	Date cannot be in the past	Date cannot be in the past	Pass
UPDATE with trainer already booked during mentioned time and date for different client	Trainer already booked	Trainer already booked	Pass
UPDATE with trainer already booked during mentioned time and date for mentioned client	Booking successfully replaced	Booking successfully replaced	Pass
UPDATE with correct syntax	Update successful	Update successful	Pass

Conclusion

Summary

The ER-Diagram designed at the start of the project helped a lot in designing the classes, the features of the program and also to implement the database. The program has interesting features such as complex validations, an algorithm designed to check existing booking times, and automatic bookingID generation.

Limitations

Console program is not reliable as the staff will have to remember the whole syntax of the command. Some commands such as the add and update has 5 arguments, therefore becomes difficult to write. If the staff wrongly enters an information, the program will display an error but will not allow the staff to edit the previously typed command. He/she will have to type the whole command again.

As for the delete part of the console program, the program does not ask the user for confirmation of the selected bookingID. Therefore, if the staff enters a wrong bookingID and by coincidence it matches with someone else's bookingID, the latter's booking will be deleted.

Future approach

An automatic backup could be implemented to avoid data loss and wrong deletion or update of data. A proper complete GUI program can be developed to allow full feature of the database and allow staff to even add new customers.

References

beginnersbook.com. (2020). *Advantages of DBMS over file system*. [online] Available at: https://beginnersbook.com/2015/04/dbms-vs-file-system/ [Accessed 31 Jan. 2020].

Anon, (2020). [online] Available at: https://www.datanamic.com/support/lt-dez005-introduction-db-modeling.html [Accessed 31 Jan. 2020].

LIANG, Y. (2017). *Intro to java programming, brief version plus pearson mylab programming with pearson etext, ... global edition*. [Place of publication not identified]: PEARSON EDUCATION Limited.

Hackr.io. (2020). *Normalization in DBMS: 1NF, 2NF, 3NF and BCNF with Examples*. [online] Available at: https://hackr.io/blog/dbms-normalization [Accessed 31 Jan. 2020].

Visual-paradigm.com. (2020). *What is Unified Modeling Language (UML)?*. [online] Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/ [Accessed 31 Jan. 2020].