AQA A2 Computing

Gates-Tutoring center

Name:Ahamed Samir Sarker

1.Analysis

1.1 Introduction

1.1.1 Background:

Gates is a tuition centre based in Barking, London. Unlike other tuition centres, Gates mainly focuses on tutoring people who would like to have a career as an entrepreneur or in a profession which revolves around science subjects. Because of this the mentors provide sessions on entrepreneurship at various levels and also sessions at the A Level on Chemistry, Physics, Biology and Mathematics. Each of the 15 mentors specialises in one of the listed above fields and are qualified at the highest standard. Every mentor is assigned a room where each student is tutored privately one at the time, because of this each mentor usually gets to teach about 4 to 7 students per day depending on how what sessions the clients decide to book as different sessions have different durations which will affect the number of students that can book a session on the same day.

1.1.2 Problem Definition:

The problem with the centre is that its booking system is outdated and inefficient as the mentors have to rely on their own memory or write down every session that they have with the clients on paper. It is also time-consuming when a client would like to change the time or date of the booking as the mentor would have to look through their notebook and change the booking by rewriting over the details of the session which leads to a messy notebook making it harder for the mentors not to make any mistakes.
Beside this is also a major problem as the notebook used could easily get damaged or it could also get lost resulting in losing all the data. It can also cause problems if there are many sessions booked with different clients as it would be hard to keep track.

1.1.3 Users

The users will consist of the 14 mentors currently working at Gates, the admin /manager which also a mentor and the clients.
In the current booking system, the mentors are either contacted through e-mail or by a phone call by the clients about when they would like to book a session, the mentor then stores the data by writing it down on paper. The mentors can check the bookings from the notebook they wrote it on and edit/delete the bookings, however, if a client wants to change the time/date of the session or if they want to cancel it they must contact the specific mentor they booked the session with. At the end of the all the mentors have to tell the manager/admin all of the session they had for that day, the manager then calculates some statistics such total revenue for that specific day and records them separately.

1.2 Investigation of User Needs and Acceptable Limitations

1.2.1 The current System Analysis

1.2.1.1 Initial interview: Danny Joestar

1. What is the current system used to book sessions?

All the sessions are recorded on notebooks.

2. How many sessions do you usually have per day?

Around 6 sessions per mentor. However as each sessions have a different duration this varies constantly also as there are different subjects being taught some sessions for some specific subjects for example entrepreneurship have a longer duration.

3. How many mentors are there?

We have 12 mentors at the moment. We are also hiring new ones as we look to expand.

4. What type of sessions do you do?

We mainly focus on teaching students who are currently studying for A Levels and at the University level. We have sessions on Mathematics and Science subjects at the A Level. We recently started providing sessions on entrepreneurship which have been very popular among people aged 20-25 years. However each mentor also teach extra classes in fields that they are specialized in.

5. How many different subjects can each mentor teach?

As we want to provide tutoring of the highest standard each mentors only tutor on the the subject they are specialised in.

6. How much do you charge per session?

This varies per subject and how long the session is. Prices per session can vary from £20 upto $50.

7. Do the mentors and pupils have accounts to record their work/data?

No.

8. Can the mentors review their previous work ?

They are able to if they can store their old notebook, however as this is quite hard it is hard for them to do so.

9. Do you think this system is efficient?

No, I don't think it is. It's very hard for me to keep track of all the data and sessions, also because they are written on paper its very easy for it to get lost. It's also hard and frustrating if you want to work at home because you have to drag the notebooks front and back from the office and home.

10. Do you think the current system you are using needs improvements? If so what main improvements

Yes. I think the booking system should be more efficient by being able to store the data online somehow so it is kept safely and can be accessed easily from anywhere.

Conclusion:
The system used is outdated. Storing data on paper(notebooks) can be time-consuming and inefficient, also the notebooks can be easily lost or get damaged, which results in re-writing the data lost in another Notebook. It also very hard for the mentors to review previous work days as they would have to keep all of the notebooks they had used in the past.

1.2.1.2 Questionnaire

1. Do you think the way data stored from the booking system is inefficient?

YES NO

2. Do you think the current booking system provides customer satisfaction?

YES NO

3. Would you prefer to have an online booking system?

YES NO

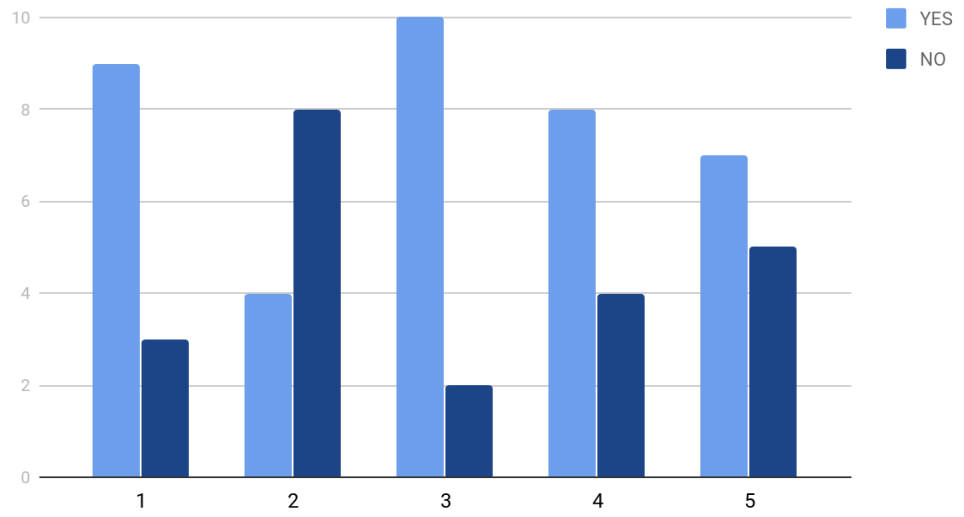
4. Do you think it's easy to lose your data about sessions and clients by using your current system?

YES NO

5. Do you think it would be hard to store data if you had more clients?

YES NO

Results:



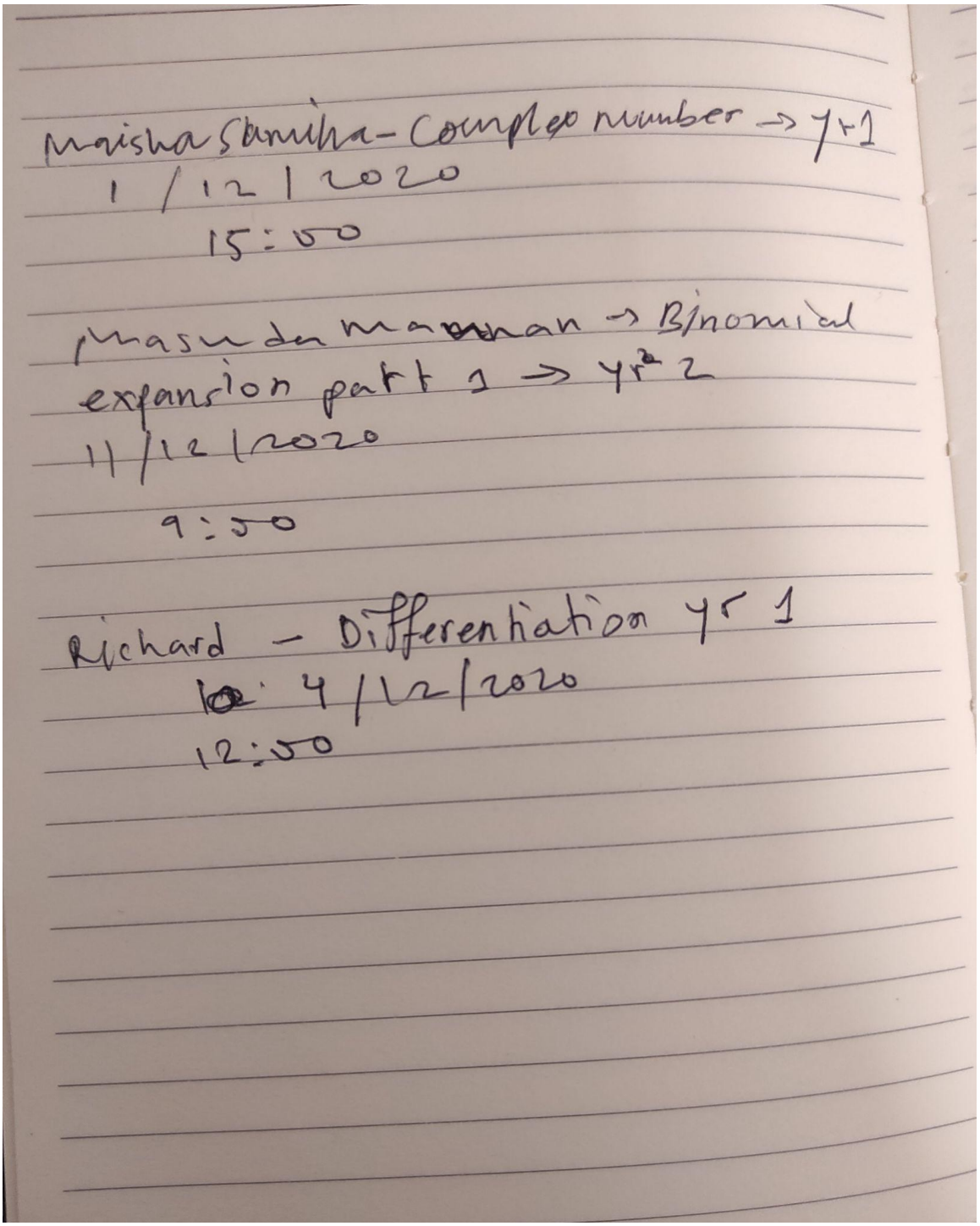
Summary:
The questionnaire was given to the 15 mentors working at Gates (1 is the admin). We can see that the majority of the mentors dislike the current booking system and think it is inefficient. The majority of them also thought that customer satisfaction is low because of it, this could mean that they think the booking system is impeding them on the growth of the company. Most of them would like to have an online booking system as it is easier to track the sessions and the clients, it would also mean that they won't lose data anymore compared to the current booking system and they would also be able to review their previous work.

1.2.1.3 Observation

To see how the current booking system work I asked Danny, one of the mentors if he could show me the
process:

Initially, a client will contact the mentor they would like to teach them by either calling or emailing them. If
the client already knows about the sessions and their prices, then they would just give their details(Name, Surname and phone number), the date and time of the session and the name of the
session as well. The mentor would then reply if that specific time slot is available and arrange for another
time if it isn't.
If its a new client then they would either first have to come to the centre and get the information about the mentors and session or call the receptionist at the front desk.

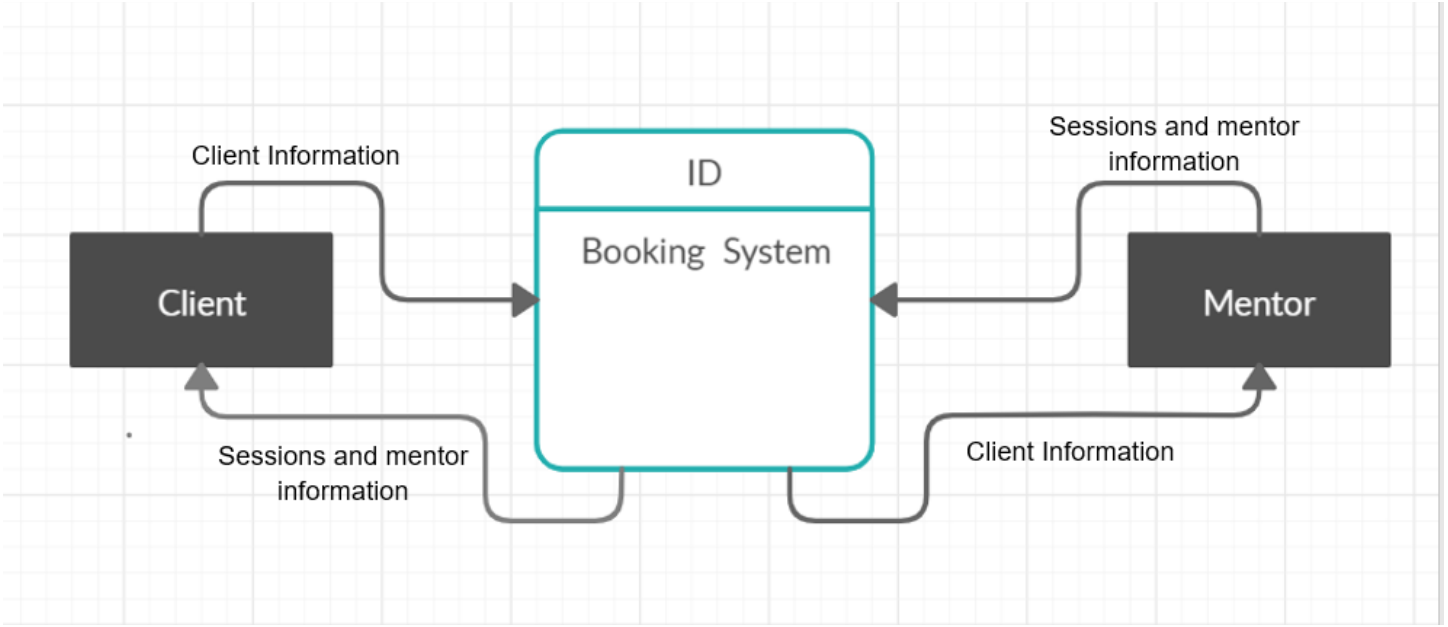
1.2.1.4 Investigation of documentation



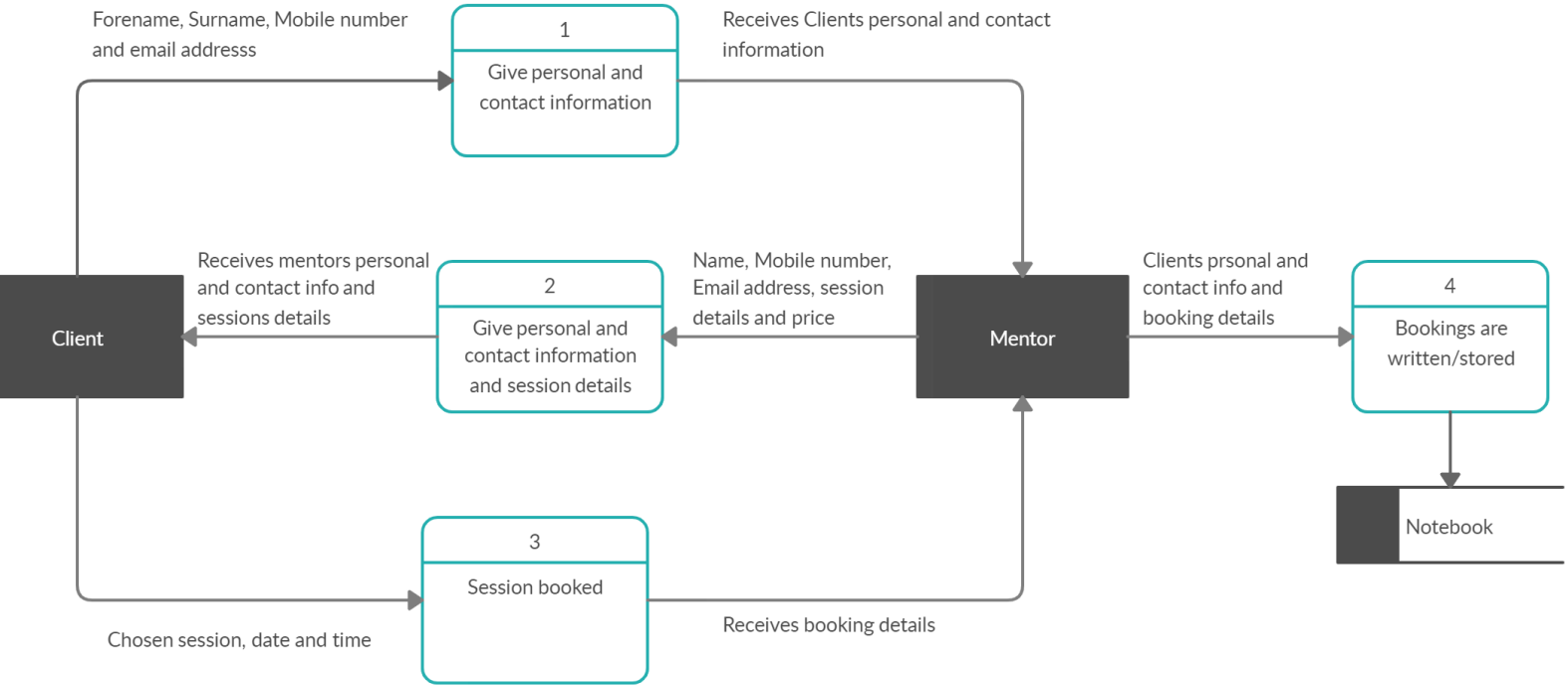
The mentors record the name of the client, the name of the session they will be tutoring and the time and date. I have been told that for completed appointments the user does a tick beside the appointment which shows that it has been completed.

1.2.2 DFD of the current system

Level 0



Level 1



1.2.3 Data Sources and Destinations

Data	Explanation	Source	Destination
Customer name	First name and surname of the customer.	Customer/ client	Mentor
Mobile number	The customer's phone number.	Customer/ client	Mentor
Email address	The customer's email address.	Customer/ client	Mentor
Session	The session the customer has chosen.	Customer/ client	Mentor
Date and time	The date and time the customer would like to book the session.	Customer/ client	Mentor
Mentor information	Information about mentors. The client can get this information by asking any mentor.	Mentor	Customer/ client
Booking	The customers and session information.	Mentor	Notebook

1.2.4 Entity-relationship diagram of current system

The current system does not use any database.

1.2.5 Discussion of problems

1. Because the mentors store the booking details in their notebook it is hard for them to keep track of the clients, this would become an even greater problem if there will be an increase in clients.
2. The booking system is inefficient and time-consuming, especially when the mentor needs to change pieces of information about the booking.
3. The notebooks can be easily damaged or lost, this would result in data being lost.
4. The mentors need to take their notebooks home if they want to do any type of work at home, which can be frustrating.
5. The clients can not keep track of the sessions they had unless if they themself write the information down or rely on their own memory.

6. The clients have to contact or go to center physically if it is their first and they want information about the sessions or mentors

1.2.6 The Proposed new system analysis

1.2.6.1 Second interview - Danny Joestar(mentor)

Should the mentors have their own accounts which they can manage?
Yes
.
Should new mentors be able to add themselves?
Yes. They should add their personal and contact information along with the sessions they provide and their prices

Should the clients be able to use this system?
Yes. It would be easier for us if they could book a session without contacting us, they should also be able to check which time slots a mentor is available and other information about a session or a mentor.

Should the clients have their own accounts which they can add to the system by signing up?
Yes. They should be able to provide their forename, surname, mobile number and email address when they sign up so its easier for us as all the required personal details needed will already be provided.

Should the information about each mentor be displayed on the system to the clients?
Yes. It would make it easier for a client to choose a mentor as the information is visual and they will be able to check it anytime they want if they might want to change mentor.

Is there any data which the mentors should have easier access to?
The clients booking should take priority.

Should the clients be able to cancel their bookings or change its time or date without contacting the mentors?
Yes, this would be much more efficient. It would also make it easier for the customers and will take much less time. However, it would be better if they could only do this 1 day prior the day of the session.

What types of input will be there?
The details of both the clients and personal trainers,sessions and prices of the session should be in the new system.

Should the customer receive some sort of confirmation regarding their booking?
It would be helpful if there is a place in the new system where the clients can see their bookings and edit/ cancel them accordingly. They should also be able to tell if the payment for the session was successful or not.

Would you like to see some sort of reports? If so what should they be based on and how frequently would you like to receive them?

If the admin gets reports automatically on the mentors performance such as their total revenue it would help us better understand what part should be improved to further progress.

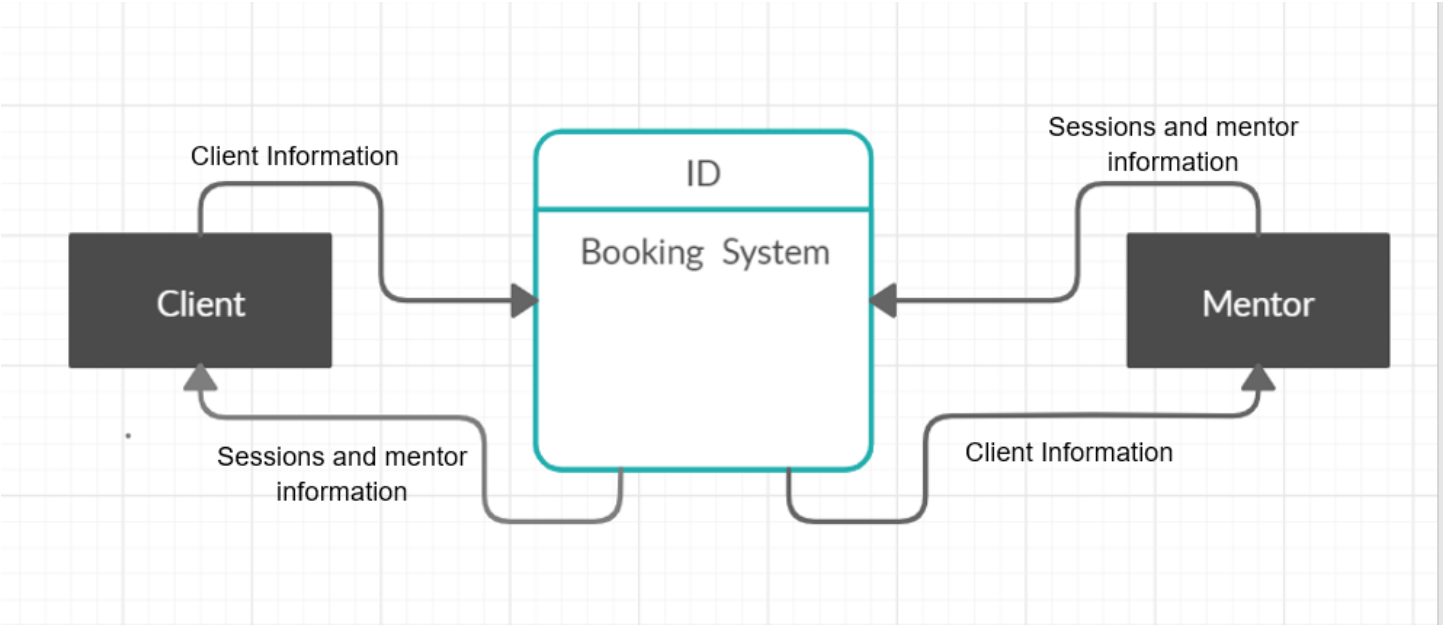
1.2.6.2 User needs

The users will be the mentor at Gates, the manager/admin (also a mentor) and the clients.
The system should be able to allow the users to:

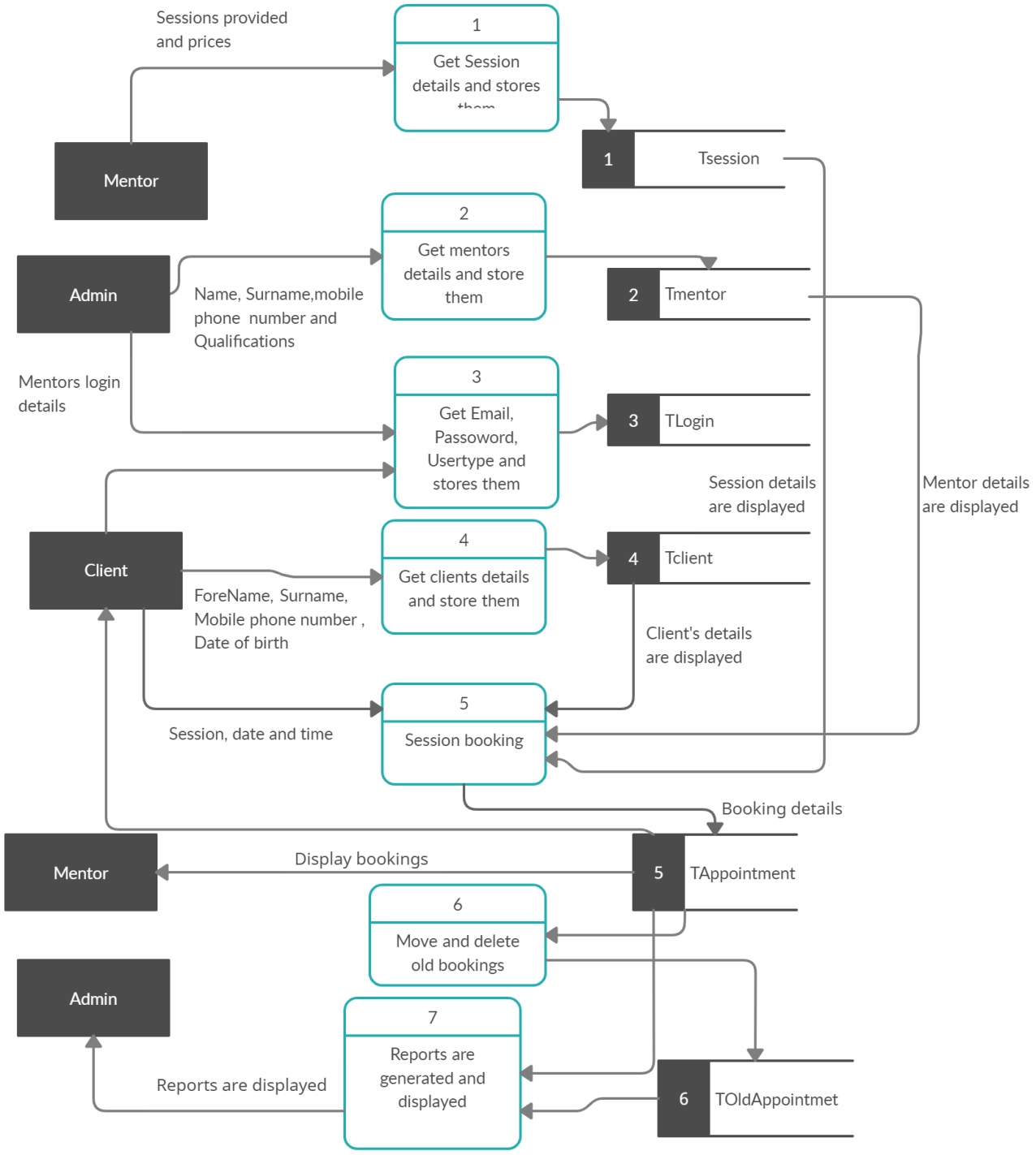
- Access a form which the clients can fill with their personal details and create an account for them when they finish the form and sign up. Give only access to the admin to make an account for the mentors.
- The client and mentor should be able to view their bookings and be able to cancel or change them a day before the day of the booking.
- The client should have a secure login access to the system.
- The mentors should have a secure access to the system.
- The clients should be able to change any of their details.
- The mentors should be able to change their personal details
- The mentors should be able to add new sessions and alter existing ones.
- When a booking is created the cost should already be calculated.
- Calculate the data from each month(only data needed for the reports) and display them as a report to the mentors.
- Store the clients details and booking information in an organised database.
- Display to the users information about the sessions and mentors if needed.

1.2.7 DFD of proposed new system

Level 0



Level 1


Sessions provided and prices
1
Get Session details and stores them
Mentor
1
Tsession
2
Get mentors details and store them
Admin
Name, Surname,mobile phone number and Qualifications
2
Tmentor
Mentors login details
3
Get Email, Passoword, Usertype and stores them
3
TLogin
Session details are displayed
Mentor details are displayed
Client
4
Get clients details and store them
4
Tclient
ForeName, Surname, Mobile phone number , Date of birth
Client's details are displayed
5
Session booking
Session, date and time
Booking details
Mentor
Display bookings
5
TAppointment
6
Move and delete old bookings
Admin
7
Reports are generated and displayed
Reports are displayed
6
TOldAppointmet

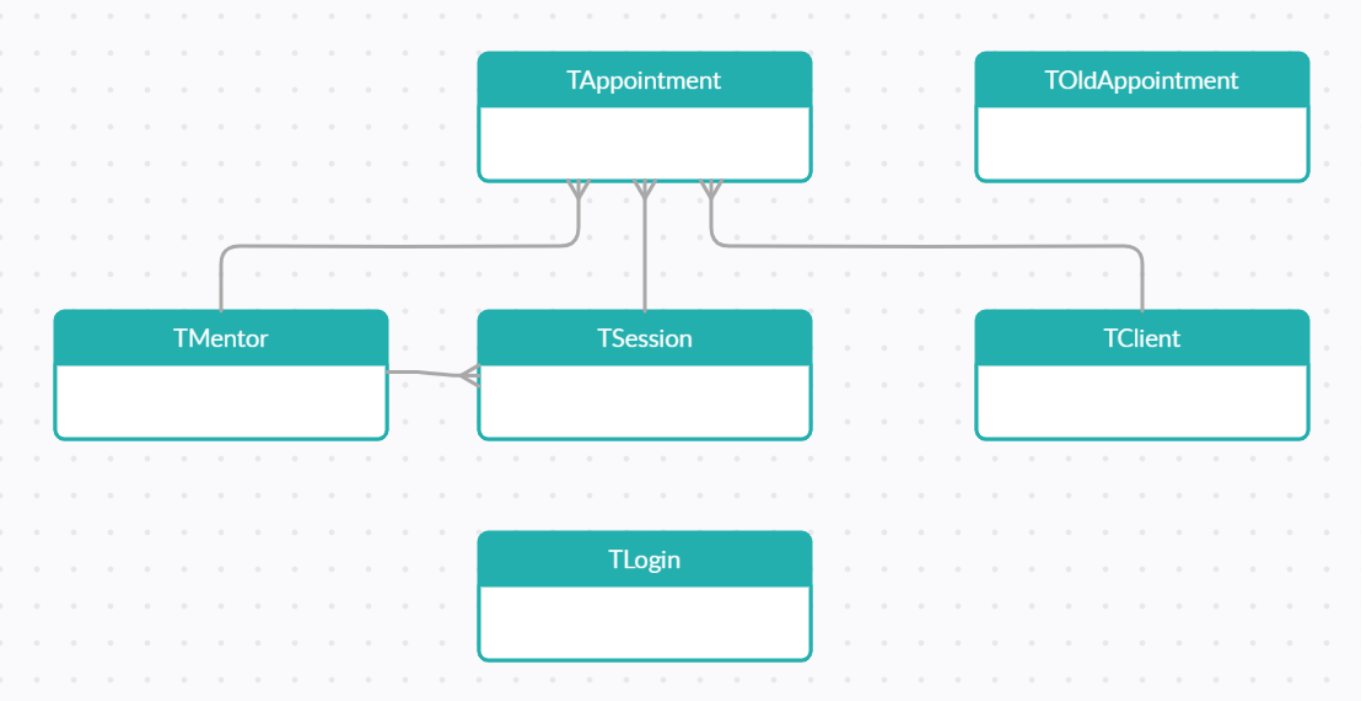
Description of DFD

The admin registers the mentors into the system by entering their personal details and they are stored on the mentors table, their email and password and the them being mentor is stored on the login table, The clients register themselves, their personal information is stored on the clients table and their email, password and them being a client is stored on the login tables.
The mentors add sessions and they are stored on the sessions table.
The clients select the session they want to book and add a date and time which creates a booking, this booking is stored on the Appointments table. The booking can then be viewed by the mentors and the clients. Some reports are created from the data from the appointments table and are displayed to the admin, while older records from the appointments table are moved to another table for older appointments. This table is also used to generate reports and display them to the admin.

1.2.8 Data sources and destinations

Data	Explanation	Source	Destination
Clients forename and surname	The clients first and last name	Client	Tclient
Clients mobile number	The clients phone number	Client	Tclient
Clients Date of birth	The clients date of birth	Client	Tclient
Clients Email	The clients email	Client	TLogin
Clients password	The clients password	Client	Tlogin
Mentors details	The mentors name, mobile number and Qualifications	Admin	Tmentor
Mentors Email	The mentors email	Admin	Tlogin
Mentors Password	The mentors password	Admin	Tlogin
Session details	The sessions name, duration,price and type	Mentor	Tsession
Session, Date, Time	Details for booking(These are details the client selects, their personal details are added automatically)	Client	TAppointment
Bookings	All the data needed for a booking from older appointments	TAppointment	TOldAppointment

1.2.9 Entity-Relationship Diagram



TClient(ClientID, CForename, CSurname, CMobileNumber,DateOfB)
TMentor(MentorID, MForename, MSurname, MMobilenumber, Qualifications)
TLogin(MainID, ID, Email, Password, UserType)
TSession(SessionID, **MentorID**, SessionName, SessionLength, Sessiondetails, Price)
TAppointment(AppointmentID, **ClientID**, **SessionID**, **MentorID,** Time, Endtime, Price, Date)
TOldAppointment(ID, AptID, ClientID, SessionID, MentorID, Price, Time, End, Date)

Key
Primary Key.
Foreign Key

1.2.10 Analysis Data Dictionary

Data name	Purpose	Data type	Example
CForename	Records the first name of the client.	String/Varchar	John
CSurname	Records the last name of the client	String/Varchar	Moore
CMobilenumber	Stores the clients mobile phone number.	String/Varchar	0745729846
DateOfB	Records the client's date of birth	Date	12/03/1995
MForename	Records the first name of the mentor.	String/Varchar	Nicholas
MSurname	Records the last name of the mentor.	String/Varchar	Wilson
MMobileNumber	Stores the mobile phone number of the mentor	String/Varchar	07126999452
Qualifications	Stores the mentor's qualifications.	String/Varchar	Hons. in English Literature, Oxford University
Email	The users email	String/Varchar	samir@gmail.com
Password	The users password	String/Varchar	1236543
ID	ClientID OR MentorID	Integer	2
SessionName	Stores the name of the session.	String/Varchar	Entrepreneurship-Introduction
Sessiondetails	Stores the details of the session	String/Varchar	The following sessions consist of complex numbers.
SessionLength	Stores the length of the session(Hours)	Integer	2
Price	Stores the cost of the session.	Double	£25
Time	The chosen time of the session	Date/time	17:00
Date	The chosen date of the session.	Date	27/06/2020

1.2.11 Data volumes

There are 15 mentors at the moment but since they are looking to expand, the maximum number of mentors can be upto 20, so the new system should be able to store 20 mentor's names, mobile phone numbers, email addresses, the sessions they provide and their prices of the sessions on the database.Each mentor has about 30 sessions per week so all the mentors together have about 450 sessions, if they expand the they could reach up to 600-650 sessions per week in total, so the system should be able to store 650 booking details. Considering each client book about two sessions per week each mentor has about 15 different clients so the new system should be able store the details of at most 200 clients. Each client's name, email address, mobile phone number and date of birth must be stored in the database.

1.3 Constraints

1.3.1 Hardware Constraints

Recommended hardware requirements:

- Processor 2 Hz
- 4 GB RAM
- 10GB of hard disk space
- Graphic card - supports directx10
- Monitor -1280x800

1.3.2 Software Constraints

To be able to change the code or to run the system smoothly the user needs to meet these software requirements:

- Visual Studio Code
- Latest browser
- SQL
- HTML
- .net 5 or above
- Microsoft access 64 bit
- Microsoft access drivers
- OS windows 10

1.3.3 Time Constraints

Design stage - 31st December
Testing - 15th January
Maintenance - 25th January
User manual - 10th February
Appraisal - 28th February

1.3.4 User's knowledge of information technology

In the current booking system the users would have to write down their session's details manually on paper or rely on their own memory. Also the clients had to call or message a mentor to book, change or cancel a session and to give their personal and contact details or to ask for a mentor's details. With the new system the users would just have to fill a registration form giving their personal and contact information at the start, and their details will be stored on the new systems database. The mentors details and login info should also be able to be added by the admin, the admin should be able to delete these details as well as clients details from the database. The users will also be able to view the bookings, change or delete them accordingly. The proposed system should be easily used by the users due to its user-friendly forms and simple layout when displaying the booked sessions.

1.3.5 Who will be allowed to use the various parts of the system

The new system will be used by the clients and the mentors.The clients and mentors will have their own type of registration forms as the two users are required to input different data. The mentors and client will also have their own login pages - unique usernames and passwords. The client will be able to see information about sessions and mentor after they login and be able to book a session. The mentors can then see the sessions booked with them. The clients will be able to change their bookings or cancel them a day before the session and the mentors will receive a message/notification. A mentor will be selected as admin. He will be allowed to use the system the same as all the other mentors however he will also have the power to remove and add a client or personal trainer from the system.

1.5 Objectives

1.5.1 General objectives

The new should be more efficient to both the clients and mentors and be less time consuming
The mentors should be able to receive information about the clients and store them in a secure way without wasting time. It should also be able to provide accurate reports to the mentors.
The clients should be able to book sessions and cancel or change them easily.
The new system should be user friendly and not confusing for the users.

1.5.2 Specific objectives

1)The system should be easy to navigate
- 1.1) There should be a navigation bar with links to the various part of the website.
- 1.2) There should be a footer with a link to a contact page
- 1.3) There should be a homepage with details about the company.
- 1.4) There should be a mentors page displaying all of the mentors.
- 1.5) There should be a sessions page displaying all the available.
- 1.6) There should be specific pages for when users log in which can be easily be navigated to
- 1.7) Links within pages should make sense in why they are there.

2) Only the clients and the mentors(plus admin) should be allowed to use the system, so there should be some sort of authorisation/login:
- 2.1) The clients and mentors will have to fill different forms and an account will be created for them.
- 2.2) The clients will need to login with their email and passwords to use the system.
- 2.3) The mentors should only be able to be added to the system by the admin.
- 2.3) The mentors and the admin will also have to use their emails and password to login.
- 2.4) There should be some sort of security check that check if the data entered by the user when registering is correct
- 2.5)Depending on the user type there should be different features available after logging in.

3)The system should be able to store and display clients data and allow them to change it later:
- 3.1) It should be able to store their name.
- 3.2)It should be able to store their mobile phone number.
- 3.3)It should be able to store their date of birth.
- 3.4)It should be able to store their email address.
- 3.5) It should be able to store their password.
- 3.6) The password should be hashed.
- 3.7) It should be able to store the fact they are clients.
- 3.8) The clients should be able to edit their personal information

3.9) The client should be able to able to edit their password

4) The system should allow the clients to book sessions through a booking form and be able to change the details of the booking or cancel the booking:

4.1) The client should not be able to book a session with a time in which the mentors are not working.
4.2) The client should not be able to book a session with a mentor that has already a booked session at the same time and date.
4.3) The clients should not be able to book a session when they have another session booked at the same time and date.
4.3) The client should only be allowed to book a session after entering all the required details in the correct fields of the booking form.
4.4) A session only be booked when payment has been received.
4.5) Clients should be able to edit the booked session or cancel it within a time restriction (1 day before the booking date)
4.6) The mentors should also be able to to edit or cancel booked sessions a day prior to the booking Date
4.7) Clients should only be able to book a session after they log in

5) The system should store the mentors details and display specific details to other users:
5.1) It should store and display the mentor's name.
5.2) It should store and display the mentor's mobile number.
5.3) It should store and display the mentor's Qualifications.
5.4) It should be able to store the mentor's email address.
5.5) It should be able to store the mentor's password.
5.6) The password should be hashed.
5.7) It should show the fact that they are mentors.
5.8) it should store the same information for the admin except it should store the fact that they are admin instead of mentors.
5.9) The mentor's should be able to change their details.
5.10)The mentor(and admin) should be able to edit their password
5.11) The mentors name, mobile number and qualifications should be displayed to the client

6) The mentors and admin should be able interact with their own sessions:
6.1) They should be able to to add new sessions
6.2) They should be able to edit existing sessions.
6.3) They should be able to remove a session.

7) Clients and mentors(and admin) should be able to view booked the sessions:
7.1) The clients and mentors should only be able to view their own booked sessions.
7.2) The admin should be able to view all of the booked sessions.

8) One of the mentors' accounts will also act as the administrator

8.1) The account should be able to do what all the other mentors' accounts can do.
8.2) The administrator should be able to view all of the mentors.
8.3) The administrator should be able to add a new mentor to the system.
8.4)The administrator should be able to remove a mentor from the system.
8.5)The administrator should be able to view all of the clients.
8.6)The administrator should be able to remove a client from the system.

9)The admin and the mentors should receive reports:
9.1)There should be a form which will add constraints when creating the reports.
9.2) The mentors should be able to use the form to receive a report on past bookings.
9.3) The admin should be able to create a report for a specific mentor or for the whole company
9.4) There should be graph which display the projected revenues for the future.

10) There should be a contact page
10.1) It should display when the center is open
10.2) It should display where the center is located.
10.3) It should have a form which can be used to contact the company

11) There should be a checkout system
11.1) The checkout system should be secure
11.2) The client should be able to pay by card

2. Design

2.1 Overall system design

2.1.1 IPSO Chart

Client

Input ● Client ClientID ● Client forename ● Client surname ● Client phone number. ● Client date of birth. ● Client email address ● Client password ● Session date and time.	Process ● Log in to the website. ● Change personal details. ● Search for sessions. ● Book sessions. ● Edit booking ● Cancel booking.
Storage ● Client table ● Login table ● Appointments table	Outputs ● Display personal details ● Display sessions. ● Display mentors. ● Display booked sessions.

Mentor

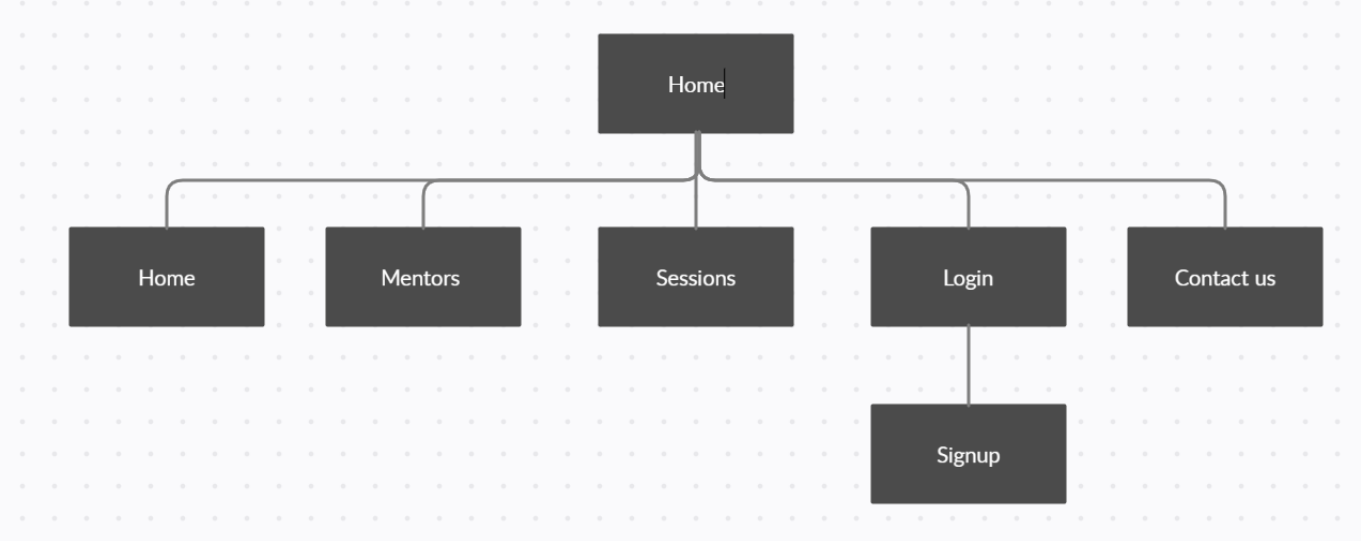
Input ● Mentor ID. ● Mentor forename ● Mentor surname ● Mentor mobile number ● Mentor Qualifications ● Mentor email address ● Mentor password ● Session name ● Session length ● Session details. ● Session prices.	**Process** ● Log in to the website. ● Change personal details. ● Change session details ● Edit and remove sessions. ● Edit bookings. ● Cancel bookings.
Storage ● Mentor table ● Login table ● Session table ● Appointment table	**Output** ● Display personal details ● Display own sessions ● Display bookings.

Admin

Input ● Same as a mentor	**Process** ● Add mentors to the system ● Change personal details. ● Change session details ● Edit and remove sessions. ● Remove mentors ● Remove clients ● Edit bookings. ● Cancel bookings. ● Generate reports
Storage ● Mentor table ● Client table ● Session table ● Login table ● Appointment table ● Old appointment table	**Output** ● Display personal details ● Display own sessions ● Display client's details ● Display mentor's details ● Display bookings. ● Display statistics

2.2 Description of Modular Structure of the system

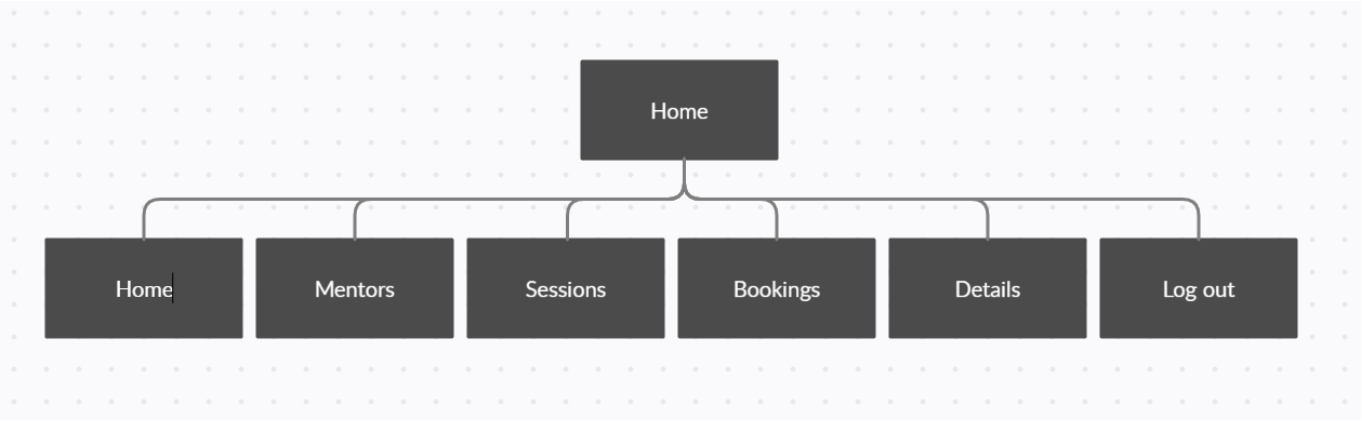
Initial



The system starts with the homepage. The following is the navigation bar before a user log in, (note the contact us is on the footer and is on every page which is why i will not include it on the following diagrams).
Before log in the navigation bar consists of links leading to the homepage, the mentors page, the sessions table and the login page. The sign up page can be accessed from a link on the login page.
After logging in the navigation bar will change depending on the user but the footer containing the link to the contact us page doesn't change.

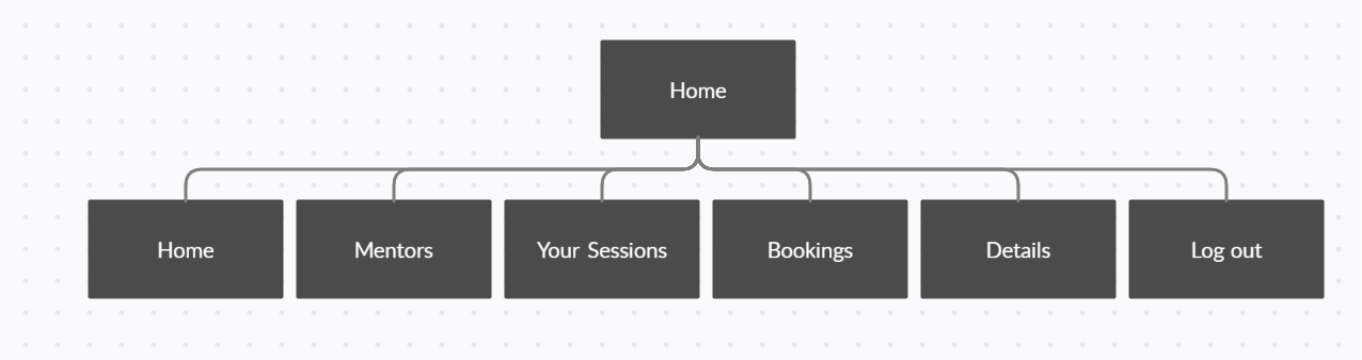
After login(navigation bar)

User: Client



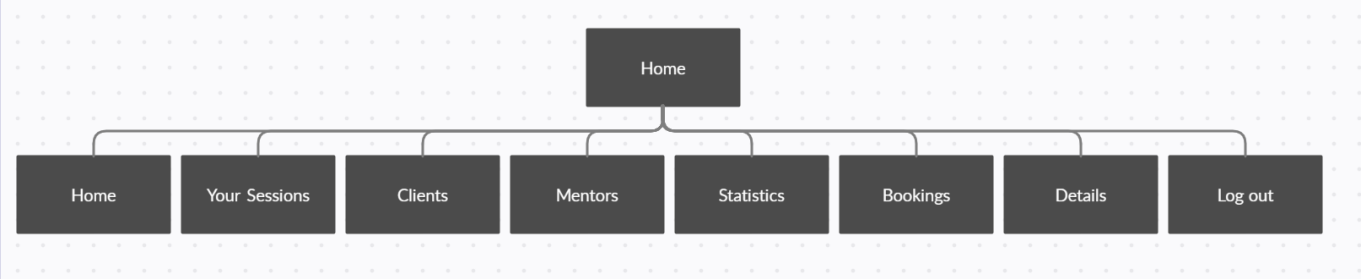
After a client logs in this is how the navigation bar changes. It now includes a booking link which leads to the bookings page, a details link leading to the clients personal details and log out button.

User: Mentor



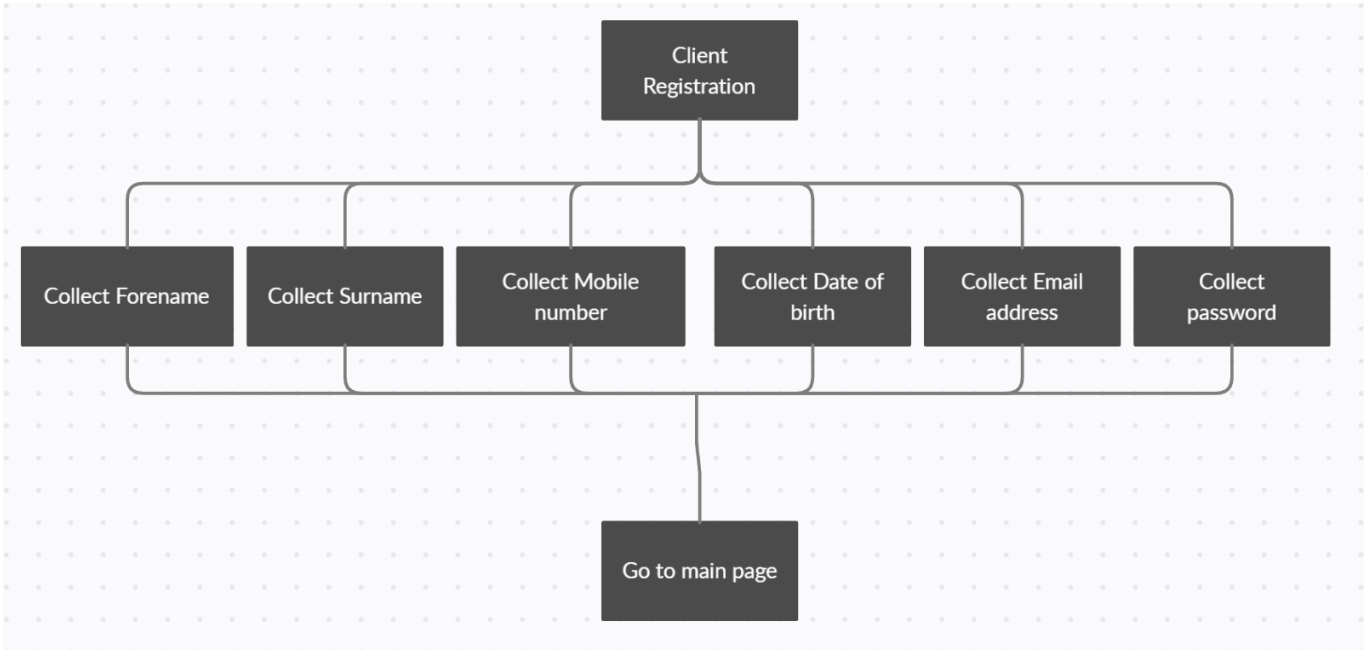
After a mentor logs in they will have a similar navigation bar to the clients except, instead of “Sessions” now it's “Your Sessions” which is a different page.

User: Admin

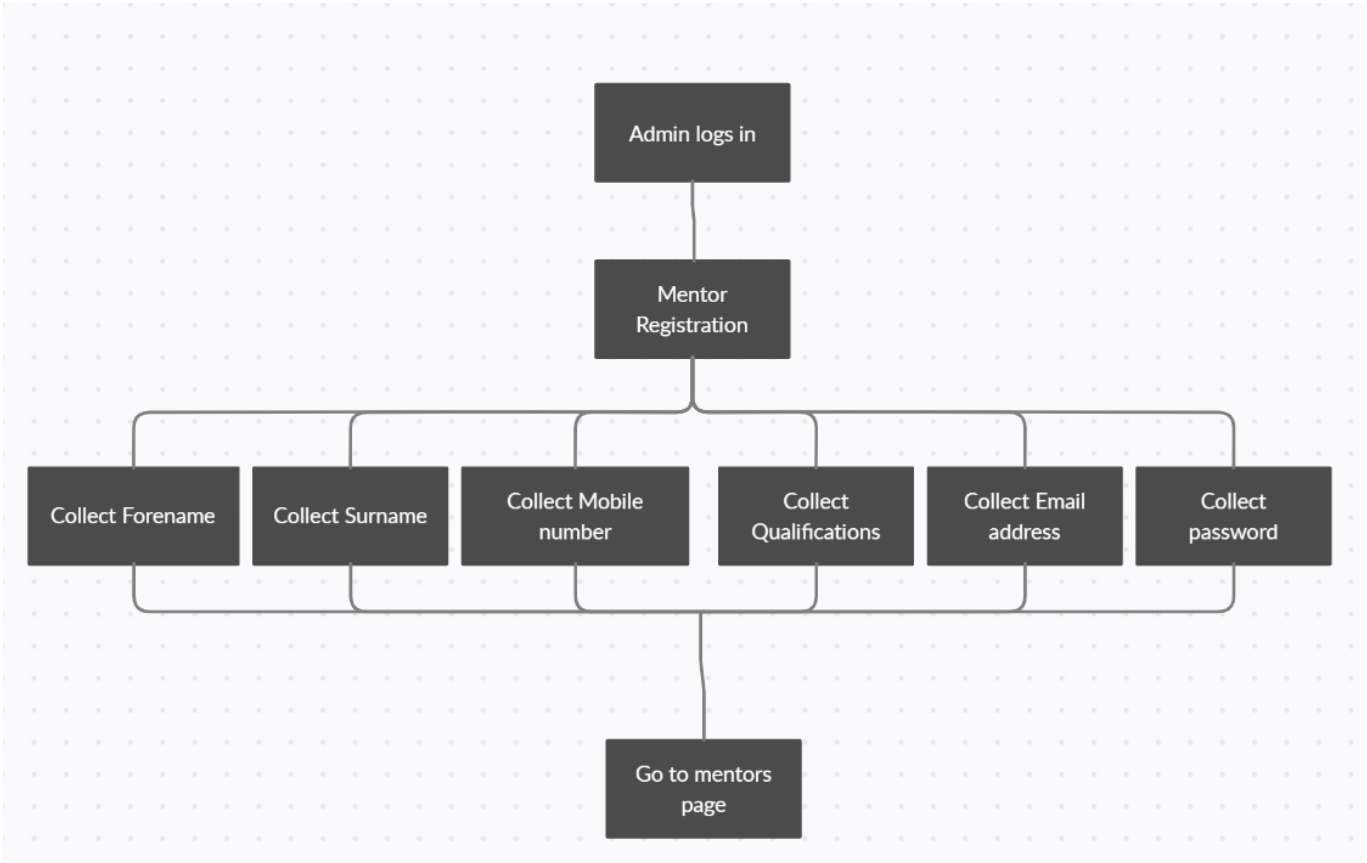


The navigation bar after the admin logs in. (Note: The Mentors link on this navigation bar links to a different page than the one on the other versions of the navigation bar. That page is exclusively for the admin). There is also a clients link and statistics page linking to the clients page and the statistics page respectively.

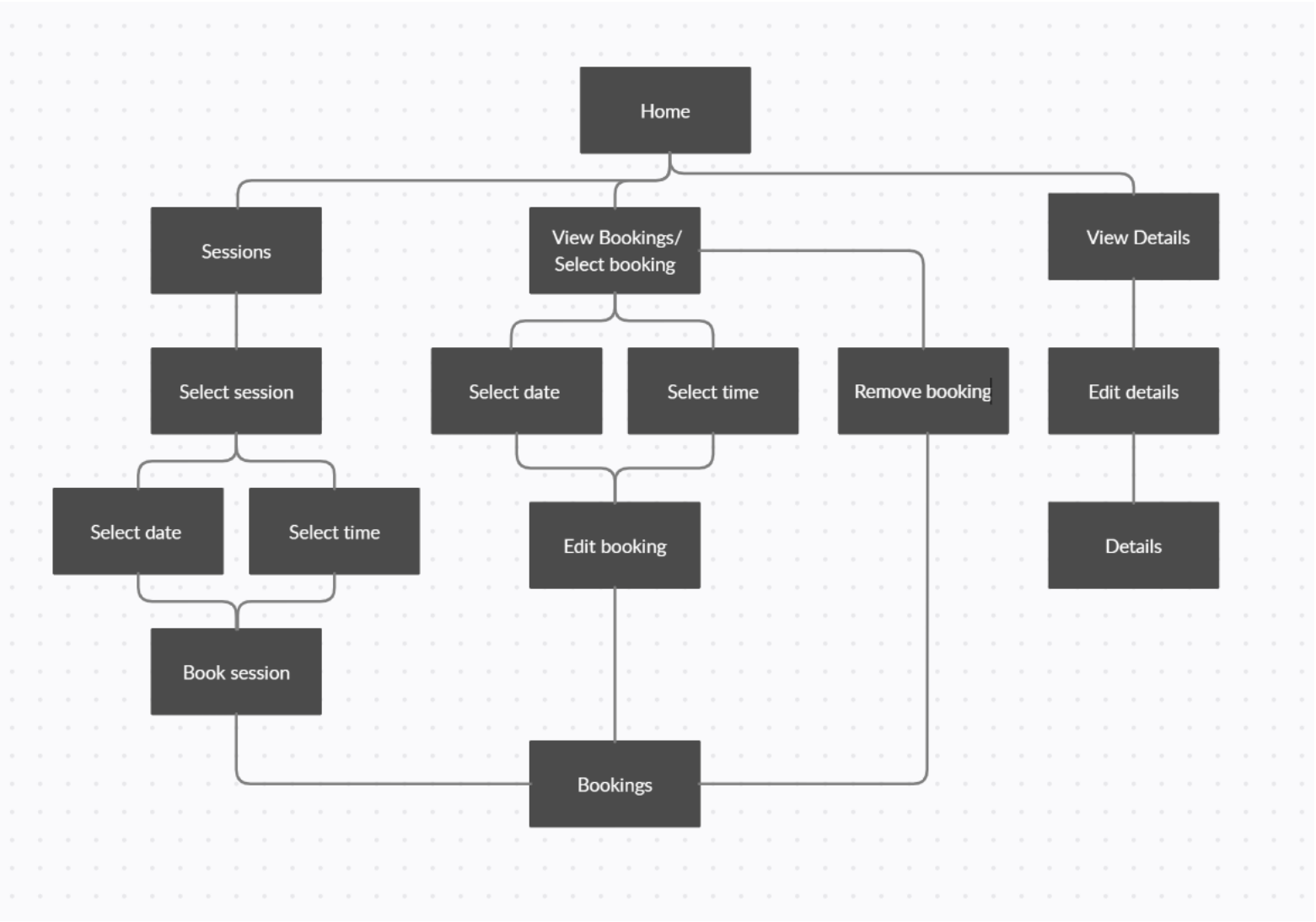
Client Registration



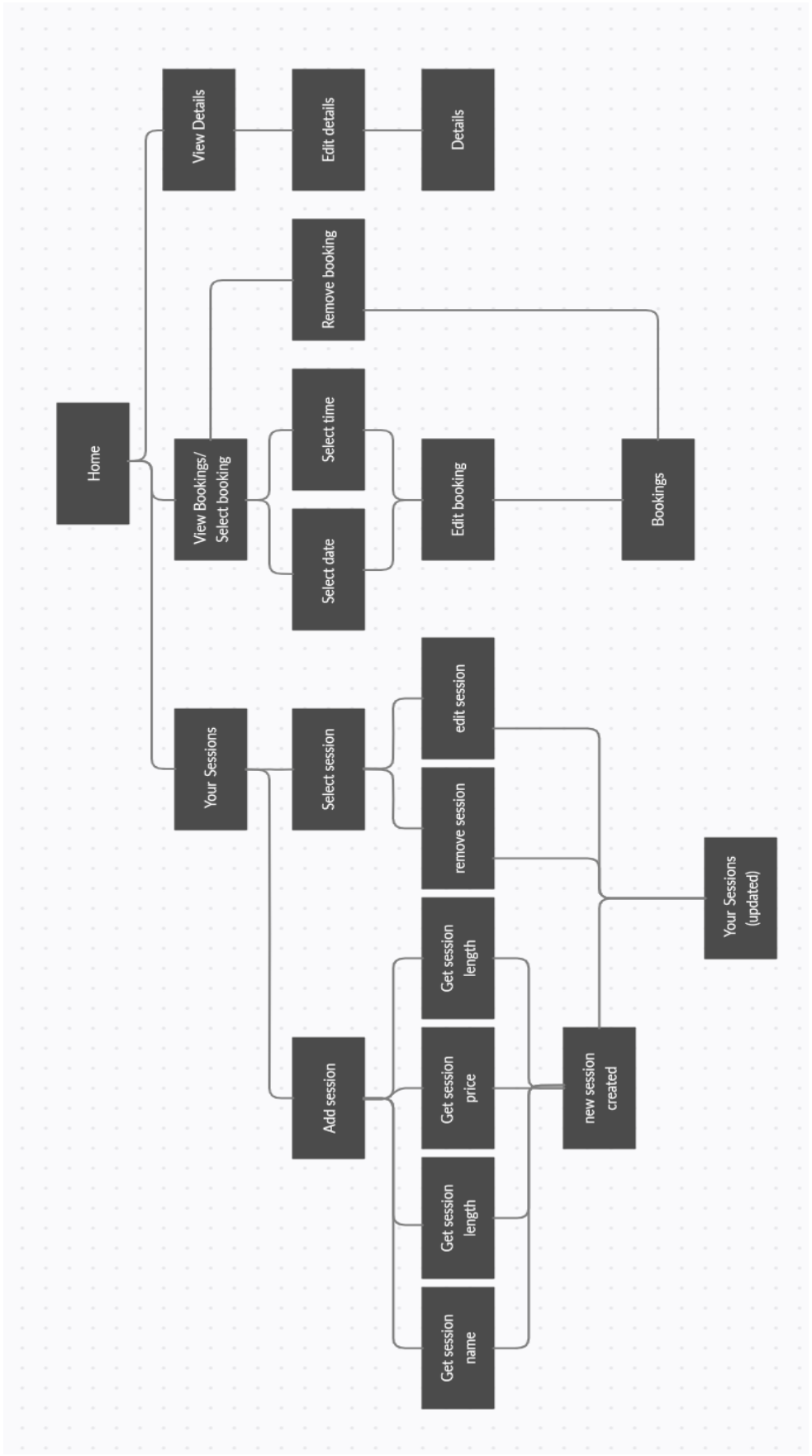
Mentor Registration



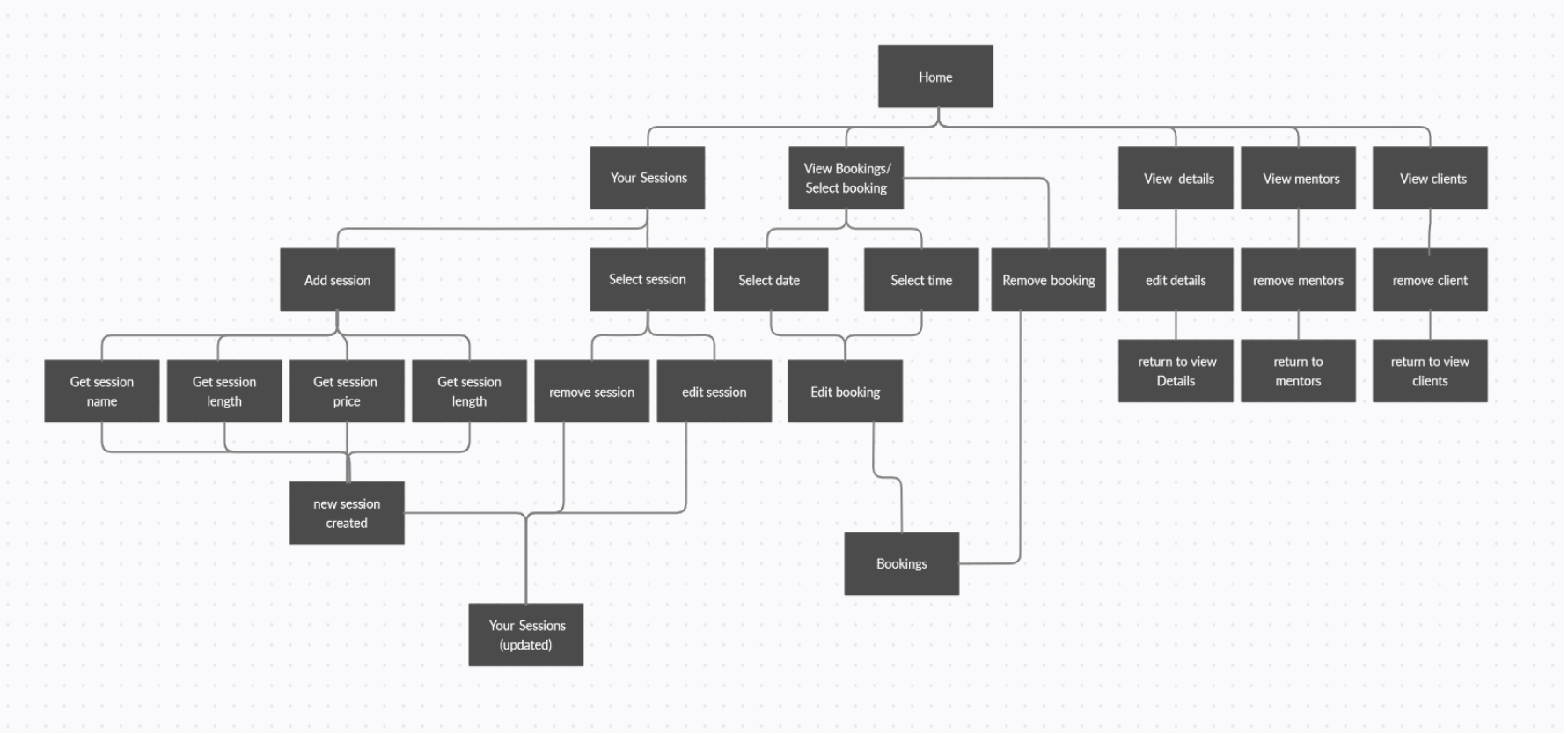
User: Client


Home
Sessions
View Bookings/
Select booking
View Details
Select session
Select date
Select time
Remove booking
Edit details
Select date
Select time
Edit booking
Details
Book session
Bookings

User: Mentor


Home
View Details
Edit details
Details
View Bookings/
Select booking
Remove booking
Select time
Select date
Edit booking
Bookings
Your Sessions
Select session
edit session
remove session
Add session
Get session
length
Get session
price
Get session
length
Get session
name
new session
created
Your Sessions
(updated)

User: Admin


Home
Your Sessions
View Bookings/ Select booking
View details
View mentors
View clients
Add session
Select session
Select date
Select time
Remove booking
edit details
remove mentors
remove client
Get session name
Get session length
Get session price
Get session length
remove session
edit session
Edit booking
return to view Details
return to mentors
return to view clients
new session created
Bookings
Your Sessions (updated)

2.3 Design Data Dictionary

TClient							
Field Name	Sample Data	Data Type	Validation (State Type)	Validation Error message	Requirement	Key	Description
ClientID	5	Integer	None	None	Yes	Primary	Identify the client
CForename	John	Varchar	Presence Check	“Please fill in the required field”	Yes	None	Records the forename of the client.
			Format check	“Forename should only be letters”			
CSurname	Ryder	Varchar	Presence Check	“Please fill in the required field”	Yes	None	Records the last name of the client.
			Format check	“Surname should only be letters”			
CMobileNumber	07865778991	Varchar	Presence Check	“Please fill in the required field”	Yes	None	Records the client mobile phone number.
			Format check	“Mobile number should only be numbers”			
			Length check	“Mobile number should be 11 digits”			
DateOfB	12/03/1995	Date/Time	Format Check	“Please enter a valid date of birth”	Yes	None	Stores the client’s date of birth

TMentor

Field Name	Sample Data	Data Type	Validation (State Type)	Validation Error message	Requirement	Key	Description
MentorID	5	Integer	None	None	Yes	Primary	Identify the mentor.
MForename	Mohammad	Varchar	Presence Check	“Please fill in the required field”	Yes	None	Records the forename of the mentor.
			Format check	“Forename should only be letters”			
MSurname	Ali	Varchar	Presence Check	“Please fill in the required field”	Yes	None	Records the last name of the mentor.
			Format check	“Surname should only be letters”			
MMobileNumber	07455778991	Varchar	Presence Check	“Please fill in the required field”	Yes	None	Records the mentor’s mobile phone number.
			Length Check	“Mobile number should be 11 digits”			
			Format check	”Mobile number should be only numbers”			
Qualifications	Hons. in Mathematics	Varchar	Presence check	“Please fill in the required field”	No	None	Stores the mentor’s qualifications

TSession

Field Name	Sample Data	Data Type	Validation (State Type)	Validation Error message	Requirement	Key	Description
SessionID	2	Integer	None	None	Yes	Primary	To identify the different sessions
SessionName	Algebra yr 11 intro	Varchar	Presence Check	“Please enter the name of the session”	Yes	None	Records the name of the session.
Sessiondetails	The session is about the use of differentiation in mechanics.	Varchar	Presence Check	“Please enter the name of the session”	Yes	None	Records the details of the session.
SessionLength	1	Integer	Presence Check	“Please enter the length of the session.”	Yes	None	Records the length of the session
			Value check	“Sessions should be 1,2 or 3 hours long”			
Price	20	Double	Presence Check	“Please enter the price of the session”	Yes	None	Records the price of the session.
			Format check	“The price should only be a number”			
MentorID	2	Integer	None	None	Yes	Foreign	Links the mentor table to this table

TAppointment

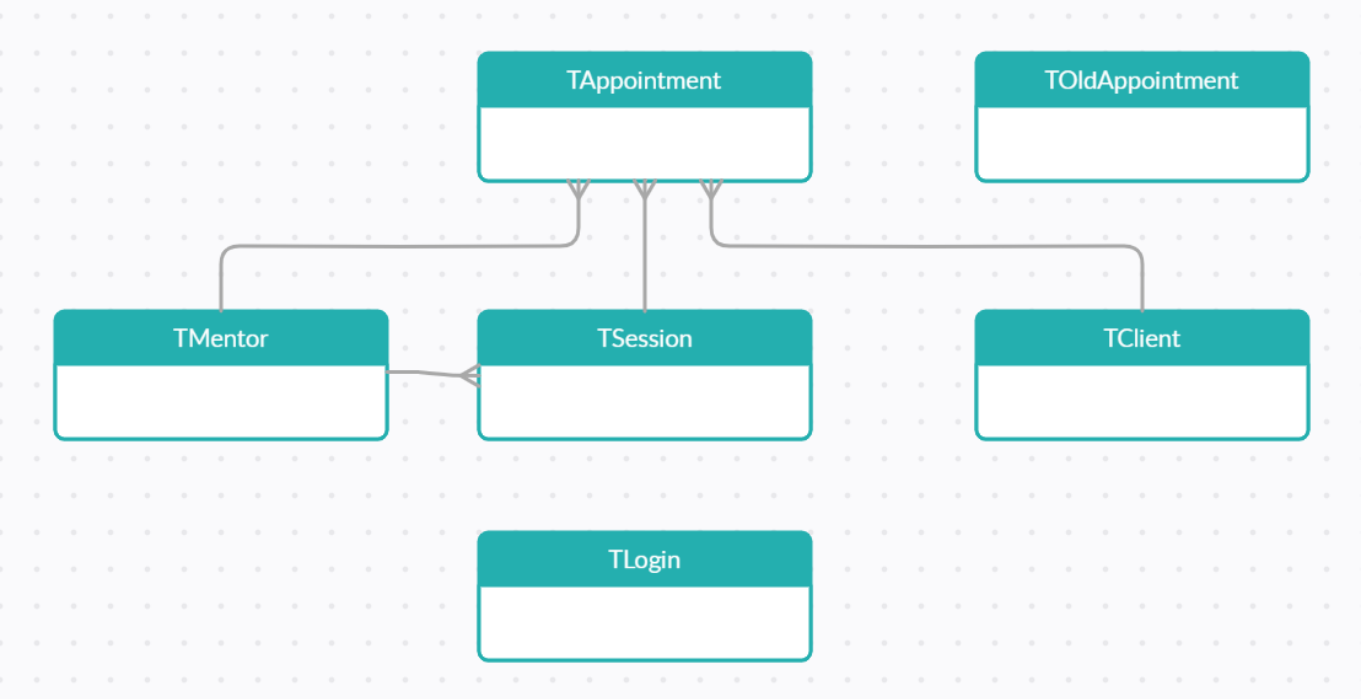
Field Name	Sample Data	Data Type	Validation (State Type)	Validation Error message	Requirement	Key	Description
AppointmentID	2	Integer	None	None	Yes	Primary	To identify the booked session
ClientID	3	Integer	None	None	Yes	Foreign	Links the client table to this table
SessionID	69	Integer	None	None	Yes	Foreign	Links the session table to this table
MentorID	67	Integer	None	None	Yes	Foreign	Links the mentor table to this table
Time	14:30	Date/Time	Presence Check	“Please enter the time you would like the session to be”	Yes	None	Stores the time the client has chosen for the session
EndTime	15:30	Date/Time	None	None	Yes	None	Records when the session ends.
Date	12/04/2020	Date	Presence Check	“Please enter the date of when would you like the session to be booked.”	Yes	None	Stores the date the client has chosen for the session.
Price	20	Double	None	None	Yes	No	The price for the appointment /session

TOldAppointment

Field Name	Sample Data	Data Type	Validation (State Type)	Validation Error message	Requirement	Key	Description
ID	2	Integer	None	None	Yes	Primary	To identify the old booked session/ appointment
AptID	2	Integer	None	None	Yes	None	The id from the TAppointment table
ClientID	3	Integer	None	None	Yes	None	Links the client table to this table
SessionID	69	Integer	None	None	Yes	None	Links the session table to this table
MentorID	67	Integer	None	None	Yes	None	Links the mentor table to this table
Time	14:30	Date/Time	Presence Check	“Please enter the time you would like the session to be”	Yes	None	Stores the time the client has chosen for the session
EndTime	15:30	Date/Time	None	None	Yes	None	Records when the session ends.
Date	12/04/2020	Date	Presence Check	“Please enter the date of when would you like the session to be booked.”	Yes	None	Stores the date the client has chosen for the session.
Price	20	Double	None	None	Yes	No	The price for the appointment /session

TLogin							
Field Name	Sample Data	Data Type	Validation (State Type)	Validation Error message	Requirement	Key	Description
MainID	2	Integer	None	None	Yes	Primary	To identify the users login details
ID	2	Integer	None	None	Yes	None	The ClientID or MentorID
UserType	Client	Varchar	None	None	Yes	None	So it's possible to tell if the user is a client, mentor or the admin
Email	ahamed@gmail.com	Varchar	Presence check	“Please fill the required field”	Yes	None	The email the user need to login
			Format check	“Please use a proper email”			
			Double email check	“This email is already registered with us”			
Password	123127(on the database the hashed version of the password is stored)	Varchar	Presence Check	“Please fill the required field”	Yes	None	The password the user needs to enter to login

2.4 Database Design(Normalised ERD)



- 1 Mentor can have multiple bookings/appointments and 1 appointment have only 1 mentor
- 1 Mentor can teach multiple sessions and 1 session can only have 1 mentor
- 1 Session can be on multiple bookings, 1 booking has only 1 session
- 1 Client can have multiple bookings/ appointments and 1 appointment only has 1 client

TClient(ClientID, CForename, CSurname, CMobileNumber,DateOfB)
TMentor(MentorID, MForename, MSurname, MMobilenumber, Qualifications)
TLogin(MainID, ID, Email, Password, UserType)
TSession(SessionID, **MentorID**, SessionName, SessionLength, Sessiondetails, Price)
TAppointment(AppointmentID, **ClientID**, **SessionID**, **MentorID,** Time, Endtime, Price, Date)
TOldAppointment(ID, AptID, ClientID, SessionID, MentorID, Price, Time, End, Date)

Keys
Primary Key,
Foreign Key

Sample of SQL queries:
SELECT * FROM TClient WHERE ClientID=1
INSERT INTO TLogin(MainID, ID, Email, Password, UserType) VALUES("1", "3","AHAMED@gmail.com", "Client")
DELETE * FROM TMentor WHERE MentorID = 2

2.5 File Organising and Processing

TClient			
Field Name	Max Amount of records	Max data size per record	Total size
ClientID	250	3 characters	750b
CForename	250	30 characters	7500b
CSurname	250	30 characters	7500b
CMobileNumber	250	11 characters	2750b
DateOfB	250	10 characters	2500b

Total size of TClient Table = (75+7500+7500+2750+2500)/1024=20.5kb

TMentor			
Field Name	Max Amount of records	Max data size per record	Total size
MentorID	20	3 characters	60b
MForename	20	30 characters	600b
MSurname	20	30 characters	600b
MMobileNumber	20	11 characters	220b
Qualifications	20	100 characters	2000b

Total size of TMentor Table=(60+600+600+220+2000)/1024 = 3.4kb

TSession			
Field Name	Max Amount of records	Max data size per record	Total size
SessionID	150	3 characters	450b
SessionName	150	30 characters	4500b
Sessiondetails	150	200 characters	30000b
SessionLength	150	1 characters	150b
Price	150	2 characters	300b
MentorID	150	3 characters	450b

Size of TSession Table=(450+4500+30000+150+300+450)/1024= 35kb

TLogin			
Field Name	Max Amount of records	Max data size per record	Total size
MainD	270	3	810b
UserType	270	7	1890b
Email	270	50	13500b
Password	270	255	68850b
ID	270	3	810b

Size of TLogin table=(810+1890+13500+68850+810)/1024= 83.85kb

TAppointment			
Field Name	Max Amount of records	Max data size per record	Total size
AppointmentID	500	3 characters	1500b
ClientID	500	3 characters	1500b
MentorID	500	3 characters	1500b
SessionID	500	3 characters	1500b
Time	500	8 characters	4000b
EndTime	500	8 characters	4000b
Date	500	10 characters	5000b
Price	500	1 characters	500b

Size of TAppointment Table=(1500+1500+1500+1500+1500+4000+4000+5000+500)/1024= 19.04kb

TOldAppointment			
Field Name	Max Amount of records	Max data size per record	Total size
ID	500	3 characters	1500b
AptID	500	3 characters	1500b
ClientID	500	3 characters	1500b
MentorID	500	3 characters	1500b

SessionID	500	3 characters	1500b
Time	500	8 characters	4000b
End	500	8 characters	4000b
Date	500	10 characters	5000b
Price	500	1 characters	500b

Size of TOldAppointment Table=(1500+1500+1500+1500+1500+4000+4000+5000+500)/1024= 20.51 kb

Total size of the database=(20.51+19.04+83.85+35+3.4+20.5) = 182.3kb

2.8 User interface rationale

For the website I have planned, the website would have a professional look so that it can attracts customer. All of the pages should have a similar layout and colour scheme, I have decided to use colours such as orange to make the website pop and attract clients, for the navigation bar and the footer I will use a dark shade of grey. So these two colours are the main colours throughout the website. The main goals in terms of how the website will look is to make it look simplistic so that it looks modern and at the same time professional.

In every page there is a header which will tell the user what the page is about, this header will have a larger font and will be positioned in the center
For the homepage and contact page the pages will be split in half with different information on each sides, these will make the pages look more appealing to the eye. To display the mentors and the sessions available to book I will make them into cards and will be displayed in their respective pages. This time the main colour will be the shade of grey mentioned previously with accents of orange.

The different forms throughout the website will only vary in the fields, otherwise they will all look the same. The forms will have a similar looking button with orange as the main colour, and chocolate as the secondary colour when the user clicks the button. The forms will all be positioned at the center of the page as this will make the page look more beautiful. Messages in case of eros will be positioned just below the form and will also be positioned in the center so that it makes it easier to catch the users attention.

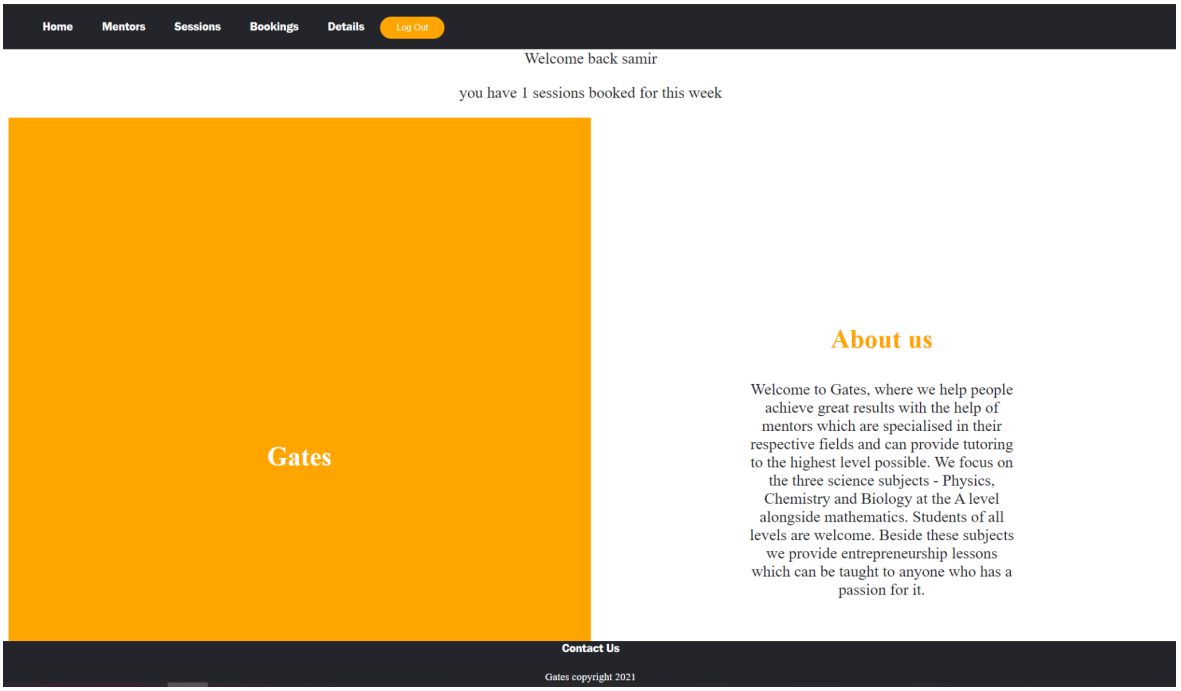
There will be various tables throughout the many pages of the website, for example a bookings table, mentors table, clients table and etc. all these tables should look the same. I am planning on them having a simplistic design, which goes with the rest of the website. The tables will also be positioned at the center so that they catch the users attention easily and will have orange accents. Inside the tables there will be buttons, such as the edit or remove buttons, these buttons will have colours which will make them stand out.

Every links inside the pages beside links on the navigation bar and footer will be in orange so the user can differentiate from normal text to the links. Buttons which are used to navigate to different pages will be positioned in the center and will span to the whole width of the page. These buttons will also be orange.

2.9 UI sample of planned data capture and entry design &

2.10 UI sample of planned output design

1. Design for homepage



Description

This is the page where the user is first redirected to when they search for the website. It displays the main information about Gates, and if the user is logged in it displays a welcome message and message informing the user if they have any sessions booked for the following week.

Variables

Variable name	**Description**
ID	The id of the user
usertype	The user type. Eg: Mentor
restriction	Creates a restriction used to get bookings upto a week from current date
getname	Query used to get the name if user
getbookings	Query used to get the bookings of user with the restriction

sessionnum	The number of bookings the user has for the following week

Pseudocode

Starts session

IF user is logged in THEN
ID := SESSION['ID']
usertype:= SESSION['UserType']

IF usertype=Client THEN

Query database to get name according to ID from clients table
Display welcome message
Creates restriction
Query to database to get bookings according to ID and restriction

ELSE

Query database to get name according to ID from mentors table
Display welcome message
Creates restriction
Query to database to get bookings according to ID and restriction

ENDIF

Get number of bookings
Display message with number of bookings

ENDIF

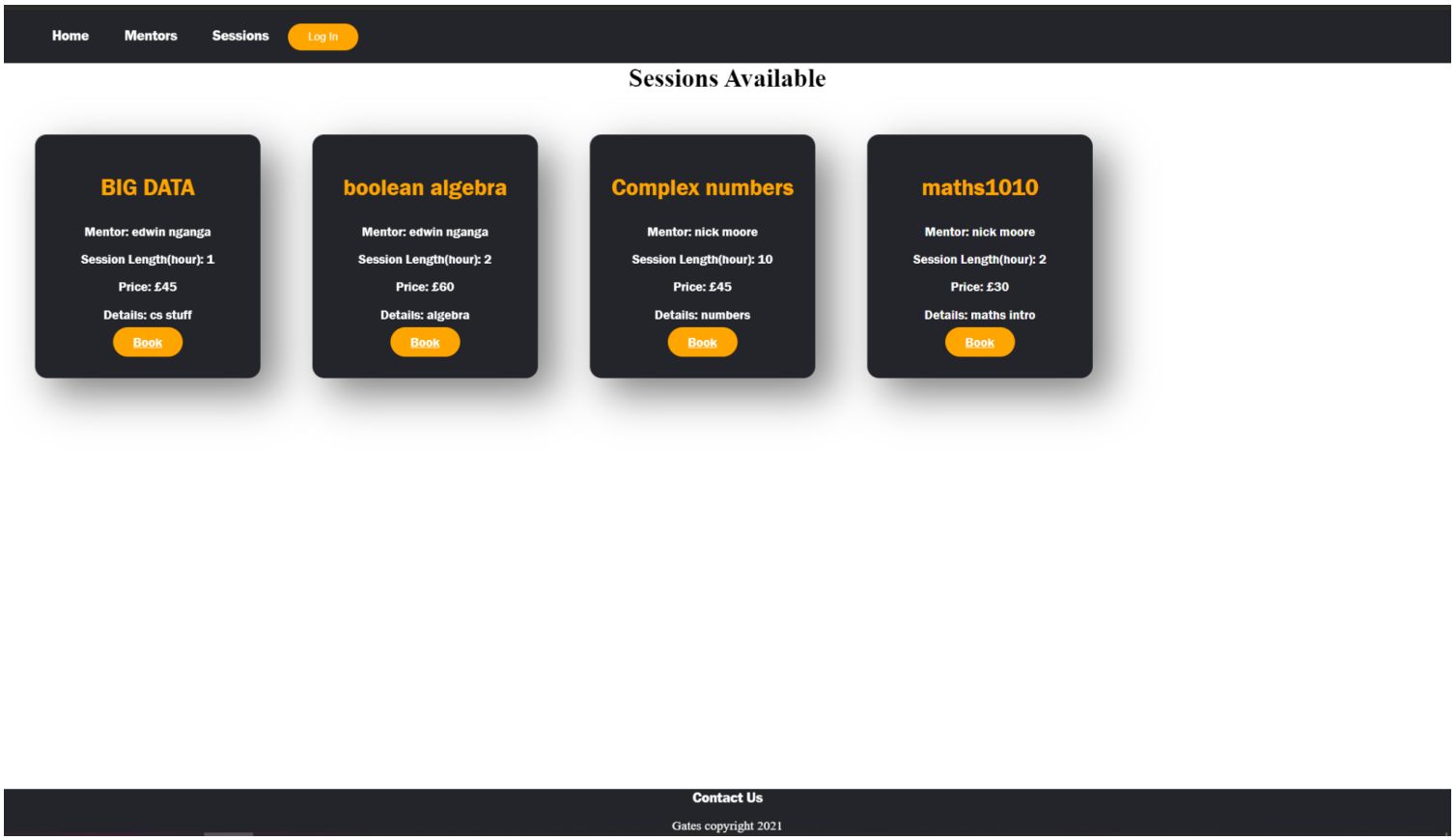
SQL

```
SELECT CForename FROM TClient WHERE ClientID = $ID
SELECT MForename FROM TMentor WHERE MentorID = $ID

SELECT * FROM TAppointment WHERE ClientID = $ID AND Date < '".$restriction."'
SELECT * FROM TAppointment WHERE MentorID = $ID AND Date < '".$restriction."'
```

Description of the SQL

The first 2 queries select the name of the client or mentor using the their ID from the clients or mentors table respectively
The last 2 queries select all the bookings within a week of the current date and based on the id.

2. Design for the display sessions page



Description

This page displays all of the sessions available to book. There is a book button that redirects the user to a booking form if they are logged in and redirects them to the login page if they aren't.

Variables

Variable name	Description
usertype	The usertype of the user
sql	Query to database to get the sessions data
result	Result from executing the query

Pseudocode

Starts session

IF user is logged in THEN

usertype:= SESSION[‘UserType’]

ENDIF

Query database to get session data

IF Query return more than 0 values THEN

Display sessions

ELSE

Output “No sessions found”
ENDIF

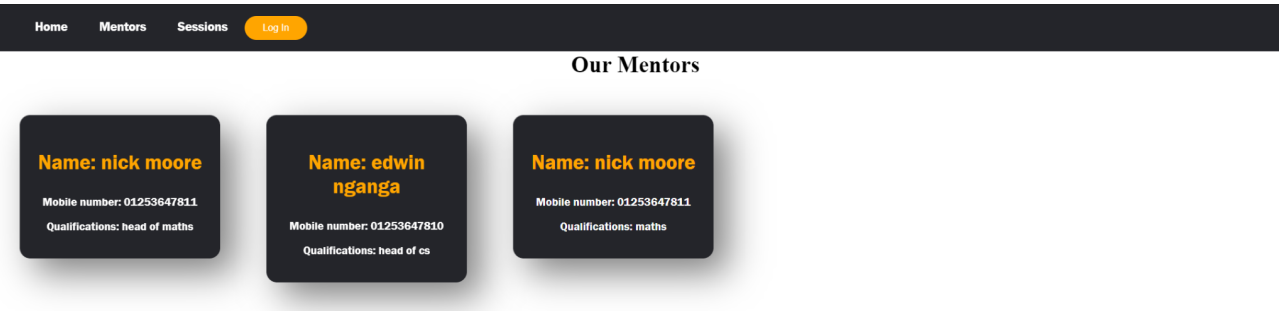
SQL

```
SELECT * , MForename, MSurname FROM TSession INNER JOIN TMentor where
TSession.MentorID = TMentor.MentorID
```

Description of the SQL

Get all the sessions data and the mentors’ name from the sessions table and mentor table correspondingly according to the mentorID.

3. Design for display mentors page



Description

This page displays all of the mentors at gates.

Variables

Variables	Description
sql	Query database to select the mentors data
result	The result from executing the query

Pseudocode

Query database to select the mentors data
IF query returns more than 0 values THEN
  Display the mentors details
ELSE
   Output “no mentors found”
ENDIF

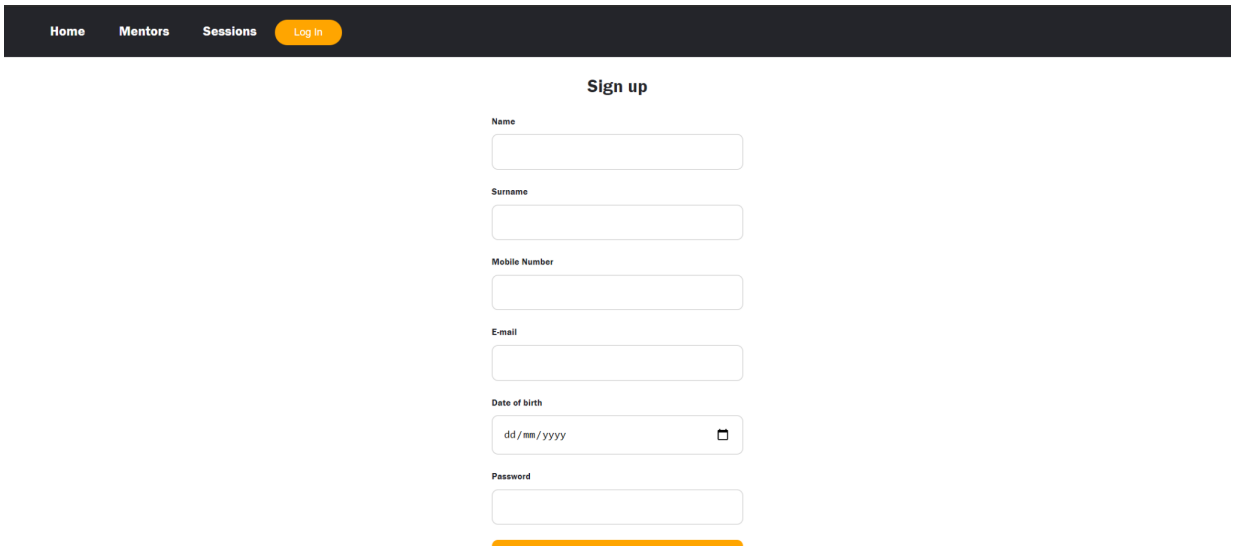
SQL

```
SELECT * FROM TMentor
```

Description

Selects all the mentors data on the database.

4. Design for the signup



Description of sign up and pseudo code

The Clients sign up starts by checking if the user clicks the sign up button.
Then it checks if every field has been entered by a presence check, if that's the case the data gets validated. There is a check to make sure the Forename and Surname contains only letters. There is check to make sure that the mobile number contain only numbers and that it is 11 characters long. The email entered is checked against other emails on the database through a query which checks if the email already exists. The email is also validated using an inbuilt function. The date of birth is checked against a restriction to make sure that the user is old enough to have an account. All the error messages are stored in an array which is used to display the error if there are any.
If there are no errors entered by the user personal details are entered into the TClient table and login details are entered into the TLogin table.

Variables

Variable name	Description
Error	Array which stores error messages
Forename	The forename the user has entered in the form
Surname	The surname the user has entered in their form
MobileNumber	The mobile number the user has entered in the form
Email	The email the user has entered in the form which will be needed to log in.
DateOfB	Establishes the connection to the database
Password	Hashed version of the password the user entered in the form
Emailcheck	Sql query to check if the email entered by the user already exists on the database
resultemail	Store all of the data fetched from the databases from the Emailcheck query
Agerequired	A date which acts as restriction for users below a certain age
sql	Contains the SQL statement that will be used to query the database
result	Store all of the data fetched from the databases
clientID	The ClientID after registering.

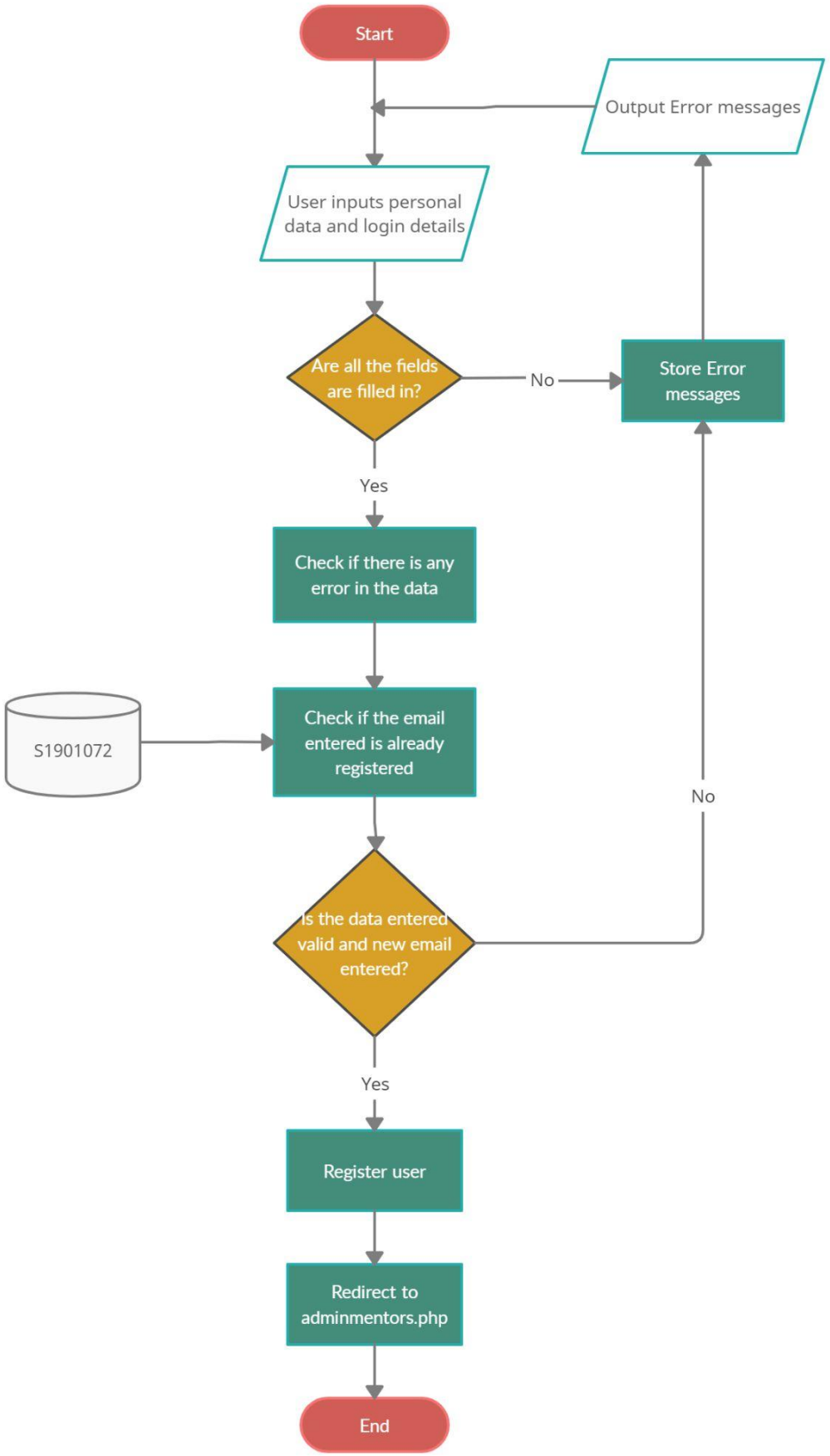
con	This variable is used to establish a connection between the web page and the database.

Pseudocode for signup

```
IF user click “Signup” THEN

    IF Forename  OR Surname  OR MobileNumber OR Email OR DateOFB OR Password is empty THEN
        Error := “ Please fill all of the fields”

    ELSE

        IF Forename is not Letters THEN
          Error[CForename] := “Forename should only contain letters”
        ENDIF

        IF Surname is not Letters THEN
          Error[CSurname] := “Surname should only contain letters”
        ENDIF

        IF MobileNumber is not Numbers THEN
          Error[CMobileNumber] := “Mobile number should only contain number”
        ELSEIF MobileNumber is not 11 characters long THEN
          Error[CMobilenumlength] :=”Mobile Number should be 11 characters”
        ENDIF
        Query To check if the email entered already exists on the system

        IF Query returns a result THEN
          Error[Emailexist] :=”You are already registered with us”
        ENDIF

        IF Email does not validate THEN
          Error[WrongEmail] :=”There seems to be something wrong with you email”
        ENDIF

        IF DateOfB> Age required THEN
          Error[DateOfB] :=”You are too young to have an account”
       ENDIF
       IF Password doesn’t have at least 1 Uppercase letter, 1 lowercase letter, 1 number and a special character THEN
         Error[Password] :=”Password need to have at least 1 Uppercase letter, 1 lowercase letter, 1 number and a special
         character”
       ENDIF
     IF Number of Error = 0 THEN
```

```
            Connect to the database
            Query database to insert personal details on the TClient table
            Query database to select the ClientID of the client who has just been added
            Query database to insert login details on the TLogin Table using the ClientID from previous query

          IF Query is executed successfully THEN

              Redirect User to Login page

            ELSE

              Output “Error: There was an error while registering user”

          ENDIF

        ENDIF

      ENDIF

ENDIF

  IF Number of Error > 0 THEN

  FOR Each Error as Error

        Output Error

      ENDFOR

  ENDIF
```

Flowchart for sign up



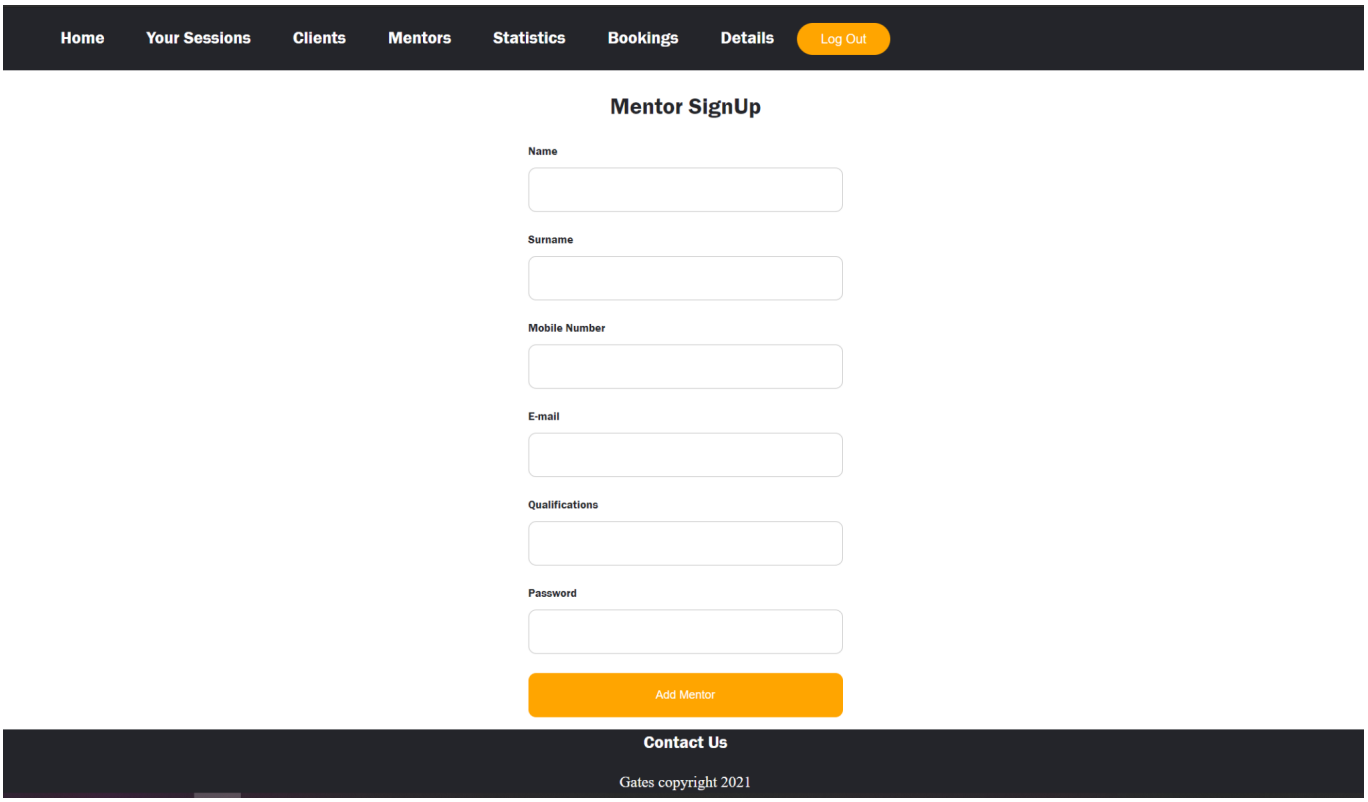
SQL

```
SELECT *FROM TLogin WHERE Email = '$Email'

INSERT INTO TClient(CForename,CSurname,CMobileNumber,DateOfB)
        VALUES('$Forename','$Surname','$MobileNumber','$DateOfB')

SELECT ClientID FROM TClient ORDER BY ClientID DESC

INSERT INTO TLogin(UserType, Email, Password, ID)
VALUES('Client','$Email','$Password', '$clientID')
```

Description of SQL

The first SELECT query select all the records from the database the have the same email entered by the user
The first INSERT INTO query store the personal data of the user in the TClient Table
The second SELECT query gets the newest record which would be the person who just signed up
The second INSERT INTO query stores the login data of the user on the TLogin table

5. Design for the mentors registration/signup



Description of the page and pseudocode

The Admin registers the mentors to the system and gives the login to their account later.
This page can be accessed only by the admin so beside them no one can register a mentor.
The signup process start when the admin clicks the Add Mentor button.
Then it checks if every field has been entered by a presence check, if that's the case the data gets validated. There is a check to make sure the Forename and Surname contains only letters. There is check to make sure that the mobile number contain only numbers and that it is 11 characters long. The email entered is checked against other email on the database through a query which checks if the email already exists. The email is also validated using an inbuilt function. The date of birth is checked against a restriction to make sure that the user is old enough to have an account. All the error messages are stored in an array.
If there are no errors entered by the admin, the personal details are entered into the TMentor table and login details are entered into the TLogin table.
If there are errors they are outputted.

Variables

Variable name	Description
Error	Array which consists of error messages
Forename	The forename of the mentor the admin has entered in the form

Surname	The surname of the mentor the admin entered in the form
MobileNumber	The mobile number of the mentor the admin has entered in the form
Email	The email of the mentor entered by the admin in the form
Qualification	The qualifications of the mentor entered by the admin
Password	Hashed version of the password the user entered in the form
Emailcheck	Sql query to check if the email entered by the user already exists on the database
resultemail	Result from the Emailcheck query
sql	Contains the SQL statement that will be used to query the database
result	Its the result from the executing the SQL query
mentorID	The mentorID after registering.
con	This variable is used to establish a connection between the web page and the database.

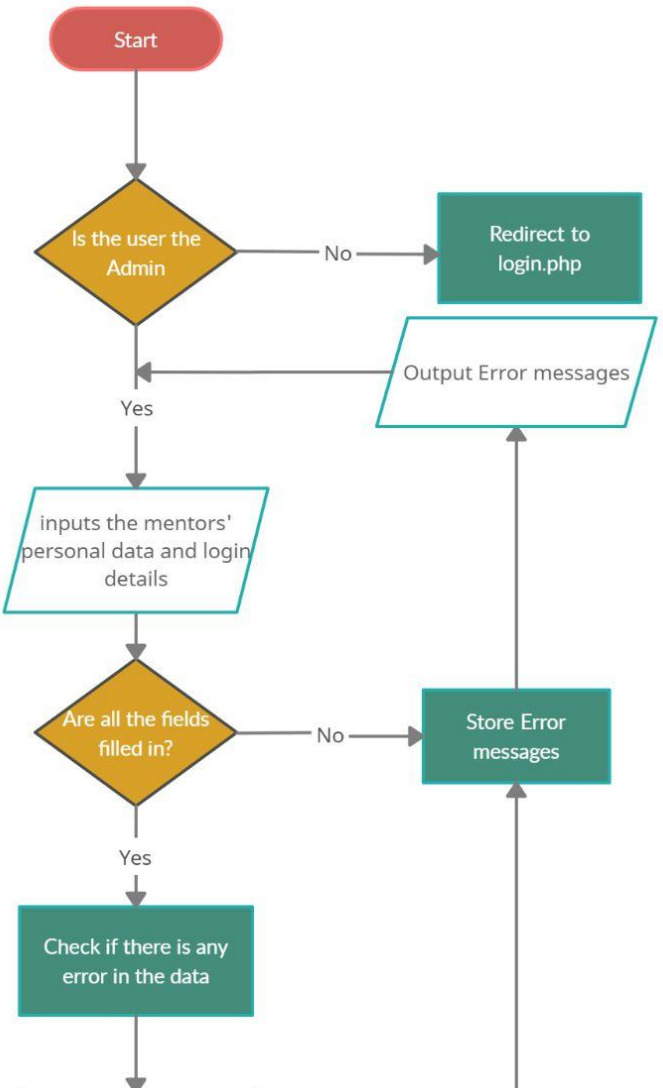
Pseudocode for mentor registration/signup

```
IF user is logged in AND user is the admin THEN

    IF user click “Signup” THEN

        IF Forename  OR Surname  OR MobileNumber OR Email OR Qualifications OR Password is empty
        THEN

                Error[‘Empty’] := “ Please fill all of the fields

        ELSE

                IF Forename is not Letters THEN

                  Error[MForename] := “Forename should only contain letters”
                ENDIF
                IF Surname is not Letters THEN

                   Error[MSurname] := “Surname should only contain letters”
                ENDIF
                IF MobileNumber is not Numbers THEN
```

```
    Error[MMobileNumber] := “Mobile number should only contain number”
  ENDIF
  ELSEIF MobileNumber is not 11 characters long THEN

    Error[MMobilenumlength] :=”Mobile Number should be 11 characters”

  Query To check if the email entered already exists on the system
  ENDIF
  IF Query returns a result THEN

    Error[Emailexist] :=”You are already registered with us”
  ENDIF
  IF Email does not validate THEN

    Error[WrongEmail] :=”There seems to be something wrong with you email”
  ENDIF
  IF Password doesn’t have at least 1 Uppercase letter, 1 lowercase letter, 1 number and a special character
  THEN
    Error[Passowrd] :=”Password need to have at least 1 Uppercase letter, 1 lowercase letter, 1 number and a
                       special  character”
  ENDIF
 IF Number of Error = 0 THEN

  Connect to the database

   Query to insert personal details on the TClient table

   Query to select the MentorID of the mentor who has just been added

   Query to insert login details on the TLogin Table with the MentorID from previous Query

   IF Query is executed successfully THEN

      Redirect User to Login page

   ELSE

      Output “Error: There was an error while registering user”
   ENDIF
  ENDIF
IF Number of Error > 0 THEN

 FOR Each Error as Error

  Output Error
```

```
        ENDFOR
    ENDIF
  ENDIF
ELSE

Redirect user to login page

ENDIF
```

Mentor Registration flowchart



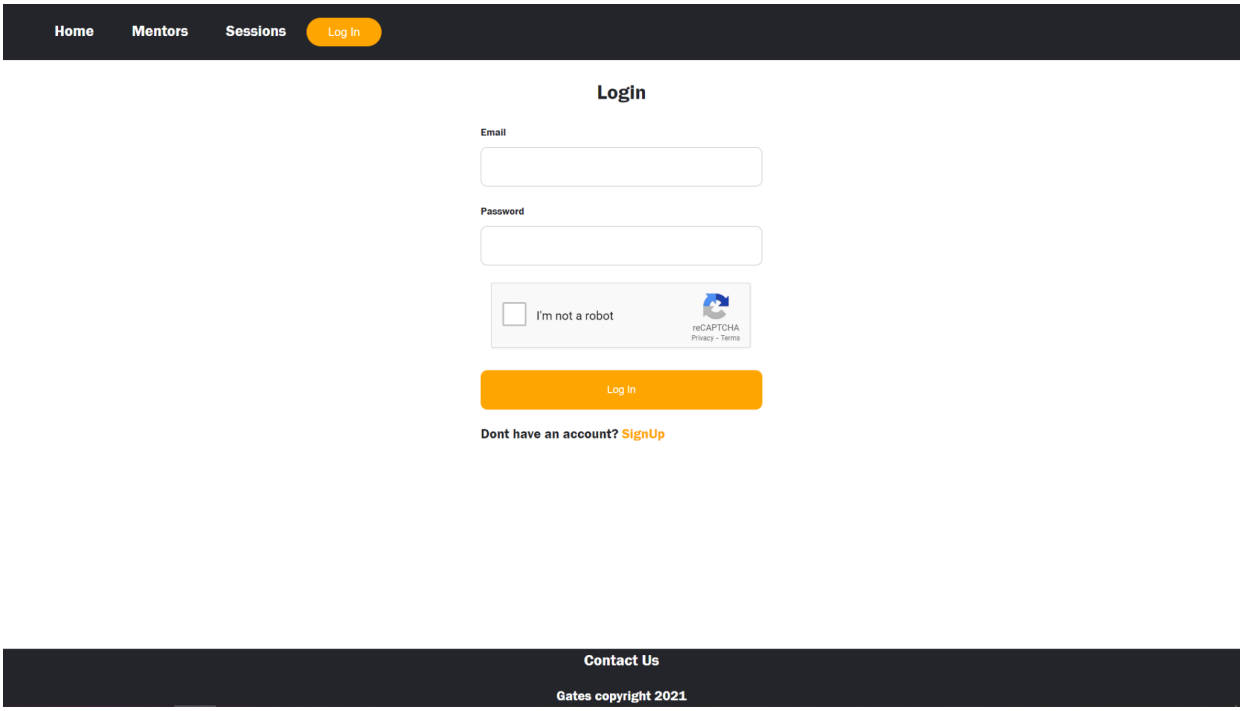
SQL

```
SELECT *FROM TLogin WHERE Email = '$Email'

INSERT INTO TMentor(MForename,MSurname,MMobileNumber,Qualifications)
            VALUES('$Forename','$Surname','$MobileNumber','$Qualifications')

SELECT MentorID FROM TMentor ORDER BY MentorID DESC"

INSERT INTO TLogin(UserType, Email, Password, ID)
VALUES('Mentor','$Email','$Password', '$mentorID')
```

Description of SQL

The first SELECT query select all the records from the database the have the same email entered by the admin
The first INSERT INTO query store the personal data of the mentor in the TMentor Table
The second SELECT query gets the newest record which would be the mentor just signed up by ordering the records with the newest one being the first one.
The second INSERT INTO query stores the login data of the mentor in the TLogin table

6. Design for login



Description

If the user is already logged in they are redirected to to the homepage.
To login the user has to enter their email address and password and pass the captcha test.
The pseudocode start with checking if the user clicked the login button. Then a presence check takes place to check if all the fields have been entered. If all fields have been entered then there is a check to see if the the user passes the captcha test. If the user passes the test the email is checked against the database to see if it exists, if it exist it compares the password on the same row from the database with the hashed password entered by the user. If the two passwords match the user is logged in and redirected to the homepage.
If there were any errors in the process they are stored in an array and then displayed.

Variables

Variable Name	Description
Errors	Array which stores error messages
SecretKey	Secret key needed for captcha
IP	Ip address for captcha
url	Url for captcha

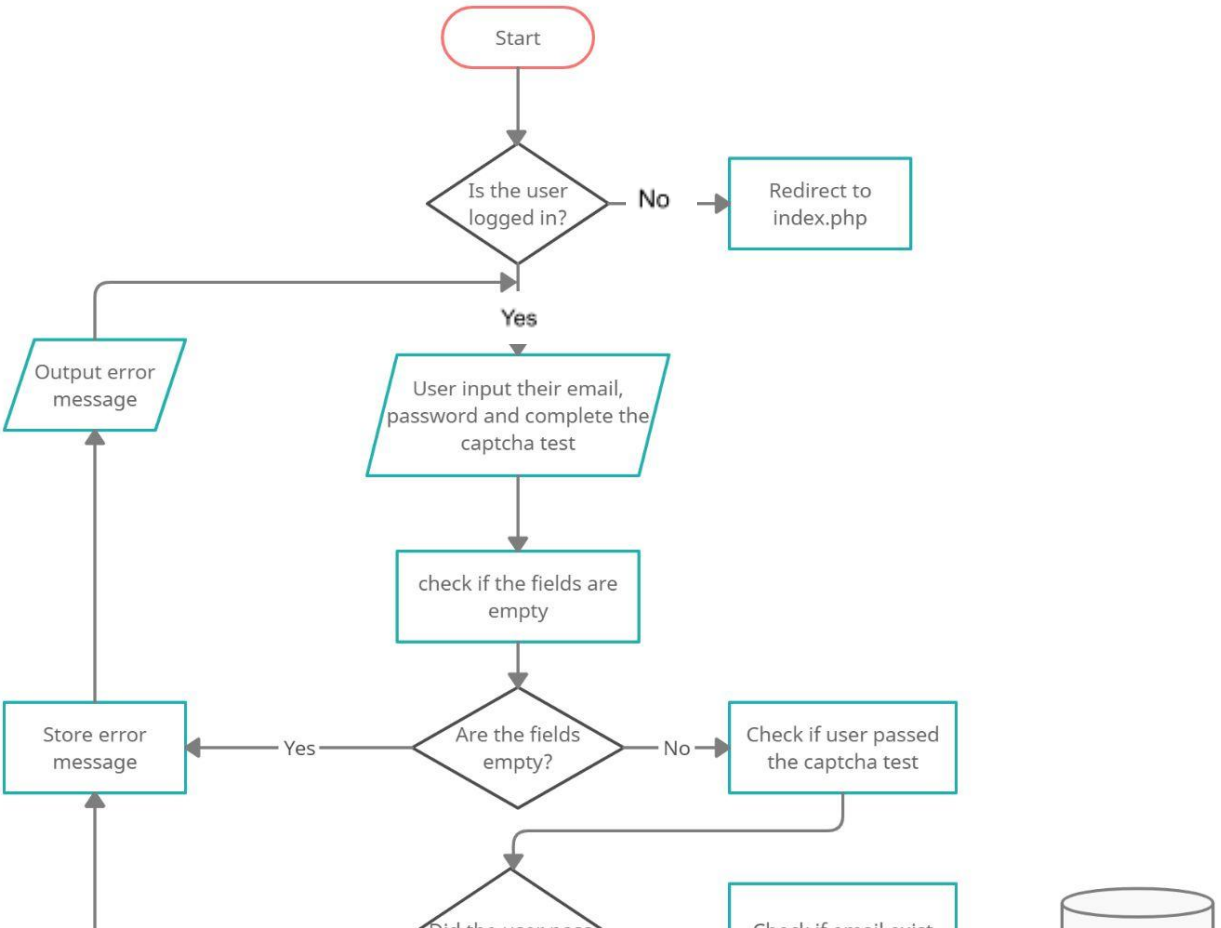
response	Decoded Result from captcha test
Email	Email entered by the user in the login form
Password	Password entered by the user in the login form
Emailcheck	Query to check if the email entered by the user exists on the database
_Session[“ID”]	Starts a session with the user’s ID
_Session[“UserType”]	Starts a session with the users’ usertype
usertype	Stores what kind of user is logged in. eg: Client, Mentor or Admin

Pseudocode for Login

```
IF user clicks the login button THEN
  IF Email field is empty or Password field is empty
    Errors[‘Empty’] := “Please fill out all the required fields”
  ELSE
    IF result from captcha is successful THEN
      Query to check if email exists on the database
      IF email exists on the database
        IF hashed password= password from database with same email entered
            Starts ID and UserType Session(logs in user)
        ELSE
          Errors[nomatch] := “Password and email do not match”
```

```
            ENDIF
        ELSE

          Errors[nonexistent] := “You are not registered with us”
        ENDIF
    ELSE

      Errors[verification] := “Please verify yourself”
   ENDIF
  ENDIF
 ENDIF
 IF Number of Errors > 0 THEN

        FOR Each Errors as Errors

          Output Errors

        ENDFOR
  ENDIF
```

Flow chart for Login



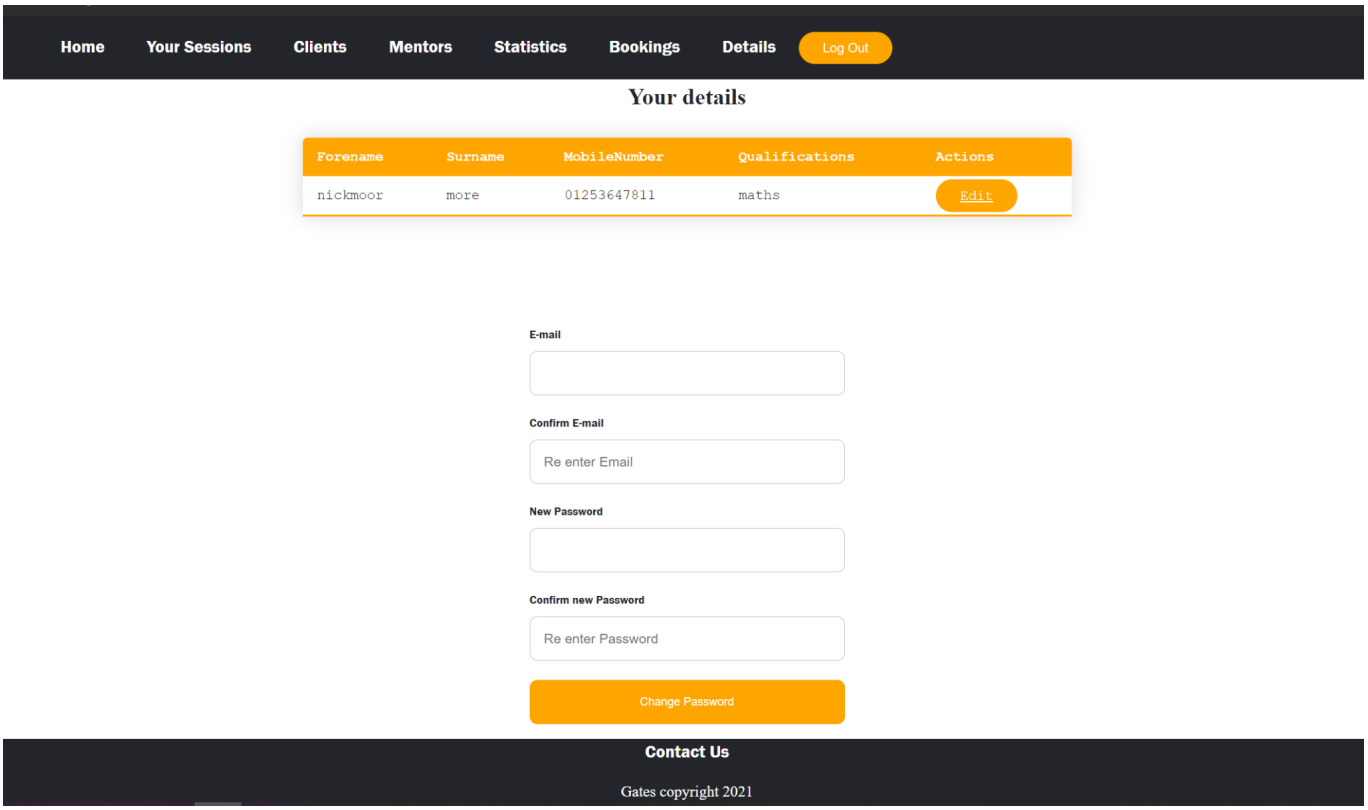
SQL

```
SELECT *FROM TLogin WHERE Email = '$Email'
```

Description OF SQL

The Query is first used to check if the email address entered in the login form exist on the TLogin table. If it exist then it is used to match the hashed version of the password the user entered with the matching password from the database on the same row of the email.

7. Design for displaying details and change password



Description

On this page the users details are displayed in a table with an edit button which will lead them to a form to change the personal details. The form below the table is to change the password.

The pseudocode starts by checking if the user is logged in or not. If its l ogged in it gets the the ID of the user and gets the details from the appropriate table and display the details in table form. For the gorm to change the password first there is a presence check to see if the user filled all of the data fields, if thats the case it then checks if the email entered is of the user logged in right now to confirm its safe. The email is checked against the confirmation email to check if they are the same and the same goes for the password. If there are any errors, error messages are stored in the Errors array and are displayed if there are any. If there are no errors then the password is hashed and the password of the user on the database gets updated.

Variables

Variable Name	Description
id	The users' ID
usertype	The type of user. Eg: Client
con	Establishes connection to the database
Error	Array to store error messages
sql	Query to database

result	Executes the query
Password	Hashed version of the password the user inputted

Pseudocode for displaying details and changing the password

```
Starts session

IF user is logged in THEN

  Establish connection to database

  IF UserType=Admin OR UserType=Mentor THEN

    Query to select mentor/admin details from TMentor table according to Mentor/Admin s ID

  Else

    Query to select details from TClient table according to the Clients ID
  ENDIF

  Execute query and display data on the table


  IF user clicks Confirm THEN

    IF Email or ConfirmEmail OR Password OR ConfirmPassword field is empty THEN

      Error['empty'] :="Please fill all of the fields"

    ELSE

      IF UserType=Client THEN

        Query to select Login details from the TLogin table according to the Clients' ID and

        UserType=Client

      ELSEIF UserType=Mentor THEN

        Query to select login details from the Tlogin table according to the mentors ID and

        UserType="Mentor"

      ELSE
```

Query to select the login details from TLogin according to the admins ID and

UserType = “Admin”

IF executing the query returns a result THEN

IF Email entered by user != Email from Query
Error[‘wrongemail’] := “Wrong Email entered”
ENDIF
IF Email != ConfirmEmail
ENDIF
Error[‘Emailnomatch’] := “Emails do not match”
ENDIF
IF Password doesn’t have at least 1 Uppercase letter, 1 lowercase letter, 1 number and a special character THEN
Error[Password] :=”Password need to have at least 1 Uppercase letter, 1 lowercase letter, 1 number and a special character”
ENDIF
IF Password := ConfirmPassword THEN

Error[‘passwordnomatch’] := “Passwords do not match”
ENDIF
IF Number of Error = 0 THEN

Hash password

IF UserType=Client THEN

Query to update password on Tlogin table according to users ID and UserType=”Client”

ELSE IF UserType=Mentor THEN

Query to update password on Tlogin table according to users ID and UserType=”Mentor”

ELSE IF UserType=Admin THEN

Query to update password on Tlogin table according to users ID and UserType=”Admin”
ENDIF

IF query is executed successfully THEN

Redirect User to homepage

ELSE

Output “There were error while updating the data”

ENDIF

ENDIF
IF Number of Errors > 0 THEN

FOR Each Errors as Errors

Output Errors

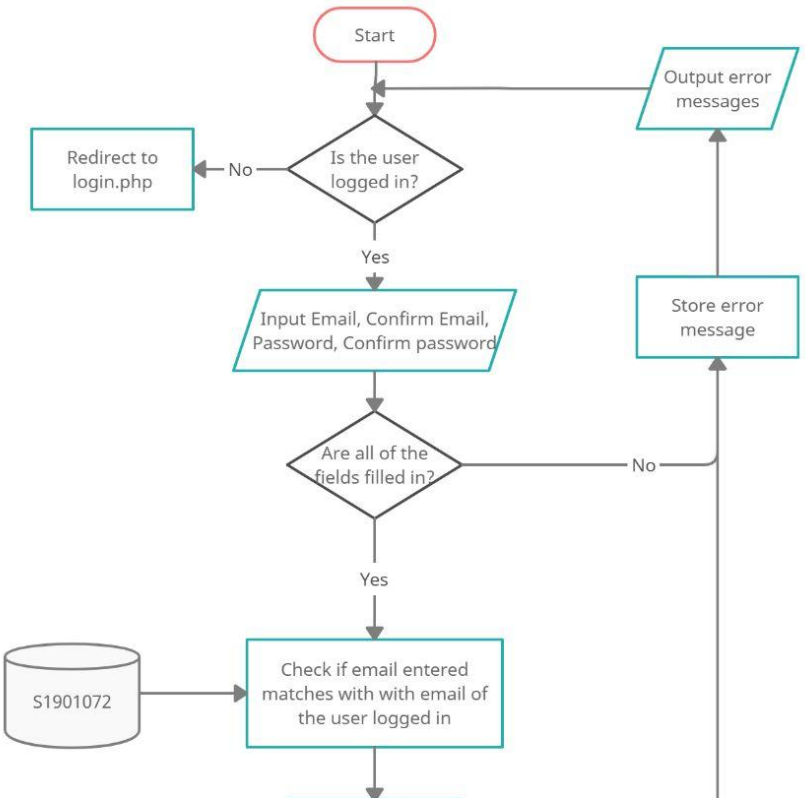
ENDFOR

ENDIF

ELSE

Redirect user to login page

Flowchart for changing the password



SQL

```
SELECT * FROM TMentor WHERE MentorID =  '$id'
SELECT * FROM TClient WHERE ClientID =  '$id'

SELECT * FROM TLogin WHERE ID =  '$id' AND UserType='Client'
SELECT * FROM TLogin WHERE ID =  '$id' AND UserType='Mentor'
SELECT * FROM TLogin WHERE ID =  '$id' AND UserType= 'Admin'

UPDATE TLogin SET Password='{$Password}' WHERE ID= '$id' AND UserType='Client'
UPDATE TLogin SET Password='{$Password}' WHERE ID= '$id' AND UserType='Mentor'
UPDATE TLogin SET Password='{$Password}' WHERE ID= '$id' AND UserType='Admin'
```

Description:

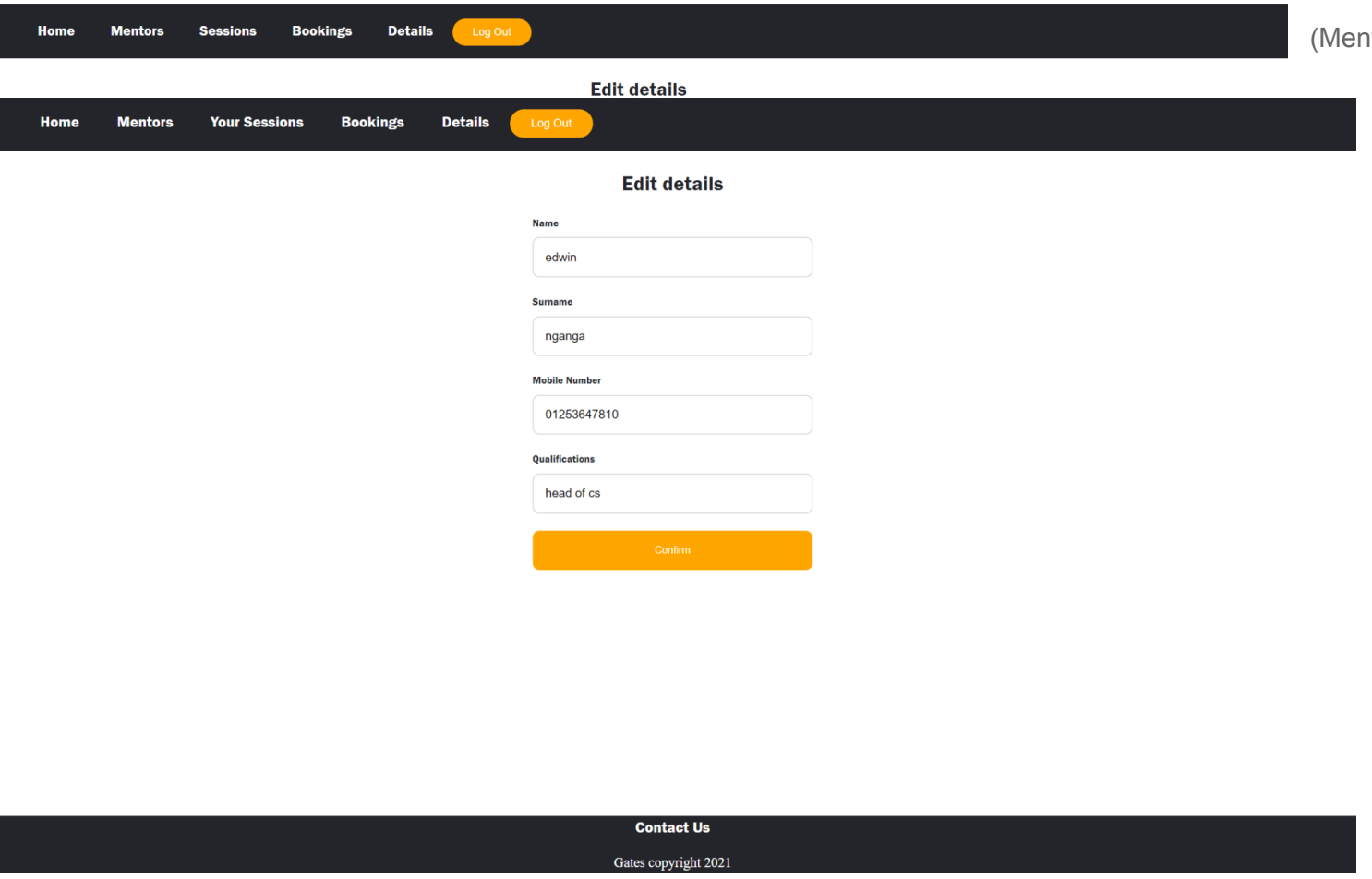
The first 2 queries are used to get the personal details of the user according their id

Then the other select queries are used to select all the login details, specifically the email of the user from the TLogin table according to the users ID and their usertype

The last 3 queries are used to change/ update the password on the the database of the user according to the their ID and Usertype.

8. Design for Edit details page

(Client)



(Mentor)

Description of edit details page and pseudo code

In this page the user can use the form to edit their personal details.
The pseudo code starts by checking if the user is logged in otherwise they are redirected to the login page. If they are logged in then it check the type of user and creates different forms filled already with the existent details. The user can edit details and click confirm to update them on the database. If the query is executed successfully they are redirected to the homepage.

Variables

Variable name	Description
id	The id of the user logged in
usertype	The type of user logged in eg:Mentor
Error	Array which stores error messages
Forename	The forename of the user
Surname	The Surname of the user
MobileNumber	The mobile number of the user
Qualifications	The qualification of the user
sql	Query to database
result	Result from execution the query

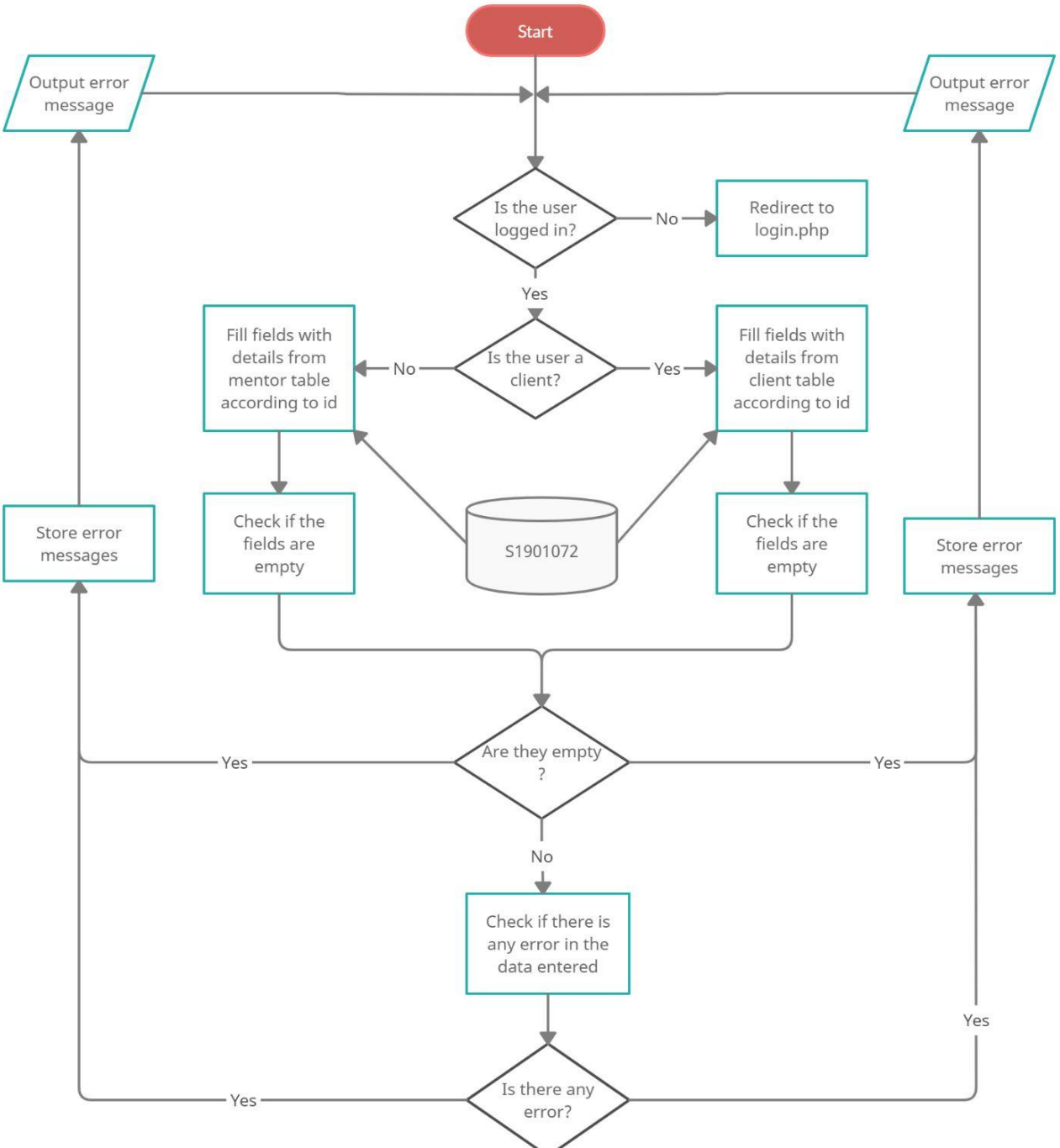
Pseudocode (Process of updating details)

```
Starts session

IF user is logged in THEN

    id := SESSION['id']
    usertype := SESSION['UserType']
    Error:= array()

    IF usertype = "Client" THEN

        IF user click "update" THEN

            IF Forename  OR Surname  OR MobileNumber is empty THEN
                Error := " Please fill all of the fields

            ELSE

                IF Forename is not Letters THEN
                  Error[CForename] := "Forename should only contain letters"
                ENDIF

                IF Surname is not Letters THEN
                   Error[CSurname] := "Surname should only contain letters"
                ENDIF
```

```
        IF MobileNumber is not Numbers THEN
          Error[CMobileNumber] := “Mobile number should only contain number”
        ELSEIF MobileNumber is not 11 characters long THEN
          Error[CMobilenumlength] :=”Mobile Number should be 11 characters”
        ENDIF

    IF Number of Error = 0 THEN

        Connect to the database
        Query database to update personal details on the TClient table according to id


        IF Query is executed successfully THEN

            Redirect User to details page

        ELSE

            Output “Error: There was an error while updating user info”

        ENDIF
  ENDIF

  IF Number of Error > 0 THEN

            FOR Each Error as Error

               Output Error

            ENDFOR

          ENDIF

ELSEIF usertype=”Mentor” OR usertype=”Admin” THEN

  IF user click “update” THEN

    IF Forename  OR Surname  OR MobileNumber OR Qualifications is empty THEN
       Error := “ Please fill all of the fields

    ELSE

        IF Forename is not Letters THEN
          Error[MForename] := “Forename should only contain letters”
        ENDIF

        IF Surname is not Letters THEN
```

```
    Error[MSurname] := “Surname should only contain letters”
  ENDIF

  IF MobileNumber is not Numbers THEN
    Error[MMobileNumber] := “Mobile number should only contain number”
  ELSEIF MobileNumber is not 11 characters long THEN
    Error[MMobilenumlength] :=”Mobile Number should be 11 characters”
  ENDIF

IF Number of Error = 0 THEN

  Connect to the database
  Query database to update personal details on the TClient table according to id


  IF Query is executed successfully THEN

     Redirect User to details page

  ELSE

     Output “Error: There was an error while updating user info”

  ENDIF
  ENDIF
  IF Number of Error > 0 THEN

       FOR Each Error as Error

          Output Error

       ENDFOR
  ENDIF


ELSE

  Redirect user to login page

ENDIF
```

Flowchart for edit details



SQL

```
UPDATE TClient SET CForename='{$Forename}', CSurname = '{$Surname}',
  CMobileNumber = '{$MobileNumber}' WHERE ClientID= '$id'

UPDATE TMentor SET MForename='{$Forename}', MSurname = '{$Surname}', MMobileNumber
= '{$MobileNumber}', Qualifications = '{$Qualifications}'
WHERE MentorID= '$id'

SELECT * FROM TClient WHERE ClientID = '$id'
SELECT * FROM TMentor WHERE MentorID = '$id'
```

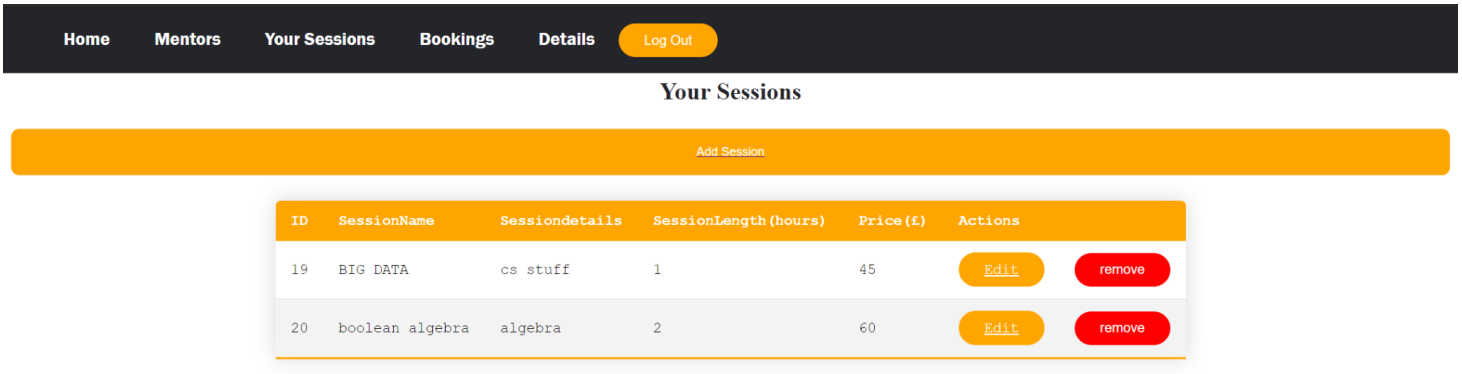
Description of the SQL

The first 2 queries update the details to corresponding id on the clients table or the mentors table depending on if the user is a Client or a Mentor or Admin

The last 2 queries select the details of the user from the database by using their id

9. Design for display mentors' own session and remove session + add session + edit session

Design for display mentors own session



Description

This is how sessions are displayed for mentors. The session displayed on the are only of that specific mentor. The mentor is allowed to edit the existing sessions, remove them and add new sessions.
The pseudocode below is for displaying the sessions and delete them. The pseudocode starts by checking if the user is logged in and that the user's user type is either mentor or the admin, otherwise they are redirected to the login page and index page accordingly.
After this the sessions are selected corresponding to the ID of mentor/Admin logged in and are displayed in a table such as in the picture above.
When the mentor/admin clicks the remove button a query is used to remove that session from the database using the corresponding SessionID.

Variables

Variable Name	Description
ID	MentorID of the mentor whos logged in
sessionid	The id of the session the mentor selected
Sql	Contains the SQL statement that will be used to query the database
con	Variable used to establish a connection to the database
result	Result from connecting to the database and executing the query
row	Specific row from results fetched by executing the query

Pseudocode for displaying sessions and removing sessions

```
Starts session

IF the user is logged in THEN

  IF the user is either an Admin or a Mentor THEN

    IF user click remove THEN

        Get according sessionid from the table

        Query to database to delete the records of the session which corresponds to the sessionid

       IF Connecting to the database and executing the query is successful

         Redirect the mentor to the own sessions page

       ENDIF
```

ENDIF

Query database to select all the sessions from the database according to the mentors ID

Connect to database and execute the query

IF there are records of sessions THEN

Output them into the table

ENDIF

ELSE

Redirect user to homepage

ENDIF

ELSE

Redirect user to login page

ENDIF

SQL

```
DELETE FROM TSession WHERE SessionID = $sessionid
```

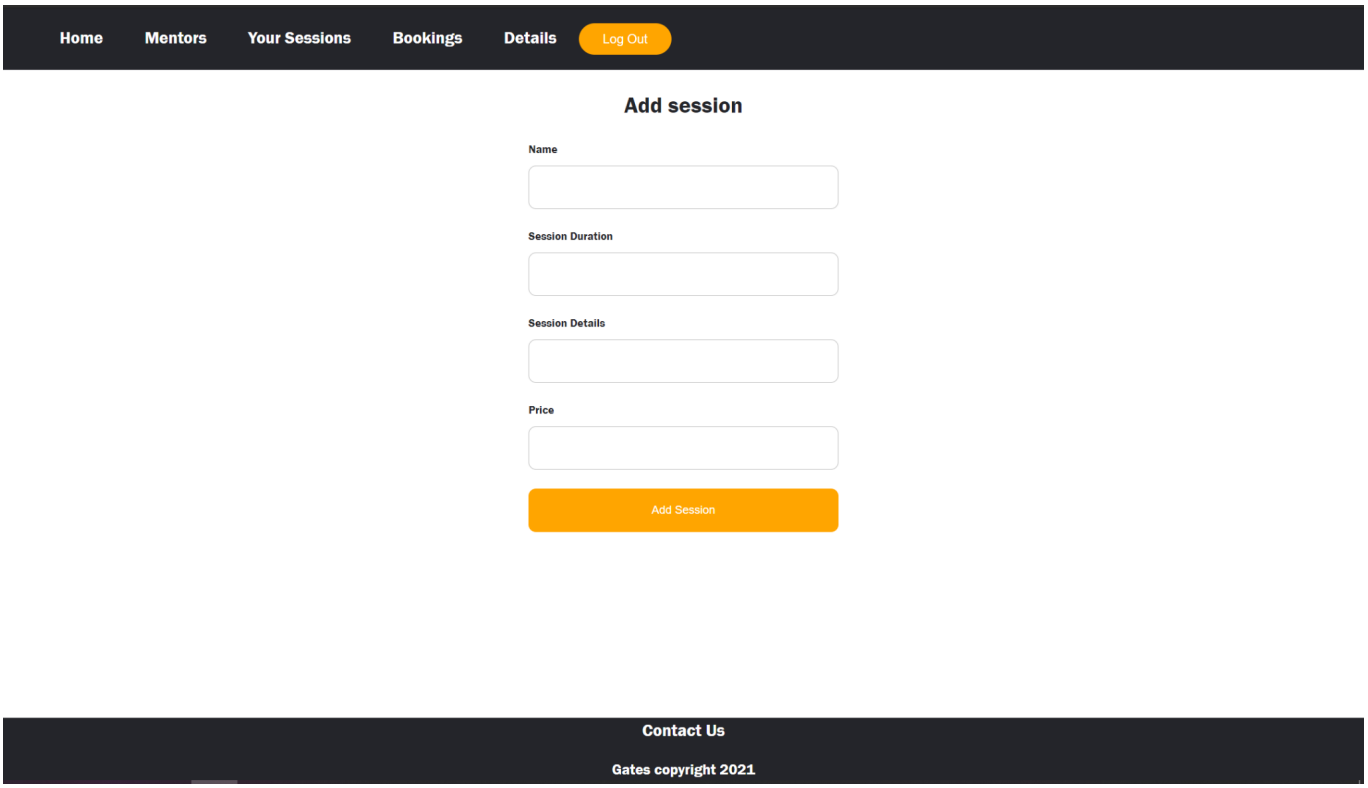
```
SELECT * FROM TSession WHERE TSession.MentorID=$ID
```

Description

The DELETE statement is used to delete the records of a session of which id's corresponds to the one the mentor had selected in the table.

The SELECT statement is used to select all the sessions records on the database that the mentor does by using the mentorID.

Design for adding session



Description of the page and pseudo code

The mentor can reach this page by clicking the the add session button from the own sessions page/Your Sessions(previous page)
As the previous page the pseudocode starts by checking if the user is logged in and check if the if the user is a mentor or admin, if thats not the case they are redirected to login page and index [age respectively.
In terms of validation, there is first a presence check to see if all the fields have been filled. Then there is a check to see if the Session Length is between 1 and 3 hours and lastly there is a check to see if only numerical characters are entered on the Price field. If there are any errors they are stored and then outputted all at once. If there are no errors the session is added.

Variables

Variable name	Description
usertype	What type the user logged in is
id	The id of the user logged in
Error	Array that stores error message

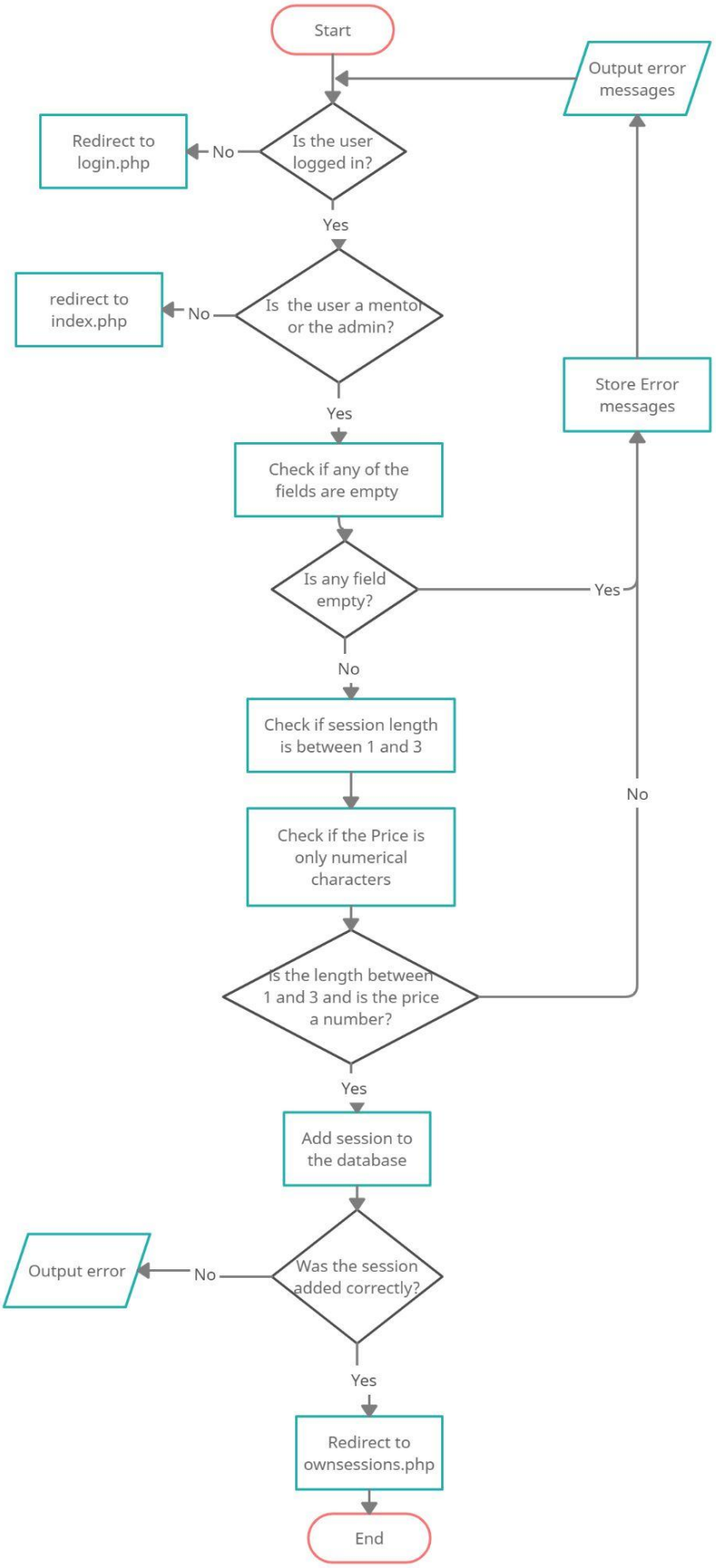
SessionName	Variable that store the name of the session entered in the form
SessionLength	Variable that store the length of the session entered in the form
Price	Variable that stores the price of the session entered in the form
Sessiondetails	Variable that stores the details of the session entered in the form
sql	Query to the database

Pseudocode

```
Starts session

IF user is logged in THEN

   usertype:= SESSION['UserType']

   IF usertype = "Admin" OR usertype="Mentor" THEN

      Id := SESSION['ID']
      Error:= array()

      IF SessionName OR SessionLength OR Price OR Sessiondetails is EMPTY THEN

         Error['Empty']:="Please fill all of the fields"

      ELSE

         IF SessionLength<1 OR SessionLength>3 THEN

            Error['Length'] := "Session should be 1,2 or 3 hours long"

         ENDIF

         IF Price is NOT only numerical characters THEN

            Error['Price'] := "Price should only be a numerical value"

         ENDIF

         IF Number of Error = 0 THEN

            Query database to add session to the session table

            IF Query is executed successfully THEN

               Redirect User to own sessions pagepage
```

```
            ELSE
                Output “Error: There was an error while updating user info”
            ENDIF
          ENDIF
        ENDIF
        IF Number of Error > 0 THEN
            FOR Each Error as Error
              Output Error
            ENDFOR
        ENDIF

    ELSE
      Redirect to homepage
    ENDIF
ELSE
  Redirect to login page
ENDIF
```

Flowchart for Add session



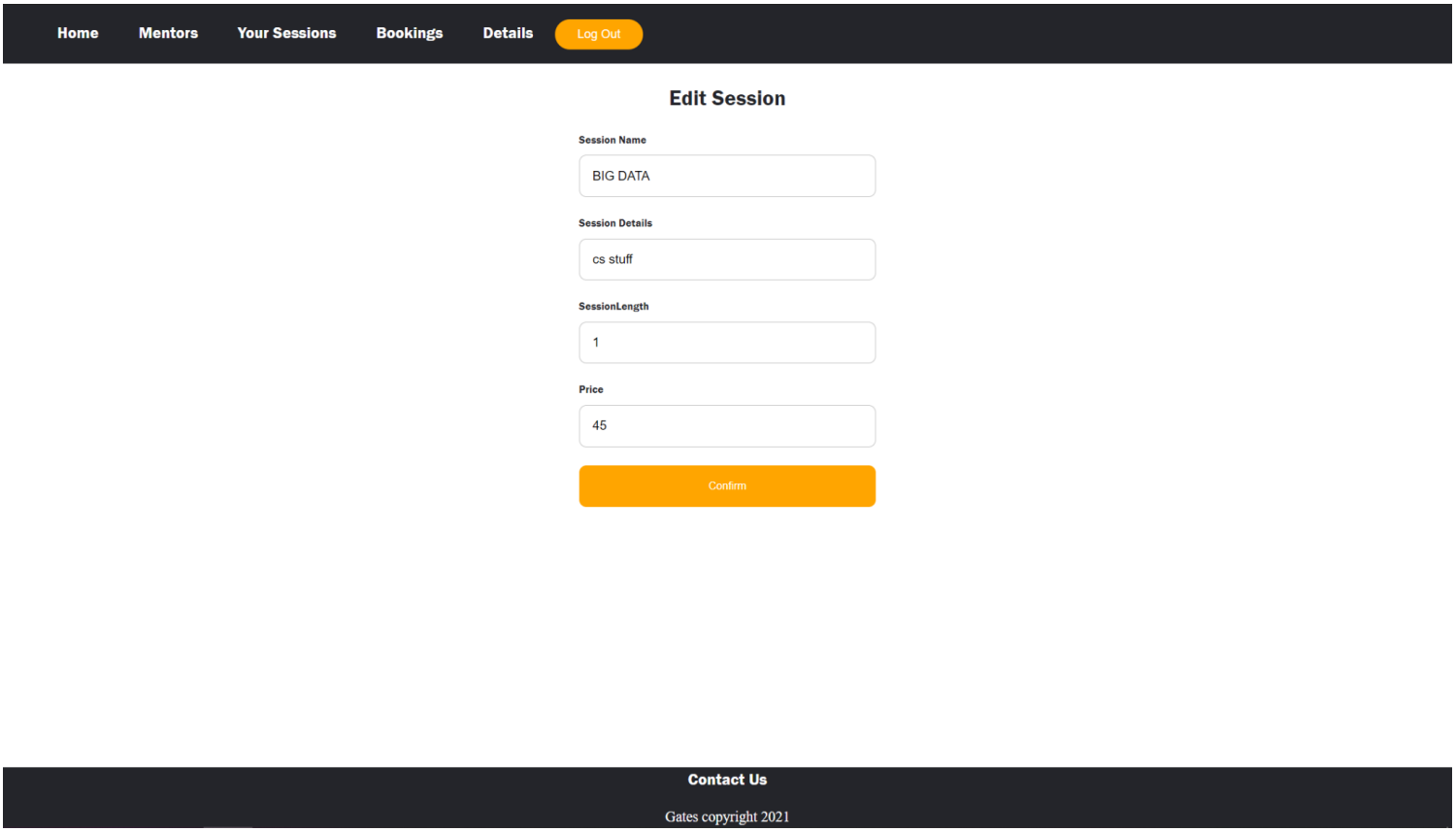
SQL

```
INSERT INTO TSession(SessionName,SessionLength,Price,Sessiondetails,MentorID)
VALUES('$SessionName','$Price','$SessionLength','$Sessiondetails','$id')
```

Description of the SQL

Insert a new session on the sessions table with the data entered on the form by the user

Design for edit session



Description of page and pseudocode

The user can use the form on this page to edit the session they had selected.
Process: It gets the session id when the edit button is clicked on the previous page. Then it gets that session's data from the database using that id and fills the fields of the form with that data. The user then changes the the data on the fields as they please. Then there is a check to see if all of the fields are empty and other checks to see if there is no error in the data entered. If there are no errors the in the data entered then the data of that specific session on the database is updated and if there are errors they are displayed to the user.

Variables

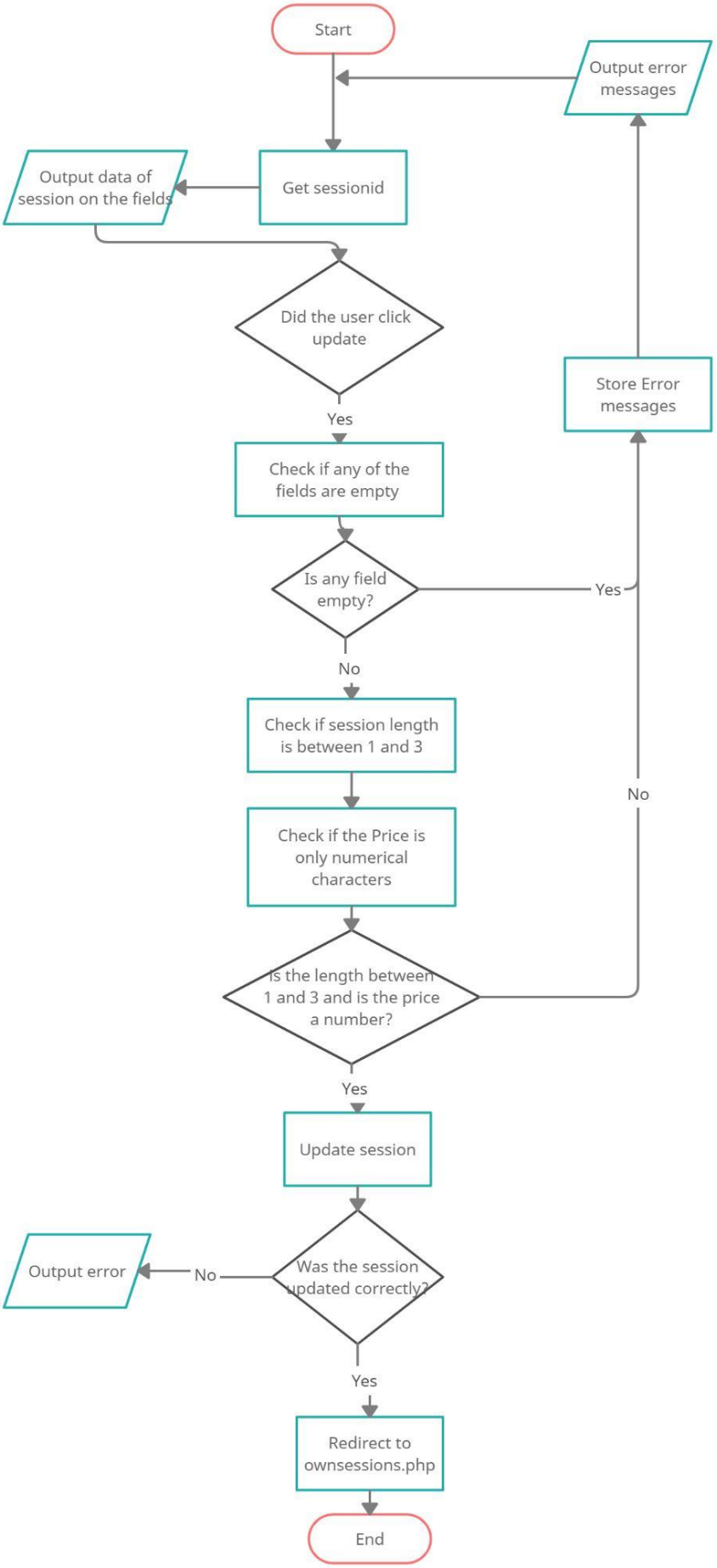
Variable name	Description
id	The id of the session selected
Error	Array that stores error message
SessionName	Variable that store the name of the session entered in the form
SessionLength	Variable that store the length of the session entered in the form
Price	Variable that stores the price of the session entered in the form
Sessiondetails	Variable that stores the details of the session entered in the form
sql	Query to the database
result	Result from executing the query

Pseudocode(Process)

```
Starts session

Error =array()

IF user clicks update THEN

    IF SessionName OR SessionLength OR Price OR Sessiondetails is EMPTY THEN

        Error['Empty']:="Please fill all of the fields"

     ELSE

        IF SessionLength<1 OR SessionLength>3 THEN

            Error['Length'] := "Session should be 1,2 or 3 hours long"

        ENDIF

        IF Price is NOT only numerical characters THEN

           Error['Price'] := "Price should only be a numerical value"

        ENDIF

        IF Number of Error = 0 THEN
```

```
            Query database to edit session according to the sessionid

            IF Query is executed successfully THEN

                Redirect User to own sessions pagepage

             ELSE

                 Output “Error: There was an error while updating user info”

            ENDIF

          ENDIF


    ENDIF

ENDIF

Get the id of the session selected
Query database to get session data corresponding to the id
Fill the fields with the session data

IF Number of Error > 0 THEN

            FOR Each Error as Error

              Output Error

            ENDFOR

ENDIF
```

Flowchart for edit session(Process)



SQL

```
UPDATE TSession SET SessionName='{$SessionName}', SessionLength =
'{$SessionLength}', Price = '{$Price}', Sessiondetails = '{$Sessiondetails}'
WHERE SessionID= " . $_POST['SessionID']

SELECT * FROM TSession WHERE SessionID={$id}
```

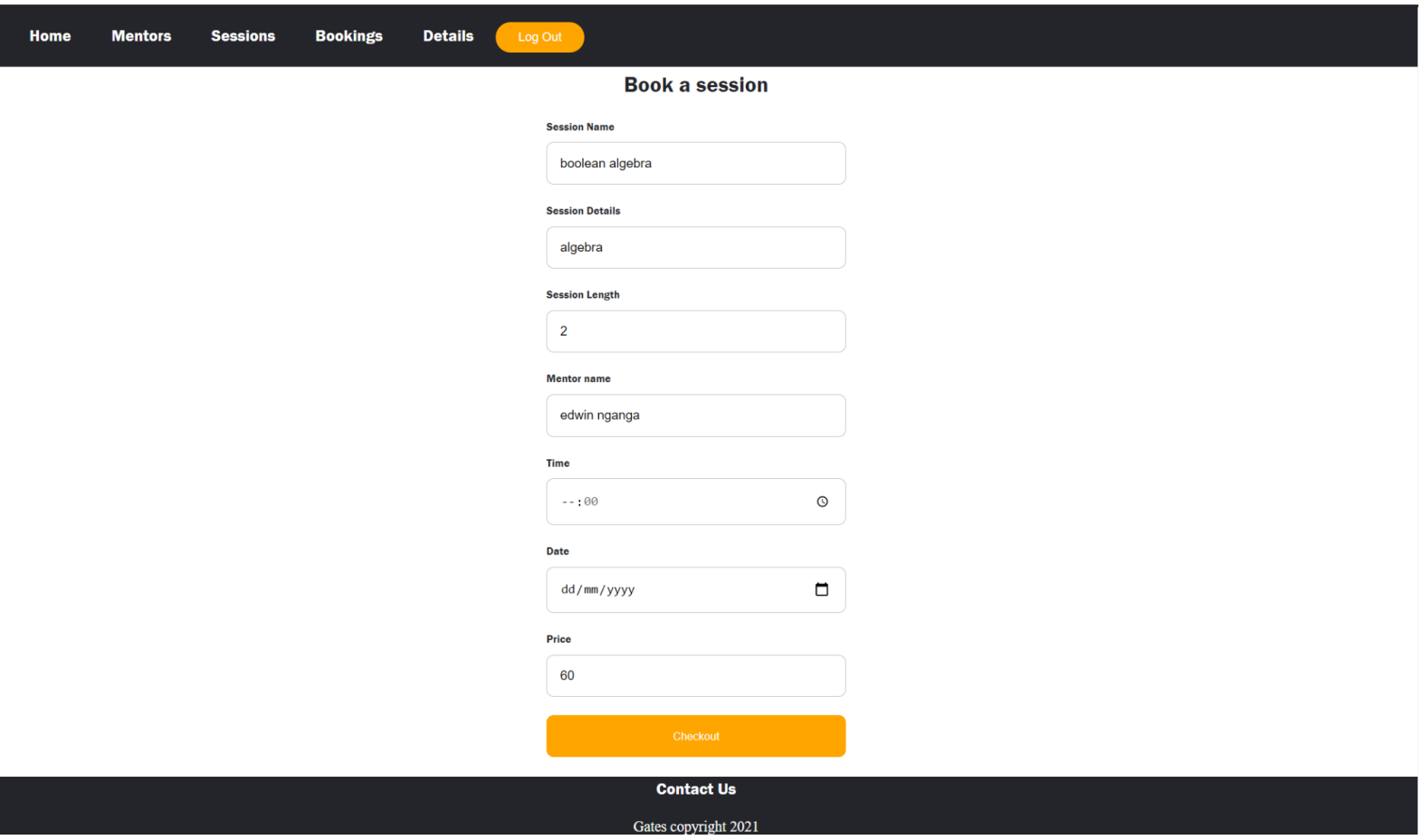
Description of the SQL

The first query updates the session with the data entered in the form by the user.

The 2nd query gets the session data from the database using the session id fetched from the previous page when the edit button was clicked.

10. Design for book session, Display booking + cancel booking and edit booking

Design for book session



Description

The user is redirected to the booking page if they are logged in and click the book button of a session from the session page. The form is filled with all the session data and the client choose the time and date for thor booking. Afterwards they are redirected to a checkout page where if the payments is successful their booking/appointments is added to the database.
Pseudocode: First it checks if the user clicked the Checkout button, if so it checks if the time and date field are empty, if they arent then there are various checks such check to see if the client booked a future date, or that the did not book on a Saturday or Sunday. There are also 4 different checks for double booking. Two which check on the clients side to see if they have another booking which collide with the one they are trying to book and two for the mentors side. If there are no errors then the booking data are passed to a checkout session where they will be used to create the appointment if the payment is successful, if there are errors then the error messages are displayed to the client.

Variables

Variable Name	**Description**
ClientID	The id of the client logged in
Errors	Array which stores error messages
Price	The price of the session the client selected
MentorID	The ID of the mentor of the session the client selected
SessionID	The ID of the session the client selected
Name	The name of the session the client selected
Time	The time entered by the client on the booking form
Date	The date entered by the client on the booking form
Length	The length of the session the client selected
session	Creates a session for the checkout
day	Variable that stores the day of the date the client entered on the booking form
Duration	Calculates the time it takes for the session
Endtime	Use the duration and stores the time the session will end
query, query1, query2, query3	Query database to check for double booking
query_result, query_result1, query_result2, query_result3	Results from the queries used to check for double booking

restriction	Stores the current date
restriction_time	Stores the current time

Pseudocode for booking a session

```
Starts session

ClientID:= SESSION['ID']
Errors:= array()

IF client clicked the checkout button THEN

  Create variables storing the booking data
  Create checkout session and pass the name and price of the sesion

  IF Time field is empty or Date field is empty THEN

    Errors['Empty'] := "Please fill all of the fields"

  ELSE

    IF day of booking is Saturday OR Sunday THEN

      Errors['Weekend'] :="Can not book during the weekend"

    ENDIF

    IF Length of session = 1 THEN

      Duration := Time user selected + 60 minutes

    ELSEIF Length of session = 2 THEN

        Duration := Time user selected + 120 minutes

    ELSE

        Duration := Time user selected + 180 minutes

    ENDIF

    Set the Endtime for when the session will end


    Query database multiple times with different conditions to check for double booking
    IF there is more than 0 results THEN
```

Errors[‘dbookingM’] :=”This Mentor is not free at this time”
Or Errors[‘dbookingC’] :=”You already have another session booked at this time”
(Note: Depending on which query returns a result greater than 0 the error message is stored if the same client has another session with the same mentor then both error messages are displayed)

ENDIF

IF Date entered is a past date THEN

Errors[‘date’] := “Error: Please book a date in the future”

ENDIF

IF date entered is the same as restriction(Current date) and the time selected is past the current time THEN

Errors[‘time’] := “Error: Please book later in the day”

ENDIF

IF Number of errors = 0 THEN

Create sessions for data which is going to be used for adding the appointment to database

ENDIF

ENDIF

ENDIF

Get the id of the session selected
Query database to get session data and the mentor name from the sessions table and mentors tables accordingly corresponding to the id
Fill all of the fields beside the Time and Date field

IF Number of Error > 0 THEN

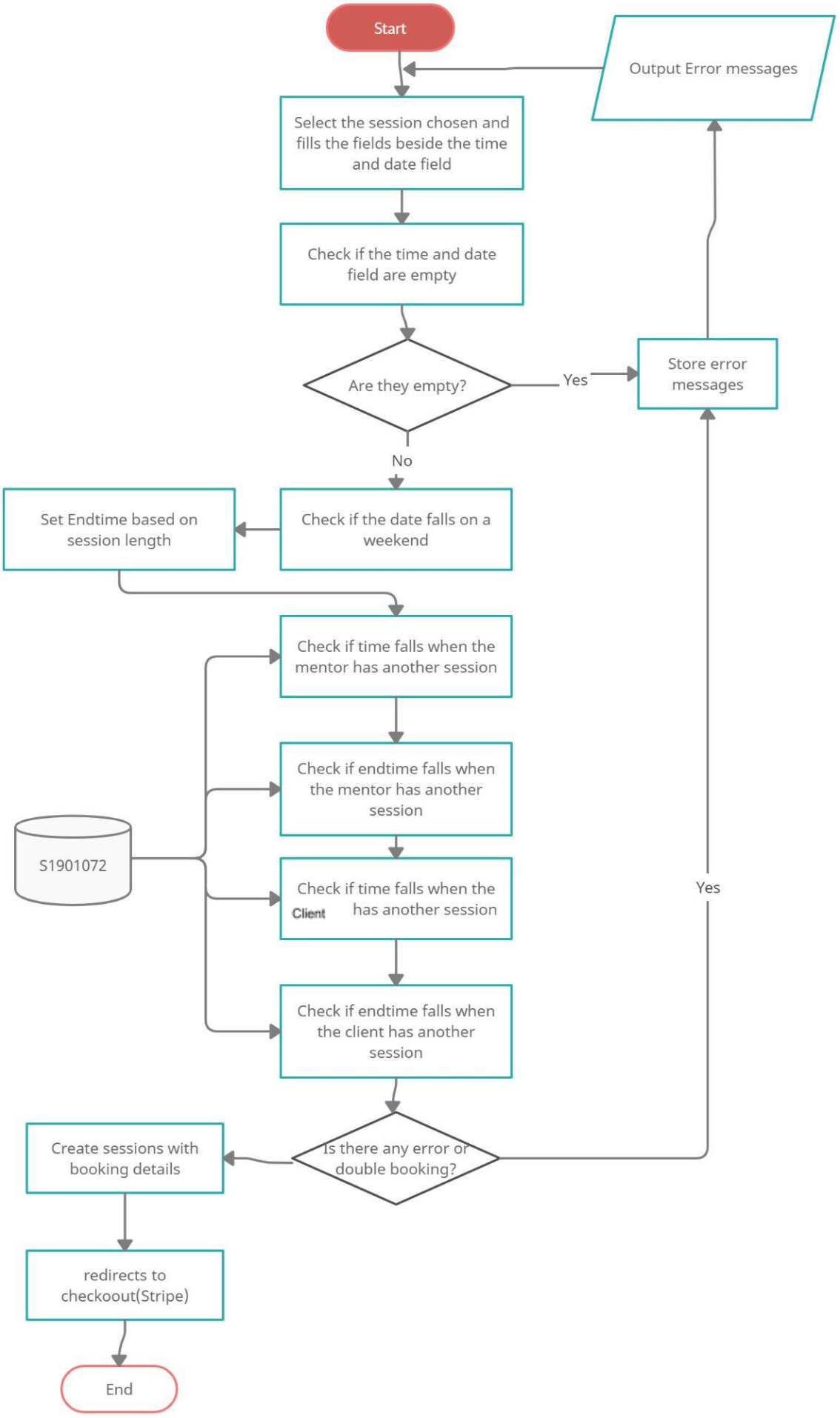
FOR Each Error as Error

Output Error

ENDFOR

ENDIF

Flowchart for booking a session



SQL

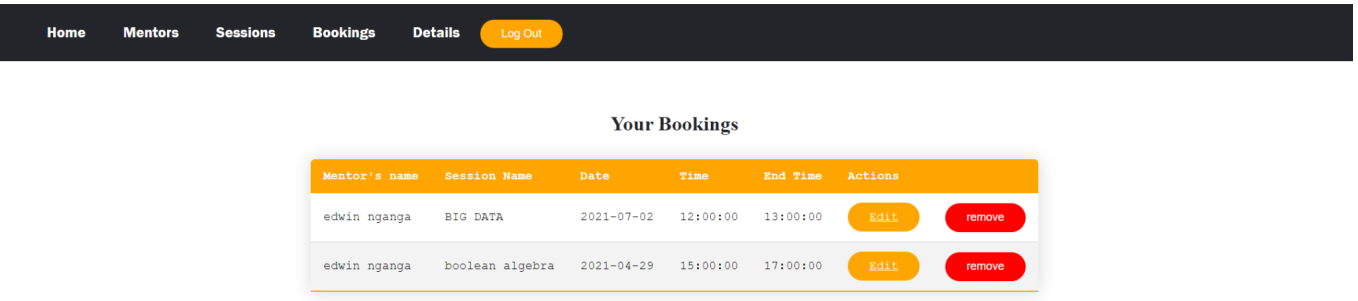
```
SELECT COUNT(*) AS COUNT FROM Tappointment WHERE Date = '".$Date."' And MentorID =
'".$MentorID."' AND Time<='".$Time."' and Endtime>'".$Time."'

SELECT COUNT(*) AS COUNT FROM Tappointment WHERE Date = '".$Date."'
And MentorID = '".$MentorID."' AND Time<'".$Endtime."' and Endtime=>'".$Endtime."'

SELECT COUNT(*) AS COUNT FROM Tappointment WHERE Date = '".$Date."'
And ClientID = '".$ClientID."' AND Time<='".$Time."' and Endtime>'".$Time."'

SELECT COUNT(*) AS COUNT FROM Tappointment WHERE Date = '".$Date."'
And ClientID = '".$ClientID."' AND Time<'".$Endtime."' and Endtime=>'".$Endtime."'

SELECT *, MForename, MSurname FROM TSession JOIN TMentor WHERE
TSession.SessionID={$_SESSION['bookID']} and TSession.MentorID = TMentor.MentorID
```

Description of the SQL

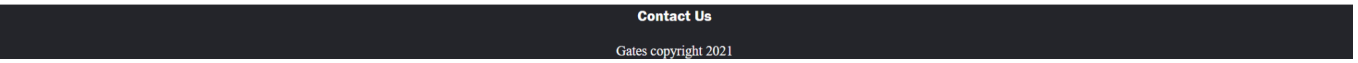
The first 4 queries essentially check for double booking, the first query check if the mentor has a session which collides with the start time of the session the client entered and the second query check if thats the case with the endtime of the session the client is trying to book.
The third and fourth query does the same thing as the first and second query respectively however this time the restriction is on the client instead of the mentor.

The last query is used to get the session data and the name of the mentor which teach that sessions using the sessionid fetched from the previous page where the client selected the session by clicking the book button.

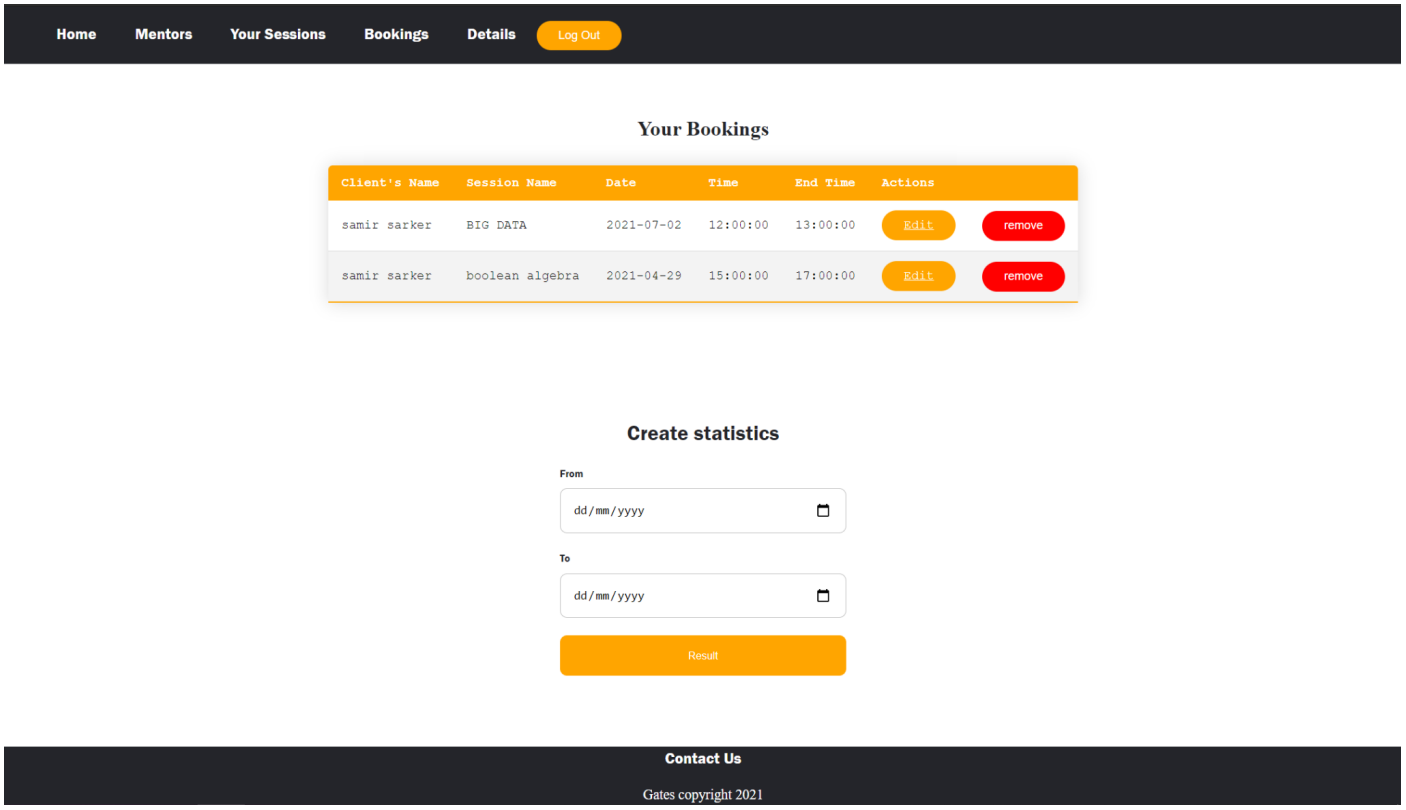
Design for displaying the bookings



(Clients)



(Mentors)



Description of the page and pseudocode

On this page all of the bookings which the user has in the future are displayed. There is an edit button which will redirect the user to the edit booking page where they will able alter their bookings and a remove button if they want to cancel their booking.
If the user is a mentor or the admin then there is a form which can be used to generate statistics on bookings they had in the past.
Pseudocode: Starts by moving all of the old appointments to the old appointments table and then deleting them from the appointments table.
If the user is not logged in they are redirected to the login page.
If the user click the remove button the session is only removed if it is 1 day in advance
Then it checks if the user is a client or the admin or mentors and gets the booking data and the mentor s name or clients name accordingly.
If the user is the admin or a mentor it displays the statistics form which they can use to create the statistics
There is a presence check, a check to see if the to date is not before the from date and another check to see if they are both past dates.
If there are no errors then statistics are calculated and displayed to them otherwise the error messages are displayed.
I have divided the pseudocode in two parts, the first part shows the pseudocode for view bookings and deleting them, the second part of the pseudocode shows how the statistics are created.

Variables(For both the view bookings and create statistics part)

Variable Name	Description
Errors	Array which stores error messages
Error	Array which stores error messages
today	Stores the current date
insertrecord	Query database to move older appointments to another table
deleterecord	Query database to delete older appointments
usertype	The usertype of the user logged in
ID	The id of the user logged in
appointmentid	The appointmentid which the user selected to edit or remove
restriction	Yesterdays date
delete	Query database to cancel booking selected
sql	Query database to get the booking details
From_Date	Restriction when calculating statistics entered by the user
To_Date	Restriction when calculating statistics entered by the user
sessions	Query database to select all the old appointments the user had within the restrictions(From and To date)
getrevenue	Query database to calculate the the total price from all the sessions within the restrictions(From and To date)
sessionnum	The number if sessions the user had within the restrictions(From and To date)
revenue_with_vat	The total revenue from sessions within the restrictions(From and To date)
revenue_without_vat	The total revenue without vat from sessions within the restrictions(From and To date)

Pseudocode for displaying bookings + delete bookings

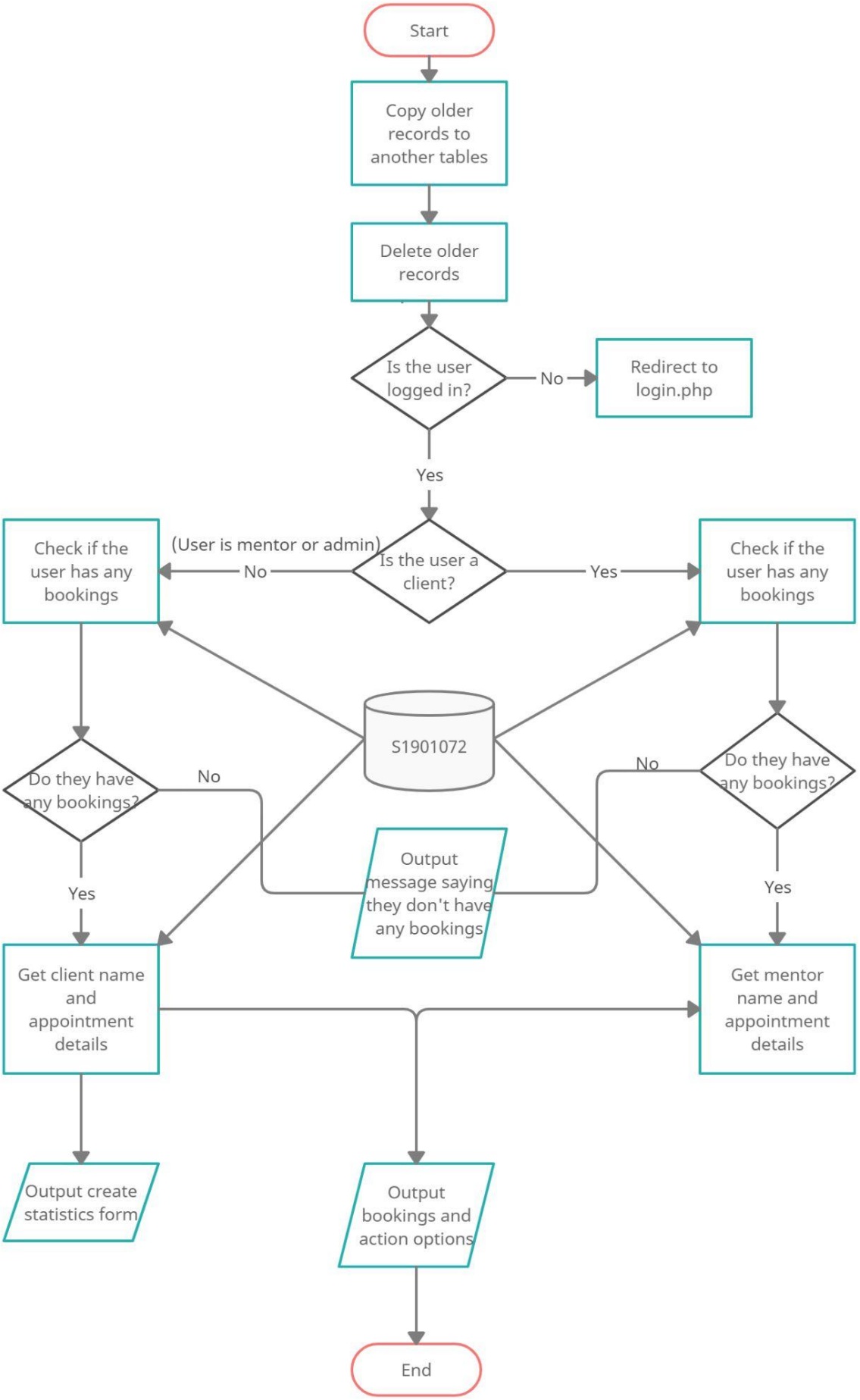
Query database to insert old appointments from the TAppointment table to the TOldAppointment table

IF query is executed successfully THEN
  Query database to delete old bookings from the TAppointment table

```
ENDIF

IF user is NOT logged in THEN
  Redirect user to the login page
ENDIF

SET usertype and ID (of user)

IF user clicks Remove button

 IF  current date is less than 1 day prior to booking date THEN
  Errors['date'] :="You can cancel booking upto 1 day prior"
 END

 IF number of Errors=0 THEN
  Query database to remove booking from TAppointment table based on booking ID

  IF query is executed successfully THEN
    Redirect user to the bookings page
  ENDIF

 ENDIF

ENDIF
IF usertype = "Mentor" OR usertype="Admin" THEN
  Query database to get client name, surname, session name and booking/appointment details
ELSEIF usertype = "Client" THEN
  Query database to get mentor name, surname, session name and booking/appointment details
ENDIF
IF query returns more than 0 rows (user has any booking) THEN
  WHILE there are rows
    Display rows data on the table
  ENDWHILE
ELSE
  Output message saying user doesn't have any session booked
ENDIF
        IF Number of Error > 0 THEN
            FOR Each Error as Error
              Output Error
            ENDFOR
```

ENDIF

Flowchart for displaying bookings(Not delete booking)



SQL

```
INSERT INTO TOldAppointment(AptID, Time, Date, ClientID, MentorID, SessionID, End, Price)  SELECT * FROM Tappointment WHERE Date<'".$today."'

DELETE FROM TAppointment WHERE Date<'".$today."'

DELETE  FROM TAppointment WHERE AppointmentID =$appointmentid

SELECT *,CForename, CSurname, SessionName FROM TAppointment  join TSession join TClient on TAppointment.MentorID=$ID AND TAppointment.SessionID=TSession.SessionID AND TAppointment.ClientID = TClient.ClientID AND TAppointment.Date>$today

SELECT *,MForename, MSurname, SessionName FROM TAppointment  join TSession join TMentor on TAppointment.ClientID=$ID AND TAppointment.SessionID=TSession.SessionID AND TAppointment.MentorID = TMentor.MentorID AND TAppointment.Date>$today
```

Description of SQL

The first query inserts/copy old bookings from the Appointments table to the Old appointment table table, The second query deletes the old bookings from the appointment table(Bookings which were just moved)

The third query deletes a booking which the user user selected from the table using the booking id

The fourth query gets the booking details, client name, surname and session name by using the id of the user logged in and then using the clientid, sessionid on the same row to get the other data.

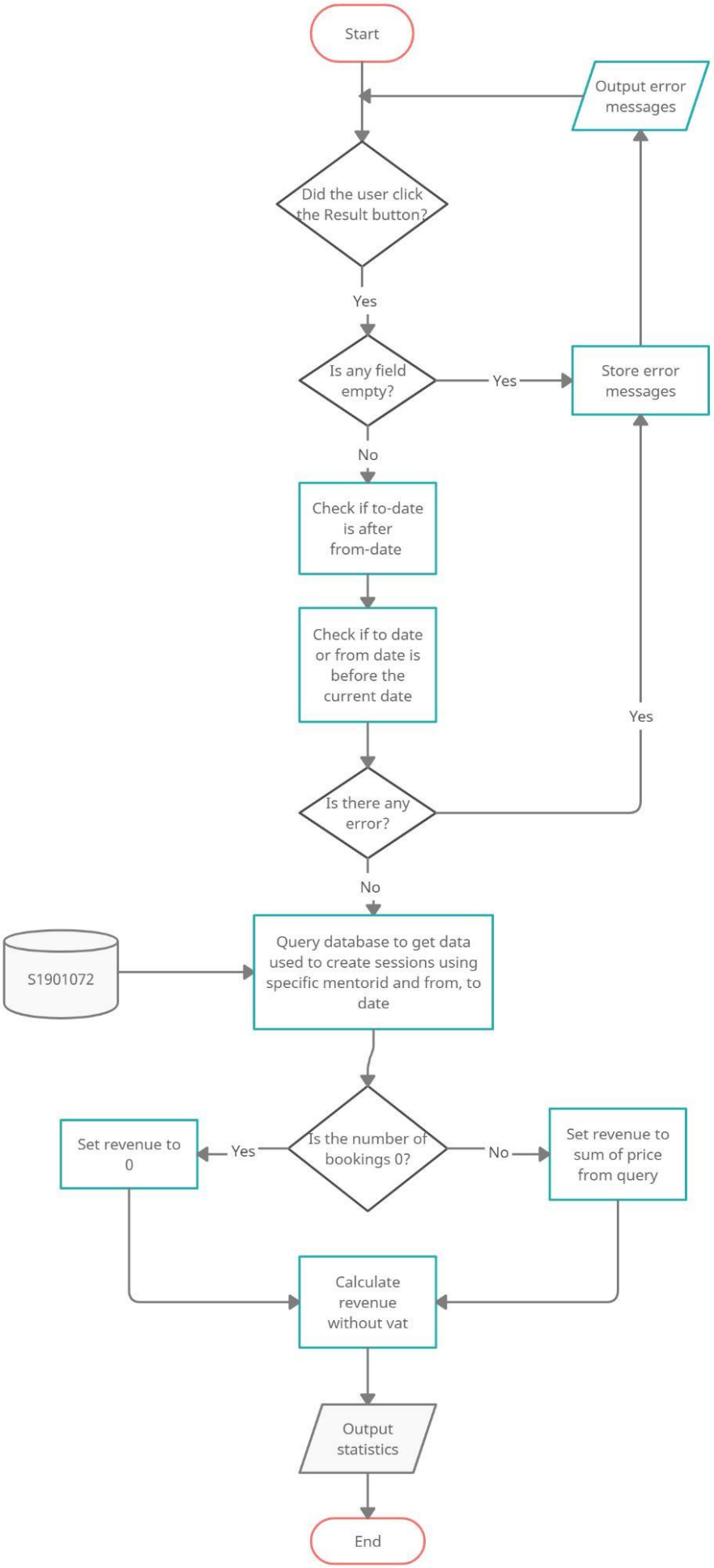
The fifth query is the same as the fourth query except now the mentors name and surname are fetched as this is a client looking at its booking.

Pseudocode for creating statistics

```
IF usertype=”Mentor” OR usertype=”Admin” THEN
  IF From_Date or To_Date field is Empty THEN
    Error[‘empty’] := “Please fill all of the fields”
  ELSE
    IF To_Date<From_Date THEN
      Error[‘date’]:=”To date needs to be after from date”
    ENDIF
```

```
 IF From_Date>Current Date OR To_Date>Current Date THEN

   Error['range']:= "Please select past dates"

 ENDIF

 IF number of errors=0 THEN

  Query database bookings based on dates selected and ID of user logged in
  Query database to find SUM of bookings in range of the dates and mentoID
  Stores number of bookings
  IF number of bookings = 0 THEN
     revenue_with_vat= 0
  ELSE
     revenue_with_vat=Result from query used to find SUM of price
  ENDIF
  Calculate and stores the revenue without vat

ENDIF

IF Number of Error > 0 THEN

        FOR Each Error as Error
          Output Error
        ENDFOR

 ELSE

  Output statistics

 ENDIF

ENDIF
```

Flowchart for creating statistics(Mentor)



SQL

```
"SELECT * FROM TOldAppointment WHERE Date>'".$From_Date."' AND Date<'".$To_Date."'
AND MentorID= $ID
```

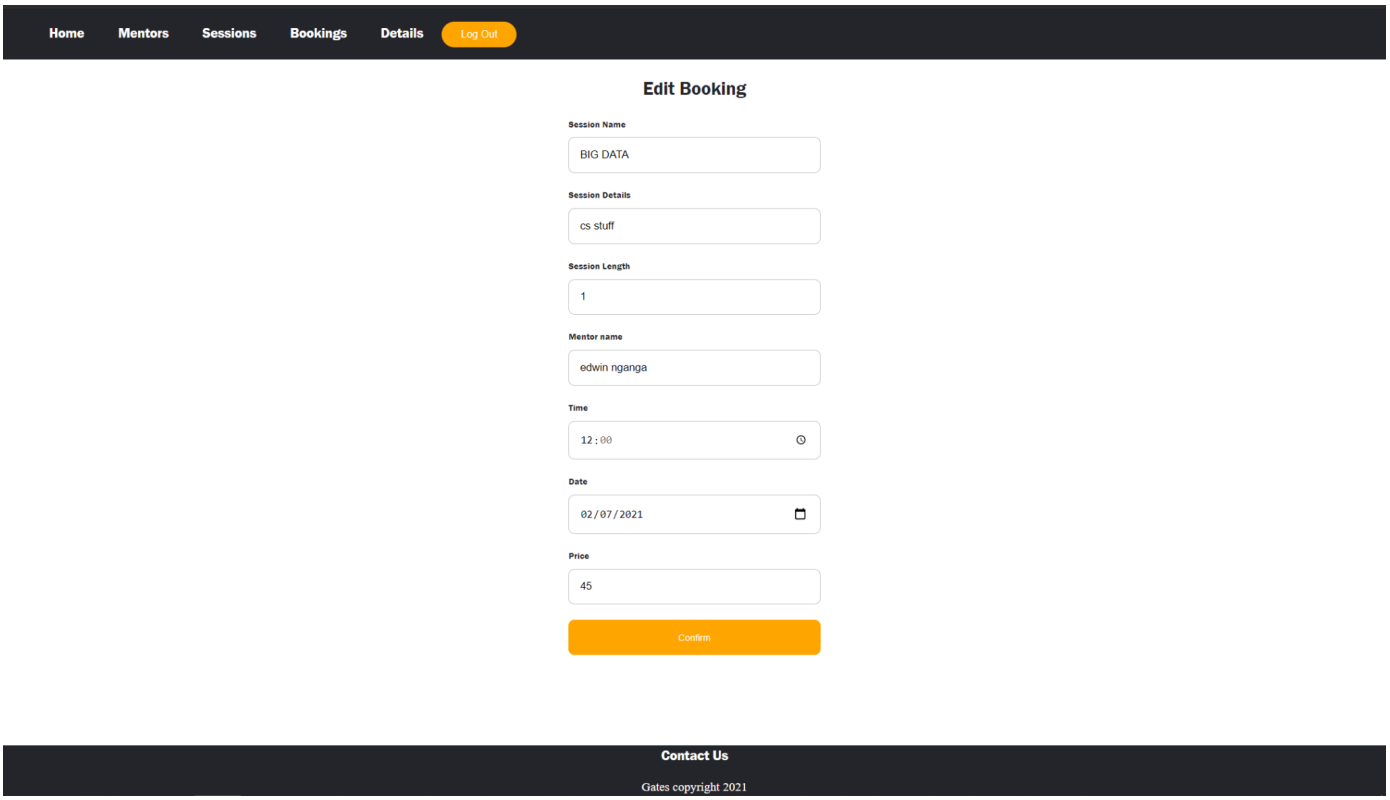
```
SELECT SUM(Price) FROM TOldAppointment WHERE Date>'".$From_Date."' AND
Date<'".$To_Date."' AND MentorID= $ID
```

Description of SQL

The first query selects every records from the Old appointments table using the from-date and to-date as restriction and using them as range as well as the mentorid of the user which is logged in.

The second query calculates the sum of the price of all the records which are in range using the from-date and to-date the user had entered and the mentorid of the user logged in

Design for edit booking



Description

This page contains a form which the user can use to edit the appointment time or date of the booking selected.
Description of pseudocode: The id of the booking is used to get the booking data which is then used to fill the fields the user then can change the time or date of the booking.If there is any error then it is displayed to the user if there is no errors then the time and date are updated.

Variables

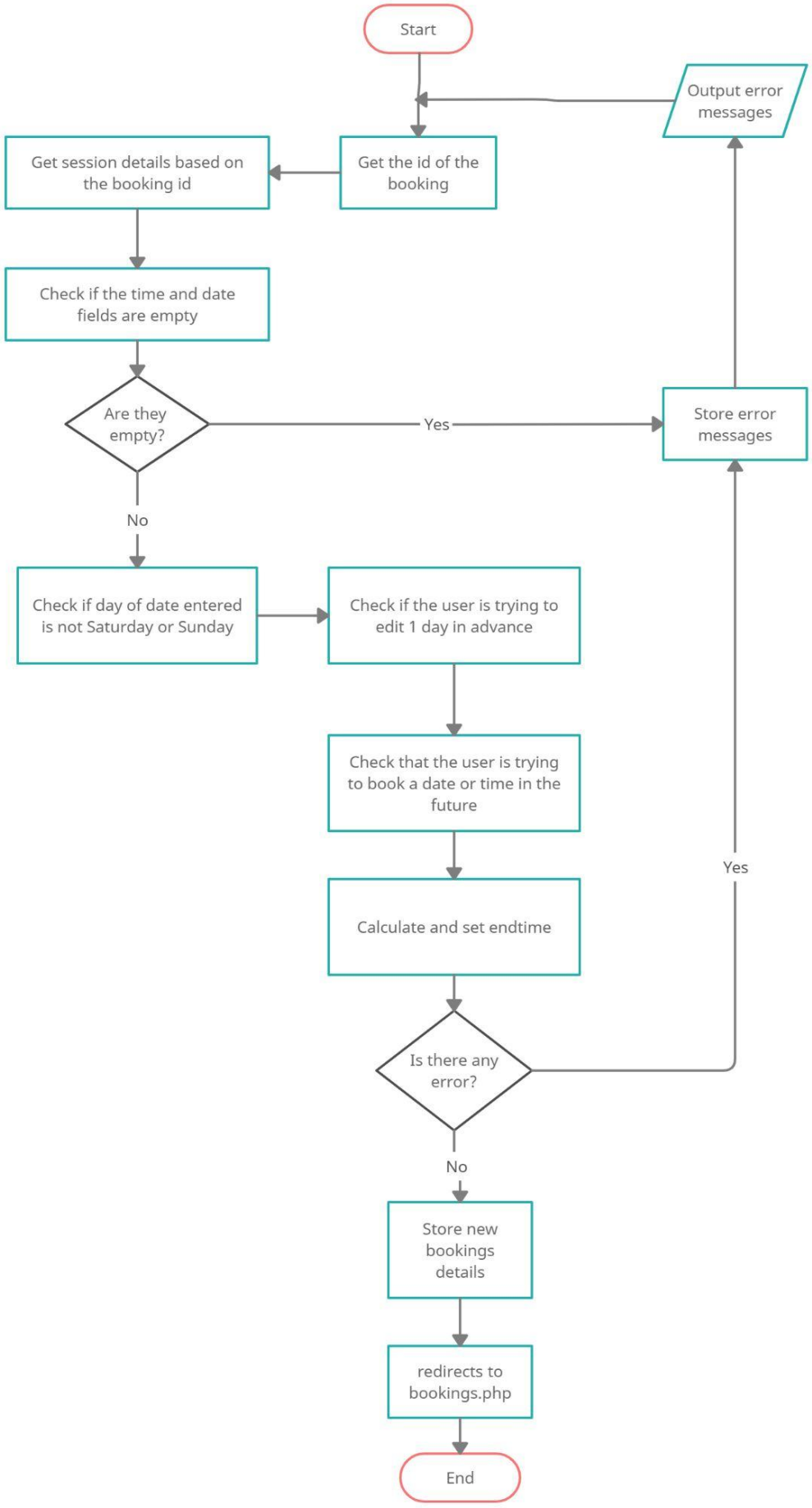
Variable Name	Description
Errors	Array used to store error messages
Time	Stores the time entered by the user in the form
Date	Stores the date entered by the user in the form
Length	Stores the length of the session
day	Stores the day of the date entered by the user
restriction	Stores a restriction used to check if the user is changing the details 1 day prior

Duration	Calculates in minutes the length of the session
Endtime	Stores the time the booking will end
editappointment	Query used to update the booking
id	Stores the booking details of the appointment selected
MentorID	Stores the id of the mentors of which the booking is with
SessionID	Stores the id of the session of the booking

Pseudocode for edit booking

```
IF user clicks update button
  IF Time or Date field is empty THEN
    Errors['Empty']:="Please fill all of the fields"
  ELSE
     IF Day of booking= "Saturday" OR Day of booking="Sunday" THEN
        Errors=['Weekend']:="Can not book during the weekend"
     ENDIF
     IF restriction>Date THEN
        Errors['date']:= "You can change booking details 1 day prior to booking date"
     ENDIF
     IF Length of session = 1 THEN
       Duration := Time user selected + 60 minutes
     ELSEIF Length of session = 2 THEN
       Duration := Time user selected + 120 minute
     ELSE
       Duration := Time user selected + 180 minutes
     ENDIF
     Set the Endtime for when the session will end
    IF number of errors=0 THEN
      Query database to update booking details based on the booking id
  ENDIF
ENDIF

Set id of booking selected
Query database to select appointment details based on that bookingid
Query database select the mentor name, surname , session name and booking details
Output result from query into form
IF Number of Error > 0 THEN

          FOR Each Error as Error
            Output Error
          ENDFOR
ENDIF
```

Flowchart edit booking



SQL

```
UPDATE Tappointment SET Time='{$Time}', Date = '{$Date}', Endtime ='{$Endtime}'
            WHERE AppointmentID= " . $_POST['AppointmentID']

SELECT * FROM  TAppointment WHERE AppointmentID={$id}

SELECT *, MForename, MSurname FROM TSession JOIN TMentor WHERE
TSession.SessionID='".$SessionID."' and TMentor.MentorID ='".$MentorID."'
```

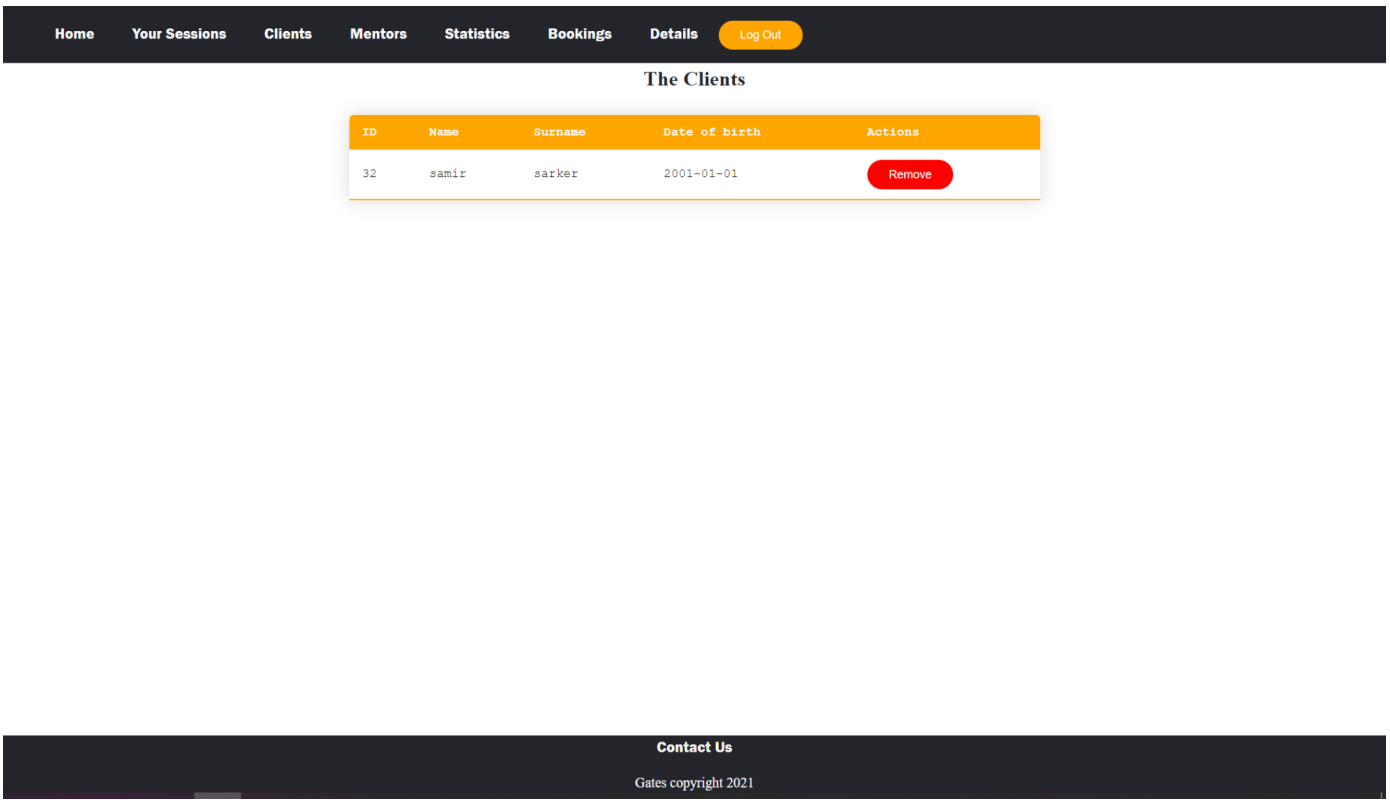
Description of the SQL

The first query is used to update the time and date of the booking using the id of the booking selected

The second query is used to select all of the appointment data of the booking selected

The last query is used to select the all the booking data as well as the mentor name,surname and session name using the MentorID and sessionID in the same row

Design for displaying clients



Description:

This page contains all the clients so that the admin can view them and remove them from the system if necessary
Description of the pseudocode:Check if the user logged in is the admin, if it isn't they are redirected to the login page. All of the data from the Tclient table is selected and then displayed on the table.
If the admin clicks the remove button the data of the client in that row is deleted from the login and client table

Variables

Variable Name	**Description**
usertype	Stores the usertype of the user logged in
ID	Stores the id of the user logged in
id	Stores the id of the client selected

Pseudocode

```
IF user is logged in THEN

  IF usertype=”Admin” THEN

    IF User clicks remove button THEN

      Query database to remove records from login and client table using id of  client selected
      IF query is executed successfully THEN

        Redirect user to Clients page

      ENDIF

    ENDIF

   Query database to select every record from the Client table

   IF query returns more than 0 rows THEN

     WHILE there are rows

       Display each row data on the table

     ENDWHILE

   ELSE
```

Output message saying there are no records

ENDIF

ELSE

Redirect to Login page

ENDIF

ELSE

Redirect to Login page

ENDIF

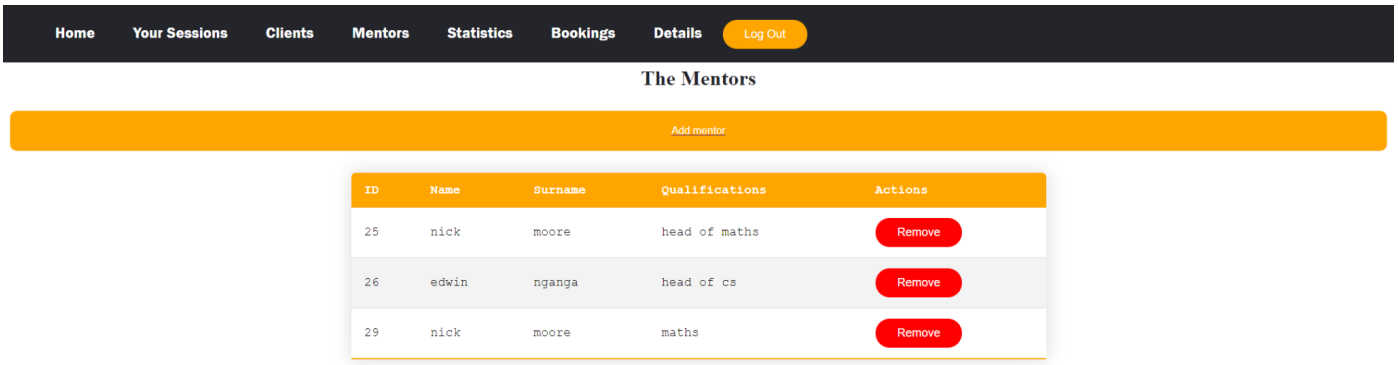
SQL

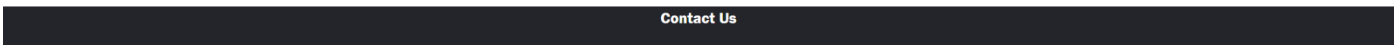
```
DELETE FROM TClient WHERE ClientID = $id
DELETE FROM TLogin WHERE UserType = 'Client' AND ID = $id
SELECT * FROM TClient
```

Description of SQL

The first query delete the row of data/record from the Client table using the id of the client the admin had selected
The second query delete the row of data/record from the login table using the id of the client the admin had selected and UserType=”Client”
The last query selects all the records from the client table

Design for displaying mentors(Admin view)





Description

This page contains all the mentors so that the admin can view them and remove them from the system if necessary
Description of the pseudocode:Check if the user logged in is the admin, if it isn't they are redirected to the login page. All of the data from the TMentor table is selected and then displayed on the table.
If the admin clicks the remove button the data of the client in that row is deleted from the login and Mentor table as well all of the mentors session on the sessions table. There is also a button which can be used to register a new mentor which i have shown on page 50.

Variables

Variable Name	**Description**
usertype	Stores the usertype of the user logged in
ID	Stores the id of the user logged in
id	Stores the id of the mentor selected

Pseudocode

```
IF user is logged in THEN

  IF usertype=”Admin” THEN

    IF User clicks remove button THEN

      Query database to remove records from login and mentor table and sessions from the session table
      using id  of  mentor selected
      IF query is executed successfully THEN

        Redirect user to Mentors page (adminmentors.php)

      ENDIF

    ENDIF

   Query database to select every record from the Mentors table

   IF query returns more than 0 rows THEN

     WHILE there are rows

       Display each row data on the table
```

ENDWHILE

ELSE

Output message saying there are no records

ENDIF

ELSE

Redirect to Login page

ENDIF

ELSE

Redirect to Login page

ENDIF

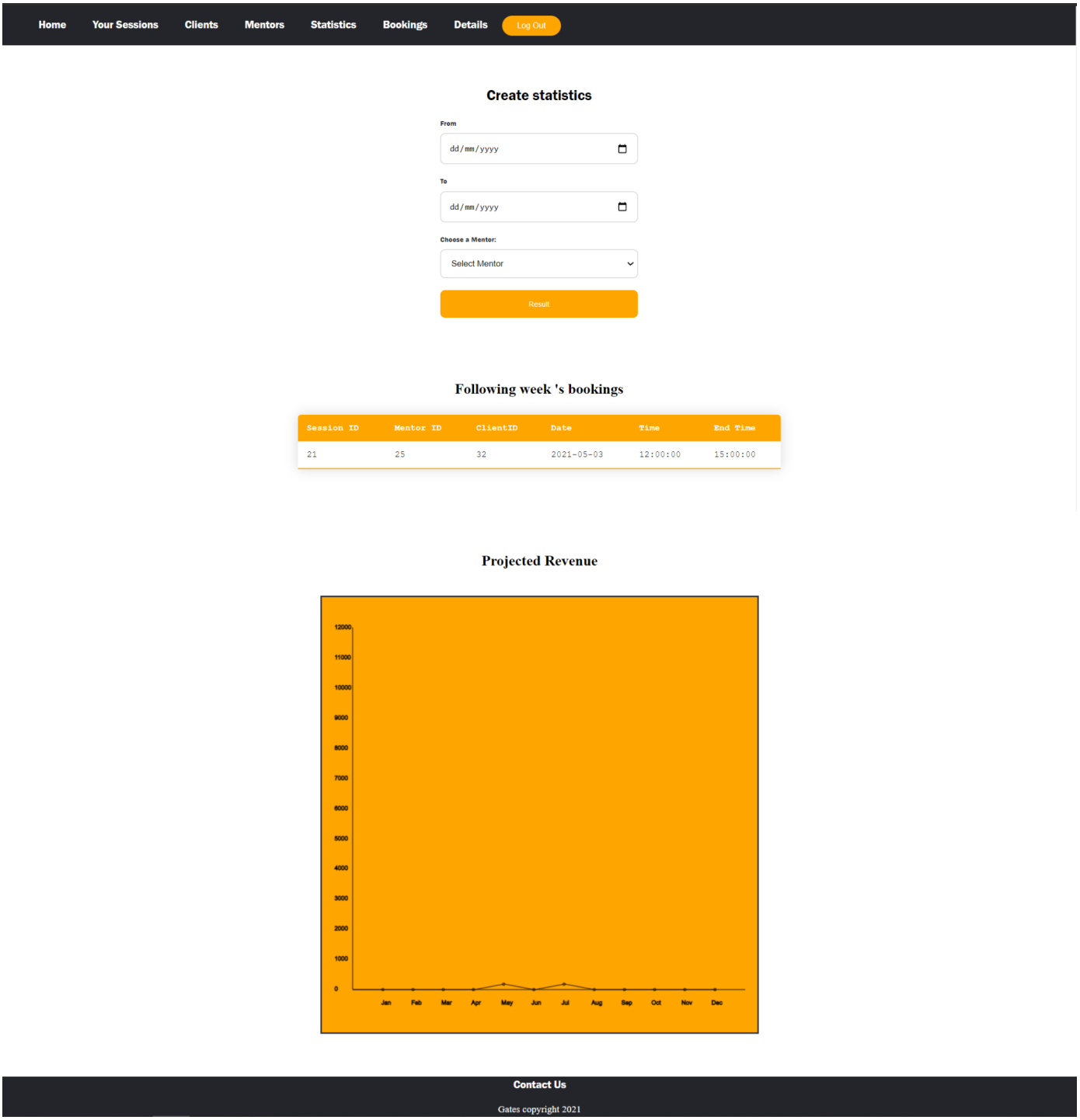
SQL

```
DELETE FROM Tmentor WHERE MentorID = $id
DELETE FROM TLogin WHERE UserType = 'Mentor' AND ID = $id
DELETE FROM TSession WHERE  MentorID = $id
SELECT * FROM TMentor
```

Description of the SQL

The first query is used to delete records from the mentors table using the id of the mentor selected
The second query is used to delete records from the login table using the id of the mentor selected and UserType=”Mentor”

Design for statistics page



Description

This page can only be accessed by the admin.
It shows a form which can be used to create statistics using old data for a specific mentor or for every mentor together
It also shows a table which displays all of the bookings for the following week.
Lastly there is graph which shows the projected revenue for the whole centre.

Description of pseudocode:First it is checked that the user is the admin otherwise they will be redirected to the login page.The rest is split in two parts, the first part it consists of how the statistics are created which is what i will write about now, the second part shows how the bookings are displayed.
How statistics are created:
The admin enters the from and to date on the form. They select a mentor from the drop down menu which shows options for every mentor and an option to choose everyone in Gates.
There is then a presence check, if this passes there are more error checks such as checking if the from and to date are past dates or if the from and to date are in order and etc. If there are no eros then query the database using the conditions from the form as restrictions else if there are errors they are displayed to the user.

Variables

Variable Name	**Description**
usertype	Stores the usertype of the user
today	Stores the current date
insertrecord	Query database to move older records
deleterecord	Query database to delete older records from appointments table
Error	Array storing error messages
Mentor	Stores the id of the mentor selected or “everyone”
From_Date	Stores a restriction entered by the user(From date used for range)
To_Date	Stores a restriction entered by the user.(To date used for range)
sessions	Query database to get all of the bookings using the restrictions
getrevenue	Query database to find the sum of the price of the bookings in the range
sessionnum	Stores the number of bookings in the range
revenue_with_vat	Stores the price (sum from getrevenue)
revenue_without_vat	Stores 80% of revenue_with_vat

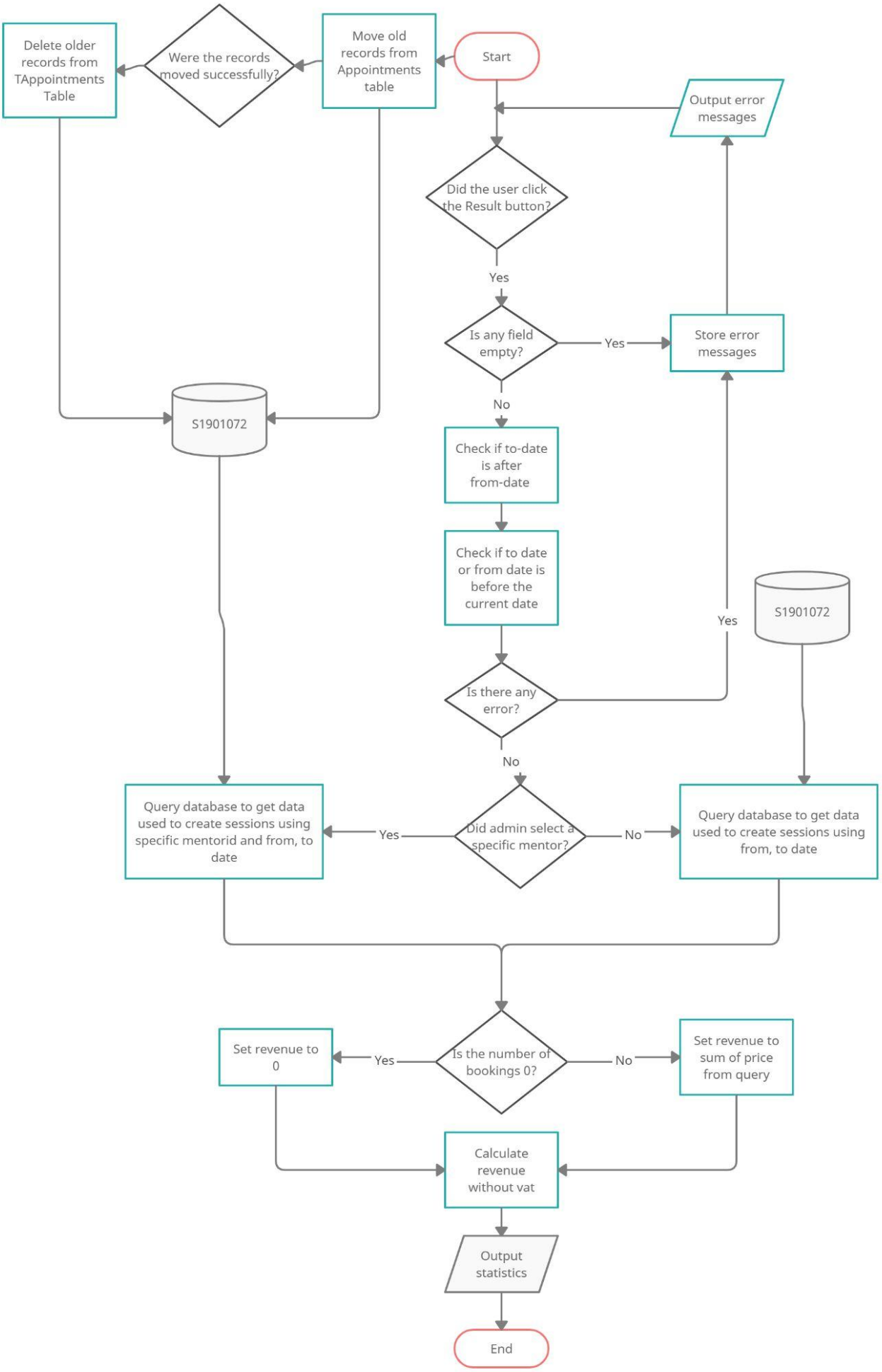
Pseudocode (Process of creating statistics, does not include check if usertype is admin)

Query database to insert old appointments from the TAppointment table to the TOldAppointment table

IF query is executed successfully THEN
  Query database to delete old bookings from the TAppointment table
ENDIF

```
IF From_Date field OR To_Date field OR Mentors-statistics field is Empty THEN

  Error[‘empty’] := “Please fill all of the fields”

ELSE

  IF To_Date<From_Date THEN

    Error[‘date’]:=”To date needs to be after from date”

  ENDIF

  IF From_Date>Current Date OR To_Date>Current Date THEN

    Error[‘range’]:= “Please select past dates”

  ENDIF

  IF number of errors=0 THEN
      IF Admin selected specific mentor
        Query database to find bookings based on dates selected and ID of mentor selected
        Query database to find SUM of bookings in range of the dates and mentorID
      ELSE
        Query database to find bookings based on dates selected
        Query database to find SUM of bookings in range of the dates
      ENDIF
    Stores number of bookings
    IF number of bookings = 0 THEN
       revenue_with_vat= 0
    ELSE
       revenue_with_vat=Result from query used to find SUM of price
    ENDIF
    Calculate and stores the revenue without vat
ENDIF

IF Number of Error > 0 THEN

          FOR Each Error as Error
            Output Error
          ENDFOR

 ELSE

   Output statistics

 ENDIF
```

Flowchart for creating statistics



SQL

```
INSERT INTO TOldAppointment(AptID, Time, Date, ClientID, MentorID, SessionID, End,
Price) SELECT * FROM Tappointment WHERE Date<'".$today."'
DELETE FROM TAppointment WHERE Date<'".$today."'

SELECT * FROM TOldAppointment WHERE Date>'".$From_Date."' AND Date<'".$To_Date."'
SELECT SUM(Price) FROM TOldAppointment WHERE Date>'".$From_Date."' AND
Date<'".$To_Date."’

SELECT * FROM TOldAppointment WHERE Date>'".$From_Date."' AND Date<'".$To_Date."'
AND MentorID= '".$Mentor."'
SELECT SUM(Price) FROM TOldAppointment WHERE Date>'".$From_Date."' AND
Date<'".$To_Date."' AND MentorID= '".$Mentor."'
```

Description of the SQL

The first query inserts old bookings from the Appointments table to the old appointments tables, the second query deletes those records from from the appointments table.

The third query selects every record from the old appointment table using the from and to date the admin entered in the form as restriction/range
The fourth query calculates the the sum of the price for those bookings/records

The fifth and sixth query essentially do the same thing as the third and fourth query however now there is another restriction which is the MentorID

Display coming weeks booking

Variables

Variables name	Description
restriction	Stores date 7 days from now
bookings	Query database to select bookings using the restriction
appointments	Result from executing the query

Pseudocode

Set restriction as 7 days after current date

Query database to select all of the records from the TAppointment table using the restriction

IF query returns more than 0 rows THEN

    WHILE there are rows

        Display each row data on the table

    ENDWHILE

ELSE

  Output message saying there are no bookings for the following week

ENDIF

SQL

```
SELECT * FROM TAppointment WHERE Date < '".$restriction."'
```

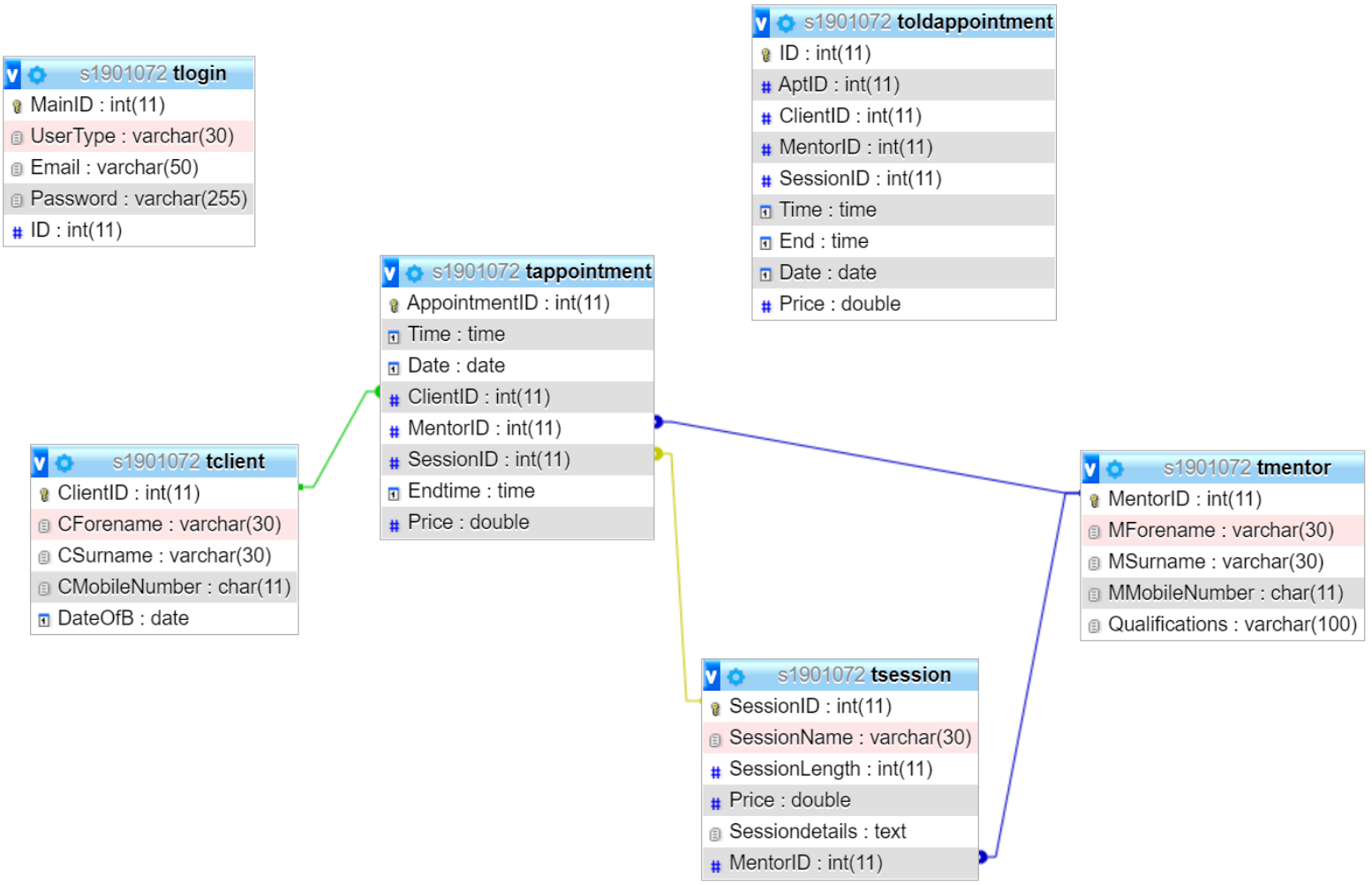
Description of the SQL

Select all of the appointments using the restriction as range(selects all of the appointments upto 1 week from current date)

3. Technical solution

3.1 Database

3.1.1 Entity relationship diagram of tables on the database



3.1.2 Tables

3.1.2.1 Tlogin

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**MainID**	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
2	**UserType**	varchar(30)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
3	**Email**	varchar(50)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
4	**Password**	varchar(255)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
5	**ID**	int(11)			Yes	*NULL*			Change Drop More

SQL used to create TLogin table

```
CREATE TABLE TLogin(
MainID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
UserType VARCHAR(30),
Email VARCHAR(50),
Password VARCHAR(255),
ID INT
);
```

3.1.2.2 Tclient

Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
ClientID	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
CForename	varchar(30)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
CSurname	varchar(30)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
CMobileNumber	char(11)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
DateOfB	date			Yes	*NULL*			Change Drop More

SQL used to create TClient table

```
CREATE TABLE TClient(
ClientID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
CForename VARCHAR(30),
CSurname VARCHAR(30),
CMobileNumber CHAR(11),
DateOfB date
);
```

3.1.2.3 TMentor

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**MentorID**	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
2	**MForename**	varchar(30)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
3	**MSurname**	varchar(30)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
4	**MMobileNumber**	char(11)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
5	**Qualifications**	varchar(100)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More

SQL used to create TMentor table

```
CREATE TABLE TMentor(
MentorID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
MForename VARCHAR(30),
MSurname VARCHAR(30),
MMobileNumber CHAR(11),
Qualifications VARCHAR(100)
);
```

3.1.2.4 TSession

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**SessionID**	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
2	**SessionName**	varchar(30)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
3	**SessionLength**	int(11)			Yes	*NULL*			Change Drop More
4	**Price**	double			Yes	*NULL*			Change Drop More
5	**Sessiondetails**	text	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
6	**MentorID**	int(11)			Yes	*NULL*			Change Drop More

SQL used to create TSession Table

```
CREATE TABLE TSession(

SessionID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,

SessionName VARCHAR(30),

SessionLength INT,

Price DOUBLE,

Sessiondetails TEXT,

MentorID INT,

CONSTRAINT fk_MentorID FOREIGN KEY (MentorID) REFERENCES TMentor(MentorID)

);
```

3.1.2.5 TAppointment

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**AppointmentID**	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
2	**Time**	time			Yes	*NULL*			Change Drop More
3	**Date**	date			Yes	*NULL*			Change Drop More
4	**ClientID**	int(11)			Yes	*NULL*			Change Drop More
5	**MentorID**	int(11)			Yes	*NULL*			Change Drop More
6	**SessionID**	int(11)			Yes	*NULL*			Change Drop More
7	**Endtime**	time			Yes	*NULL*			Change Drop More
8	**Price**	double			Yes	*NULL*			Change Drop More

SQLused to create TAppointment table

```
CREATE TABLE TAppointment (
AppointmentID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
Time time,
Date date,
ClientID INT,
CONSTRAINT fk_ClientID FOREIGN KEY (ClientID) REFERENCES TClient(ClientID),
    MentorID INT,
CONSTRAINT fk_MentorID FOREIGN KEY (MentorID) REFERENCES TMentor(MentorID),
    SessionID INT,
CONSTRAINT fk_SessionID FOREIGN KEY (SessionID) REFERENCES TSession(SessionID),
Endtime time,
Price DOUBLE
);
```

3.1.2.6 TOldAppointment

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**ID**	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
2	**AptID**	int(11)			Yes	*NULL*			Change Drop More
3	**ClientID**	int(11)			Yes	*NULL*			Change Drop More
4	**MentorID**	int(11)			Yes	*NULL*			Change Drop More
5	**SessionID**	int(11)			Yes	*NULL*			Change Drop More
6	**Time**	time			Yes	*NULL*			Change Drop More
7	**End**	time			Yes	*NULL*			Change Drop More
8	**Date**	date			Yes	*NULL*			Change Drop More
9	**Price**	double			Yes	*NULL*			Change Drop More

SQL used to create TOldAppointment Table

```
CREATE TABLE TOldAppointment (

ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,

AptID INT,

ClientID INT,

MentorID INT,

SessionID INT,

Time time,

End time,

Date date,

Price DOUBLE

);
```

3.2 Objectives

Number and page	Objective	Objective	Complexity	Page
1.Connect.php	Connects to the database	NA	Low	120
2.footer.php	Will be at the bottom of every page. It has a link which lead the user to the contact page and shows copyright	1	Low	120
3.header.php	Will be at the top of every page. This includes link to many pages and its the main way the user will be able to navigate the website. The header/navigation bar changes depending the user type.	1	Medium	121
4.index.php	It is the landing page when the user initially enter the website	1	Medium	123
5.mentors.php	A Page which will display the details of all the	1	Low	126

	mentors at Gates			
6.sessions.php	A Page which will display all the sessions available from which the client can select and book a session.	1	Medium	128
7.contact.php	A Page which displays the days and times at which Gates will be open as well the location of the center. There is also a contact form which a user can use to ask information about Gates.	10	High	130
8.signup.php	A page which contains a form where a client enters personal details and their login information to register on the system.	3	High	132
9.msignup.php	A Page only accessible by the admin which contains the form which is used to register mentors on the system	5	High	135
10.login.php	Page which contains a form and captcha test to login the user to the system	2	High	138
11.logout.php	Logs the user out of the system	2	Low	141
12.details.php	A page which displays the details of the user logged in and a button which will lead to a page where the user can edit their details and a form which will allow them to change the password.	4,5	High	141
13.edit.php	Page containing a form which allows the user to edit their personal details.	4,5	Medium	145
14.ownsessions.php	A page which contains all of the sessions which the mentor logged in provides. There are 3 buttons, one which will lead to a page where the mentor can edit the session selected and the other which will let them delete the selected session. There is another button which will allow the mentor to add a session.	6	High	149
15.sessionmentor.php	This Page contains a from which will allow the mentor to add a new session.	6	Medium	152
16.editsession.php	The page contains a form filled with the information of the session selected and allow the mentor to change those details hence editing the session.	6	Medium	154
17.book.php	A page which contains a form which will allow the user to book the session by providing the time and date. It connects to the Stripe API which will allow the user to make a payment so that they can book the session.	4	High	157

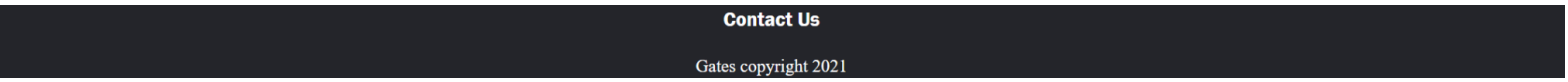
18.success.php	this page will add the booked appointment to database as the payment is successful	4	Low	161
19.bookings.php	This Page will display the bookings the user has and allow them to change the bookings or remove them. If the user is a mentor it also includes a form which will create statistics for the specific mentor.	7,9	High	162
20.editbooking.php	The page contains a form which will allow the user to edit the time or date of the selected appointment/booking	4	Medium	167
21.adminmentors.php	The page can only be accessed by the admin, it displays all of the mentors at Gates and allows the admin to remove them from the system.	8	Medium	170
22.Clients.php	The page can only be accessed by the admin, it displays all of the clients of Gates and allows the admin to remove them from the system.	8	Low	172
23.statistics.php	This page can also be only be accessed by the admin. It has a form which will create statistics, It displays all of the sessions of the upcoming week and displays a graph which shows the projected revenue.	8,9	High	174
24.graph.php	A page which creates a graph on the projected revenue by getting the data from the database.	NA	High	179
25. cards.css	Styling to make the mentor and session cards	NA	Low	182
26. form.css	Styling to make the forms	NA	Low	183
27. navbar.css	Styling for the navbar and footer	NA	Medium	184
28. split.css	Style to split the screen in half	NA	Low	185
29. table.css	Styling for the tables	NA	Low	185

1. Connect.php

Connect.php

```php
<?php
$localhost = "localhost";
$username = "root";
$password = "";
$dbname = "s1901072";
//establishes connection to the database
$con = new mysqli($localhost, $username, $password, $dbname);
//If there is any error disrupt the connection to the database.
if($con->connect_error) {
    die("connection failed : " . $con->connect_error);
}
?>
```

2. footer.php



Code for footer.php

footer.php

```
1
2   <html>
3   <head>
4   <link rel="stylesheet" type="text/css" href="CSS/navbar.css">
5   </head>
6   <body>
7   <div class="footer">
8   <li><a href="contact.php">Contact Us</a></li>
9   </p>Gates copyright<?php echo " ".date('Y');?>
10  </div>
11  </body>
12  </html>
```

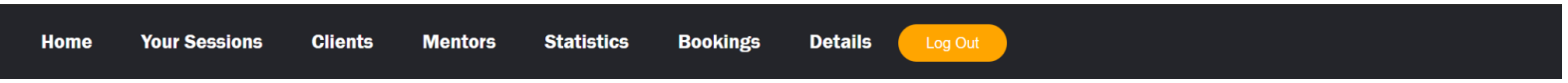
3. header.php

(Not Logged in)

(Client Logged in)

(Mentor Logged in)

(Admin Logged in)



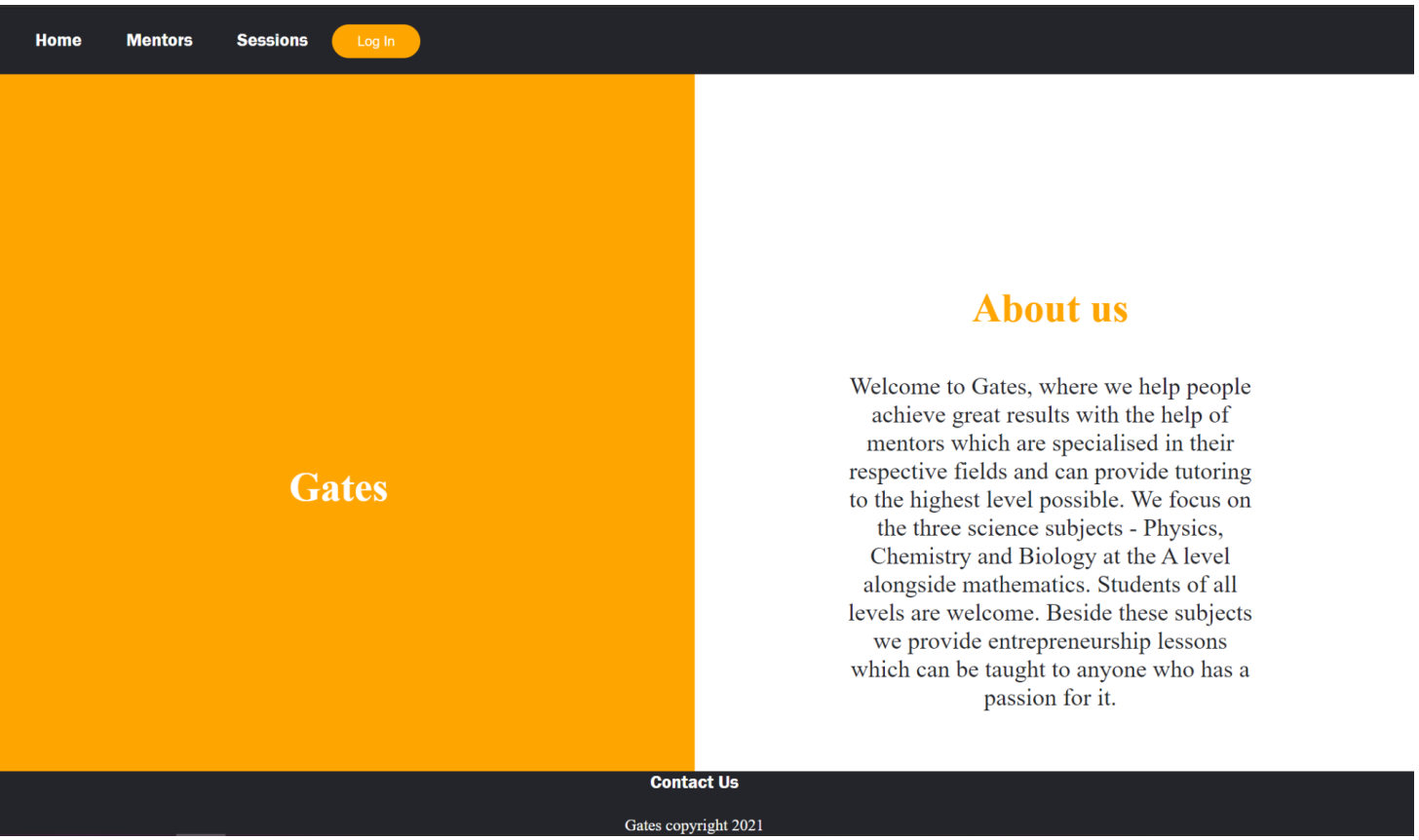
Code for header.php

header.php

```php
<!DOCTYPE html>
<html>
  <head>
  <link rel="stylesheet" type="text/css" href="CSS/navbar.css">
  </head>
  <body class="nav-header">
    <header>
      <nav>
        <ul class="links">

          <li><a href="index.php">Home</a></li>
          <?php
          //Checks if the user is not logged in
          if( !( isset($_SESSION["ID"]) ) ) {
          ?>
          <li><a href="mentors.php">Mentors</a></li>
          <li><a href="sessions.php">Sessions</a></li>
          <?php
          }
          else{
            $usertype = $_SESSION["UserType"];
            //Checks if the user is a mentor or client
            if ($usertype=="Mentor"||$usertype=="Client"){
              ?>
              <li><a href="mentors.php">Mentors</a></li>
              <?php
            }
             //Checks if the user is a client
             if ($usertype=="Client"){
              ?>
              <li><a href="sessions.php">Sessions</a></li>
              <?php
            }
            //Checks if the user is a mentor or the admin
            if ($usertype=="Admin"||$usertype=="Mentor"){
              ?>
              <li><a href="ownsessions.php">Your Sessions</a></li>
              <?php
            }
            //Checks if the user is an admin
            if ($usertype=="Admin"){
              ?>
              <li><a href="Clients.php">Clients</a></li>
              <li><a href="adminmentors.php">Mentors</a></li>
              <li><a href="statistics.php">Statistics</a></li>
              <?php
            }

            ?>
            <li><a href="bookings.php">Bookings</a></li>
```

```
                <li><a href="details.php">Details</a></li>

                <?php
              }
               ?>
            <?php
            if(isset($_SESSION["ID"])) {
            ?>
              <a class="cta" href="logout.php"><button class="navigation">Log Out</button></a>
            <?php
            }
            else{
              ?>
              <a class="cta" href="login.php"><button class="navigation">Log In</button></a>
              <?php
            }
            ?>
          </ul>
        </nav>
      </header>
    </body>
  </html>
```

4. index.php



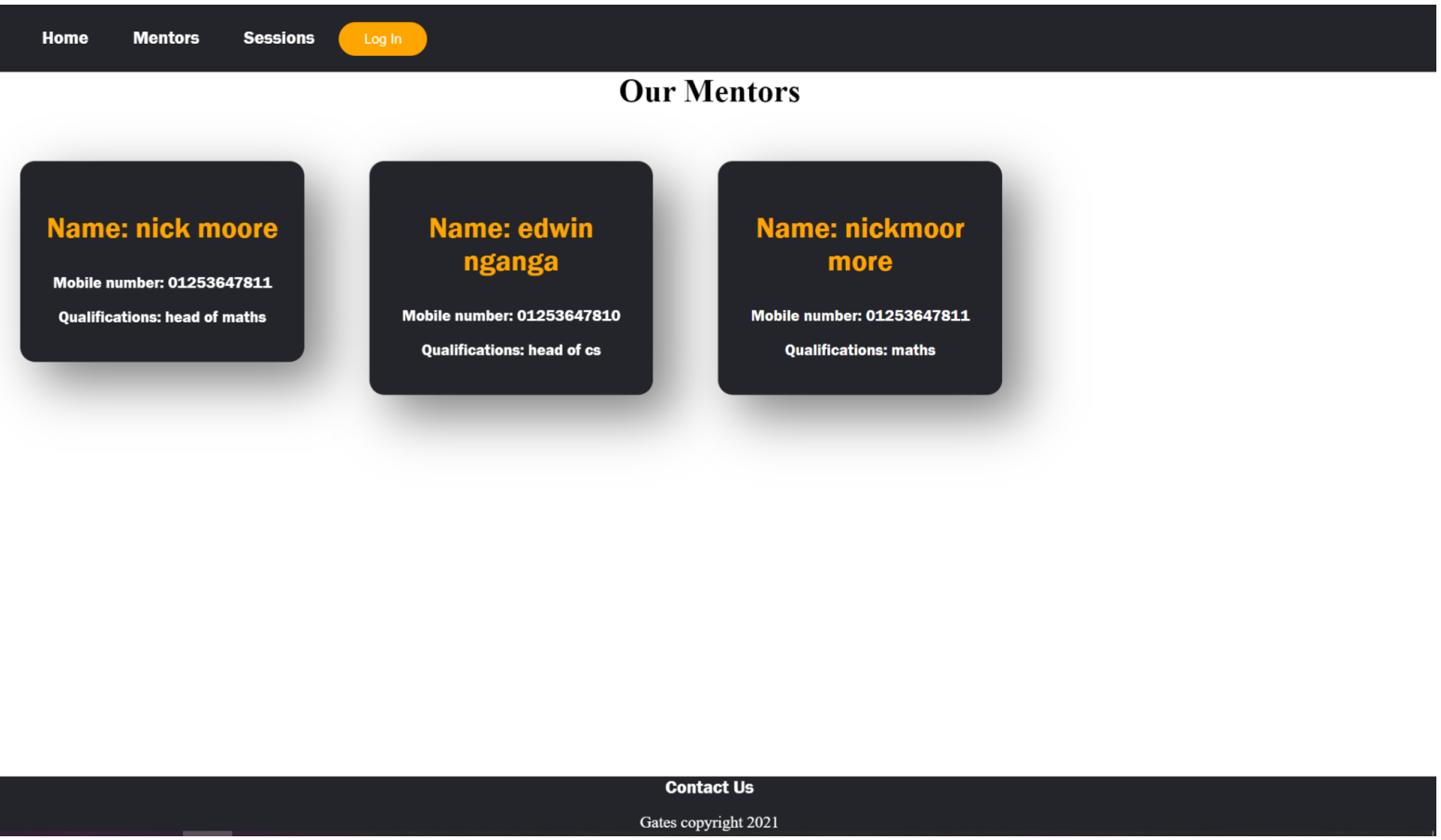
Code for index.php

index.php

```php
<?php
session_start();

require_once 'connect.php';
require_once 'header.php';

?>
<html>
    <head>
    <link rel="stylesheet" type="text/css" href="CSS/split.css">
    </head>
    <body>
        <style>
            h1{
                color:white;
                font-size: 40px;
            }
            h3{
                color:orange;
                font-size: 40px;
            }
            .welcome{
             z-index: 0;

            margin-top: 65px;
            }
            .welcome p, .About p{
                font-size: 1.5rem;

                 color: #24252A;
                 text-align:center;
            }
            .About{
                width: 400px;
                height: 400px;
                margin:100px auto;
                position: relative;

            }

        </style>
        <div class="welcome" align="center">
        <?php
        if(isset($_SESSION["ID"])  ) {
            $ID = $_SESSION["ID"];
            $usertype = $_SESSION["UserType"];
```

```
        if ( $usertype=="Client" ){
            //gets the name of the client
            $getname=mysqli_query($con,"SELECT CForename FROM TClient WHERE ClientID = $ID");
            $row = mysqli_fetch_array($getname);
        echo "<p>"."Welcome back ".$row['CForename']."</p>" ;
            // Creates a restriction by finding the day a week from the current date
            $restriction = date("Y-m-d", strtotime("+ 7 day"));
            $getbookings=mysqli_query($con," SELECT * FROM TAppointment WHERE ClientID = $ID AND Date < '".$restriction."' ");
        }
        else{
            //gets the name of the mentor
            $getname=mysqli_query($con,"SELECT MForename FROM TMentor WHERE MentorID = $ID");
            $row = mysqli_fetch_array($getname);
            echo "<p>"."Welcome back ".$row['MForename']."</p>" ;
            // Creates a restriction by finding the day a week from the current date
            $restriction = date("Y-m-d", strtotime("+ 7 day"));
            $getbookings=mysqli_query($con," SELECT * FROM TAppointment WHERE MentorID = $ID AND Date < '".$restriction."'") ;
        }
        //Checks if the user has any bookings in the following week
        $sessionnum  =mysqli_num_rows($getbookings);
        if ($sessionnum===0){
            echo "<p>"." You don't have any sessions booked for this week"."</p>";

        }
        else{
            echo"<p>"."you have ".$sessionnum." sessions booked for this week"."</p>";

        }

    }
    ?>

    </div>
  <div class="split">
    <div class="left">
        <div class="main_index">
        <h1  class="indexh1" align="center"> Gates</h1>
        </div>

    </div>
    <div class="right">
        <div class="main">
           <div class= "About">
               <h3 align="center">About us</h3>
               <p>Welcome to Gates, where we help people achieve great results with the
                   help of mentors which are specialised in their respective fields and can
                   provide tutoring to the highest level possible. We focus on the three science
                   subjects - Physics, Chemistry and Biology at the A level alongside mathematics.
                   Students of all levels are welcome. Beside these subjects we provide entrepreneurship
                   lessons which can be taught to anyone who has a passion for it.
                    </p>
                <div>

            <div>
        </div>

    </div>

    </body>


</html>
<?php
require_once 'footer.php';
?>
```

5. mentors.php



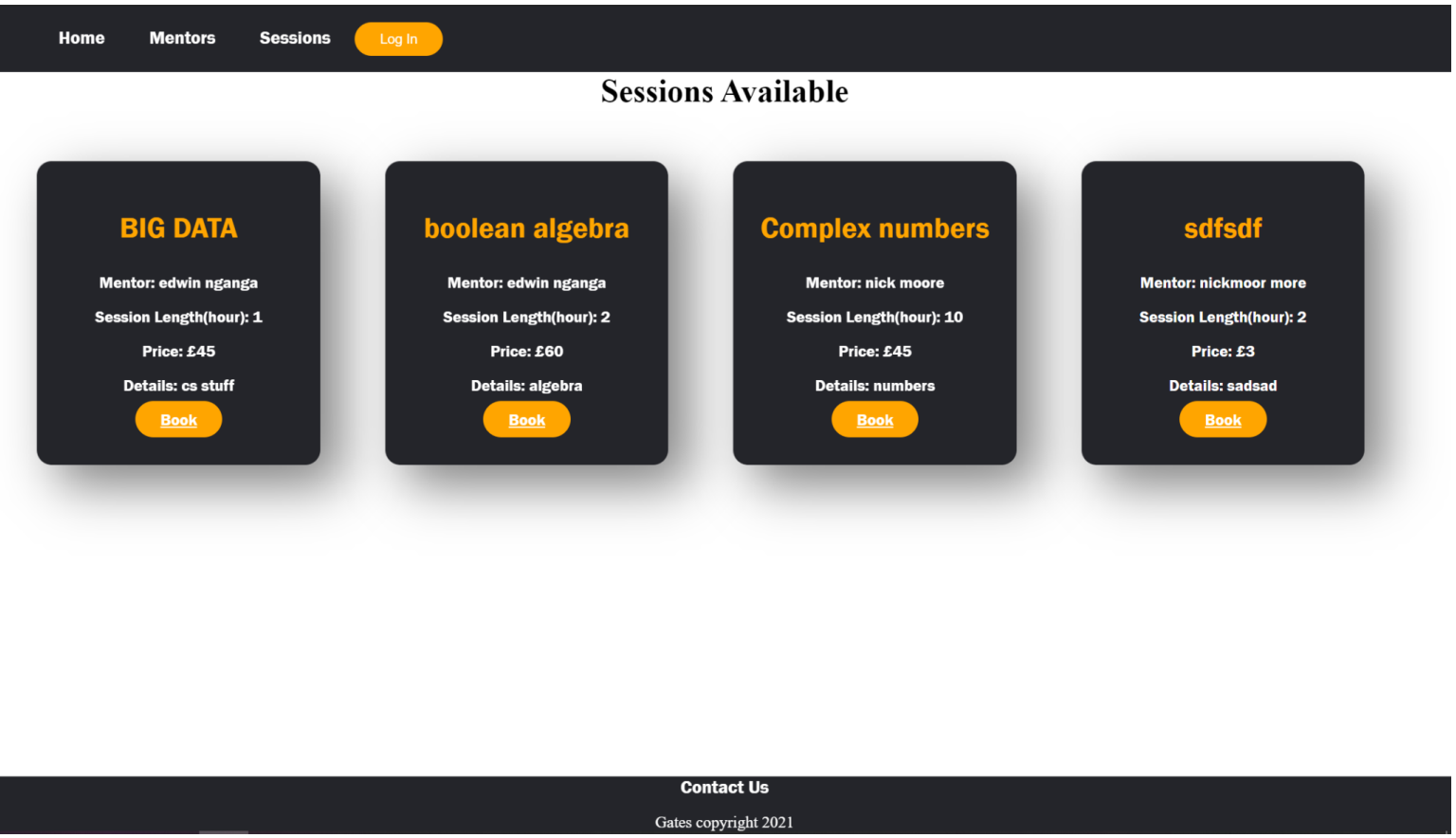
Code for mentors.php

mentors.php

```
1
2   <?php
3   //Starts session to provide specific features if the user is logged in
4   session_start();
5   //Gets the following file to connect to the database
6   require_once 'connect.php';
7   //Get the navigation bar
8   require_once 'header.php';
9   ?>
10  <html>
11    <head>
12
13    <link rel="stylesheet" type="text/css" href="CSS/cards.css">
14
15    </head>
16    <body>
17     <h1 align="center">Our Mentors</h1>
```

```php
<?php
//Query to retrieve the mentors details
$sql = "SELECT * FROM TMentor";
$result = $con->query($sql);
//Checks if there are mentor record on the mentors table from the database
if( $result->num_rows > 0){
  while( $row = $result->fetch_assoc()){
    ?>


      <div class="container">
        <div class="card">
          <div class="content">
            <h3><?php echo "Name: ".$row['MForename']." ".$row['MSurname'];?></h3>
            <p><?php echo  "Mobile number: ".$row['MMobileNumber'];?><p>
            <p><?php echo  "Qualifications: ".$row['Qualifications'];?><p>
          </div>
        </div>
      </div>

    <?php
  }

}
else{
  //Error message in case no records are found
  echo "no Mentors found";
}

?>
</body>
</html>
<?php
//Gets the footer
require_once 'footer.php';
?>
```

6. sessions.php



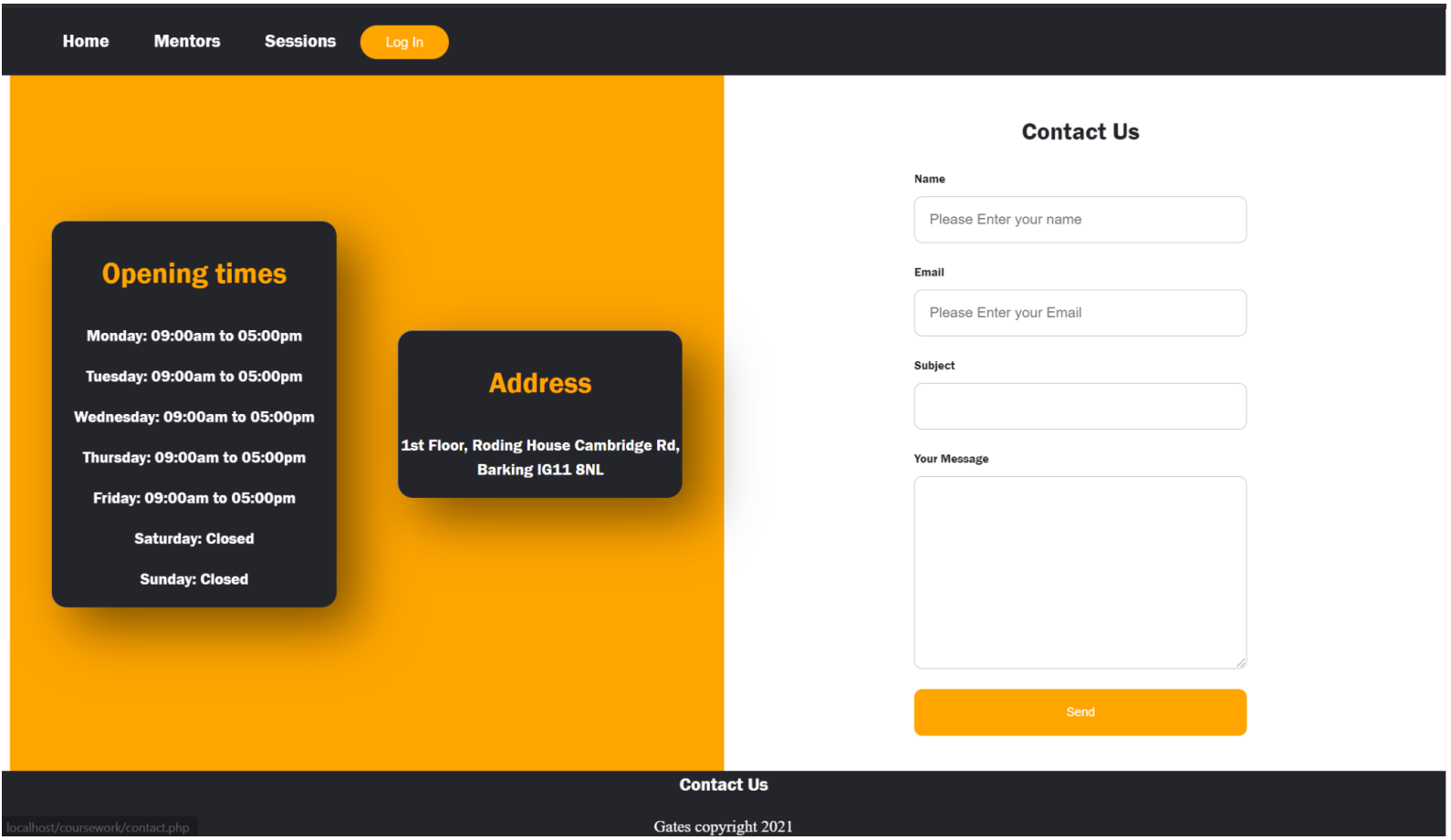
Code for sessions.php

sessions.php

```php
<?php
//Starts session to provide specific features if the user is logged in
session_start();
//Gets the following file to connect to the database
require_once 'connect.php';
//Get the navigation bar
require_once 'header.php';
if(  isset($_SESSION["ID"])  ){
$usertype = $_SESSION["UserType"];
}
?>
 <html>
    <head>

    <link rel="stylesheet" type="text/css" href="CSS/cards.css">
    <link rel="stylesheet" type="text/css" href="CSS/navbar.css">
    </head>
    <body>
    <style>

    </style>
     <h1 align="center">Sessions Available</h1>
```

```php
<?php
//Select all sessions data and the name of the mentor
$sql = "SELECT * , MForename, MSurname FROM TSession INNER JOIN TMentor where TSession.MentorID = TMentor.MentorID";
$result = $con->query($sql);
if( $result->num_rows > 0){
  //loops used to display all the sessions data and the corresponding mentor's name
  while( $row = $result->fetch_assoc()){
    ?>


      <div class="container">
        <div class="card">
          <div class="content">
            <?php
            echo "<input type='hidden' value='". $row['SessionID']."' name='SessionID' />";
            echo "<input type='hidden' value='". $row['Sessiondetails']."' name='sessiondetails' />";
            ?>
            <h3><?php echo $row['SessionName'];?></h3>
            <p><?php echo "Mentor: ".$row['MForename']." ".$row['MSurname'];?></p>
            <p><?php echo  "Session Length(hour): ".$row['SessionLength'];?></p>
            <p><?php echo  "Price: £".$row['Price'];?><p>
            <p><?php echo  "Details: ".$row['Sessiondetails'];?><p>



              <?php
              if   (!isset($_SESSION["ID"])){
                echo "<td><a href='login.php?id=".$row['SessionID'] . "' class='navigation'>Book</a></td>";

              }
              elseif ( $usertype=="Client" ){
                echo "<td><a href='book.php?id=".$row['SessionID'] . "' class='navigation'>Book</a></td>";
              }

              ?>
          </div>
        </div>
      </div>

    <?php
  }

}
else{
  //Error message displayed if there are no sessions found on the sessions table
  echo "no sessions found";
}

?>
</body>
</html>
<?php
//Gets the footer
require_once 'footer.php';
?>

```

7. contact.php



Code for contact.php

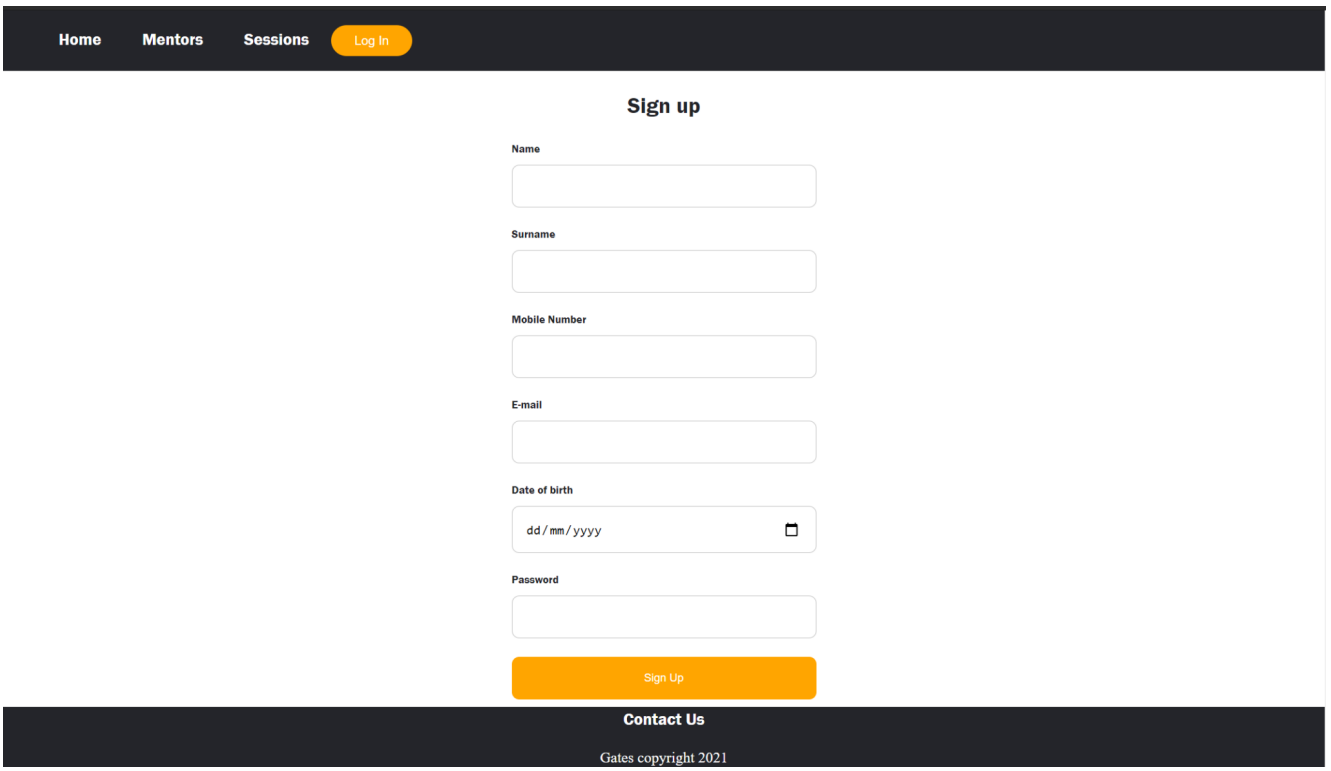
contact.php

```php
<?php
//Starts session to ensure features if user is logged in
session_start();
//Connection to databse
require_once 'connect.php';
//Gets the navigation bar/ header
require_once 'header.php';
//Creates an array to store the error messages.
$Error = array();
//Does the following the user click the confirm button.
if(isset($_POST['Confirm'])){
    //Checks if any required field is empty
    if(empty($_POST['Name']) || empty($_POST['Email']) || empty($_POST['Subject'])|| empty($_POST['Message'])){
        $Error['empty']="Please fill all of the fields" ;
    }
    else{
        // Sets variables needed and uses function to provide secury against SQL injection
        $Name = mysqli_real_escape_string($con,$_POST['Name']);
        $From= mysqli_real_escape_string($con,$_POST['Email']);
        $Subject =mysqli_real_escape_string($con,$_POST['Subject']);
        $Message = mysqli_real_escape_string($con,$_POST['Message']);
```

```
        //Check if the users name only consists if letters
        if (!preg_match ("/^[a-zA-Z]*$/", $Name )) {
            $Error['CForename']="Forename should only contain letters";
        }
        //Checks if the user provided an authentic/proper emaail
        if (!filter_var($From,FILTER_VALIDATE_EMAIL)){
            $Error['WrongEmail']= "There seems to be something wrong with your email";
        }
        //Does the following/Sends email if there are no errors in the data provided to the user
        if (count($Error)===0){
        $To = "Gates.barking@cw.net";
        $header = "From: ".$From;
        $body = "Email received from ".$Name.".\n\n".$Message;
        mail($To, $Subject, $body, $header);
        header("Location: index.php?Emailsent");
        }
    }
}
?>
<html>
    <head>
    <link rel="stylesheet" type="text/css" href="CSS/split.css">
    <link rel="stylesheet" type="text/css" href="CSS/form.css">
    <link rel="stylesheet" type="text/css" href="CSS/cards.css">
    </head>
    <body>
    <div class="split">
        <div class="left">
            <div class="main">
                <div class="time">
                    <h3 align="center">Opening times</h3>
                    <p align="center">Monday: 09:00am to 05:00pm </p>
                    <p align="center">Tuesday: 09:00am to 05:00pm </p>
                    <p align="center">Wednesday: 09:00am to 05:00pm </P>
                    <p align="center">Thursday: 09:00am to 05:00pm </P>
                    <p align="center">Friday: 09:00am to 05:00pm </P>
                    <p align="center" class="weekend">Saturday: Closed</P>
                    <p align="center" class="weekend">Sunday: Closed</P>

                </div>
                <div class="address">
                    <h3 align="center">Address</h3>
                    <p align="center">1st Floor, Roding House Cambridge Rd, Barking IG11 8NL</p>
                </div>
            </div>

        </div>
        <div class="right">
            <div class="main">
```

```
71                  <form name="formUser" method="POST" action="">
72                      <h2>Contact Us</h2>
73                      <div class= "input-container name">
74                          <label for="Name">Name</label>
75                          <input type="text" name="Name" placeholder="Please Enter your name" value="<?php echo isset($_POST["Name"]) ? $_POST["Name"] : ''; ?>">
76                      </div>
77                      <div class= "input-container email">
78                          <label for="Email">Email</label>
79                          <input type="email" name="Email" placeholder="Please Enter your Email" value="<?php echo isset($_POST["name"]) ? $_POST["Email"] : ''; ?>">
80                      </div>
81                      <div class= "input-container subject">
82                          <label for="Subject">Subject</label>
83                          <input type="text" name="Subject" value="<?php echo isset($_POST["name"]) ? $_POST["Subject"] : ''; ?>">
84                      </div>
85                      <div class= "input-container message">
86                          <label for="Message">Your Message</label>
87                          <textarea id="message" name="Message" rows="10" cols="40" value="<?php echo isset($_POST["name"]) ? $_POST["Message"] : ''; ?>">
88                          </textarea>
89                      </div>
90                      <button class="signup-button" type="submit" name="Confirm" value="submit">Send</button>
91                      <?php
92                      //outputs errors to the user
93                      if (count($Error)>0){
94                          foreach ($Error as $Error) :
95                              echo $Error ;
96                              ?>
97                              <br>
98                              <?php
99                          endforeach;
100                         }
101                     ?>
102                 </form>
103             <div>
104         </div>
105
106     </div>
107
108     </body>
109
110
111 </html>
112 <?php
113 //Gets the footer
114 require_once 'footer.php';
115 ?>
```

8. signup.php



Code for signup.php

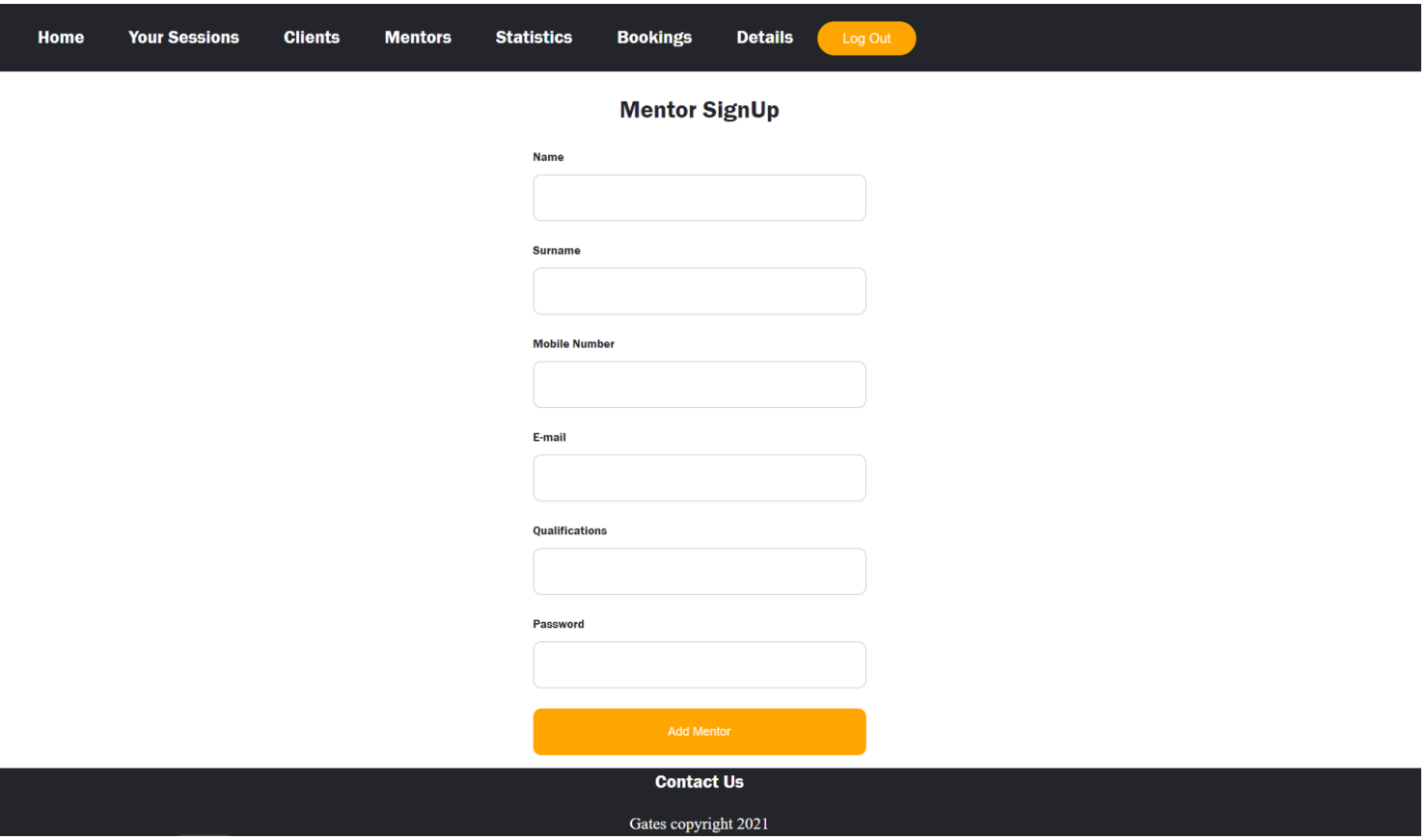
signup.php

```php
<?php
//Starts session to provide specific features if the user is logged in
session_start();
//Gets the following file to connect to the database
require_once 'connect.php';
//Get the navigation bar
require_once 'header.php';
//Creates an array to store error messages
$Error=array();
//if user click submit
if(isset($_POST['Signup'])){
        $Forename= mysqli_real_escape_string($con,$_POST['CForename']);
        $Surname =mysqli_real_escape_string($con,$_POST['CSurname']);
        $MobileNumber = mysqli_real_escape_string($con,$_POST['CMobileNumber']);
        $Email =mysqli_real_escape_string($con, $_POST["Email"]);
        $DateOfB = $_POST['DateOfB'];
        $Password =  password_hash($_POST['Password'], PASSWORD_DEFAULT);
    //check if required fields are empty
    if(empty($Forename) || empty($Surname) || empty($MobileNumber) || empty($Email)|| empty($DateOfB)|| empty($Password)){
        $Error['empty']="Please fill all of the fields" ;
    }
    else{
        //Check if the forename consists of only letters
        if (!preg_match ("/^[a-zA-Z]*$/",$Forename )) {
            $Error['CForename']="Forename should only contain letters";
        }
        //check if the surname consists of only letter
        if (!preg_match ("/^[a-zA-Z]*$/",$Surname) ) {
            $Error['CSurname']="Surname should only contain letters";
        }
        //checks if the mobile number consists of only numbers
        if (!preg_match ("/^[0-9]*$/", $MobileNumber) ) {
            $Error['CMobileNumber']="Mobile number should consist only of numbers";
        }
        //Check is the mobile number is 11 digits
        elseif (strlen($MobileNumber)!== 11){
        $Error['CMobilenumlength']="Mobile Number should be 11 digits";
        }
        //check if there is already an account registered with the same email.
        $Emailcheck = "SELECT *FROM TLogin WHERE Email = '$Email'";
        $resultemail = $con->query($Emailcheck);
        //If the query returns a result it means that there is already a similiar email registered
        if( $resultemail->num_rows > 0){
            $Error['Emailexist']= "You are already registered with us";
        }
        //Ensure that a proper email in enteres
        if (!filter_var($Email,FILTER_VALIDATE_EMAIL)){
            $Error['WrongEmail']= "There seems to be something wrong with your email";
        }
```

```
        //set the age a client is required to have by policy to be able to book sessions and compare with clients date of birth
        $Agerequired=new DateTime('-15 years');
        $DateOfBirth = new DateTime($DateOfB);
        //Compares the DOB entered by the user and the restriction
        if ($Agerequired<$DateOfBirth){
            $Error['DateOfB']= "You are too young to have an account.";
        }
        if(!preg_match('@[A-Z]@', $_POST["Password"]) || !preg_match('@[a-z]@', $_POST["Password"])
           || !preg_match('@[0-9]@', $_POST["Password"]) || !preg_match('@[^\w]@', $_POST["Password"])){
            $Error['Password']= "Password needs to have at least 1 upper letter character,
            1 lower case letter,1 number and a special character";
        }
        }


        if (count($Error)===0){
        //add the client's personal details to the clients table
        $sql = "INSERT INTO TClient(CForename,CSurname,CMobileNumber,DateOfB)
        VALUES('$Forename','$Surname','$MobileNumber','$DateOfB')";
        $con->query($sql);
        $sql = "SELECT ClientID FROM TClient ORDER BY ClientID DESC";
        $result = $con->query($sql);
        $row = mysqli_fetch_array($result);
        $clientID = $row[0];
        //adds the login details to the login table
        $sql = "INSERT INTO TLogin(UserType, Email, Password, ID) VALUES('Client','$Email','$Password', '$clientID')";
        if( $con->query($sql) === TRUE){
            header('Location: login.php');
        }
        else{
        echo "<p align=center>Error: There was an error while registering user</div>";

        }
    }
}
?>
<html>
    <title>User signup</title>
    <head>
        <link rel="stylesheet" type="text/css" href="CSS/form.css">
    </head>
    <body class="formbody">
        <div class="main">
            <form name="formUser" method="POST" action="">
                <h2>Sign up</h2>
                <div class= "input-container name">
                    <label for="CForename">Name</label>
                    <input type="text" name="CForename" value="<?php echo isset($_POST["CForename"]) ? $_POST["CForename"] : ''; ?>">
                </div>
                <div class= "input-container surname">
                    <label for="CSurname">Surname</label>
                    <input type="text" name="CSurname" value="<?php echo isset($_POST["CSurname"]) ? $_POST["CSurname"] : ''; ?>">
                </div>
                <div class= "input-container mobile">
                    <label for="CMobileNumber">Mobile Number</label>
                    <input type="tel" name="CMobileNumber" value="<?php echo isset($_POST["CMobileNumber"]) ? $_POST["CMobileNumber"] : ''; ?>">
                </div>
                <div class= "input-container email">
                    <label for="Email">E-mail</label>
                    <input type="email" name="Email" value="<?php echo isset($_POST["Email"]) ? $_POST["Email"] : ''; ?>">
                </div>
                <div class= "input-container dob">
                    <label for="DateOfB">Date of birth</label>
                    <input type="date" name="DateOfB" value="<?php echo isset($_POST["DateOfB"]) ? $_POST["DateOfB"] : ''; ?>">
                </div>
                <div class= "input-container password">
                    <label for="Password">Password</label>
                    <input type="Password" name="Password">
                </div>
```

```
119                 <button class="signup-button" type="submit" name="Signup" value="submit">Sign Up</button>
120                 <div class ="login">
121                     <p>already have an account? <a class = "a-form" href="login.php">
122                     <strong>Login</strong></a></p>
123                 </div>
124                 <?php
125                 //outputs errors to the user
126                 if (count($Error)>0){
127                     foreach ($Error as $Error) :
128                         echo $Error ;
129                         ?>
130                         <br>
131                         <?php
132                     endforeach;
133                     }
134                 ?>
135             </form>
136         </div>
137     </body>
138 </html>
139 <?php
140 // Gets the footer
141 require_once 'footer.php';
142 ?>
```

9. msignup.php



Code for msignup.php

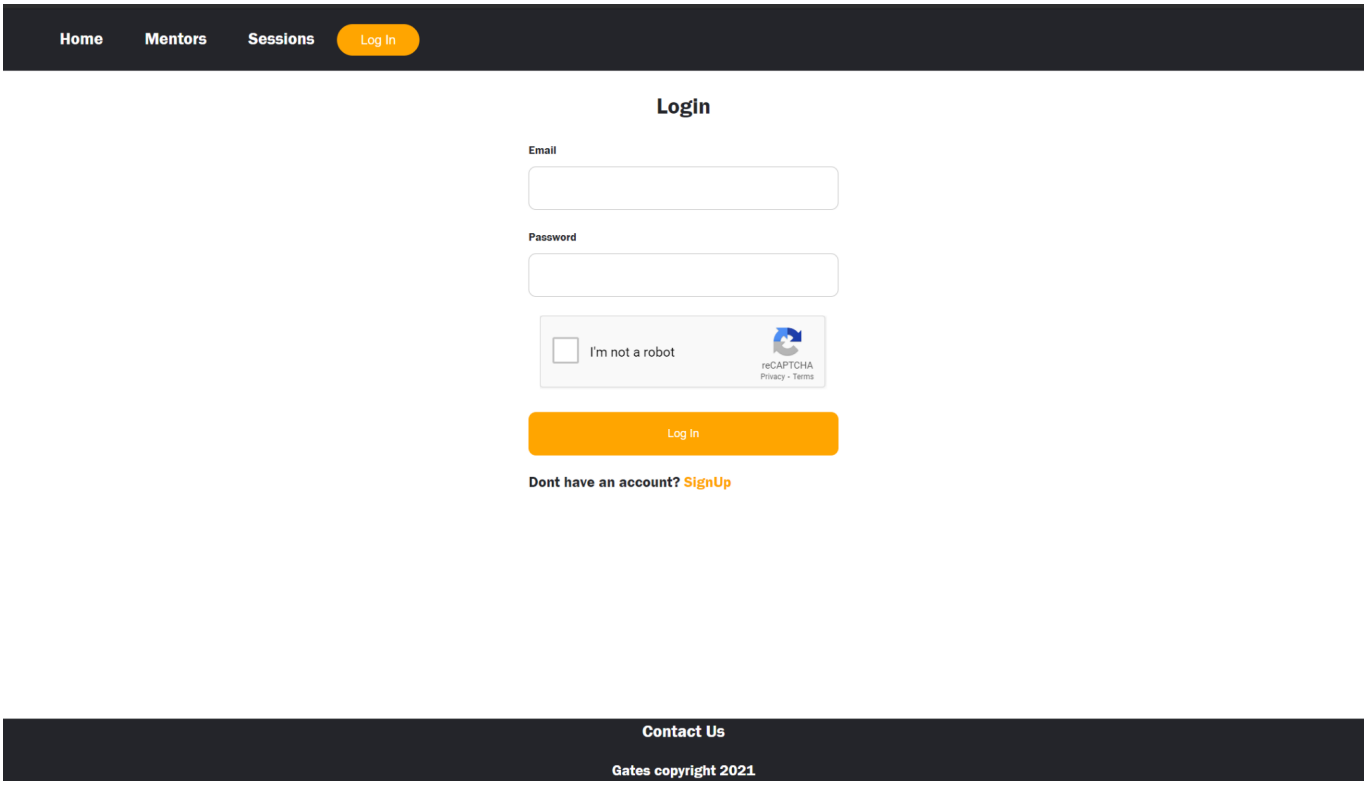
msignup.php

```php
<?php
//Starts session to provide specific features if the user is logged in
session_start();
//Gets the following file to connect to the database
require_once 'connect.php';
//Get the navigation bar
require_once 'header.php';
//Creates an array to store error messages
$Error=array();
//Check if the user the admin
if(  isset($_SESSION["UserType"]) and  $_SESSION["UserType"]=="Admin") {

    if(isset($_POST['Submit'])){
        $Forename= mysqli_real_escape_string($con,$_POST['MForename']);
        $Surname = mysqli_real_escape_string($con,$_POST['MSurname']);
        $MobileNumber = mysqli_real_escape_string($con,$_POST['MMobileNumber']);
        $Email = mysqli_real_escape_string($con,$_POST["Email"]);
        $Qualifications =mysqli_real_escape_string($con, $_POST["Qualifications"]);
        $Password =  password_hash($_POST['Password'], PASSWORD_DEFAULT);
        //Checks if there are any empty fields
        if(empty($Forename)||empty($Surname)||empty($MobileNumber )||empty($Qualifications)||empty($Email )||Sempty( $Password)){
        $Error['Empty']="Please fill all of the required fields";
        }
        else{
                    //Checks that forename consists of only letters
            if (!preg_match ("/^[a-zA-Z]*$/",$Forename )) {
                $Error['MForename']="Forename should only contain letters";
            }
            //Checks that the surname contains only letters
            if (!preg_match ("/^[a-zA-Z]*$/",$Surname) ) {
                $Error['MSurname']="Surname should only contain letters";
            }
            //Checks that the mobile number consists of only numbers
            if (!preg_match ("/^[0-9]*$/", $MobileNumber) ) {
                $Error['MMobileNumber']="Mobile number should consist only of numbers";
            }
            //Checks that mobile number is 11 digits long
            elseif (strlen($_POST['MMobileNumber'])!== 11){
                $Error['MMobilenumlength']="Mobile Number should be 11 digits";
            }
            //check if there is already an account registered with the same email.
            $Emailcheck = "SELECT *FROM TLogin WHERE Email = '$Email'";
            $resultemail = $con->query($Emailcheck);
            if( $resultemail->num_rows > 0){
                $Error['Emailexist']= "You are already registered with us";
            }
            //Checks if the user entered a valid email
            if (!filter_var($Email,FILTER_VALIDATE_EMAIL)){
                $Error['WrongEmail']= "There seems to be something wrong with your email";
            }
```

```
        if(!preg_match('@[A-Z]@', $_POST["Password"]) || !preg_match('@[a-z]@', $_POST["Password"])
        || !preg_match('@[0-9]@', $_POST["Password"]) || !preg_match('@[^\w]@', $_POST["Password"])){
        $Error['Password']= "Password needs to have at least 1 upper letter character,
        1 lower case letter,1 number and a special character";
        }

    }
    //Checks if there arent any error
    if (count($Error)===0){
        //Proceeds to do the following if there arent any errors
        //Query that inserts the peronal details of the mentor on the mentors table
        $sql = "INSERT INTO TMentor(MForename,MSurname,MMobileNumber,Qualifications)
        VALUES('$Forename','$Surname','$MobileNumber','$Qualifications')";
        $con->query($sql);
        $sql = "SELECT MentorID FROM TMentor ORDER BY MentorID DESC";
        $result = $con->query($sql);
        $row = mysqli_fetch_array($result);
        $mentorID = $row[0];
        //Query that inserts the mentor's email and password, id and usertype(mentor) on the login table
        $sql = "INSERT INTO TLogin(UserType, Email, Password, ID) VALUES('Mentor','$Email','$Password', '$mentorID')";
        if( $con->query($sql) === TRUE){
            //redirects to the admin version of the mentors table if the query is executed properly
            header("location:adminmentors.php");
        }
        else{
            //Error message displayed if there is an error when executing the query
            echo "<p align=center>Error: There was an error while registering the mentor</div>";

        }
    }
}

    ?>

<html>
<title>Mentor Signup</title>
    <head>
        </html><link rel="stylesheet" type="text/css" href="CSS/form.css">
    </head>
    <body class= "formbody">
        <div class="main">
            <form name="formUser" method="POST" action="">
                <h2>Mentor SignUp</h2>
                <div class= "input-container name">
                    <label for="MForename">Name</label>
                    <input type="text" name="MForename" value="<?php echo isset($_POST["MForename"]) ? $_POST["MForename"] : ''; ?>">
                </div>
                <div class= "input-container surnamename">
                    <label for="MSurname">Surname</label>
                    <input type="text" name="MSurname" value="<?php echo isset($_POST["MSurname"]) ? $_POST["MSurname"] : ''; ?>">
                </div>
                <div class= "input-container mobile">
                    <label for="MMobileNumber">Mobile Number</label>
                    <input type="tel" name="MMobileNumber" value="<?php echo isset($_POST["MMobileNumber"]) ? $_POST["MMobileNumber"] : ''; ?>">
                </div>
                <div class= "input-container email">
                    <label for="Email">E-mail</label>
                    <input type="email" name="Email" value="<?php echo isset($_POST["Email"]) ? $_POST["Email"] : ''; ?>">
                </div>
                <div class= "input-container qualifications">
                    <label for="Qualifications">Qualifications</label>
                    <input type="text" name="Qualifications" value="<?php echo isset($_POST["Qualifications"]) ? $_POST["Qualifications"] : ''; ?>">
                </div>
                <div class= "input-container Password">
                    <label for="Password">Password</label>
                    <input type="password" name="Password">
                </div>
```

```
                    <button class="signup-button" type="submit" name="Submit" value="submit">Add Mentor</button>
                    <?php
                        //Check if there is any error and display them if there are
                        if (count($Error)>0){
                            foreach ($Error as $Error) :
                                echo $Error ;
                                ?>
                                <br>
                                <?php
                            endforeach;
                        }
                        ?>
                    </form>
                </div>
            </body>
        </html>
    <?php
    }
    else{
        //redirects the user to the login page if they are npt logged in as the admin
        header("location:login.php");
    }
    //Gets the footer
    require_once 'footer.php';
    ?>
```

10. login.php



Code for login.php

login.php

```php
<?php
//Starts session to provide specific features if the user is logged in
session_start();
//Gets the following file to connect to the database
require_once 'connect.php';
//Get the navigation bar
require_once 'header.php';
//Creates an array to store error messages
$Errors=array();
//if user is already logged in they are redirected to homepage
if(  isset($_SESSION["ID"])  ){
   header("location:index.php");
}
//if user click submit
if(isset($_POST['Login'])){
   //Set the keys needed for google captcha
   $SecretKey = "6LfjYHEaAAAAANU8_hk5tPhp-M18EbhYLXOP8BHS";
   $ResponseKey = $_POST['g-recaptcha-response'];
   $IP= $_SERVER['REMOTE_ADDR'];
   $url = "https://www.google.com/recaptcha/api/siteverify?secret=$SecretKey&response=$ResponseKey&remoteip=$IP";
   $response = file_get_contents($url);
   $response = json_decode($response);
   //Checks if the required fields are empty
   if(empty($_POST['Email']) || empty($_POST['Password']) ){
      $Errors['Empty'] = "Please fillout all the required fields";
   }
   else{
      //check if the user is a human
      if($response ->success) {
         $Email = $_POST['Email'];
         $Password = $_POST['Password'];
         //check that the email exist
         $Emailcheck = "SELECT *FROM TLogin WHERE Email = '$Email'";
         $result = $con->query($Emailcheck);
         if( $result->num_rows > 0){

            $row = mysqli_fetch_assoc($result);
            //compare the password the user have enter to the one on the database by de-hashing it, if it matches logs the user in.
            if(password_verify($Password, $row['Password'])==TRUE) {
               $_SESSION["ID"] = $row['ID'];
               $_SESSION["UserType"] = $row['UserType'];
               $usertype = $_SESSION["UserType"];
               //if the password matches with the hashed password on the database, logs user in and redirects to the home/index page
               header("location:index.php");
            }
            else {
               $Errors['nomatch']="Password and email do not match";
            }

         }
         else{
            $Errors['nonexistent']="You are not registered with us";
         }
      }
      else{
         $Errors['verification']="Please verify yourself";

      }
   }
}
```

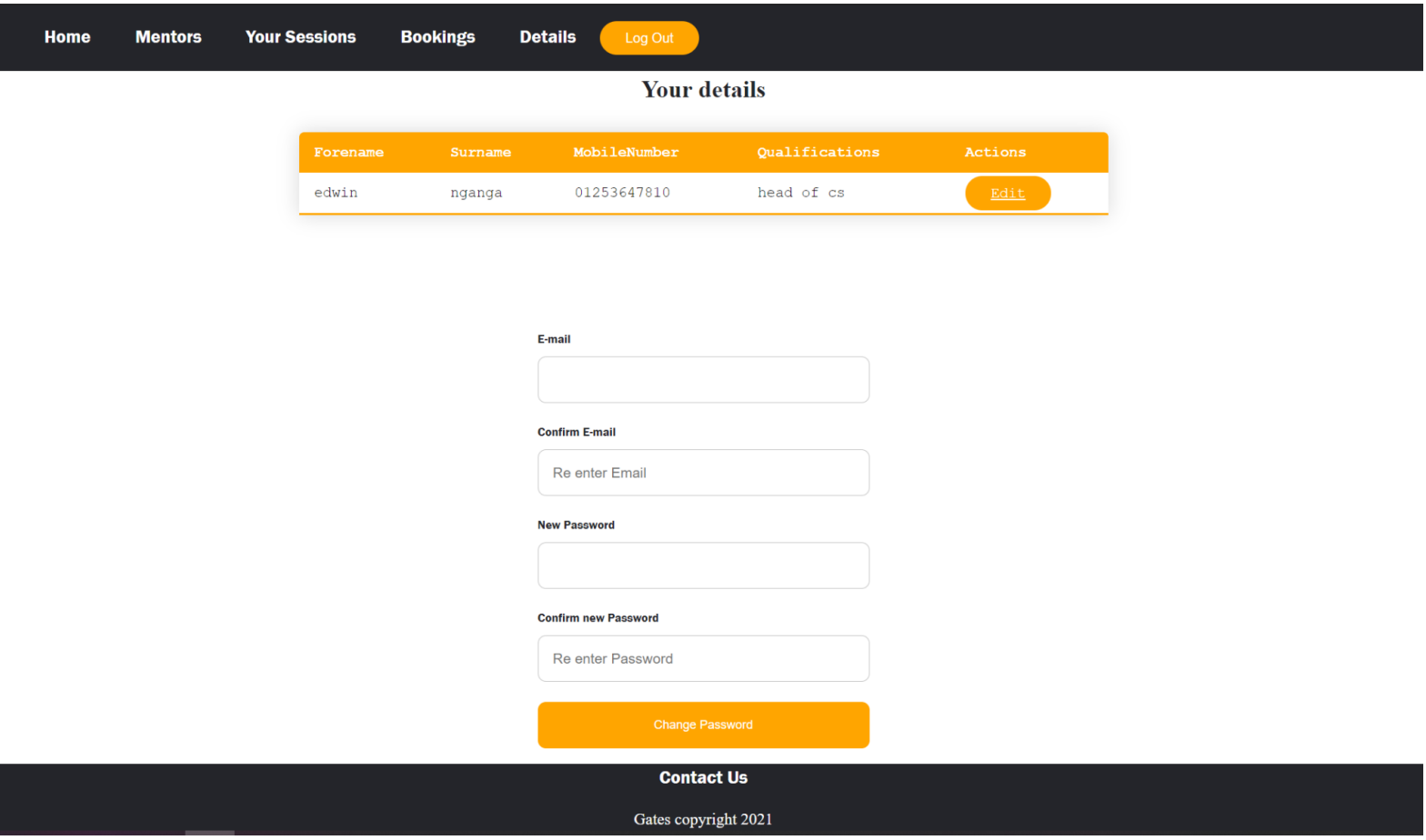
```
61  ?>
62  <html>
63      <head>
64      <title>User Login</title>
65      <script src="https://www.google.com/recaptcha/api.js" async defer></script>
66      <link rel="stylesheet" type="text/css" href="CSS/form.css">
67
68      </head>
69      <body class="formbody">
70          <div class="main">
71              <form name="frmUser" method="post" action="">
72                  <h2> Login </h2>
73                  <div class= "input-container email">
74                      <label for="Email">Email</label>
75                      <input type="email" name="Email" value="<?php echo isset($_POST["Email"]) ? $_POST["Email"] : ''; ?>">
76                  </div>
77                  <div class= "input-container password">
78                      <label for="Password">Password</label>
79                      <input type="Password" name="Password">
80                  </div>
81                  <div align ="center" class="g-recaptcha" data-sitekey="6LfjYHEaAAAAACrj0aF-QTmnBG90i2WPwR_F6x_m"></div>
82                  <br>
83                  <button class="signup-button" type="submit" name="Login" value="submit">Log In</button>
84                  <div class ="signup">
85                      <p>Dont have an account? <a class="a-form" href="signup.php">
86                      <strong>SignUp</strong></a></p>
87                      <?php
88                      //Check if there are any errors and display them if there are
89                      if (count($Errors)>0){
90                          foreach ($Errors as $Errors) :
91                              echo $Errors ;
92                              ?>
93                              <br>
94                              <?php
95                          endforeach;
96                      }
97                      ?>
98                  </div>
99              </form>
100         <div>
101     </body>
102 </html>
103 <?php
104 //Gets the footer
105 require_once 'footer.php';
106 ?>
107
108
109
```

11. logout.php

logout.php

```php
<?php
session_start();
//logs the user out of the system by unsetting their ID'S
unset($_SESSION["ID"]);
unset($_SESSION["UserType"]);
header("Location:index.php");
?>
```

12. details.php



Code for details.php

details.php

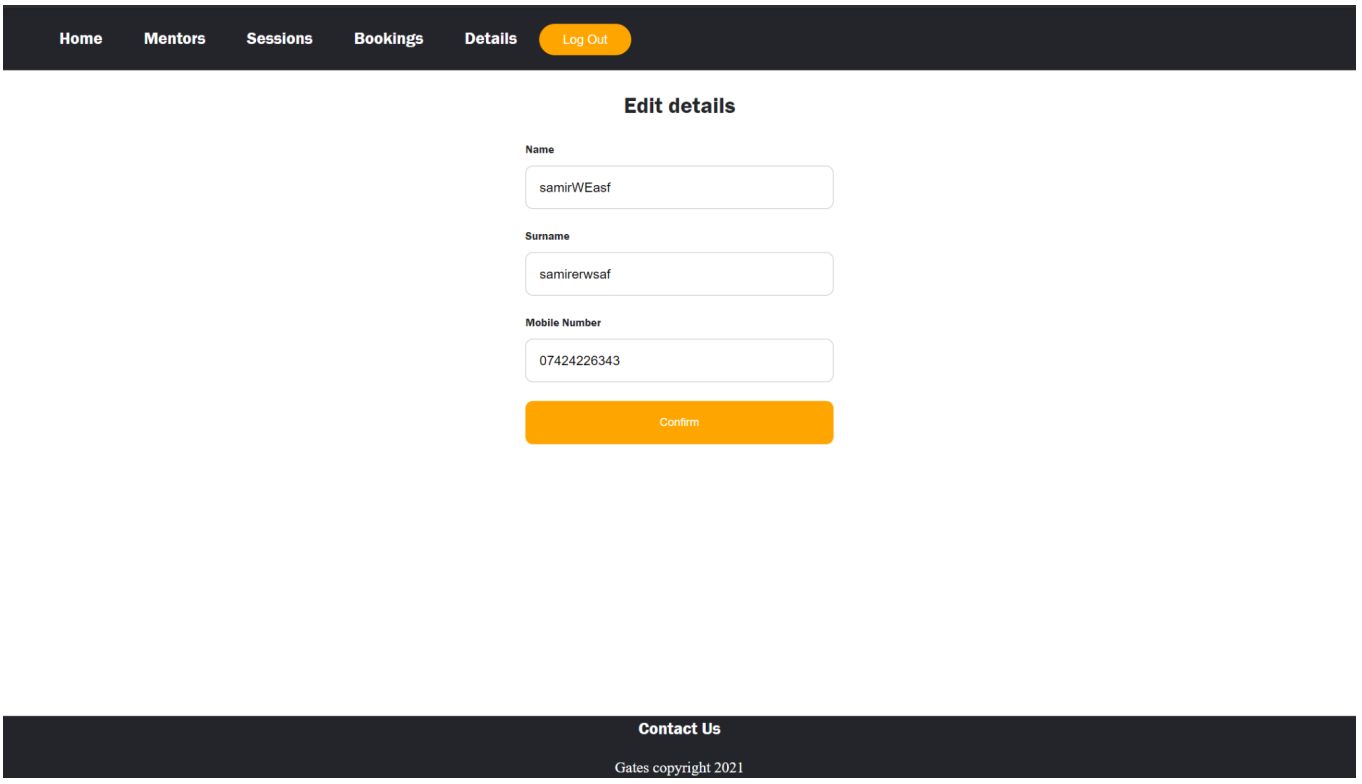
```
<html>
    <title>Details</title>
    <head>
        <link rel="stylesheet" type="text/css" href="CSS/table.css">
        <link rel="stylesheet" type="text/css" href="CSS/form.css">
        <link rel="stylesheet" type="text/css" href="CSS/navbar.css">
    </head>
<?php
//Starts session to provide specific features if the user is logged in
session_start();
//Gets the following file to connect to the database
require_once 'connect.php';
//Get the navigation bar
require_once 'header.php';
if(isset($_SESSION["ID"])){
  $id = $_SESSION["ID"];
  $usertype = $_SESSION["UserType"];
  //Sets the connection to the database
  $con = mysqli_connect('localhost','root','','s1901072') or die('Unable To connect');;
  $Error=array();
  //Check if the user is a mentor or the admin
  if ($usertype=="Mentor" || $usertype=="Admin"):
    //Gets the user's details
    $sql =" SELECT * FROM TMentor WHERE MentorID =  '$id'";
    $result = $con->query($sql);
    //check if the user is a client
  else:
    //gets the user's details
     $sql =" SELECT * FROM TClient WHERE ClientID =  '$id'";
     $result = $con->query($sql);

  endif;
  ?>
   <body>
      <h2 align = "center">Your details</h2>



  <table class="table">
  <thead>
  <tr>
  <th>Forename</th>
  <th>Surname</th>
  <th>MobileNumber</th>
  <?php
  if ($usertype=="Mentor"||$usertype=="Admin"):
  ?>
  <th>Qualifications</th>
  <?php
  elseif ($usertype=="Client"):
  ?>
  <th>Date of birth</th>
  <?php
  endif
  ?>
  <th>Actions</th>
  </tr>
  <thead>
  <tbody>
```

```
60
61      <?php
62      //Check if there are actually the required details in the database
63      if( $result->num_rows > 0):
64        //Gets the details and show them on the table
65        while( $row = $result->fetch_assoc()){
66          echo "<form action='' method='POST'>";
67          //Check if the user is a mentor or the admin
68          if ($usertype=="Mentor"||$usertype=="Admin"):
69
70          echo "<tr>";
71          echo "<td>".$row['MForename'] . "</td>";
72          echo "<td>".$row['MSurname'] . "</td>";
73          echo "<td>".$row['MMobileNumber'] . "</td>";
74          echo "<td>".$row['Qualifications'] . "</td>";
75          //check if the user is a client
76          elseif($usertype=="Client"):
77
78            echo "<tr>";
79            echo "<td>".$row['CForename'] . "</td>";
80            echo "<td>".$row['CSurname'] . "</td>";
81            echo "<td>".$row['CMobileNumber'] . "</td>";
82            echo "<td>".$row['DateOfB'] . "</td>";
83
84          endif;
85          //button to redirect to edit page where personal details can be changed
86          echo "<td><a href='edit.php' class='navigation'>Edit</a></td>";
87
88        echo "</tr>";
89        echo "</form>";
90          }
91        ?>
92         </tbody>
93        </table>
94
95        <br>
96        <?php
97
98      else:
99      // Error message in case there are no details on the databse
100     echo "<br><br>No Record Found";
101       endif;
102       ?>
103
104     <?php
105     if (isset($_POST['Confirm'])){
106       if(empty($_POST["Email"])|| empty($_POST["ConfirmEmail"]) || empty($_POST['Password']) ||empty($_POST['ConfirmPassword'])){
107         $Error['empty']="Please fill all of the fields" ;
108       }
109         else{
110           if($usertype=="Client"){
111           $sql =" SELECT * FROM TLogin WHERE ID =  '$id' AND UserType='Client'";
112           }
113           elseif($usertype=="Mentor"){
114             $sql =" SELECT * FROM TLogin WHERE ID =  '$id' AND UserType='Mentor'";
115           }
116           else{
117             $sql =" SELECT * FROM TLogin WHERE ID =  '$id' AND UserType='Admin'";
118           }
119           $result = $con->query($sql);
```

```
        if( $result->num_rows > 0){
          $row = $result->fetch_assoc();

            if( $_POST["Email"]!==$row['Email']){
              $Error['wrongemail']= "Wrong email entered";
            }
            if($_POST["Email"]!==$_POST["ConfirmEmail"]){
              $Error['Emailnomatch']= "Emails do not match";
            }
            if(!preg_match('@[A-Z]@', $_POST["Password"]) || !preg_match('@[a-z]@', $_POST["Password"])
              || !preg_match('@[0-9]@', $_POST["Password"]) || !preg_match('@[^\w]@', $_POST["Password"])){
            $Error['Password']= "Password needs to have at least 1 upper letter character,
            1 lower case letter,1 number and a special character";
            }
            if($_POST['Password']!==$_POST['ConfirmPassword']){
              $Error['passwordnomatch']= "Passwords do not match";
            }
            if (count($Error)===0){
              $Password = password_hash($_POST['Password'], PASSWORD_DEFAULT);
              if($usertype=="Client"){
              $sql = "UPDATE TLogin SET Password='{$Password}' WHERE ID= '$id' AND UserType='Client'";
              }
              elseif($usertype=="Mentor"){
                $sql = "UPDATE TLogin SET Password='{$Password}' WHERE ID= '$id' AND UserType='Mentor'";
              }
              else{
                $sql = "UPDATE TLogin SET Password='{$Password}' WHERE ID= '$id' AND UserType='Admin'";
              }
              //redirects to the details page if the query is executed is executed with no errors
              if( $con->query($sql) === TRUE){
                  header('Location: index.php');
              }
              else{
                  //Error in case the query is not executed properly
                  echo "<div class='alert alert-danger'>Error: There was an error while updating user info</div>";
              }
            }
        }
      }
    }
    ?>
    <body class="formbody">
        <div class="main">

            <form name="formUser" method="POST" action="">

            <div class= "input-container email">
                    <label for="Email">E-mail</label>
                    <input type="email" name="Email">
                </div>
                <div class= "input-container confirmemail">
                    <label for="Email"> Confirm E-mail</label>
                    <input type="email" name="ConfirmEmail" placeholder="Re enter Email">
                </div>
                <div class= "input-container password">
                    <label for="Password"> New Password</label>
                    <input type="Password" name="Password">
                </div>
                <div class= "input-container password">
                    <label for="Password"> Confirm new Password</label>
                    <input type="Password" name="ConfirmPassword" placeholder="Re enter Password">
                </div>
```

```
                    <button class="signup-button" type="submit" name="Confirm" value="submit">Change Password</button>
                    <?php
                  //outputs errors to the user
                  if (count($Error)>0){
                      foreach ($Error as $Error) :
                          echo $Error ;
                          ?>
                          <br>
                          <?php
                      endforeach;
                      }
                  ?>
                    </form>
              </div>
          </body>
      <?php
    ?>
    <body>
    </html>
    <?php
}
else{
    //redirects to the login page if user is not logged in.
    header("Location: login.php");
}
//Gets the footer
require_once 'footer.php';
?>
```

13. edit.php



Code for edit.php

edit.php

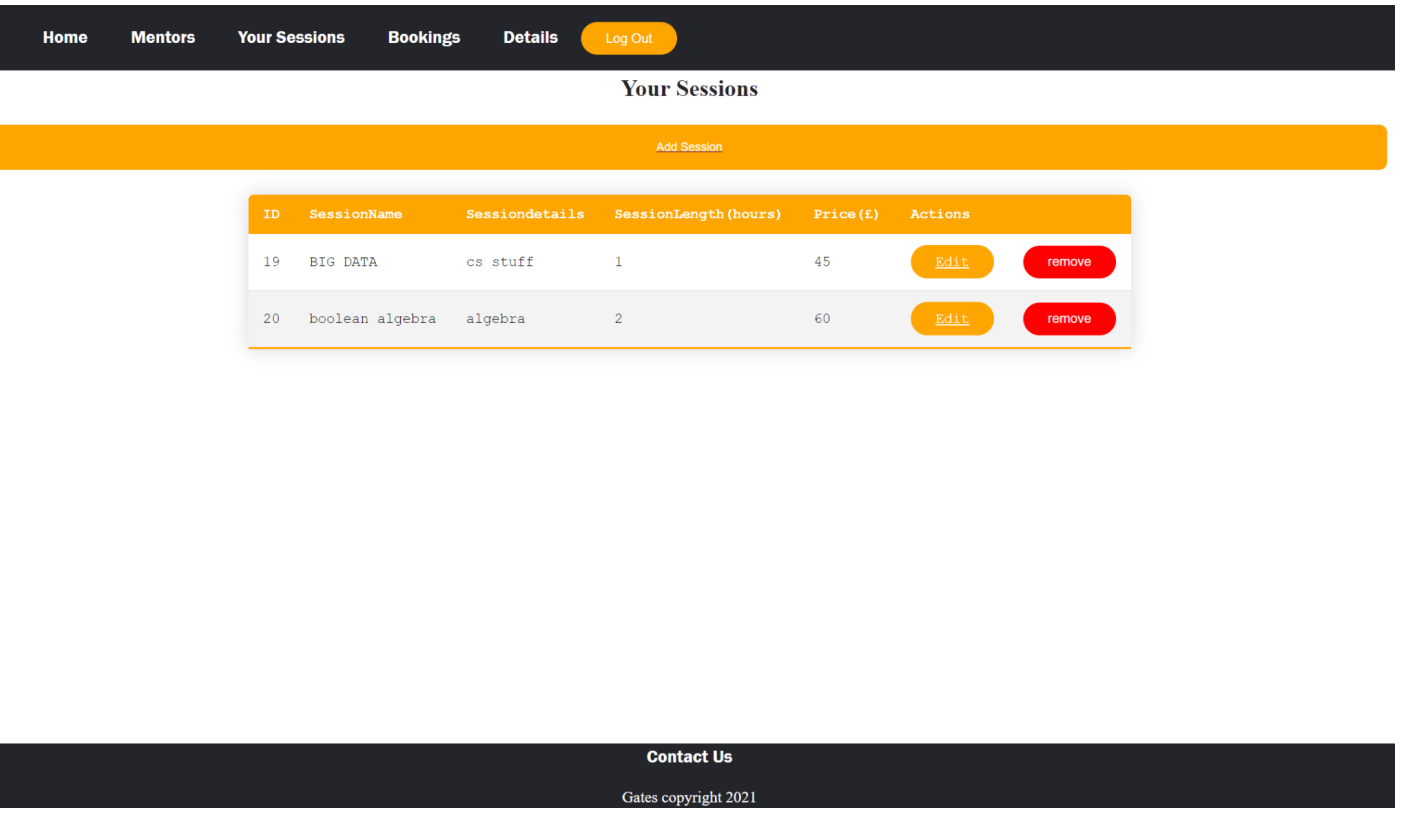
```php

<?php
//Starts session to provide specific features if the user is logged in
session_start();
//Gets the following file to connect to the database
require_once 'connect.php';
//Get the navigation bar
require_once 'header.php';
if(isset($_SESSION["ID"])){
    $id = $_SESSION["ID"];
    $usertype = $_SESSION["UserType"];
    //Creates an array storing error messages
    $Error = array();
    ?>
    <html>
        <title>Edit details</title>
        <head>
            <link rel="stylesheet" type="text/css" href="CSS/form.css">
            <link rel="stylesheet" type="text/css" href="CSS/navbar.css">
        </head>
    <?php
    //Check if the user is a client
    if ($usertype=="Client"){

        if(isset($_POST['update'])){
            //Does the following if the user clicks the update button
            //Checks if any required fields are empty
            if( empty($_POST['CForename']) || empty($_POST['CSurname']) || empty($_POST['CMobileNumber'])){
                $Error['Empty']="Please fill all of the fields";
            }
            else{

                $Forename = mysqli_real_escape_string($con,$_POST['CForename']);
                $Surname = mysqli_real_escape_string($con,$_POST['CSurname']);
                $MobileNumber = mysqli_real_escape_string($con,$_POST['CMobileNumber']);

                //Checks if the forename contains only letters
                if (!preg_match ("/^[a-zA-Z]*$/",$Forename )) {
                    $Error['CForename']="Forename should only contain letters";
                }
                //checks if the surname only contain letters
                if (!preg_match ("/^[a-zA-Z]*$/",$Surname) ) {
                    $Error['CSurname']="Surname should only contain letters";
                }
                //Check if the mobile number only contains numbers
                if (!preg_match ("/^[0-9]*$/", $MobileNumber) ) {
                    $Error['CMobileNumber']="Mobile number should consist only of numbers";
                }
                //Checks if the mobile number is 11 digits
                elseif (strlen($_POST['CMobileNumber'])!== 11){
                $Error['CMobilenumlength']="Mobile Number should be 11 digits";
                }
                //Proceeds if there are no erros in the data entered by the user
                if (count($Error)===0){
                    //Query that update the users details
                    $sql = "UPDATE TClient SET CForename='{$Forename}', CSurname = '{$Surname}',
                    CMobileNumber = '{$MobileNumber}'
                    WHERE ClientID= '$id'";
                    //redirects to the details page if the query is executed is executed with no errors
                    if( $con->query($sql) === TRUE){
                        header('Location: details.php');
                    }
```

```
                else{
                    //Error in case the query is not executed properly
                    echo "<div class='alert alert-danger'>Error: There was an error while updating user info</div>";
                }
            }


        }
    }
        //Retrieve the Client details from the database
        $sql = "SELECT * FROM TClient WHERE ClientID = '$id'";
        $result = $con->query($sql);
        //If there are no personal details of th user on the clients table the user is redirected to the index page
        if($result->num_rows < 1){
        header('Location: index.php');
        }

    $row = $result->fetch_assoc();
    ?>
    <div class = "main">
        <form name="formUser" method="POST" action="">
            <h2>Edit details</h2>
            <div class= "input-container name">
            <label for="CForename">Name</label>
            <input type="text" name="CForename" value="<?php echo $row['CForename']; ?>" class="form-control" >
            </div>
            <div class= "input-container surname">
            <label for="CSurname">Surname</label>
            <input type="text" name="CSurname" value="<?php echo $row['CSurname']; ?>" class="form-control">
            </div>
            <div class= "input-container mobile">
            <label for="CMobileNumber">Mobile Number</label>
            <input type="tel" name="CMobileNumber" value="<?php echo $row['CMobileNumber']; ?>" class="form-control"  >
            </div>
            <button class="signup-button" type="submit" name="update" value="update">Confirm</button>
            <?php
            // If there are errors they are displayed to the user
            if (count($Error)>0){

                foreach ($Error as $Error) :
                    echo $Error ;
                    ?>
                    <br>
                    <?php
                endforeach;
            }
            ?>
        </form>

    </div>

    <?php
}
//Checks if the user is a mentor or the admin
elseif ($usertype=="Mentor" or $usertype=="Admin"){
    if(isset($_POST['update'])){
        //Does the following if the user clicks the update button
        //Checks if any required field is empty
```

```
        if( empty($_POST['MForename']) || empty($_POST['MSurname']) || empty($_POST['MMobileNumber']) || empty($_POST['Qualifications']) ){
            $Error['Empty']="Please fill all of the fields";
        }
        else{
            $Forename = mysqli_real_escape_string($con,$_POST['MForename']);
            $Surname = mysqli_real_escape_string($con,$_POST['MSurname']);
            $MobileNumber = mysqli_real_escape_string($con,$_POST['MMobileNumber']);
            $Qualifications = mysqli_real_escape_string($con,$_POST['Qualifications']);
            //Checks if the forename only contains letters
            if (!preg_match ("/^[a-zA-Z]*$/",$Forename )) {
                $Error['MForename']="Forename should only contain letters";
            }
            //checks if the surname only contain letters
            if (!preg_match ("/^[a-zA-Z]*$/",$Surname) ) {
                $Error['MSurname']="Surname should only contain letters";
            }
            //check if the mobile number only contains numbers
            if (!preg_match ("/^[0-9]*$/", $MobileNumber) ) {
                $Error['MMobileNumber']="Mobile number should consist only of numbers";
            }
            //checksi the mobile number is 11 digits
            elseif (strlen($_POST['MMobileNumber'])!== 11){
            $Error['MMobilenumlength']="Mobile Number should be 11 digits";
            }

            if (count($Error)===0){
                //proceed if the are no errors in the data enterd by the user
                //Query that update the users details
                $sql = "UPDATE TMentor SET MForename='{$Forename}', MSurname = '{$Surname}',
                MMobileNumber = '{$MobileNumber}', Qualifications = '{$Qualifications}'
                WHERE MentorID= '$id'";
                //If the query is executed properly the user is redirected to the details page
                if( $con->query($sql) === TRUE){
                    header('Location: details.php');
                }
                else{
                    //display error message s if the query was not executed properly
                echo "<div class='alert alert-danger'>Error: There was an error while updating user info</div>";
                }
            }
        }
    }
    //Retrieve the mentor/admin details
    $sql = "SELECT * FROM TMentor WHERE MentorID = '$id'";
    $result = $con->query($sql);
    //If there are no peronal details on the mentor table/(no results) the user is redirected to home page
    if($result->num_rows < 1){
    header('Location: index.php');
    }

    $row = $result->fetch_assoc();
    ?>
    <div class = "main">
    <form name="formUser" method="POST" action="">
        <h2>Edit details</h2>
        <div class= "input-container name">
        <label for="MForename">Name</label>
        <input type="text" name="MForename" value="<?php echo $row['MForename']; ?>" class="form-control">
        </div>
        <div class= "input-container surname">
        <label for="MSurname">Surname</label>
        <input type="text" name="MSurname" value="<?php echo $row['MSurname']; ?>" class="form-control">
```

```
            <input type="text" name="MSurname" value="<?php echo $row['MSurname']; ?>" class="form-control">
            </div>
            <div class= "input-container mobile">
            <label for="MMobileNumber">Mobile Number</label>
            <input type="tel" name="MMobileNumber" value="<?php echo $row['MMobileNumber']; ?>" class="form-control">
            </div>

            <div class= "input-container Qualifications">
            <label for="Qualifications">Qualifications</label>
            <input type="text" name="Qualifications" value="<?php echo $row['Qualifications']; ?>" class="form-control">
            </div>
            <button class="signup-button" type="submit" name="update" value="update">Confirm</button>
            <?php
            //Checks if there are nay errors and display them if there are
            if ($Error>0){

                foreach ($Error as $Error) :
                    echo $Error ;
                    ?>
                    <br>
                    <?php
                endforeach;
            }
            ?>
        </form>

        </div>
        <?php
    }
}
else{
    header('Location: login.php');
}
//Gets the footer
require_once 'footer.php';
?>
```

14. ownsessions.php



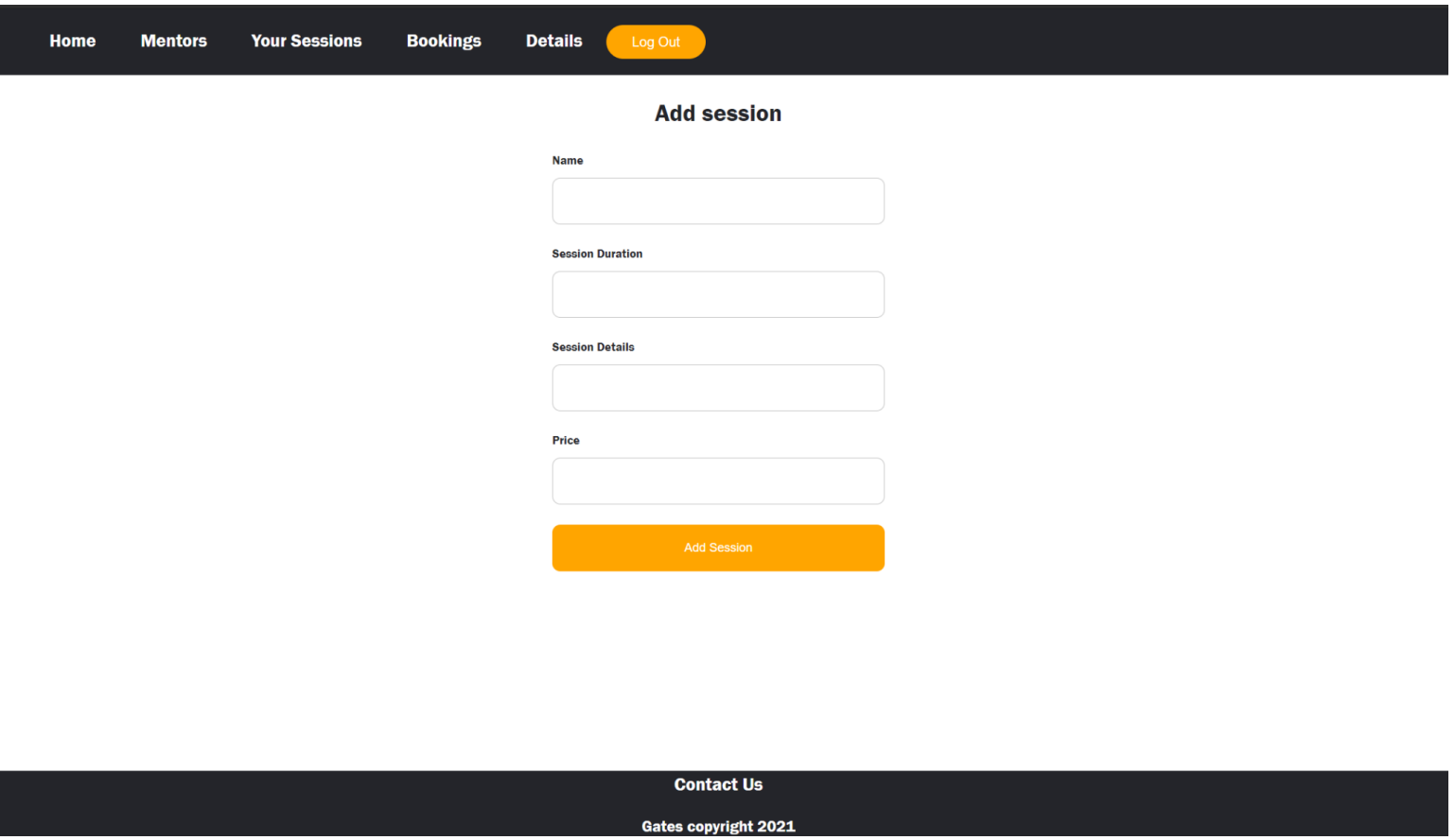
Code for ownsessions.php

ownsessions.php

```
<html>
        <head>
        <link rel="stylesheet" type="text/css" href="CSS/table.css">
        <link rel="stylesheet" type="text/css" href="CSS/form.css">
        <link rel="stylesheet" type="text/css" href="CSS/navbar.css">
        </head>
<?php
//Starts session to provide specific features if the user is logged in
session_start();
if(  isset($_SESSION["ID"])  ){
    $usertype = $_SESSION["UserType"];
    //Checks that the user is a mentor or the admin
    if ($usertype=="Mentor"||$usertype=="Admin"){

        //Gets the following file to connect to the database
        require_once 'connect.php';
        //Get the navigation bar
        require_once 'header.php';
        $ID = $_SESSION["ID"];
        if( isset($_POST['remove'])){
            //Does the following if the user clicks the remove button
            $sessionid =  $_POST['SessionID'];
                //Query that deletes the session data of the selected session
                $sql = "DELETE FROM TSession WHERE SessionID = $sessionid";
                if($con->query($sql) === TRUE){
                    //user is redirected to the ownsessions page if the query is executed correctly
                    header("Location:ownsessions.php");
                echo "<div class='alert alert-success'>Successfully removed session </div>";
                }

        }
        // Selects all the sessions taught by corresponding the mentor from the database
        $sql = "SELECT * FROM TSession WHERE TSession.MentorID=$ID ";
        $result = $con->query($sql);
        ?>
        <h2 align = "center">Your Sessions</h2>


            <a class="cta" href="sessionmentor.php"><button class="signup-button">Add Session</button></a>

            <?php
        if( $result->num_rows > 0):
            ?>

            <table class="table">
            <thead>
            <tr>
            <td>ID</td>
            <td>SessionName</td>
            <td>Sessiondetails</td>
            <td>SessionLength(hours)</td>
            <td >Price(£)</td>
            <td>Actions</td>
            <td></td>
            </tr>
            </thead>
```

```
57
58              <tbody>
59              <?php
60              //loop used to show all the corresponding sessions
61              while( $row = $result->fetch_assoc()){
62              echo "<form action='' method='POST'>";
63              echo "<input type='hidden' value='". $row['SessionID']."' name='SessionID' />";
64              echo "<tr>";
65              echo "<td>".$row['SessionID'] . "</td>";
66              echo "<td>".$row['SessionName'] . "</td>";
67              echo "<td>".$row['Sessiondetails'] . "</td>";
68              echo "<td>".$row['SessionLength'] . "</td>";
69              echo "<td>".$row['Price'] . "</td>";
70              //Action buttons the mentor can use to edit and remove their own sessions
71              echo "<td><a href='editsession.php?id=".$row['SessionID']."' class='navigation'>Edit</a></td>";
72              echo "<td><input type='submit' name='remove' value='remove' class='remove' </td>";
73              echo "</tr>";
74              echo "</form>";
75              }
76
77          else:
78          //Display massage if the mentor doesnt have any sessions
79          echo "<br><br>No Record Found";
80          endif;
81          ?>
82          </tbody>
83          </table>
84          <?php
85      }
86      else{
87          header("location:index.php");
88
89      }
90
91  }
92  else{
93      header("location:login.php");
94
95  }
96  //Gets the footer
97  require_once 'footer.php';
98
99  ?>
```

15. sessionmentor.php



Code for sessionmentor.php

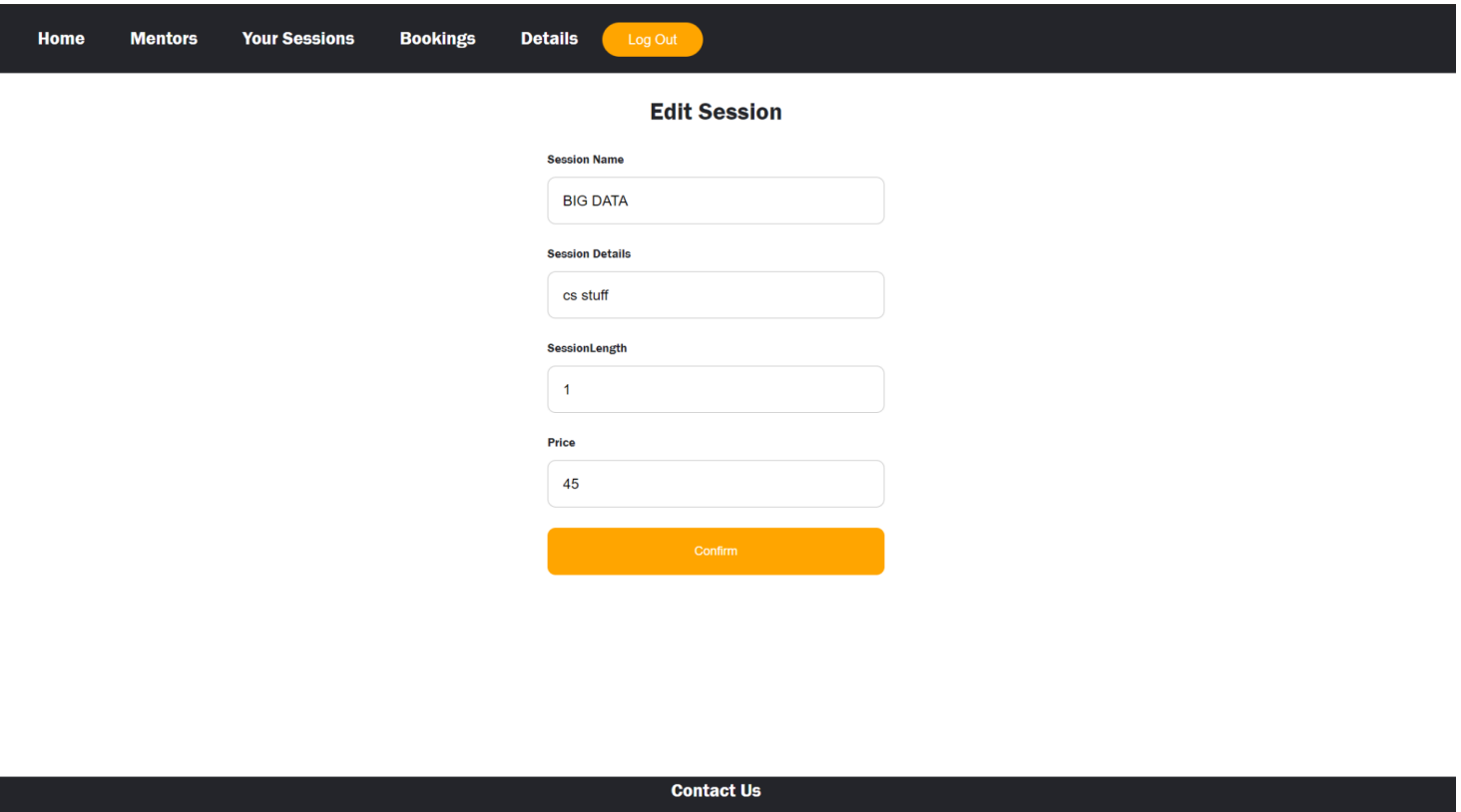
sessionmentor.php

```php
<?php
session_start();
//Gets the following file to connect to the database
require_once 'connect.php';
//Get the navigation bar
require_once 'header.php';
if(  (isset($_SESSION["ID"]))) {
   $usertype = ($_SESSION["UserType"]) ;
   //checks if the user logged in is the admin or a mentor
   if ($usertype=="Admin"|| $usertype== "Mentor"){

      $id = $_SESSION["ID"];
      $Error = array();
      if(isset($_POST['Submit'])){
         //check if any field is empty
         if(empty($_POST['SessionName']) || empty($_POST['Price']) || empty($_POST['Sessiondetails']) || empty($_POST['SessionLength']) ){
            $Error['Empty']="Please fill all of the required fields";
         }
         else{
            $SessionName= $_POST['SessionName'];
            $SessionLength = $_POST['SessionLength'];
            $Price = $_POST["Price"];
            $Sessiondetails = $_POST['Sessiondetails'];
```

```php
            //check if the session is between 1 and 3 hours
            if ($SessionLength<"1"||$SessionLength>"3") {
               $Error['Length']="Session should be 1, 2 or 3 hours long";
            }
            if (!preg_match ("/^[0-9]*$/",$Price )) {
               $Error['Price']="Price should only be a numerical value";
            }
           //If there are no errors add the session to the database
            if (count($Error)===0){
               $sql = "INSERT INTO TSession(SessionName,SessionLength,Price,Sessiondetails,MentorID)
                VALUES('$SessionName','$Price','$SessionLength','$Sessiondetails','$id')";

               if( $con->query($sql) === TRUE){
                  header("location:ownsessions.php");
               }
               else{
               echo "<p align=center>Error: There was an error while adding session</div>";
               echo $sql;
               }
            }
      }
   }
   ?>

   <html>
      <title>Add session</title>
      <head>
         <link rel="stylesheet" type="text/css" href="CSS/form.css">
      </head>
         <body class="formbody">
            <div class="main">
               <form name="formUser" method="POST" action="" >
                  <h2>Add session</h2>
                  <div class= "input-container name">
                  <label for="SessionName">Name</label>
                  <input type="text" name="SessionName" value="<?php echo isset($_POST["SessionName"]) ? $_POST["SessionName"] : ''; ?>">
                  </div>
                  <div class= "input-container duration">
                  <label for="SessionLength">Session Duration</label>
                  <input type="number" name="SessionLength" value="<?php echo isset($_POST["SessionLength"]) ? $_POST["SessionLength"] : ''; ?>">
                  </div>
                  <div class= "input-container details">
                  <label for="Sessiondetails">Session Details</label>
                  <input type="text" name="Sessiondetails" value="<?php echo isset($_POST["Sessiondetails"]) ? $_POST["Sessiondetails"] : ''; ?>">
                  </div>
                  <div class= "input-container price">
                  <label for="Price">Price</label>
                  <input type="number" name="Price" value="<?php echo isset($_POST["Price"]) ? $_POST["Price"] : ''; ?>">
                  </div>
                  <button class="signup-button" type="submit" name="Submit" value="submit">Add Session</button>
                  <?php
                  //If there are errors, display them to the user
                  if (count($Error)>0){
                    echo"Unable to add session";
                    foreach ($Error as $Error) :
                       echo $Error ;
                       ?>
                       <br>
                       <?php
                    endforeach;
```

```
                    }
                    ?>
                </form>
            <div>
        </body>
    </html>
    <?php
    }
    else{
        header("location:index.php");
    }
}
else{
 header("location:login.php");
}

require_once 'footer.php';
?>

```

16. editsession.php



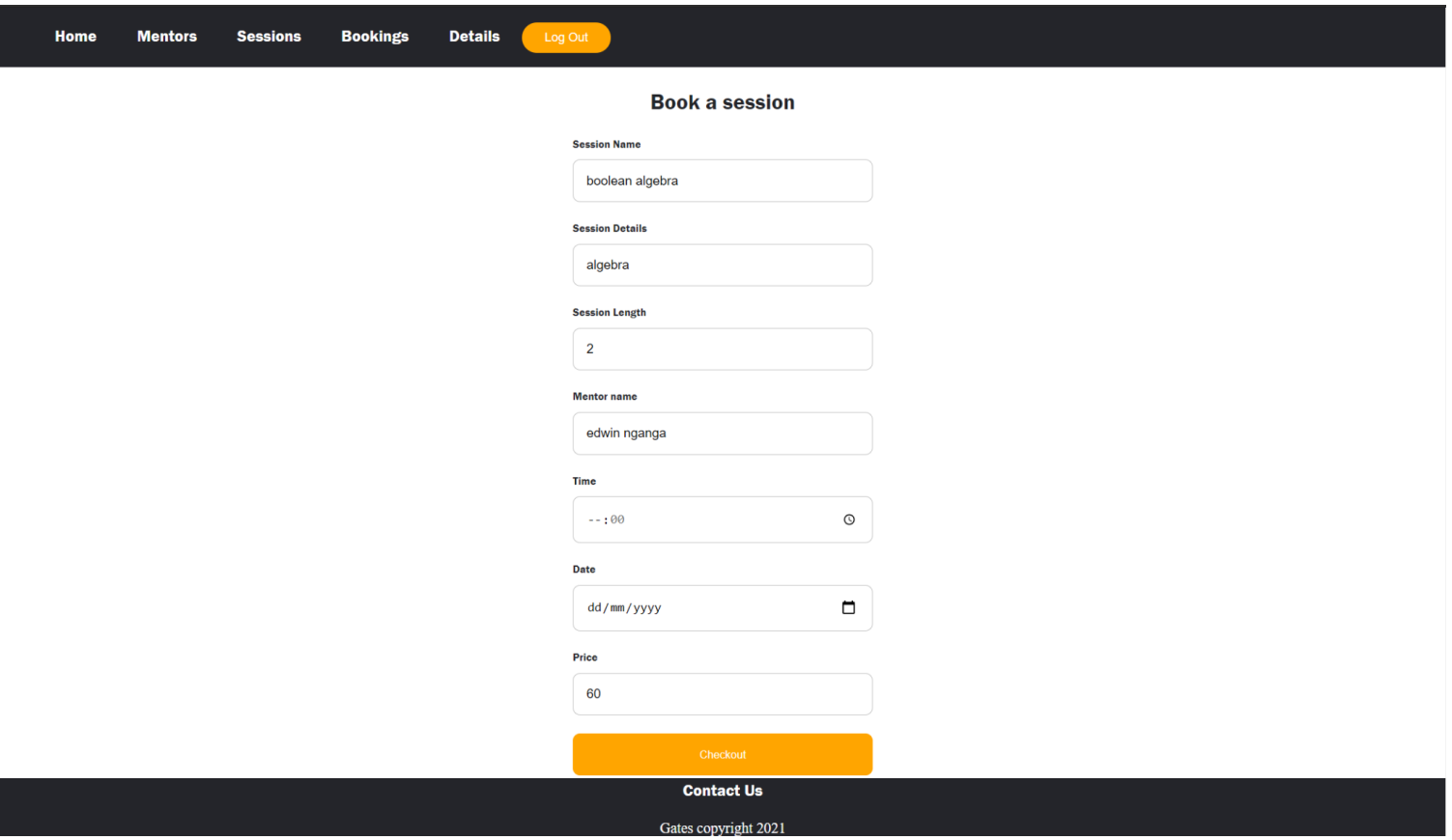
Code for editsession.php

editsession.php

```php
<?php
//Starts session to provide specific features if the user is logged in
session_start();
//Gets the following file to connect to the database
require_once 'connect.php';
//Get the navigation bar
require_once 'header.php';
//Creates an array to store error messages
$Error=array();
?>
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="CSS/form.css">
    </head>
<title> Edit session</title>
<?php

if(isset($_POST['update'])){
    //Does the following if the user clicks the update button
    //Checks if any required fields are empty
    if( empty($_POST['SessionName']) || empty($_POST['SessionLength']) ||
        empty($_POST['Sessiondetails']) || empty($_POST['Price']) )
        {
            $Error['Empty']="Please fill all the required fields";
    }
    else{
        $SessionName = mysqli_real_escape_string($con,$_POST['SessionName']);
        $SessionLength =mysqli_real_escape_string($con, $_POST['SessionLength']);
        $Price = mysqli_real_escape_string($con,$_POST['Price']);
        $Sessiondetails = mysqli_real_escape_string($con,$_POST['Sessiondetails']);

        //Checks that session length is either 1,2 or 3
        if (!preg_match ("/^[1-3]*$/",$SessionLength )) {
            $Error['Length']="Session should be 1, 2 or 3 hours long";
        }
        //Checks if the price consist only of numbers
        if (!preg_match ("/^[0-9]*$/",$Price )) {
            $Error['Price']="Price should only be a numerical value";
        }
        //Checks if there any errors and proceed if there arent any
        if (count($Error)===0){
            //Query which update the selected session
            $sql = "UPDATE TSession SET SessionName='{$SessionName}', SessionLength = '{$SessionLength}',
            Price = '{$Price}', Sessiondetails = '{$Sessiondetails}'
            WHERE SessionID= " . $_POST['SessionID'];
            if( $con->query($sql) === TRUE){
            echo "<div class='alert alert-success'>Successfully updated session</div>";
            //redirects to the ownsessions page if query is executed succeessfully.
            header('Location: ownsessions.php');
            }
            else{
                //eror message in case the query is not executed properly
            echo "<div class='alert alert-danger'>Error: There was an error while updating user info</div>";
            }
        }
    }
}
```

```php
$id = isset($_GET['id']) ? (int) $_GET['id'] : 0;
//Gets the session details
$sql = "SELECT * FROM TSession WHERE SessionID={$id}";
$result = $con->query($sql);
if($result->num_rows <1  ){
//redirects to the homepage of there are any errors
header('Location: index.php');
exit;
}
$row = $result->fetch_assoc();
?>
    <div class="main">
      <body class="formbody">
          <form  action="" method="POST">
              <h2>Edit Session</h2>
              <input type="hidden" value="<?php echo $row['SessionID']; ?>" name="SessionID">
              <div class= "input-container name">
                  <label for="SessionName">Session Name</label>
                  <input type="text" id="SessionName" name="SessionName" value="<?php echo $row['SessionName']; ?>" class="form-control">
              </div>
              <div class= "input-container details">
                  <label for="Sessiondetails">Session Details</label>
                  <input type="text" name="Sessiondetails" id="Sessiondetails" value="<?php echo $row['Sessiondetails']; ?>" class="form-control">
              </div>
              <div class= "input-container length">
                  <label for="SessionName">SessionLength</label>
                  <input type="number" id="SessionLength" name="SessionLength" value="<?php echo $row['SessionLength']; ?>" class="form-control">
              </div>
              <div class= "input-container Price">
                  <label for="Price">Price</label>
                  <input type="number" name="Price" id="Price" value="<?php echo $row['Price']; ?>" class="form-control">
              </div>
              <button class="signup-button" type="submit" name="update" value="update">Confirm</button>
              <?php
              //Checks if there are any errors and display them if there are
                if (count($Error)>0){
                  foreach ($Error as $Error) :
                      echo $Error ;
                      ?>
                      <br>
                      <?php
                  endforeach;
              }
              ?>
          </form>

      </body>
    </div>
</html>
<?php
//Gets the footer
require_once 'footer.php';
?>
```

17. book.php



Code for book.php

book.php

```php
<?php
//Starts the session if user is logged in.
session_start();
//Gets files need for the checkout API and set the secret key
require_once('C:\xampp\htdocs\coursework\stripe-php-7.75.0\init.php');
\Stripe\Stripe::setApiKey('sk_test_51ISlNDCAIdyEfTc7wGdj9E8OLyyD7O0uTg6SHA3HFSW82D7GlJV7k4u7JCCJMhr3ziYRyJb6PxgFOEsggcQRCUZr00mkx1ugFP');
//connection to the databse
require_once 'connect.php';
//Gets the navigation bar/header.
require_once 'header.php';
$ClientID = $_SESSION["ID"];
//Creates an array which will consist of error messages
$Errors = array();

//Will do the following when user click submit.
if(count($_POST)>0){
```

```php
// Using the following functions to provide security against SQL injection
$Price = mysqli_real_escape_string($con,$_POST['Price']);
$MentorID = mysqli_real_escape_string($con,$_POST['MentorID']);
$SessionID = mysqli_real_escape_string($con,$_POST['SessionID']);
$Name = mysqli_real_escape_string($con,$_POST['SessionName']);
$Price = mysqli_real_escape_string($con,$_POST['Price']);
//creates the checkout session
$session = \Stripe\Checkout\Session::create([
'payment_method_types' => ['card'],
'line_items' => [[
 'price_data' => [
  'currency' => 'gbp',
  'product_data' => [
   "name" => $Name,
  ],
  'unit_amount' => $Price*100,
 ],
 'quantity' => 1,
]],
'mode' => 'payment',
'success_url' => 'http://localhost/coursework/success.php' ,
'cancel_url' => 'http://localhost/coursework/sessions.php',
]);
// Check if the time and date fields are empty
if(empty($_POST['Time']) || empty($_POST['Date']) ){
 $Errors['Empty'] = "Please fillout all the required fields";
}
else{
 $Time= mysqli_real_escape_string($con,$_POST['Time']);
 $Date = mysqli_real_escape_string($con,$_POST['Date']);
 $Length = $_POST['SessionLength'];
 //Converts the date and check what the day of the week it is
  $day = date("D", strtotime($Date));
  //Check if the user chose a day in the weekend(Closed on Saturday and Sunday)
  if($day == 'Sat' || $day == 'Sun'){
   $Errors['Weekend'] = "Can not book during weekend";
  }
 //check the duration of the session and add to the start time to find when the session ends.
   if($Length==1){
    $Duration = strtotime($Time) + 60*60;
   }
   elseif($Length==2){
    $Duration = strtotime($Time) + 2*60*60;
   }
   else{
    $Duration = strtotime($Time) + 3*60*60;
   }
   $Endtime = date('H:i', $Duration);

   //check for double booking(Mentor)
  $query = "SELECT COUNT(*) AS COUNT FROM Tappointment WHERE Date = '".$Date."' And MentorID = '".$MentorID."'
            AND Time<='".$Time."' and Endtime>'".$Time."'";
  $query_result  = mysqli_query($con , $query);
   if ($query_result == TRUE){
    while($row= mysqli_fetch_assoc($query_result)){
     $numrow = $row['COUNT'];
      if($numrow>0){
       $Errors['dbookingM'] = "This mentor is not free at this time";
      }
```

```
        }
    }
    //check for double booking(Mentor)
$query1 = "SELECT COUNT(*) AS COUNT FROM Tappointment WHERE Date = '".$Date."'
        And MentorID = '".$MentorID."' AND Time<'".$Endtime."' and Endtime=>'".$Endtime."'";
$query_result1  = mysqli_query($con , $query1);
    if ($query_result1 == TRUE){
        while($row1= mysqli_fetch_assoc($query_result1)){
         $numrow1 = $row['COUNT'];
          if($numrow1>0){
           $Errors['dbookingM'] = "This mentor is not free at this time";
           }
        }
    }
    //Check for double booking(Client)
$query2 = " SELECT COUNT(*) AS COUNT FROM Tappointment WHERE Date = '".$Date."'
        And ClientID = '".$ClientID."' AND Time<='".$Time."' and Endtime>'".$Time."'";
$query_result2  = mysqli_query($con , $query2);
    if ($query_result2 == TRUE){
        while($row2= mysqli_fetch_assoc($query_result2)){
         $numrow2 = $row2['COUNT'];
            if($numrow2>0){
             $Errors['dbookingC'] = "You already have another session booked for this time1";
            }
        }
    }
     //Check for double booking(Client)
$query3= "SELECT COUNT(*) AS COUNT FROM Tappointment WHERE Date = '".$Date."'
        And ClientID = '".$ClientID."' AND Time<'".$Endtime."' and Endtime=>'".$Endtime."'";
$query_result3  = mysqli_query($con , $query3);
    if ($query_result3 == TRUE){
        while($row3= mysqli_fetch_assoc($query_result3)){
         $numrow3 = $row3['COUNT'];
            if($numrow3){
             $Errors['dbookingC'] = "You already have another session booked for this time";
            }
        }
    }
//Creates restrictions which can be used to compare the time and date.
$restriction = date("Y-m-d",time());
$restrictiontime = date("H-i-s",time());
//Check if the client is trying to book on a past date
    if( $restriction>$Date){
      $Errors['date'] = "Error:Please book a date in the future";//have to write something better
    }
//Check if the client is trying too book on the current date but at a past time
    elseif($restriction==$Date && $restrictiontime>$Time){
    $Errors['time'] = "Error:Please book later in the day";
    }
}
//Do the following if there were no errors in the data entered by the user.
if(count($Errors)===0){
  $_SESSION['SessionID'] = $SessionID;
  $_SESSION['SessionTime'] = $Time;
  $_SESSION['SessionDate'] = $Date;
  $_SESSION['SessionEndTime'] = $Endtime;
  $_SESSION['SessionMentorID'] = $MentorID;
  $_SESSION['SessionPrice'] = $Price;
  ?>
```

```
    <script src="https://js.stripe.com/v3/"></script>
    <script>
var stripe = Stripe('pk_test_51ISlNDCAIdyEfTc7SjzcMFEzzoMWWPIJeZ7pyv6RTKf1iTA3Of00agTRZOJzqogIZSuxobu9x1k2pG1MU3Yh7SF700sJcQJC2h');
      //Will be redirected to the checkout page
      stripe.redirectToCheckout({
        sessionId: "<?php echo $session->id; ?>"

      });


    </script>
    <?php
  }

}

//Get the SessionID
$_SESSION['bookID'] = isset($_GET['id']) ? (int) $_GET['id'] : $_SESSION['bookID'];
//Get the name of the mentor,session name,details,length,price of the session the client wanted to book
$sql = "SELECT *, MForename, MSurname FROM TSession JOIN TMentor WHERE TSession.SessionID={$_SESSION['bookID']}
      and TSession.MentorID = TMentor.MentorID";
$result = $con->query($sql);
if($result->num_rows <1  ){
  $_SESSION['bookID'] = null;

}
//Shows the result from thr query.
$row = $result->fetch_assoc();
?>
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="CSS/form.css">
  </head>
  <title>book</title>
  <body class="formbody">
    <div class="main">
      <form action="book.php" method="POST">
        <h2>Book a session</h2>
        <!--Fill in the details of the session-->
        <input type="hidden" value="<?php echo $row['SessionID']; ?>" name="SessionID">
        <input type="hidden" value="<?php echo $row['MentorID']; ?>" name="MentorID">
        <div class= "input-container name">
          <label for="SessionName">Session Name</label>
          <input type="text" id="SessionName" name="SessionName" value="<?php echo $row['SessionName']; ?>" class="form-control" readonly>
        </div>
        <div class= "input-container details">
          <label for="Sessiondetails">Session Details</label>
          <input type="text" id="Sessiondetails" name="Sessiondetails"  value="<?php echo $row['Sessiondetails']; ?>" class="form-control" readonly>
        </div>
        <div class= "input-container length">
          <label for="SessionLength">Session Length</label>
          <input type="number" id="SessionLength" name="SessionLength" value="<?php echo $row['SessionLength']; ?>" class="form-control" readonly>
        </div>
        <div class= "input-container mname">
          <label for="MForename">Mentor name</label>
          <input type="text" id="MentorName" name="MForename" value="<?php echo $row['MForename']. " ".$row['MSurname'] ; ?> "class="form-control" readonly>
        </div>
        <div class= "input-container time">
          <label for="Time">Time</label>
          <!--Set min and max time so that clients can only book during work hours and only at the beginning of the hour(at 0 minutes)-->
          <input type="time" step="3600"  min = "09:00:00" max = "17:00:00" name="Time">
        </div>
        <div class= "input-container date">
          <label for="Date">Date</label>
          <input type="date"  name="Date">
        </div>
        <div class= "input-container price">
          <label for="Price">Price</label>
          <input type="number" name="Price" id="Price" value="<?php echo $row['Price']; ?>" class="form-control" readonly>
        </div>
```

```
205
206
207         <button class="signup-button" type="submit" name="checkout" value="submit">Checkout</button>
208         <!-- <script>
209             var stripe = Stripe('pk_test_51ISlNDCAIdyEfTc7SjzcMFEzzoMWWPIJeZ7pyv6RTKf1iTA3Of00agTRZOJzqogIZSuxobu9x1k2pG1MU3Yh7SF700sJcQJC2h');
210             const btn = document.getElementById("checkout-button")
211             btn.addEventListener('click', function(e) {
212               e.preventDefault();
213               stripe.redirectToCheckout({
214                 sessionId: "<?php echo $session->id; ?>"
215               });
216             });
217           </script> -->
218         <?php
219         //display errors to the user
220           if (count($Errors)>0){
221             echo"Unable to book session.<br>";
222             foreach ($Errors as $Errors) :
223             echo $Errors ;
224             endforeach;
225           }
226         ?>
227       </form>
228     </div>
229   </body>
230 </html>
231 <?php
232
233 require_once 'footer.php';
234 ?>
```

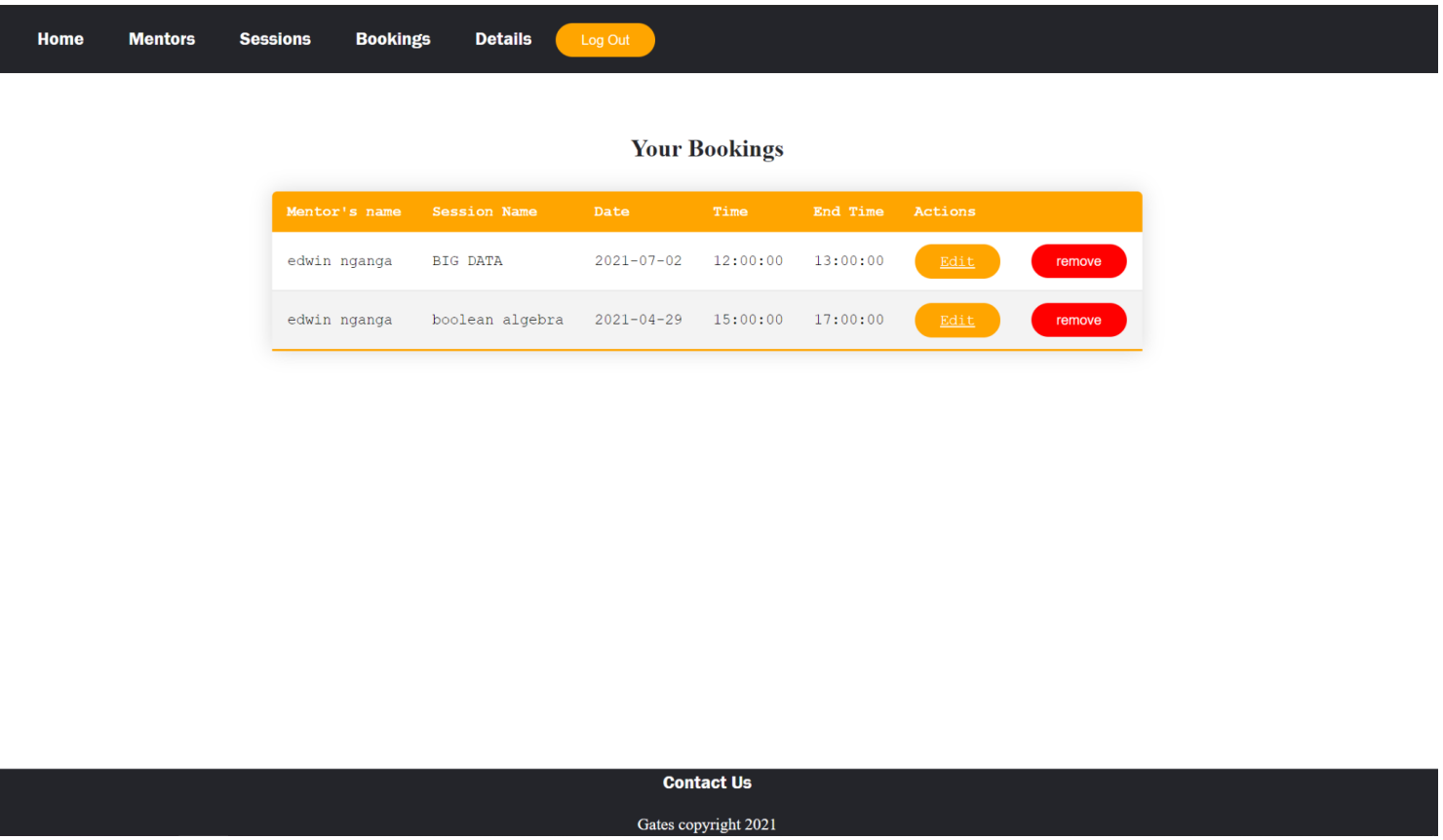
18. success.php

success.php

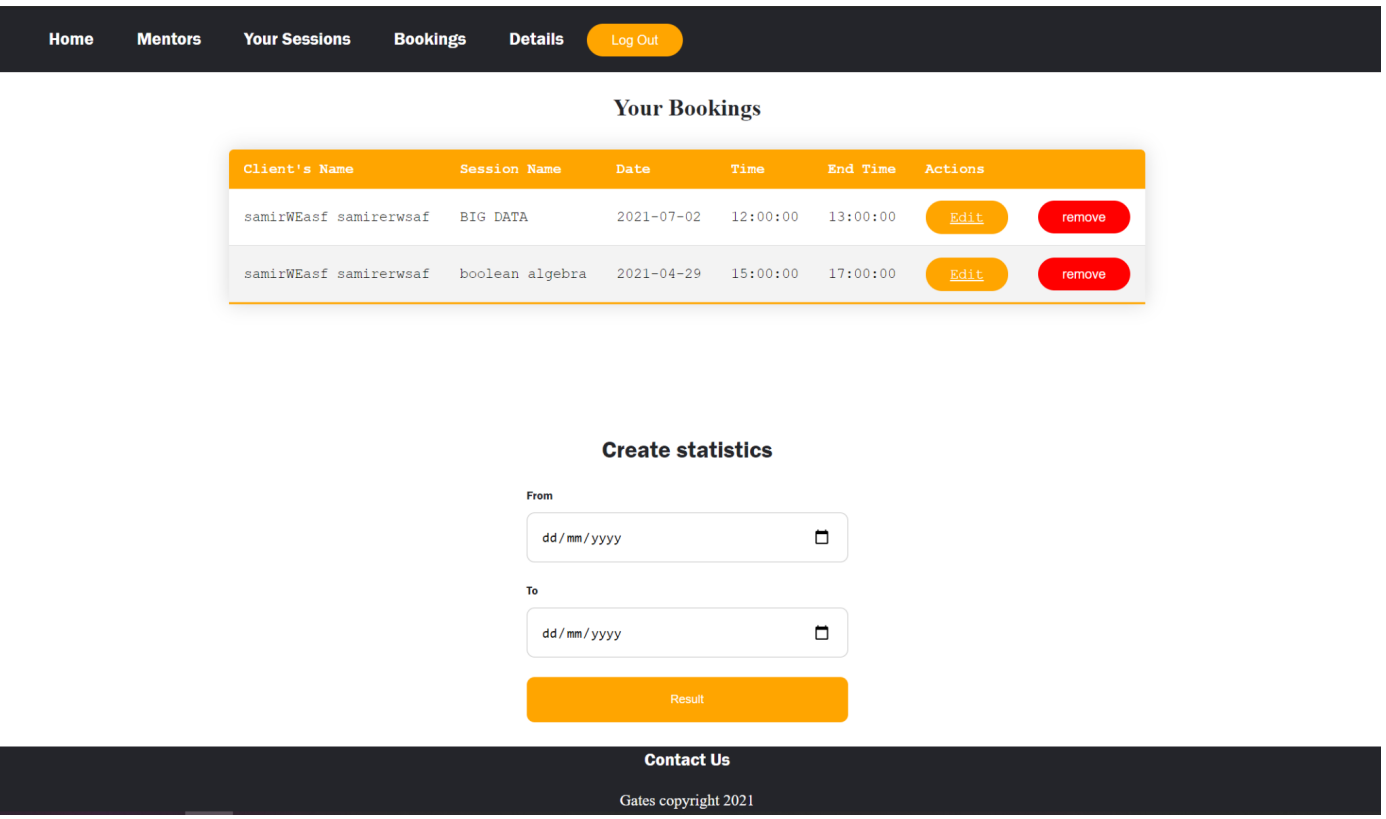
```
1   <?php
2   require_once 'connect.php';
3   session_start();
4   $ClientID = $_SESSION["ID"];
5   if (empty($_SESSION['SessionID']) || empty($_SESSION['SessionDate']) || empty($_SESSION['SessionTime']) ||
6       empty($_SESSION['SessionEndTime']) || empty($_SESSION['SessionMentorID']) || empty($_SESSION['SessionPrice'])){
7       header("Location: index.php");
8       exit();
9   }else{
10      $Time = $_SESSION['SessionTime'];
11      $Date = $_SESSION['SessionDate'];
12      $SessionID = $_SESSION['SessionID'];
13      $Endtime = $_SESSION['SessionEndTime'];
14      $MentorID = $_SESSION['SessionMentorID'];
15      $Price = $_SESSION['SessionPrice'];
16
17      $sql = "INSERT INTO tappointment(Time, Date, EndTime, MentorID, ClientID, SessionID, Price)
18              VALUES('$Time','$Date','$Endtime','$MentorID', '$ClientID','$SessionID','$Price')";
19      $result =  $con->query($sql);
20      header("Location: bookings.php");
21
22  }
23  ?>
```

19. bookings.php

(Client logged in)



(Mentor logged in)



Code for bookings.php

bookings.php

```
<html>
    <title>Bookings</title>
    <head>
        <link rel="stylesheet" type="text/css" href="CSS/table.css">
        <link rel="stylesheet" type="text/css" href="CSS/form.css">
        <link rel="stylesheet" type="text/css" href="CSS/navbar.css">
    </head>
    <style>
    .formbody{
        z-index: 0;

    margin-top: 1px;
    }
    </style>
<?php
//Starts the session if user is logged in.
session_start();
//connection to the database
require_once 'connect.php';
//gets the navigation bar/header
require_once 'header.php';
//creates and array to store error messages
$Errors=array();
$Error=array();
//create variables with the current date
$today = date("Y-m-d",time());
//Move older appointments to the old appointments table
$insertrecord = "INSERT INTO TOldAppointment(AptID, Time, Date, ClientID, MentorID, SessionID, End, Price)
                 SELECT * FROM Tappointment WHERE Date<'".$today."';";
if( $con->query($insertrecord)===TRUE){
    //Delete old appointments from the Appointments table.
    $deleterecord = "DELETE FROM TAppointment WHERE Date<'".$today."'";
    $result = $con->query($deleterecord);
}
//check if the user is logged in
if(  !isset($_SESSION["ID"])  ){

    //user is redirected to the login page if theyre not logged in
    header("Location:login.php");
}
    $usertype = $_SESSION["UserType"];
    $ID = $_SESSION["ID"];
    //Gets the current date
    $appointmentid=  $_POST['AppointmentID'];
    //Does the following if the user click the remove button
    if(isset($_POST['remove'])){
        $date=$_POST['Date'];
        //only allowed to edit booking 1 day prior
        $restriction = date("Y-m-d", strtotime("+ 1 day"));
        if($restriction>$date){
            $Errors['date'] = "You can only cancel booking upto 1 day prior.";
        }
        //Does the following if the are no errors
        if (count($Errors)===0){
            //Remove the chosen appointment from the appointmnets table
            $delete = "DELETE  FROM TAppointment WHERE AppointmentID =$appointmentid";
            //redirects to the bookings page if the query is executed.
```

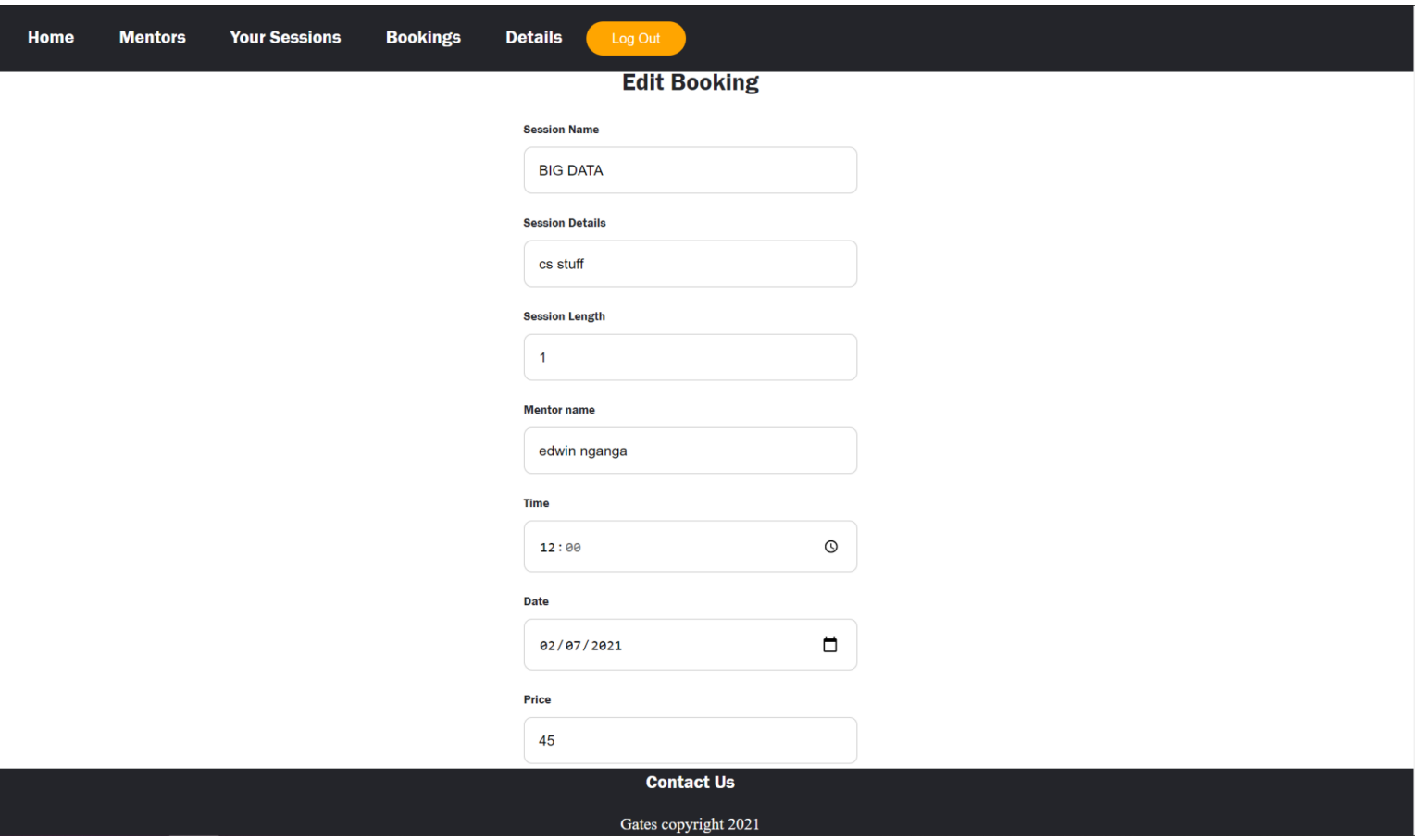
```
            //redirects to the bookings page if the query is executed.
            if($con->query($delete) === TRUE){
                header("Location:bookings.php");
                echo "<div class='alert alert-success'>Successfully removed session </div>";
            }
        }
    }
    // Check if th user is a mentor or the admin
    if (  $usertype=="Mentor" ||$usertype=="Admin"){
        //Get appointments details with customer detaiils(name) and the session name
        $sql = "SELECT *,CForename, CSurname, SessionName FROM TAppointment  join TSession join TClient on
                TAppointment.MentorID=$ID AND TAppointment.SessionID = TSession.SessionID
                AND TAppointment.ClientID = TClient.ClientID AND TAppointment.Date>$today";
    }
    //check if the user is a client
    elseif (  isset($_SESSION["UserType"]) and ($usertype=="Client")){
        // Get the appointment details with the mentor details(name) and the session name
        $sql = "SELECT *,MForename, MSurname, SessionName FROM TAppointment  join TSession join TMentor on
                TAppointment.ClientID=$ID AND TAppointment.SessionID = TSession.SessionID
                AND TAppointment.MentorID = TMentor.MentorID AND TAppointment.Date>$today";
    }
    $result = $con->query($sql);
    //Checks if the user has any bookings
    if( $result->num_rows > 0){

        ?>

        <h2 align = "center">Your Bookings</h2>

        <table class="table">
        <thead>
        <tr>
        <?php
        if ($usertype=="Client"){
            ?>

            <th>Mentor's name</th>
            <?php
        }
        elseif ($usertype=="Mentor"||$usertype=="Admin"){
            ?>

            <th>Client's Name</th>
            <?php
        }
        ?>

        <th>Session Name</th>
        <th>Date</th>
        <th>Time</th>
        <th>End Time</th>
        <th>Actions</th>
        <th></th>
        </tr>
        <thead>
        <tbody>

        <?php
        //Loop which will show all the bookings on the table
```

```
        //Loop which will show all the bookings on the table
        while( $row = $result->fetch_assoc()){
            echo "<form action='' method='POST'>"; //added
            echo "<input type='hidden' value='". $row['AppointmentID']."' name='AppointmentID' />"; //added
            echo "<tr>";
            if ($usertype=="Client"){
                echo "<td>".$row['MForename'] ." ".$row['MSurname'] . "</td>";
            }
            elseif ($usertype=="Mentor" || $usertype=="Admin"){
                echo "<td>".$row['CForename'] ." ".$row['CSurname'] . "</td>";
            }
            echo "<td>".$row['SessionName'] . "</td>";
            echo "<td>".$row['Date'] . "</td>";
            echo "<input type='hidden' value='". $row['Date']."' name='Date' />"; //added
            echo "<td>".$row['Time'] . "</td>";
            echo "<td>".$row['Endtime'] . "</td>";
            //Button which allow the user to edit the booking
            echo "<td><a href='editbooking.php?id=".$row['AppointmentID']."' class='navigation'>Edit</a></td>";
            //button that allow the user remove the booking from the database
            echo "<td><input type='submit' name='remove' value='remove' class='remove' /></td>";
            echo "</tr>";
            echo "</form>";
        }
        ?>
        </tbody>
        </table>

        <?php

        //outputs errors to the user
        if (count($Errors)>0){
            foreach ($Errors as $Errors) :
                echo $Errors ;
                ?>
                <br>
                <?php
            endforeach;
        }
    }
    else{
    //Error message in case the suer does not have any sessions booked with him
    echo "<br><br> You dont have any sessions booked ";
    }
if($usertype=="Mentor"||$usertype=="Admin"){
    if(isset($_POST['Result'])){
        //Checks if the any required field is empty
        if(empty($_POST['From_Date']) || empty($_POST['To_Date'])){
            $Error['empty']="Please fill all of the fields" ;
        }
        else{
            $From_Date = mysqli_real_escape_string($con,$_POST['From_Date']);
            $To_Date = mysqli_real_escape_string($con,$_POST['To_Date']);
            //Ensures that the admin selects the from and to dates in order
            if($To_Date<$From_Date){
                $Error['date']="To-date needs to be after the From date" ;
            }
            //Makes sure that the user selects past dates
            if($From_Date>$today||$To_Date>$today){
                $Error['range']="Please select past dates." ;
            }
```

```php
            //Checks if there arent any errors in the data submitted by the user
            if(count($Error)===0){


                $sessions = mysqli_query($con,"SELECT * FROM TOldAppointment WHERE
                        Date>'".$From_Date."' AND Date<'".$To_Date."' AND MentorID= $ID");
                $getrevenue = mysqli_query($con,"SELECT SUM(Price) FROM TOldAppointment WHERE
                          Date>'".$From_Date."' AND Date<'".$To_Date."' AND MentorID= $ID" );

               $sessionnum  =mysqli_num_rows($sessions);
               //Makes sure that revenue is = 0 if there are no records found in the database
               if($sessionnum===0){
                   $revenue_with_vat=0;
               }
               else{
                   $revenue_with_vat = mysqli_fetch_array($getrevenue);
                   $revenue_with_vat = $revenue_with_vat[0];
               }
                   $revenue_without_vat = 0.8 * $revenue_with_vat;

            }
        }
    }
    ?>
     <body class="formbody">
        <div class="main">
            <form name="statistics" method="POST" action="">
            <h2>Create statistics</h2>
                <div class= "input-container from_date">
                    <label for="From_Date">From</label>
                    <input type="date" name="From_Date" value="<?php echo isset($_POST["From_Date"]) ? $_POST["From_Date"] : ''; ?>">
                </div>
                <div class= "input-container to_date">
                    <label for="To_Date">To</label>
                    <input type="date" name="To_Date" value="<?php echo isset($_POST["To_Date"]) ? $_POST["To_Date"] : ''; ?>">
                </div>
                <button class="signup-button" type="submit" name="Result" value="submit">Result</button>
               <?php
                if(isset($_POST['Result'])){
                    //Checks if there are any errors and display them to the user if there are any
                    if (count($Error)>0){
                        foreach ($Error as $Error) :
                            echo $Error ;
                            ?>
                            <br>
                            <?php
                        endforeach;
                    }
                    else{
                        //Display statistics to the user if there arent any errors
                        echo"revenue with VAT: " . $revenue_with_vat."<br>";
                        echo"revenue without VAT: " . $revenue_without_vat."<br>" ;
                        echo "Total sessions: " . $sessionnum."<br>";

                    }
                }
               ?>
                </form>
        </div>
    <?php
}
//gets the footer
require_once 'footer.php';
?>
```

20. editbooking.php

(Note a button is currently hidden by the footer, so need to scroll down to click on it)



Code for editbooking.php

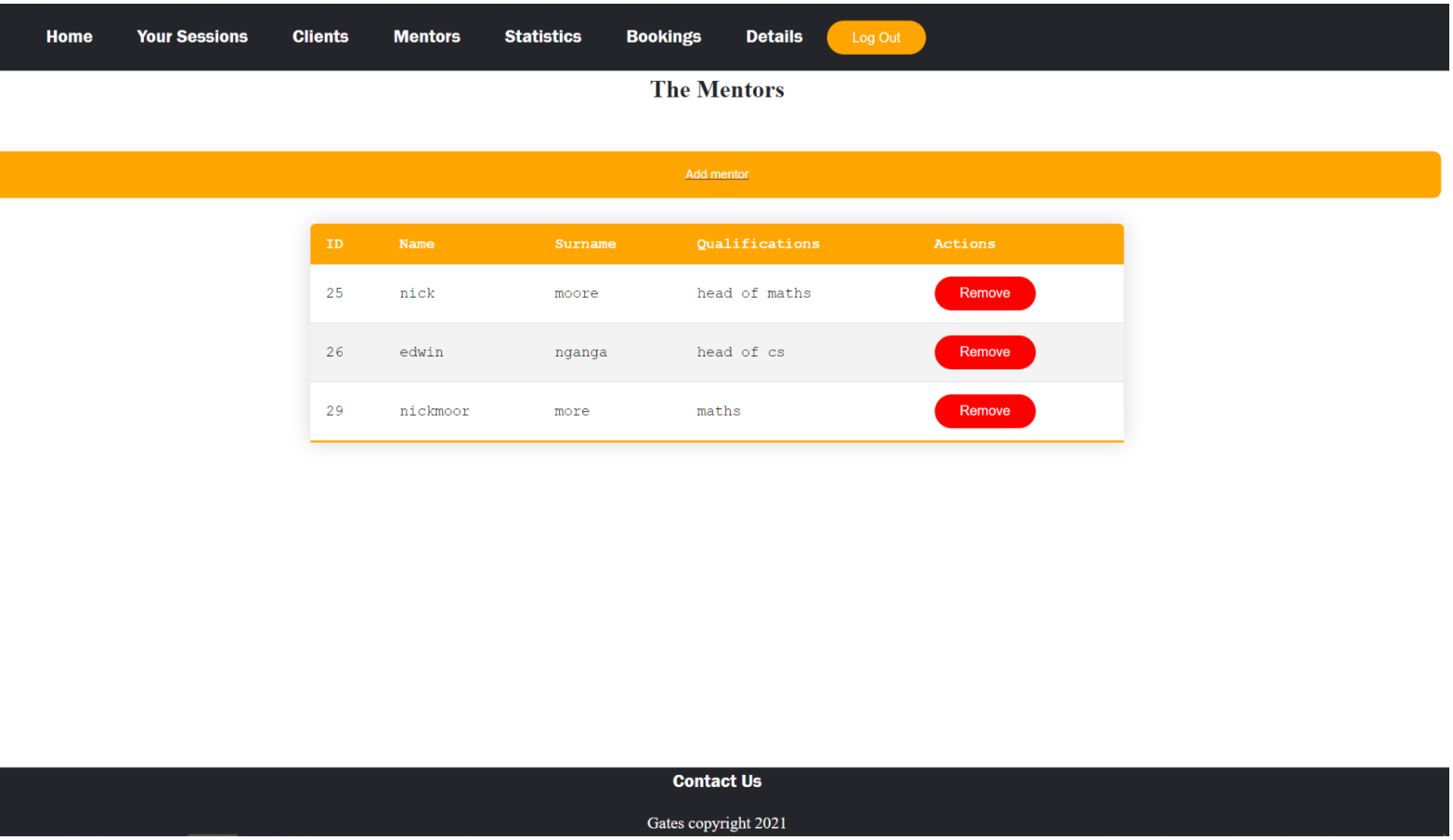
editbooking.php

```php
<?php
//Starts session to provide specific features if the user is logged in
session_start();
//Gets the following file to connect to the database
require_once 'connect.php';
//Get the navigation bar
require_once 'header.php';
//Creates an array to store error messages
$Errors=array();
if(isset($_POST['update'])){
    //Cheks if the time and date fields are empty
    if( empty($_POST['Time']) || empty($_POST['Date'])){
        $Errors['Empty'] = "Please fill the time and date field";
    }
```

```
    else{
        $Time = mysqli_real_escape_string($con,$_POST['Time']);
        $Date = mysqli_real_escape_string($con,$_POST['Date']);
        $Length = mysqli_real_escape_string($con,$_POST['SessionLength']);
         //Converts the date and check what the day of the week it is
        $day = date("D", strtotime($Date));
        //Check if the user chose a day in the weekend(Closed on Saturday and Sunday)
        if($day == 'Sat' || $day == 'Sun'){
          $Errors['Weekend'] = "Can not book during weekend";
        }
        //creates a time restriction
        $restriction = date("Y-m-d", strtotime("+ 1 day"));
        //Checks so that user can edit only one day in advance
        if( $restriction>$Date){
            $Errors['date'] = "You can only change booking details upto 1 day prior.";
        }
        if($Length==1){
         $Duration = strtotime($Time) + 60*60;
        }
        elseif($Length==2){
         $Duration = strtotime($Time) + 2*60*60;
        }
        else{
         $Duration = strtotime($Time) + 3*60*60;
        }
        $Endtime = date('H:i', $Duration);
        //Proceeds if there arent any errors in the data entered by the user
        if (count($Errors)===0){
            //Query that updates the appointments details on the appointment table
            $editappointment = "UPDATE Tappointment SET Time='{$Time}', Date = '{$Date}', Endtime ='{$Endtime}'
            WHERE AppointmentID= " . $_POST['AppointmentID'];

            if( $con->query($editappointment) === TRUE){
            echo "<div class='alert alert-success'>Successfully updated session</div>";
            //redirects to the bookings page if the query is executed properly
            header('Location: bookings.php');
            }
            else{
              //error message in case the query was not executed properly
            echo "<div class='alert alert-danger'>Error: There was an error while updating user info</div>";
            }
        }
    }
}

$id = isset($_GET['id']) ? (int) $_GET['id'] : 0;
// Get the details of the selcted appointment
$sql = "SELECT * FROM  TAppointment WHERE AppointmentID={$id}";
$result = $con->query($sql);
//Redirect user to homepage if there isn't a result
if($result->num_rows <1  ){
header('Location: index.php');
exit;
}
$row = $result->fetch_assoc();
$MentorID=$row['MentorID'];
$SessionID=$row['SessionID'];
//Gets the session details and the mentors name
$sql2= "SELECT *, MForename, MSurname FROM TSession JOIN TMentor WHERE TSession.SessionID='".$SessionID."'
        and TMentor.MentorID ='".$MentorID."'";
$result2 = $con->query($sql2);
if($result2->num_rows <1  ){
header('Location: index.php');
```

```php
exit;
}
$row2 = $result2->fetch_assoc();
?>
<html>
    <title>Edit booking</title>
    <head>
        <link rel="stylesheet" type="text/css" href="CSS/form.css">
    </head>
    <body class="formbody">
        <div class="main">
            <form name="formUser" method="POST" action="">
                <h2>Edit Booking</h2>
                <input type="hidden" value="<?php echo $row['AppointmentID']; ?>" name="AppointmentID">
                <div class= "input-container name">
          <label for="SessionName">Session Name</label>
          <input type="text" id="SessionName" name="SessionName" value="<?php echo $row2['SessionName']; ?>" class="form-control" readonly>
        </div>
        <div class= "input-container details">
          <label for="Sessiondetails">Session Details</label>
          <input type="text" id="Sessiondetails" name="Sessiondetails"  value="<?php echo $row2['Sessiondetails']; ?>" class="form-control" readonly>
        </div>
        <div class= "input-container length">
          <label for="SessionLength">Session Length</label>
          <input type="number" id="SessionLength" name="SessionLength" value="<?php echo $row2['SessionLength']; ?>" class="form-control" readonly>
        </div>
        <div class= "input-container mname">
          <label for="MForename">Mentor name</label>
          <input type="text" id="MentorName" name="MForename" value="<?php echo $row2['MForename']. " ".$row2['MSurname'] ; ?> "class="form-control" readonly>
        </div>
        <div class= "input-container time">
          <label for="Time">Time</label>
          <!--Set min and max time so that clients can only book during work hours and only at the beginning of the hour(at 0 minutes)-->
          <input type="time" step="3600"  min = "09:00:00" max = "17:00:00" name="Time" value="<?php echo $row['Time']; ?>" class="form-control">
        </div>
        <div class= "input-container date">
          <label for="Date">Date</label>
          <input type="date"  name="Date" value="<?php echo $row['Date']; ?>" class="form-control">
        </div>
        <div class= "input-container price">
          <label for="Price">Price</label>
          <input type="number" name="Price" id="Price" value="<?php echo $row2['Price']; ?>" class="form-control" readonly>
        </div>
        <button class="signup-button" type="submit" name="update" value="submit">Confirm</button>
                <?php
                //outputs errors to the user
                if (count($Errors)>0){
                    foreach ($Errors as $Errors) :
                        echo $Errors ;
                        ?>
                        <br>
                        <?php
                    endforeach;
                    }
                ?>
            </form>
        </div>
    </body>
</html>
<?php
//Gets the footer
require_once 'footer.php';
?>
```

21. adminmentors.php



Code for adminmentors.php

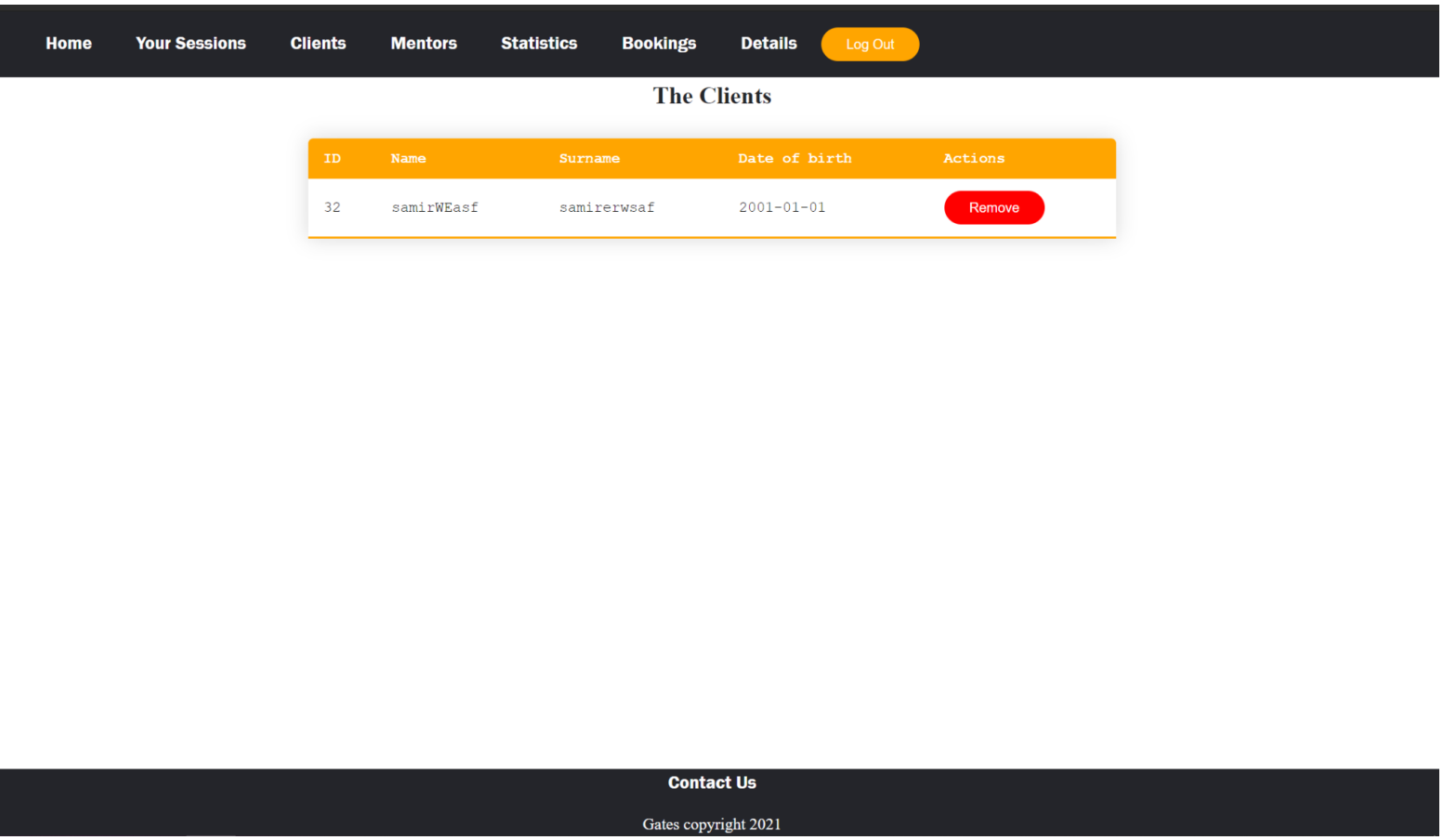
adminmentors.php

```
<html>
    <title>Mentors</title>
    <head>
        <link rel="stylesheet" type="text/css" href="CSS/table.css">
        <link rel="stylesheet" type="text/css" href="CSS/form.css">
        <link rel="stylesheet" type="text/css" href="CSS/navbar.css">
    </head>
<?php
//Starts the session if user is logged in.
session_start();
//connection to the database
require_once 'connect.php';
//Get the navigation bar
require_once 'header.php';
//checks if user is logged in and check if the user is the admin
if(isset($_SESSION["ID"])  ) :
    $usertype = $_SESSION["UserType"];
    if ( $usertype=="Admin" ){
        $ID = $_SESSION["ID"];


```

```php
//do the following when the remove button is pressed
if( isset($_POST['Remove'])){
    $id =  $_POST['MentorID'];
    // the 2 sql queeries below delete the seleceted mentors detail from the login and mentors table
    $sql = "DELETE FROM Tmentor WHERE MentorID = $id;";
    $sql.= "DELETE FROM TLogin WHERE UserType = 'Mentor' AND ID = $id;";
    $sql.= "DELETE FROM TSession WHERE  MentorID = $id";
    if($con->multi_query($sql) === TRUE){
        header("Location:adminmentors.php");
    echo "<div class='alert alert-success'>Successfully removed mentor </div>";

    }

}


//Get the mentors details
$sql = "SELECT * FROM TMentor";
$result = $con->query($sql);
if( $result->num_rows > 0){

    ?>

    <h2 align = "center">The Mentors</h2>
    <!-- button that which redirects to the mentor sign up page -->
    <a class="cta" href="msignup.php"><button class="signup-button">Add mentor</button></a>


    <table class="table">
    <thead>
    <tr>
    <th>ID</th>
    <th>Name</th>
    <th>Surname</th>
    <th>Qualifications</th>
    <th>Actions</th>
    </tr>
    <thead>
    <tbody>
    <?php
    //loop which will show every mentors on the table
    while( $row = $result->fetch_assoc()){
    echo "<form action='' method='POST'>";
    echo "<input type='hidden' value='". $row['MentorID']."' name='MentorID' />";
    echo "<tr>";
    echo "<td>".$row['MentorID'] . "</td>";
    echo "<td>".$row['MForename'] . "</td>";
    echo "<td>".$row['MSurname'] . "</td>";
    echo "<td>".$row['Qualifications'] . "</td>";
    //button that allows the admin to remove the mentors from the system
        echo "<td><button class='remove' type='submit' name='Remove' value='submit'>Remove</button></td>";

    echo "</tr>";
    echo "</form>";
    }
    ?>
    </tbody>
    </table>
    </html>
    <?php
}
else
```

```
84          {
85              //error message to show that there are no mentors on the database.
86          echo "<br><br>No Record Found";
87          }
88      }
89      else{
90          //if the user is not the admin they will be redirected to the login page.
91          header("Location:login.php");
92      }
93  else:
94      header("Location:login.php");
95  endif;
96  //gets the footer
97  require_once 'footer.php';
98  ?>
```

22. Clients.php



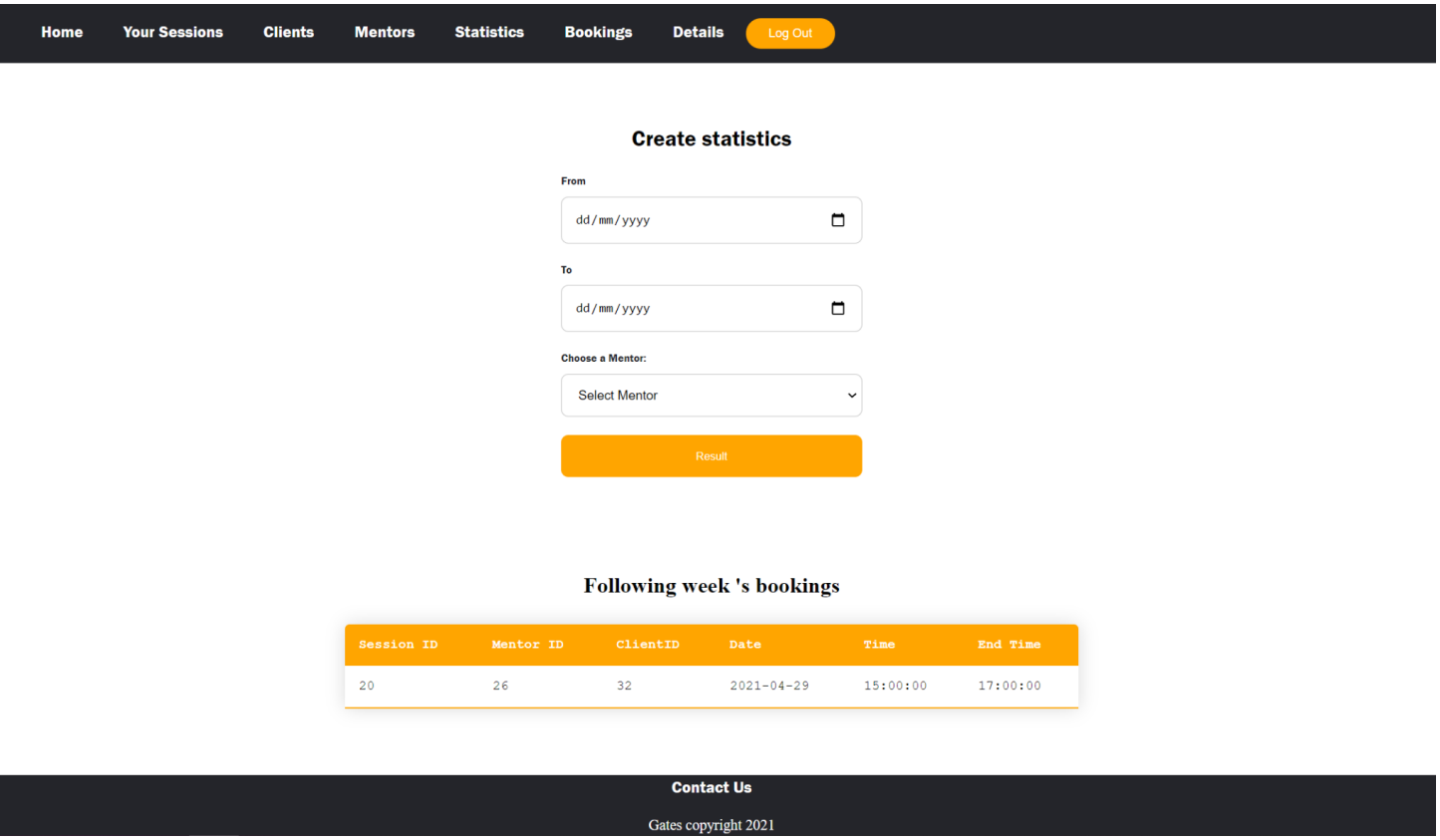
Code for Clients.php

Clients.php

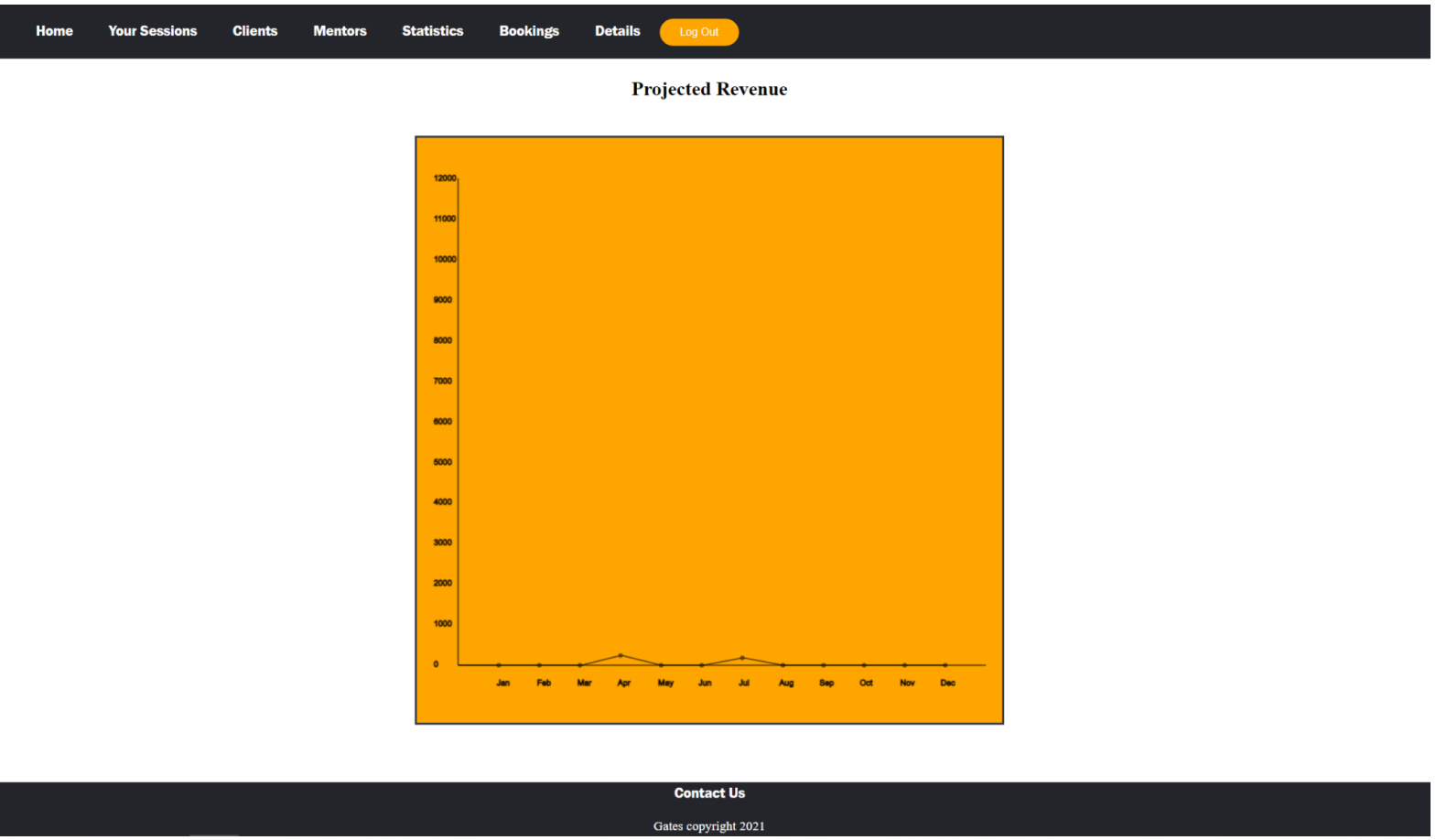
```
<html>
    <title>Clients</title>
    <head>
        <link rel="stylesheet" type="text/css" href="CSS/table.css">
        <link rel="stylesheet" type="text/css" href="CSS/form.css">
        <link rel="stylesheet" type="text/css" href="CSS/navbar.css">
    </head>
<?php
//Starts the session if user is logged in.
session_start();
//connection to the database
require_once 'connect.php';
//Get the navigation bar
require_once 'header.php';
//checks if user is logged in and check if the user is the admin
if(isset($_SESSION["ID"])  ) {
    $usertype = $_SESSION["UserType"];
    if ( $usertype=="Admin" ){
        $ID = $_SESSION["ID"];

        //do the following when the remove button is pressed
        if( isset($_POST['Remove'])){
            $id =  $_POST['ClientID'];
            // the 2 sql queeries below delete the seleceted mentors detail from the login and mentors table
            $sql = "DELETE FROM TClient WHERE ClientID = $id;";
            $sql.= "DELETE FROM TLogin WHERE UserType = 'Client' AND ID = $id";
            if($con->multi_query($sql) === TRUE){
                //if the two queries aer executed correctly redirect the admin to the clients page
                header("Location:Clients.php");
            echo "<div class='alert alert-success'>Successfully removed mentor </div>";

            }

        }

        //Get the mentors details
        $sql = "SELECT * FROM TClient";
        $result = $con->query($sql);
        if( $result->num_rows > 0){

            ?>

            <h2 align = "center">The Clients</h2>

            <table class="table">
            <thead>
            <tr>
            <th>ID</th>
            <th>Name</th>
            <th>Surname</th>
            <th>Date of birth</th>
            <th>Actions</th>
            </tr>
            <thead>
            <tbody>
            <?php
            //loop which will show every mentors on the table
            while( $row = $result->fetch_assoc()){
            echo "<form action='' method='POST'>";
            echo "<input type='hidden' value='". $row['ClientID']."' name='ClientID' />";
            echo "<tr>";
            echo "<td>".$row['ClientID'] . "</td>";
            echo "<td>".$row['CForename'] . "</td>";
            echo "<td>".$row['CSurname'] . "</td>";
```

```
65          echo "<td>".$row['DateOfB'] . "</td>";
66          //button that allows the admin to remove the mentors from the system
67          echo "<td><button class='remove' type='submit' name='Remove' value='submit'>Remove</button></td>";
68
69          echo "</tr>";
70          echo "</form>";
71          }
72          ?>
73          </tbody>
74          </table>
75          </html>
76          <?php
77      }
78      else
79      {
80          //error message to show that there are no mentors on the database.
81      echo "<br><br>No Record Found";
82      }
83  }
84  else{
85      //if the user is not the admin they will be redirected to the login page.
86      header("Location:login.php");
87  }
88  }
89  else{
90      header("Location:login.php");
91  }
92  //gets the footer
93  require_once 'footer.php';
94  ?>
```

23. statistics.php



(By scrolling down the statistics page…)



Code for statistics.php

statistics.php

```php
<?php
session_start();
require_once 'connect.php';
require_once 'header.php';
$Error=array();
//check if user is logged in.
if(  (isset($_SESSION["ID"]))) {
    $usertype = ($_SESSION["UserType"]) ;
    //check if user is the admin.
    if ($usertype=="Admin"){
        //check todays date and move Old appointments to the TOldAppointments table.
        $today = date("Y-m-d",time());
```

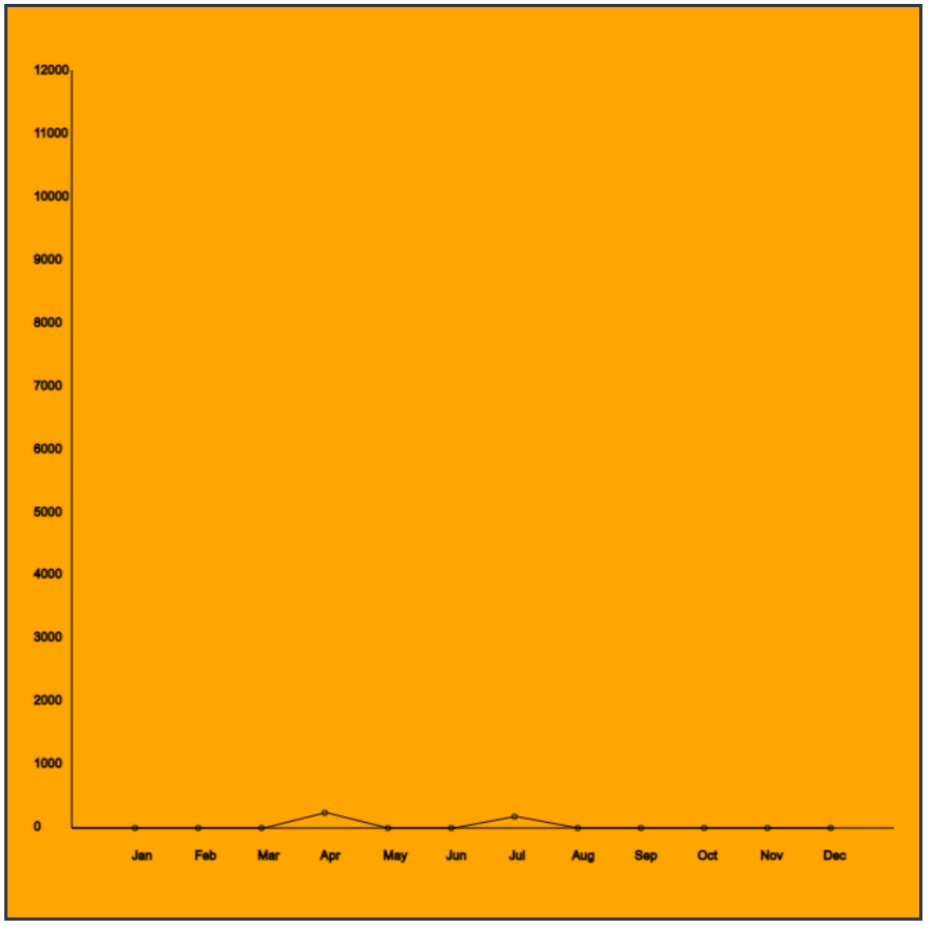
```
    $insertrecord = "INSERT INTO TOldAppointment(AptID, Time, Date, ClientID, MentorID, SessionID, End, Price)
                     SELECT * FROM Tappointment WHERE Date<'".$today."';";
    if( $con->query($insertrecord)===TRUE){
        //Delete old appointments from the Appointments table.
        $deleterecord = "DELETE FROM TAppointment WHERE Date<'".$today."'";
        $result = $con->query($deleterecord);
        if(isset($_POST['Result'])){
            //Checks if the any required field is empty
            if(empty($_POST['From_Date']) || empty($_POST['To_Date'])|| empty($_POST['Mentors-statistics'])){
                $Error['empty']="Please fill all of the fields" ;
            }
            else{
                $Mentor =  mysqli_real_escape_string($con,$_POST['Mentors-statistics']);
                $From_Date = mysqli_real_escape_string($con,$_POST['From_Date']);
                $To_Date = mysqli_real_escape_string($con,$_POST['To_Date']);
                //Ensures that the admin selects the from and to dates in order
                if($To_Date<$From_Date){
                    $Error['date']="To-date needs to be after the From date" ;
                }
                //Makes sure that the user selects past dates
                if($From_Date>$today||$To_Date>$today){
                    $Error['range']="Please select past dates." ;
                }
                //Checks if there arent any errors in the data submitted by the user
                if(count($Error)===0){
                    //If admin selects the option everyone create statistics for every mentor
                    if ($Mentor==="everyone"){
                        $sessions = mysqli_query($con,"SELECT * FROM TOldAppointment WHERE
                                    Date>'".$From_Date."' AND Date<'".$To_Date."'");
                        $getrevenue = mysqli_query($con,"SELECT SUM(Price) FROM TOldAppointment WHERE
                                      Date>'".$From_Date."' AND Date<'".$To_Date."'");
                    }
                    else{
                        //In case the admin select a specific mentor these 2 queries are used to create the statistics
                        $sessions = mysqli_query($con,"SELECT * FROM TOldAppointment WHERE Date>'".$From_Date."'
                                    AND Date<'".$To_Date."' AND MentorID= '".$Mentor."'");
                        $getrevenue = mysqli_query($con,"SELECT SUM(Price) FROM TOldAppointment WHERE Date>'".$From_Date."'
                                      AND Date<'".$To_Date."' AND MentorID= '".$Mentor."'" );
                    }
                    $sessionnum  =mysqli_num_rows($sessions);
                    //Makes sure that revenue is = 0 if there are no records found in the database
                    if($sessionnum===0){
                        $revenue_with_vat=0;
                    }
                    else{
                        $revenue_with_vat = mysqli_fetch_array($getrevenue);
                        $revenue_with_vat =$revenue_with_vat[0];
                    }
                    $revenue_without_vat = 0.8 * $revenue_with_vat;

                }
            }
        }
    }

    ?>
     <html>
<title>Statistics</title>
<head>
  <link rel="stylesheet" type="text/css" href="CSS/form.css">
  <link rel="stylesheet" type="text/css" href="CSS/table.css">
</head>
 <body class="formbody">
```

```
<div class="main">
    <form name="statistics" method="POST" action="">
    <h2>Create statistics</h2>
        <div class= "input-container from_date">
            <label for="From_Date">From</label>
            <input type="date" name="From_Date" value="<?php echo isset($_POST["From_Date"]) ? $_POST["From_Date"]:''; ?>">
        </div>
        <div class= "input-container to_date">
            <label for="To_Date">To</label>
            <input type="date" name="To_Date" value="<?php echo isset($_POST["To_Date"]) ? $_POST["To_Date"] : ''; ?>">
        </div>
        <div class ="input-container Mentors" >
            <label for="Mentors-statistics">Choose a Mentor:</label>
            <?php
            //Select the mentors from the mentor tables
             $Mentors = "SELECT MentorID, MForename, MSurname FROM TMentor";
             $result1 = mysqli_query($con, $Mentors);
            echo "<select name='Mentors-statistics' id='Mentors-statistics'>";
            echo "<option value=''>Select Mentor</option>";
            //uses a loop to make options with every mentors
            while($row = mysqli_fetch_array($result1))

            {
         echo"<option value='".$row['MentorID']."'>" .$row['MentorID']." - ".$row['MForename']." ".$row['MSurname']."</option>";
            }
            ?>
            <option value="everyone">everyone</option>
            <?php
            echo "</select>";
            ?>
        </div>
        <button class="signup-button" type="submit" name="Result" value="submit">Result</button>
       <?php
        if(isset($_POST['Result'])){
            //Checks if there are any errors and display them to the user if there are any
            if (count($Error)>0){
                foreach ($Error as $Error) :
                    echo $Error ;
                    ?>
                    <br>
                    <?php
                endforeach;
            }
            else{
                //Display statistics to the user if there arent any errors
                echo"revenue with VAT: " . $revenue_with_vat."<br>";
                echo"revenue without VAT: " . $revenue_without_vat."<br>" ;
                echo "Total sessions: " . $sessionnum."<br>";

            }

        }
       ?>
        </form>
</div>
<?php
// Creates a restriction by finding the day a week from the current dat
$restriction = date("Y-m-d", strtotime("+ 7 day"));
//Select all the bookings within a week from the current date
$bookings = "SELECT * FROM TAppointment WHERE Date < '".$restriction."' ";
$appointments = $con->query($bookings);

if( $appointments->num_rows > 0){
```

```
    ?>
    <h2 align = "center">Following week 's bookings</h2>

        <table class="table">
        <thead>
        <tr>

        <th>Session ID</th>
        <th>Mentor ID</th>
        <th>ClientID</th>
        <th>Date</th>
        <th>Time</th>
        <th>End Time</th>


        </tr>
        <thead>
        <tbody>

        <?php
        //Loop used to display all the appointments on the table
        while( $row = $appointments->fetch_assoc()){
            echo "<form action='' method='POST'>";
            echo "<input type='hidden' value='". $row['AppointmentID']."' name='AppointmentID' />"
            echo "<tr>";
            echo "<td>".$row['SessionID'] . "</td>";
                echo "<td>".$row['MentorID'] . "</td>";

                echo "<td>".$row['ClientID'] . "</td>";


            echo "<td>".$row['Date'] . "</td>";
            echo "<td>".$row['Time'] . "</td>";
            echo "<td>".$row['Endtime'] . "</td>";


            echo "</tr>";
            echo "</form>";
        }
        ?>
        </tbody>
        </table>
        </html>
        <?php
    }
    else{
        //Message displayed if no records of appointments is found on the appointments table
      echo "No booked appointments this week";
    }···
    </body>
   </html>

    <div class="graph" align="center">
    <?php
    //Gets the projected revenue graph
    include 'graph.php';
    ?>

    </div>
    <?php
}
else{
```

```
202         else{
203             header("location:index.php");
204         }
205 
206     }
207     else{
208         //Redirects the user to login page if they are not logged in
209         header("location:login.php");
210     }
211     //Gets the footer
212     require_once 'footer.php';
213     ?>
```

24. graph.php



Code for graph.php

graph.php

```
1 
2   <style>
3       canvas{margin-top:20px;border: 3px solid #303745;background:orange;}
4       h2 {color: #000;}
5       .graph{
6           z-index: 0;
7           margin-bottom: 65px;
8       }
9   </style>
```

```
62  //creates a data array to display on the graph (axis)
63  data={Jan:rev1,
64        Feb:rev2,
65        Mar:rev3,
66        Apr:rev4,
67        May:rev5,
68        Jun:rev6,
69        Jul:rev7,
70        Aug:rev8,
71        Sep:rev9,
72        Oct:rev10,
73        Nov:rev11,
74        Dec:rev12}
75  entries=Object.entries(data);
76  //adds a scale factor
77  function Cells(count){
78      return count*10
79  }
80  //creates axis
81  Axis()
82  graph()
83  //creates axis through vector graphics
84  function Axis(){
85      //sets the start point
86      yLine=0
87      yTitle=65.2;
88
89      //draws lines on the graph
90      canvas.beginPath();
91      canvas.strokeStyle="black";
92      canvas.moveTo(Cells(5),Cells(5));
93      canvas.lineTo(Cells(5),Cells(65));
94      canvas.lineTo(Cells(70),Cells(65));
95
96      canvas.moveTo(Cells(5),Cells(65));
97      //writes y axis values
98      for(i=1;i<=13;i++){
99          canvas.strokeText(yLine,20,Cells(yTitle));
100         //positioning and repositioning
101         yTitle=yTitle-4.99;
102         yLine=1000+yLine;
103     }
104     canvas.stroke();
105 }
106 function graph(){
107     canvas.beginPath();
108     canvas.strokeStyle="black";
109     canvas.moveTo(Cells(5),Cells(65));
110     //sets the start point
111     xPlot=10
112     xTitle=9.75
113     //draws graph
114     for([xvalue,yvalue] of entries){
115         //scaling
116         ycells=yvalue/50;
117         //writes on the x axis
118         canvas.strokeText(xvalue,Cells(xTitle),675)
119         //creates plot
120         canvas.lineTo(Cells(xPlot),Cells(65-ycells))
121         canvas.arc(Cells(xPlot),Cells(65-ycells),2,0,Math.PI*2,true);
122         //repositions
123         xPlot=5+xPlot
124         xTitle=4.97+xTitle
125     }
126     canvas.stroke();
127 }
128 </script>
129 </body>
```

25. cards.css

```css
.container{
    padding: 0;
    margin: 0;
    box-sizing: border-box;
    font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
    position: relative;
    display: inline-flex;
    justify-content: space-between;
    align-items: center;
    max-width: 1200px;
    flex-wrap: wrap;
    z-index: 1;
}
.container .card, .time, .address{
    position: relative;
    width: 280px;
    height: 400;
    margin: 30px;
    box-shadow: 20px 20px 50px rgba(0, 0, 0, 0.5 );
    border-radius: 15px;
    background-color: #24252A;
    overflow: hidden;
    display: inline-flex;
    justify-content: center;
    align-items: center;
    border-top: 1px solid rgba(255, 255, 255, 0.5);
    border-left: 1px solid rgba(255, 255, 255, 0.5);
    backdrop-filter: blur(5px);
}
.time, .address{

    display: block;
}
.container .card .content{
    padding: 20px;
    text-align: center;
    transition: 0.5s;
}
.container .card .content h3, .address h3, .time h3{
    font-size: 1.8em;
    color: orange;
    z-index: 1;
}
.container .card .content p, .time p, .address p{
    font-size: 1em;
    color: white;
    font-weight: 300;
}
h1{
    z-index: 0;

    margin-top: 65px;
}
```

26. form.css

```css
:root{
    font-size: 100%;
    font-size: 16px;
    line-height: 1.5;
}


.main{
    padding: 0;
    margin: 0;
    font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
    font-weight: 700;
    color: #24252A;


    display: flex;
    justify-content: center;
    align-items: center;
    z-index: 0;

  margin-top: 65px;
  margin-bottom: 100px;
}
h2{
    font-size: 1.5rem;
    font-weight: 700;
    color: #24252A;
    text-align:center;


}
h1{
    z-index: 0;

    margin-top: 65px;
}
.a-form{
    text-decoration: none;
    color:orange;
}
.a-form:hover{
    text-decoration: underline;
}

form{
    width: 328px;

}

form input[type="text"],
form input[type="email"],
form input[type="Password"],
form input[type="tel"],
form input[type="date"],
form input[type="time"],
form input[type="number"],
textarea#message,
#Mentors-statistics
{
    display: block;
    width: 100%;
    box-sizing: border-box;
    border-radius:8px;
    border: 1px solid #c4c4c4;
    padding: 1em;
    margin-bottom: 1.25rem;
    font-size:0.875rem;

}
label{
    display: block;
    margin-bottom: 0.5rem;
    font-size: 0.75rem;
}
.signup-button{
display: block;
width: 100%;
color: white;
background:orange;
font-size: 0.75rem;
border:none;
border-radius:8px;
padding:1rem;
}
.signup-button:hover{
    background: chocolate;
    cursor:pointer;
}
```

27. navbar.css

```css
nav{
    position:fixed;
    left: 0;
    top: 0;
    width:100%;
    background-color: #24252A;
    box-sizing:border-box;
    z-index: 0;
    margin-bottom: 1000px;
}


.links, .footer{
    list-style: none;
}
.links li a, .footer li a{
    transition: all 0.5s ease 0s;
}
.links li, .footer li{
    display: inline-block;
    padding: 0px 20px;
}

.links li a:hover, .footer li a:hover{
    color: orange;
}

.links li a, .footer li a {
    font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
    font-weight: 500;
    font-size: 18px;
    color: white;
    text-decoration: none;
}
.navigation, .remove {

    padding: 9px 25px;
    background-color: orange;
    color: white;
    transition: all 0.5s ease 0s;
    border: none;
    border-radius: 50px;
    cursor:pointer;
}
.remove{
    background-color: red;
}
.navigation:hover, .remove:hover{
    background-color: chocolate;
}

.footer {
    position: fixed;
    left: 0;
    bottom: 0;
    width: 100%;
    background-color: #24252A;
    color: white;
    text-align: center;
    margin-top: 10000px;
    z-index: 0;
 }
```

28. split.css

```css
.split{
    display:flex;
    flex-direction: column;
}
.left{
    height:200px;

}
.left, .right{
    display: flex;
    justify-content: center;
    align-items: center;
}
.left{
    background: orange;
    background-size: cover;
}

.left .copy{
    color: cornsilk;
    text-align: center;

}
@media screen and (min-width: 900px) {
    .split{
        flex-direction:row;
        height:100vh;
    }
    .left, .right{
        display: flex;
        width:50%;
        height:auto;
    }
}
```

29. table.css

```css

.table {
  border-collapse: collapse;
  margin: 25px 0;
  font-size: 0.9em;
  min-width: 800px;
  border-radius: 5px 5px 0 0;
  overflow: hidden;
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.15);
  margin-left: auto;
  margin-right: auto;
  font-family:'Courier New', Courier, monospace;
}

.table thead tr {
  background-color: orange;
  color: #ffffff;
  text-align: left;
  font-weight: bold;
}

.table th,
.table td {
  padding: 12px 15px;
}

.table tbody tr {
  border-bottom: 1px solid #dddddd;
}

.table tbody tr:nth-of-type(even) {
  background-color: #f3f3f3;
}

.table tbody tr:last-of-type {
  border-bottom: 2px solid orange;
}

h2{
 z-index: 0;

 margin-top: 65px;
}

```

4. Testing

4.1 Testing data

TClient

ClientID	CForename	CSurname	CMobileNumber	DateOfB
32	samir	sarker	07424226343	2001-01-01
37	Mannan	Samiha	07234589121	1999-10-20
38	elizabeth	rayner	07657384911	1999-10-10
39	zoe	payne	07856283618	1995-04-02

(Note zoe got removed afterwards which os why she is not the on the TLogin table)

TMentor

	MentorID	MForename	MSurname	MMobileNumber	Qualifications
Edit Copy Delete	26	edwin	nganga	07864562911	computer science degree from Oxford University
Edit Copy Delete	29	nick	moore	07999235811	Phd in Mathematics from Cambridge University
Edit Copy Delete	44	Waheed	Akhtar	07334657819	Degree in Computer science from Queen Mary Univers...

(Mentor Patricia which was in the video is not here as she was removed)

TLogin

MainID	UserType	Email	Password	ID
5	Mentor	edwin@gmail.com	$2y$10$x/L3Pls6IReYQrcoZm5Z/uvS5NZRPxcG.wGc3I9Tcgz...	26
104	Admin	nickm@gmail.com	$2y$10$j4mRrtn93xNDF0SM/x5pcuD2RHECOKr0y8vmDK3xvkj...	29
105	Client	sarker2@gmail.com	$2y$10$Kg7NEIaHn78HioJxnPg/B.he9Yd4v1d46gSQUvtrxOT...	32
121	Client	maisha@gmail.com	$2y$10$oMVdF57N3.IYwtWT.h6MFObwzmqa3lTtlupocEJCyCC...	37
122	Client	liz@gmail.com	$2y$10$4A0tBTzgMMXp2UQVG5KHCO3IDLtrYPRktaD1rcloHy2...	38
127	Mentor	waheed@gmail.com	$2y$10$O1ldssKoVuq0foGb2KPFCO68MK2I8WE5MxuhiAOkLjT...	44

TSession

SessionID	SessionName	SessionLength	Price	Sessiondetails	MentorID
19	BIG DATA	1	45	cs stuff	26
20	boolean algebra	2	60	algebra	26
38	maths1010	2	30	maths intro	29
39	sadbhsajk	1	2	asda	29
40	asdfasdfa	3	2	sdfsdf	29
42	functional programming 101	23	2	we look at the basics of functional programming wh...	26

TAppointment

AppointmentID	Time	Date	ClientID	MentorID	SessionID	Endtime	Price
66	14:00:00	2021-07-14	32	26	19	17:00:00	45
82	12:00:00	2021-05-03	38	26	19	15:00:00	30
83	17:00:00	2021-05-03	38	26	19	18:00:00	30

AppointmentID	Time	Date	ClientID	MentorID	SessionID	Endtime	Price
96	15:00:00	2021-05-20	32	29	38	17:00:00	30
97	15:00:00	2021-05-19	32	26	20	17:00:00	60
98	17:00:00	2021-05-05	32	26	20	19:00:00	60

4.2 Details of individual tests

1.Signup

No.	Test Description	Test Data	Test Type	Expected Result	Actual result	Time
1.1	To make sure an error message is shown if all of the fields aren't filled in	Empty fields	Absent/ Presence	Error message =”Please fill all of the fields”	As intended	0.00
1.2	To make sure error message is displayed if the forename is not only letters	Forename: ”Maisha1” Rest: All valid data	Erroneous	Error message =”Forename should only be letters”	As intended	0:20
1.3	To make sure error message is displayed if the surname is not only letters	Surname: ”Samiha1” rest:All valid data	Erroneous	Error message = ”Surname should only be letters”	As intended	1:05
1.4.1	To make sure error message is displayed if the mobile number is not only numbers	Mobile number: ”0723458912a” rest:All valid data	Erroneous	Error message = ”Mobile number should be only numbers”	As intended	1:23
1.4.2	To make sure an error message is displayed if the mobile number is not 11 digits	Mobile number: ”072345891211 ” rest:All valid data	Boundary	Error message = ”Mobile number should be 11 digits”	As intended	1:42
1.5.1	To make sure error message is displayed if the email entered is not valid	Email:”Maisha” rest:All valid data	Erroneous	Error message = “Please include @”	As intended	1:57
1.5.2	To make sure error message is displayed if they enter an already registered email	emai:”sarker2@gmail.com” Rest:all valid data	Erroneous	Error message = “ This email is already registered with us”	As intended	2:17
1.6	To make sure an error message is displayed if the user is not old enough	Date of birth:”14/10/2015” Rest: All valid	Boundary	Error message = “ “You are too young to have an account with us”	As intended	2:51

		data				
1.7	Make sure error message is displayed if password is invalid(Not a strong password is entered)	Password- Not strong enough password Rest: All valid data	Boundary	Error message = “Password should have at least 1 uppercase letter, 1 lower case letter, 1 number and 1 special character”	As intended	3:29
1.8	Make sure data is entered correctly on intended tables	All valid data	Normal	NA	As intended	4:31
1.9	To make sure the password is hashed	All valid data	Normal	Password entered is stored hashed on the database	As intended	4:39

2. Login/Logout + Contact

No.	Test Description	Test Data	Test Type	Expected Result	Actual result	Time
2.1	Make sure error message is displayed if fields are empty	Empty fields	Absent/ Presence	Error message = “Please fill all of the fields”	As intended	0:00
2.2	make sure to display error message if user fails captcha test	Email: “maisha@gmail.com” Password:”123” Captcha test- not done	Erroneous	Error message = “Please verify yourself”	As intended	0.17
2.3	Make sure error message if the user enters an unregistered email	Email:”samir@gmail.com” Password:”123” Captcha test - passed	Erroneous	Error message = “This email is not registered with us”	As intended	0.35
2.4	To make sure error message is displayed if the email and password do not match	Email:”maisha@gmail.com” Password:”1” Captcha test-passed	Erroneous	Error message = “Email and password do not match”	As intended	1.12
2.5	To make sure the user is redirected to homepage if login is successful	All valid data	Normal	Redirected to homepage	As intended	1.32
2.6	To make sure the user is	User tries to go to login	Normal	Redirected to	As	1:40

	redirected to the homepage if they are already logged in and they try to access the login page	page when they are already logged in		homepage	intended	
2.7	To make sure that specific pages are available if the user logged in is a client	Logged in as a client	Normal	Specific pages available to only clients	As intended	1:55
2.8	Make sure session is destroyed when user logs out	User clicks logout	Normal	Redirect to index.php when tries to access details.php	As intended	2:04
2.9	To make sure that specific pages are available if the user logged in is a Mentor	Logged in as the mentor	Normal	Specific pages available to only mentors	As intended	2:21
2.10	To make sure that specific pages are available if the user logged in is the admin	Logged in as the admin	Normal	Specific pages available to only admin	As intended	2:51
Contact page						
2.11	To make sure an error message is displayed if the fields aren't filled in	All empty fields	Absent/ Presence	Error message = "Please fill all of the fields"	As intended	3:35
2.12	To make sure error message is displayed if the name does not consist of only letters	Name:"Samir1" rest:"All valid data"	Erroneous	Error message = "Your name should be only letters"	As intended	3:42
2.13	To make sure an error message is displayed if the user enter an invalid email	Email:"samir" rest : all valid data	Erroneous	Error = "Please include @"	As intended	4:02
2.14	To make sure the user redirected to homepage if email is sent successfully.	All valid data	Normal	User redirected to homepage	As intended	4:23

3. Edit Details + Change password

No.	Test Description	Test Data	Test Type	Expected Result	Actual result	Time
3.1	Make sure user is redirected to the login page if they aren't logged in	NA	Normal	User redirected to login page	As intended	0:00
3.2	To make sure an error message is shown if all of the fields aren't filled in	All empty fields	Absent/ Presence	Error message = "Please fill all of the fields"	As intended	0:45
3.3	To make sure error message is displayed if the forename is not only letters	Name:"Maisha1" Rest:all valid data	Erroneous	Error message = "Forename should only be letters"	As intended	0.58
3.4	To make sure error message is displayed if the surname is not only letters	Surname: "Samiha1" Rest: all valid data	Erroneous	Error message = "Surname should only be letters"	As intended	1:01
3.5	To make sure error message is displayed if the mobile number is not only numbers	Mobile number: "0723458912a" Rest:all valid data	Erroneous	Error message = "Mobile number should be only numbers"	As intended	1:14
3.6	To make sure error message is displayed if mobile number is not 11 digits	Mobile number: "072345891211" Rest: all valid data	Boundary	Error message = "Mobile number should be 11 digits"	As intended	1:19
3.7	To make sure the user is redirected to the details page after changing their details	All valid data entered	Normal	Redirect to details page	As intended	1:35
3.8	To make sure the correct record is updated on the correct table	All valid data entered	Normal	NA	As intended	1:45
Change password						
3.9	To make sure an error message is shown if all of the fields aren't filled in	All empty fields	Absent/ Presence	Error message = "Please fill all of the fields"	As intended	1:56

3.10	To Make sure error message is displayed if the user enters the wrong email	Email:"mannan@gmail.com" Confirm email: "mannan@gmail.com " Password:"123" Confirm password:"123"	Erroneous	Error message = "Wrong email entered "	As intended	2:04
3.11	To Make sure error message is displayed if the email and the re-enter email do not match	Email:"maisha@gmail.com" Confirm email: "mannan@gmail.com " Password:"123" Confirm password:"123"	Boundary	Error message = "Emails do not match"	As intended	2:24
3.12	To make sure an error message is displayed if the password and re-enter password do not match	Email:"maisha@gmail.com" Confirm email: "maisha@gmail.com" Password:"@Mannan 123" Confirm password:"123"	Boundary	Error message ="Passwords do not match"	As intended	2:39
3.13	Make sure error message is displayed if password is invalid(Not strong enough password)	Email:"mannan@gmail.com" Confirm email: "mannan@gmail.com " Password:"123" Confirm password:"123"	Erroneous	Error message = "Password should have at least 1 uppercase letter, 1 lower case letter, 1 number and 1 special character"	As intended	2:12
3.14	To make sure user is redirected to the homepage if the changing password was successful	All valid data	Normal	Redirected to homepage	As intended	3:07
3.15	To make sure the hashed version of the password was stored on the database	All valid data	Normal	Hashed version of password stored on database	As intended	3:36

4. Mentor - Add session + Edit session + delete session

No.	Test Description	Test Data	Test Type	Expected Result	Actual result	Time
Add session						
4.1	To make sure only the admin or mentor can access this page	User logged out and user logged in as client	Normal	User is redirected to login page or index page	As intended	0:00
4.2	To make sure error message is displayed if the fields are empty	Empty fields	Absent	Error message = “Please fill all of the fields”	As intended	1:21
4.3	To make sure an error message is displayed if the session length is not between 1 and 3(1 and 3 included)	Session duration: 22 Rest: all valid data	Boundary	Error message = “Session length should be 1,2 or 3 hours long”	As intended	1:55
4.4	To make sure user is redirected to own sessions page if session is added successfully	All valid data	Normal	User is redirected to own sessions page	As intended	2:55
4.5	To make sure the session is stored on TSession Table	All valid data	Normal	NA	As intended	3:18
Remove session						
4.6	To make sure session selected is removed from TSession table after remove button is clicked	User click remove button	Normal	Session is removed from database	As intended	3:34
Edit Session						
4.7	To make sure form is pre-filled with data of session selected	User clicks edit button	Normal	Form is filled with data of session selected	As intended	3:50, 4:51
4.8	To make sure an error message is displayed if any field is empty	All empty fields	Absent	Error message = “Please fill all of the fields”	As intended	3:56
4.9	To make sure an error message is displayed if the session length is not between 1 and 3(1 and 3 included)	Session details: ”6” Rest: all valid data	Boundary	Error message = “Session length should be 1,2 or 3 hours long”	As intended	4:54
4.10	To make sure the user is	All valid data	Normal	NA	As intended	4.15,

	redirected to the own sessions page if session data has been updated successfully					4:56
4.11	To make sure the details of the specific session selected is updated	All valid data	Normal	NA	As intended	4:30, 5:03

5. Client- Book session

No.	Test Description	Test Data	Test Type	Expected Result	Actual result	Time
5.1	To make sure the user is redirected to the login page if they try to book and they aren't logged in.		Normal	User redirected to login page	As intended	
5.2	To make sure the details of the session selected are used to pre-fill in the booking form	User click edit button	Normal	Form is filled with details of the session selected	As intended	
5.3	To make sure an error message is displayed if the time or date field are empty		Absent	Error message = "Please fill all of the fields"	As intended	
5.4	Make sure error message is displayed if the user tries to book during the weekend		Erroneous	Error message = "Can not book during the weekend"	As intended	
5.5	Make sure error message is displayed if the user doesn't book at the start of the hour		Erroneous	Error message = "Can not book at this time"	As intended	
5.6	Make sure error message is displayed if user tries to book outside working hours		Erroneous	Error message = "Can not book at this time"	As intended	
5.7	To make sure an error message is displayed if the client tries to book a session when they have another session booked(The start time is checked)(Double booking)		Erroneous	Error message = "You already have another session booked at this time"	As intended	
5.8	To make sure an error message is displayed if the client tries to book a session		Erroneous	Error message = "You already have another session	As intended	

	when they have another session booked.(Check endtime) (Double booking)			booked at this time”		
5.9	To make sure an error message is displayed if the client tries to book when the mentor selected has another session booked. (Check start time)(Double booking)		Erroneous	Error message = “This mentor is not free at this time”	As intended	
5.10	To make sure an error message is displayed if the client tries to book when the mentor has another session booked.(Check Endtime)(Double booking)		Erroneous	Error message = “This mentor is not free at this time”	As intended	
5.11	Make sure error message is displayed if the user tries to book a past date		Erroneous	Error message = “Please book a date in the future”	As intended	
5.12	Make sure an error message is displayed if the user tries to book at a past time on the same day		Erroneous	Error message = “Please book later in the day”	As intended	
5.13	Make sure user is redirected to checkout page(Stripe API) if the there is no error	All valid data	Normal	Redirect to checkout page	As intended	
5.14	Testing Stripe API is working as supposed to be working					
5.14	Make sure to only add booking to the TAppointment table only after the payment is successful	All valid data	Normal	Booking added to database	As intended	

6. Client - View booking + Edit booking + delete booking

No.	Test Description	Test Data	Test Type	Expected Result	Actual result	Time
View booking						
6.1	Make sure older appointments are moved to the TOldAppointment table and are deleted from the the TAppointment table	NA	Normal	Old records transferred to TOldAppointment table	As intended	0:47
6.2	Make sure the mentor's name is displayed if its a client viewing the bookings	NA	Normal	Display the bookings on the table with the mentors name	As intended	1:10
Delete booking						
6.3	Make sure error message is displayed if the user tries to delete booking when there is less than a day from it	User clicks remove button	Erroneous	Error message = "You can only cancel booking upto 1 day prior"	As intended	1:15
6.4	Make sure when user clicks the remove button the booking is deleted from the database	User clicks remove button	Normal	Booking record is removed from the database	As intended	1:24
Edit booking						
6.5	Make sure the mentor's name is displayed on the form.	User clicks edit button	Normal	Form filled with booking data and mentors' name	As intended	1:51
6.6	To make sure an error message is displayed if the time or date field is empty		Absent/ Presence	Error message = " Please fill the time and date field"	As intended	1:57
6.7	Make sure error message is displayed if the user tries to book during the weekend	Date: 17/07/2021	Erroneous	Error message = " Can not book during the weekend"	As intended	2:08
6.8	Make sure error message is displayed if user tries to book outside working	Time: 07:00 Time:18:00	Erroneous	Error message = "Can not book at this time"	As intended	2:22

	hours					
6.9	Make sure the correct booking is updated on the TAppointment table	All valid data	Normal	NA	As intended	

7. Mentor - Edit booking + create statistics

No.	Test Description	Test Data	Test Type	Expected Result	Actual result	Time
Edit booking						
7.1	Make sure the client' s name is displayed if its a mentor editing the booking		Normal	Clients name and booking details displayed on form	Not as intended	
7.2	To make sure an error message is displayed if the time or date field are empty		Absent	Error message = "Please fill the time and date field"	As intended	
7.3	Make sure error message is displayed if the user tries to book during the weekend		Erroneous	Error message = "Can not book during the weekend"	As intended	
7.5	Make sure error message is displayed if user tries to book outside working hours		Erroneous	Error message = "Can not book at this time"	As intended	
7.6	Make sure the correct booking is updated on the TAppointment table	All valid data	Normal	NA	As intended	
Cancel booking						
7.7	Make sure error message is displayed if the user tries to delete booking when there is less than a day from it	User clicks remove button	Erroneous	Error message = "You can only cancel booking upto 1 day prior"		
7.8	Make sure when user clicks the remove button the booking is deleted from the database	User clicks remove button	Normal	Booking record is removed from the database		
Create statistics						
7.9	Make sure only the mentors		Normal	Does not display form	As intended	

	and admin can view the create statistics form			if user is not admin or mentor		
7.10	Make sure error message is displayed if any field is empty		Absent	Error message = “Please fill all of the required fields”	As intended	
7.11	Make sure error message is displayed if the from-date is after the to-date		Erroneous	Error message = “From date can not be after to date”	As intended	
7.12	Make sure error message is displayed if the from-date or to-date aren’t past dates		Erroneous	Error message = “Please select a past date”	As intended	
7.13	Make sure revenue is £0 if the mentor had no bookings between the dates selected		Normal	Revenue = 0 if no bookings between date selected	As intended	

8. Admin- Mentor registration+ View/remove mentor + View/delete client

No.	Test Description	Test Data	Test Type	Expected Result	Actual result	Time
Mentor registration						
8.1	To make sure only the admin can register a mentor(only admin can access this page)	NA	Normal	User redirected to different pages if user is not the admin	As intended	0:00
8.2	To make sure an error message is shown if all of the fields aren’t filled in	All empty fields	Absent	Error message = “Please fill all of the fields”	As intended	2:50
8.3	To make sure error message is displayed if the forename is not only letters	Name:”Waheed1” Rest:all valid data	Erroneous	Error message =”Forename should only be letters”	As intended	3:00
8.4	To make sure error message is displayed if the surname is not only	Surname: “Akhtar1” Rest: all valid data	Erroneous	Error message = ”Surname should only be letters”	As intended	3:49

	letters					
8.5.1	To make sure error message is displayed if the mobile number is not only numbers	Mobile number: “0733465781a” Rest: all valid data	Erroneous	Error message = ”Mobile number should be only numbers”	As intended	4:06
8.5.2	To make sure an error message is displayed if the mobile number is not 11 digits	Mobile number: “073346578199” Rest: all valid data	Boundary	Error message = ”Mobile number should be 11 digits”	As intended	4:22
8.6.1	To make sure error message is displayed if the email entered is not valid	Email: waheed Rest: all valid data	Erroneous	Error message = “ Email should contain @”	As intended	4:38
8.6.2	To make sure error message is displayed if they enter an already registered email	Email: “edwin@gmail.com” Rest: all valid data	Erroneous	Error message = “ This email is already registered with us”	As intended	4:55
8.7	Make sure error message is displayed if password is invalid(password is not strong enough)	password:(Multiple attempts with weak password)	Erroneous	Error message = “Password should have at least 1 uppercase letter, 1 lower case letter, 1 number and 1 special character”	As intended	5:13
8.8	To make sure the password is hashed	All valid data	Normal	Hashed version of password stored on the database	As intended	5:48
8.9	Make sure all data is entered on the correct tables	All valid data	Normal	Record stored on database	As intended	5:48
View, Delete mentor						
8.10	Make sure all of the mentors are displayed on the table		Normal	All of the mentors are displayed	As intended	6:00
8.11	Make sure when a mentor is removed their records from the TLogin table, TMentor table are removed	User clicks remove button	Normal	Remove related records from those specific tables on the database	As intended	6:05
View, Delete Client						

8.12	Make sure all of the Clients are displayed on the table		Normal	All of the clients are displayed	As intended	6:22
8.13	Make sure if the admin clicks remove all of that specifics client's records from the TClient table and the TLogin table are removed.	User clicks remove button	Normal	Remove the selected clients records from the TLogin and TClient table	As intended	6:32

9. Statistics page

No.	Test Description	Test Data	Test Type	Expected Result	Actual result	Time
9.1	Make sure user is redirected to login page if they aren't logged in	NA	Normal	User redirected to login page	As intended	
9.2	Make sure if user is a mentor or client they are redirected to the homepage	NA	Normal	User redirected to homepage	As intended	
9.3	Make sure older appointments are moved to the TOldAppointment table and are deleted from the the TAppointment table	NA	Normal	Old records are moved to the TOldAppointment table	As intended	
Create statistics						
9.4	Make sure error message is displayed if any field is empty	Empty fields	Absent	Error message = "Please fill all of the required fields"	As intended	
9.5	Make sure error message is displayed if the from-date is after the to-date		Erroneous	Error message = "From date can not be after to date"	As intended	
9.6	Make sure error message is displayed if the from-date or to-date aren't past dates		Erroneous	Error message = "Please select a past date"	As intended	
9.7	Make sure revenue is £0 if the mentor had no		Normal	Revenue = 0 if no bookings between	As intended	

	bookings between the dates selected			date selected		
9.8	Make sure if the admin selects “everyone” statistics are calculated accordingly		Normal	Calculate statistics of every mentor together	As intended	
Display bookings						
9.9	Make sure only booking for the following weeks are displayed	NA	Normal	Booking upto a week are displayed	As intended	

Note for displaying bookings i later on added a booking for that week manually in the video to show it only shows booking for that week

Evaluation

5.1 Evaluation of objectives

1.The system should be easy to navigate

I have made sure every page can be accessed easily using the links from the navigation bar for most pages and for the contact page there is a link on the footer. The user can access the three main pages - homepage, mentors page and session page without logging in. To access the sign up page i have put a link just below the login form. For other links within pages I have made sure that all of them are in a position which makes it easy to find and make sense why they are there. Because of the navigation bar and the links within pages the first objective has been achieved.

2. Only the clients and the mentors(plus admin) should be allowed to use the system, so there should be some sort of authorisation.

The clients and mentors have different forms which can be used to register them on the system, for both these forms there are various checks to validate the data as well as specific checks to see if they are already registered on the system to prevent duplicate data on the database.
I have made sure that any actions beside contacting the company can only be performed after the user logs in making sure the user is either a client, mentor or the admin.There is a login system which allows different types of users to log in.The user needs to be registered on the system to be able to log in and they have to use their email and password which they had used to register, for extra security the user must also pass a google captcha test to be able to login. After login the system automatically detects if the user is a client, mentor or the administrator making sure different features are accessible depending the usertype. Through the secure login and registration system this objective is achieved,

3. The system should be able to store and display clients data and allow them to change it later

A client can access the signup page from a link on the log in page. The sign up page contains a form which the client can use to register on the system. The form contains different fields for the user to enter personal information which is necessary for booking as well as login information which includes the password and email.I have made sure that all of the data entered on the field is validated before it is stored on the database. I have made sure that the password is hashed before is stored on the system to add more security. To help reduce data on the database i have made sure to check if the email entered by the user is already registered on the system, this prevents duplicate data on the database and avoid collisions. After they are registered the fact that they are a client is also stored on the database, this is so specific features are granted when they log in. The data is stored on the database only if there are no errors in it.
The client is also able to edit their personal information afterwards as well as their password. Through The secure signup page and the details page used to edit details and password this objective is achieved

4. The system should allow the clients to book sessions through a booking form and be able to change the details of the booking or cancel the booking:

I have made sure that the clients can very easily book sessions. The clients are only allowed to book a session after they are logged in. I have made sure to add a checkout system which the clients can use to pay for their appointments. I have made sure that the clients can only book at times the center is open by preventing them from booking outside working hours or on a Saturday or Sunday. The clients can use a form which is filled with the sessions data and they just have to enter the date and time they would like to book. I made sure to add measures against double booking. The client and mentor can both edit the

booking as long as its one day prior the booking date, same goes for cancelling a booking. As the client and mentor can both do all of the specified actions in the objective this objective has been achieved.

5. The system should store the mentors details and display specific details to other Users

I have made sure that a mentor can only be registered by the admin. The admin can use a form which they would fill with the mentors personal details and login details which include the email address and password. The admin would then give the login details to the mentor which they can use to login and change anything if needed. As the client sign up form, there are also many validations for the fields in the form to make sure that data entered on the database is correct, there is also a check which checks if the email address entered is already registered to prevent double data on the system and to also avoid collisions. The password is also hashed before it is stored on the system. The mentor can change their personal information after they log in as well their password. There is page which displays all the mentors name, mobile number and qualifications to the clients. By making the mentor registration only available to the admin and by making it secure as well as displaying only specific information to the client objective five has been achieved

6.The mentors and admin should be able interact with their own sessions

Every mentor(and admin) can access a page which shows all of the sessions they taught, from this page the mentors can delete their session, they can access a page to add new session and they can also edit their existing sessions. The mentor can use a form to add a new session where some fields go through some sort of validation depending on the necessity. The same form is used to edit a session, when editing an existing session the form is already filled with the information of that session so that it is easier for the mentor to know what they are doing. Through these functions/ actions which the mentors and admin only can perform objective 6 has been achieved,

7. Clients and mentors(and admin) should be able to view booked the sessions

The users of the system can access a page which displays all of the bookings they have. The users can then cancel the booking if they are at least 1 day prior to the booking, they are also able to alter the time of the booking if they are at least 1 day prior to the booking. The admin can view all of the bookings the company has for the coming week on the statistics page. As everything specified in the objective can be done by the users of the system objective seven has been achieved.

8. One of the mentors' accounts will also act as the administrator

There is an account which is granted all of the functionalities a mentor has as well as extra functionalities. This account can view all of the clients which are registered on the system, and is allowed to remove them as well for any specific reason. This account also has a list of all the mentors and is also allowed ot remove them. There is a specific page for this accounts which displays statistics for the company. From this page the administrator can create reports as well as displays all of the bookings for the company for the following week. As this account can do everything a mentor can do as well as monitor the clients and mentor and the revenue flow for the company objective 8 of having a specific administrator account with special features has been achieved.

9. The admin and the mentors should receive reports

The administrator can create reports on the statistics page using a form which has validations so that the administrator chooses past dates so only old records are used to create these reports. The administrator can also choose to create a report for a specific mentor or if they want to create it for all of the mentors together. The administrator can also view a graph which displays the projected revenue for the whole company for the year. The mentor can also use a form to create statistics which has the same validation so that past dates are chosen however for a mentor they can only generate a report for their own data. Since the mentors and admin can create accurate reports objective 9 has been achieved

10. There should be a contact page

There is a contact page which can be accessed from a link on the footer which is on every page. On the contact page there is card which displays when the center is open as well as a card which displays the address of the center. There is also a contact form which the user can use to write a message to company. As the contact page includes everything specified on the objective, it means that objective 10 has been achieved.

11. There should be a checkout system

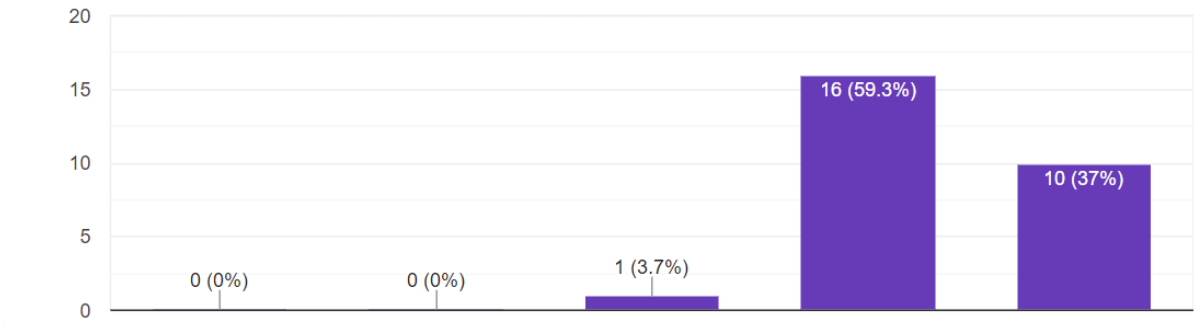
I have integrated the Stripe API on my system which allows the client to pay for the booking. This API provides a secure checkout for the customer and allows them to use different payment methods/cards.The api automatically check the cards validity and make sure its authenticity. This allows for the booking to only be added to system when the payment has been received. As the checkout provides a secure and easy experience objective 11 have been achieved

5.2 User feedback

Client feedback

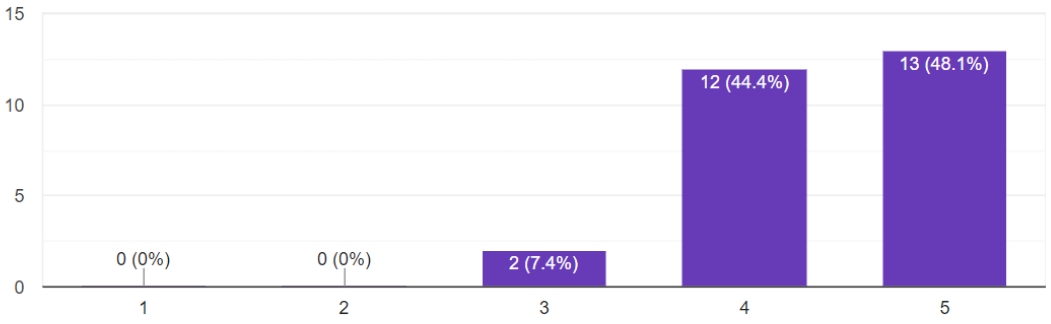
How easy is the system to navigate? 1 being very bad and 5 being very good

27 responses



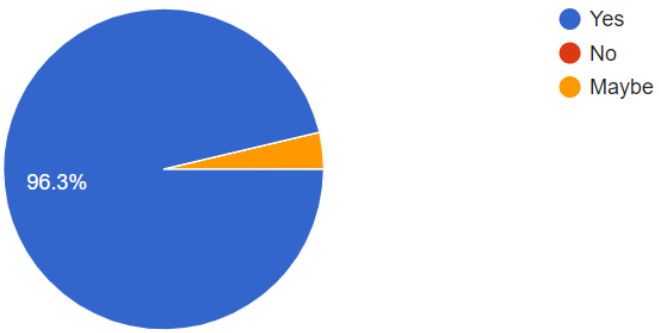
On a scale of 1 to 5, 1 being very bad and 5 being very good, how is the booking part of the system?

27 responses



Do you think the system is safe enough to store your information?

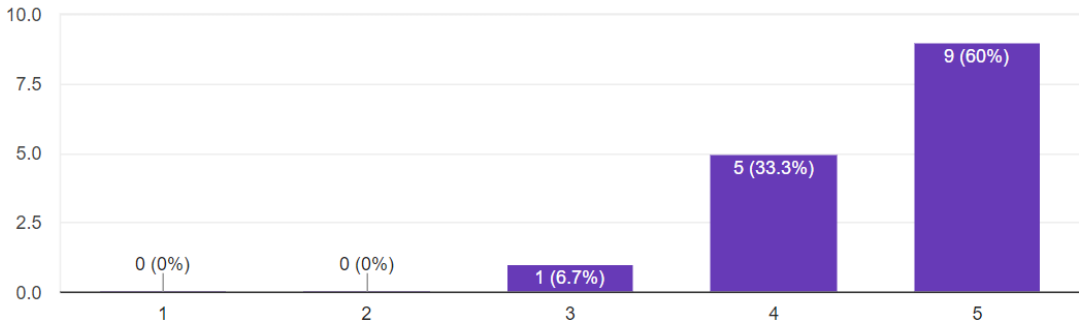
27 responses



Mentor feedback

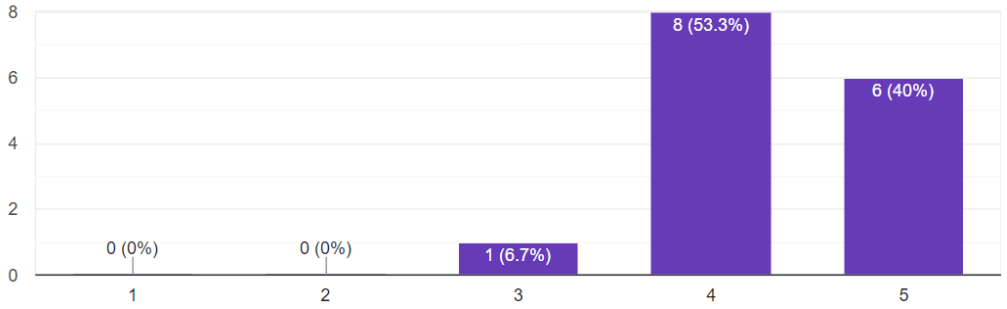
on a scale of 1 to 5, 1 being very bad and 5 being very good how easy is the system to navigate?

15 responses



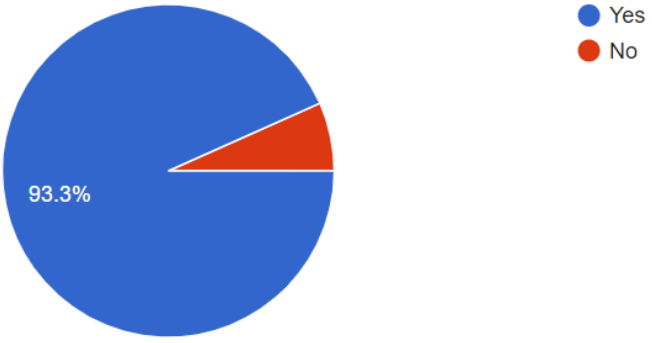
on a scale of 1 to 5, 1 being very bad and 5 being very good how good are the reports generated?

15 responses



Does the system has all of the required features you need?

15 responses



Do you feel like the system is secure enough?

15 responses

5.3 Evaluation of user feedback

Client feedback

Overall the feedback received from the clients is very positive. Most of the clients thought that the system was quite easy to navigate with only a very small number of clients thinking it was not very good, this should decrease if the sample size is increased, however there was no negative feedback for the system navigation which was quite good. Very similar results for how easy the booking part of the system is as well which is quite good, here as well there was no negative feedback, however the fact that there was some neutral feedback for both questions means that there is definitely an area for improvement.
All of the clients did not think that the system is not secure enough to store their information which quite and a very small amount wasn't really sure which is also very good.

Mentor feedback

Like the client feedback the overall feedback received from the mentors was also quite positive. We can see that a very high percentage of the mentors thought the system was easy to navigate and only one mentor was neutral, similar response was received for how good the report generated were, same as before only one person neutral. A very high percentage of the mentors though that the system had all of the features they need however the fact that this is not 100% means that some features can still be added. All of the mentors thought that the system was secure which means there shouldn't be any problems in terms of security.

5.4 Future development

The system I have created has received very good feedback and meets all the requirements which were set by the initial objectives as well as additional features which makes the system more efficient.

As I talked with the administrator they said that they would like to buy some computers and make them available for booking by the hour, so for future development I could make it possible for the system to do so.
Another development for the system could be to add an email service which would send information regarding the center automatically to all the clients.
Lastly the css could be improved so that it makes the website look more attractive to the clients.

The system meets all of the requirements/objectives set by the center and provides a much more efficient and easy to use experience for the users, with the further improvements the system can become even better.