Vanier College

Faculty of Science and Technology

System Development

420-436-VA

Deliverable 5

Brown Team

Team Leader: Hibba Qaraman

Sadaf Zakria

Nabil Ramadan

Peter Isaac Fishman

In collaboration with ***Justin Eberwein***

Next Team Leader: Peter Isaac Fishman

We, the Brown Team, certify that this assignment is our own work

I, Sadaf Zakria, [2151361], certify that I have contributed to this deliverable, S. Z.

I, Nabil Ramadan, [2195270], certify that I have contributed to this deliverable, N.R.

I, Hibba Qaraman, [1760010], certify that I have contributed to this deliverable, H.Q.

I, Peter Isaac Fishman, [1980427], certify that I have contributed to this deliverable, P.I.F.

Table of Contents

Executive Overview

In this project phase, our primary focus is to design a relational database system to boost the operational efficiency of "Just B Fitness." The absence of a dedicated database has caused inefficiencies in scheduling and client management.

1. **ER Diagram and Relational Table Structure:** We've created an Entity-Relationship (ER) diagram based on the initial class diagram, outlining the relationships and entities within the system. This diagram serves as the foundation for our relational table structure.
2. **Database Design and Physical Characteristics:** We've analyzed user stories to identify entities and their relationships, ensuring that our design aligns with the client's needs. Additionally, we've considered the physical characteristics of the database system, including hardware requirements and integration capabilities.
3. **Technical Specifications:** Our report outlines the technical specifications for data storage and retrieval, with a focus on performance, database integrity, security, and recoverability.
4. **Scalability and Accessibility:** We've also taken into account the potential growth of the database over the next three to five years and the need for rapid data access to ensure the system remains adaptable to changing requirements.

In summary, our design of the relational database system for the new booking web application is a significant step to transforming "Just B Fitness" into a more efficient and user-friendly fitness business.

Business Problem

The new booking web application is poised to address critical business challenges plaguing Justin Eberwein's "Just B Fitness." Firstly, the absence of a dedicated database and the reliance on a manual client management process has led to inefficiencies in scheduling and information management. Justin currently manages a substantial clientele of over 100 clients, and this manual approach has proven to be both time-consuming and prone to errors. **Without a centralized data repository, client information is scattered and challenging to access, update, and utilize effectively. This reduces the overall operational efficiency of the fitness business, increases the risk of scheduling conflicts, and can result in missed appointments and frustrated clients.**

Secondly, Justin's exhaustive working hours, which extend from 6:30 AM to 11:30 PM, Monday to Saturday, present a considerable business challenge. The current booking process consumes his entire Sundays as he schedules sessions individually with clients, and currently uses a third-party app “FitLog” [1] to provide dietary recommendations to his clients. This not only takes a toll on his well-being but also limits his capacity to focus on other vital aspects of growing and managing his fitness business. It is unsustainable in the long term, and Justin's dedication to his clients should be channeled into more strategic and growth-oriented activities.

Our way to resolve these problems:

The new booking web application is expected to alleviate the strain on Justin's schedule, allowing him to optimize his time and energy for broader business development while ensuring a more efficient and client-friendly booking process.

The following application will be used by both the admin (Justin) and his clientele. The app will allow clients to create an account, create/cancel a booking based on the admin’s (Justin) availability, and view their booking history. The admin Justin will be able to modify his availability, view all his previous and current bookings, cancel bookings, and view all client information.

[1] FitLog is a mobile app for managing coaching that lets businesses make personalized workout plans, meal schedules, supplement recommendations and other things.

Summary of the Client

Justin's journey in the fitness industry began in 2016 when he founded Just B Fitness. Back then, his online presence was limited to Facebook and Instagram, until 2020 when he launched his website. Although the website was created by Silo, who is no longer part of the team, Justin can still rely on him for any necessary website modifications. Today, Justin operates independently but maintains valuable connections with former colleagues Francesca and Ian. He conducts his client sessions at Monster Gym, renting their space for his fitness endeavors. Notably, Justin has collaborated with fitness models throughout his career, as evidenced by the glowing testimonials on his webpage. Despite his lack of computer and programming skills, Justin manages a substantial clientele of over 100 individuals, without the help of a database. His work days span from 6:30 AM to 11:30 PM, Monday through Saturday, with Sundays dedicated to scheduling sessions with clients individually. To streamline his clients' fitness journeys, he relies on the FitLog mobile app, using it to monitor their health and fitness goals. Additionally, Justin collaborates with nutritionists to offer tailored diet plans through the FitLog app, ensuring a comprehensive approach to his client's well-being.

Narrative Description of the Database

The database for "Just B Fitness" is structured to streamline the business's operational efficiency, focusing on client management and appointment scheduling. The database is divided into several entities: User, Client, Booking, Admin, Availability, and Timeslots, each serving a distinct purpose within the system.

Users can register, log in, and update their profiles. Clients, a subset of Users, have additional capabilities such as creating and managing bookings, choosing available timeslots, and deleting their accounts if needed. The Admin, also a type of User, has control over the system, including viewing and modifying schedules, searching and updating bookings, managing client information, and overseeing account status.

The Booking entity is central to the system, capturing all details related to the scheduling of fitness sessions. Each booking record links to a specific Client and includes a unique identifier, the date and time of the booking, and client details. This integration ensures that bookings are tightly coupled with client accounts, maintaining data integrity and providing a seamless user experience.

The Availability and Timeslots entities are crucial for managing the dynamic scheduling requirements of the business. They work together to reflect the Admin's (Justin's) available hours, allowing Clients to book sessions based on real-time availability data. This setup not only optimizes Justin's time but also provides clients with the convenience of scheduling appointments according to their preferences.

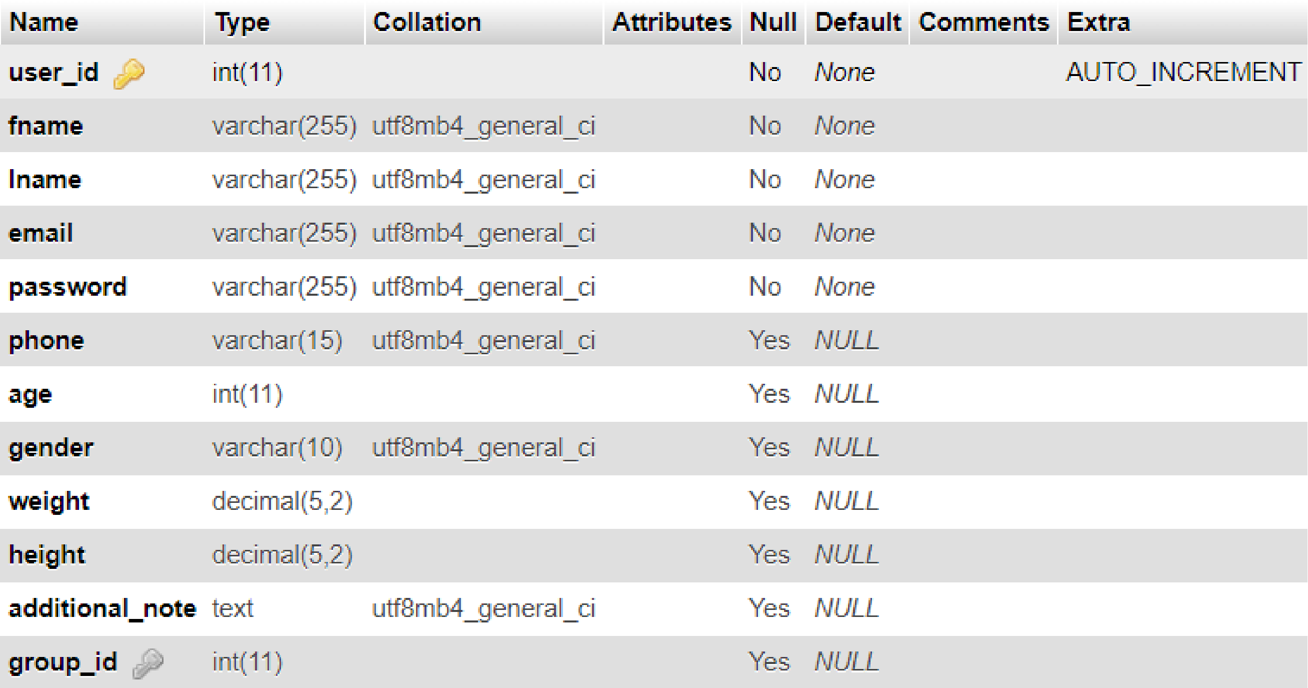
Furthermore, the database is designed for scalability, anticipating the potential inclusion of additional user types and permissions in the future. This is encapsulated within the Group entity, which categorizes users and categorizes their permissions, a feature that allows both flexibility and control over user access and system functionality.

The database design emphasizes speed and efficiency, particularly in response times during booking transactions, to ensure that Justin can rapidly access and manage bookings, which is vital given the high frequency of client interactions. To achieve this, optimization techniques such as proper indexing and selective data retrieval are employed.

Appendix 1: Data Dictionary

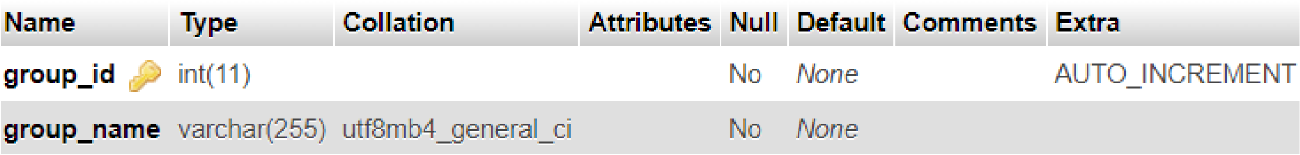
Database Name: justbfitness

Data Entity: USER



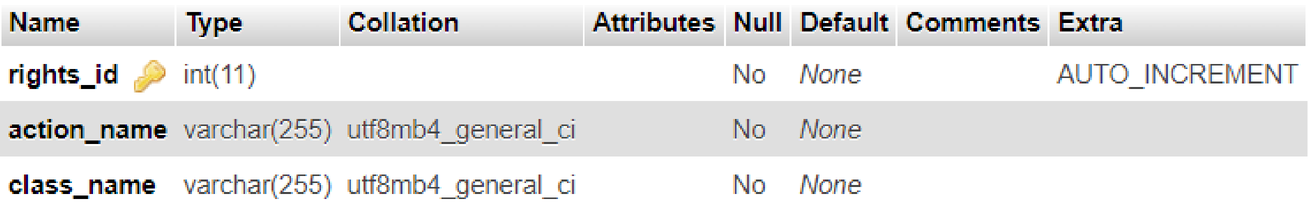
Name	Type	Collation	Attributes	Null	Default	Comments	Extra
user_id 🔑	int(11)			No	*None*		AUTO_INCREMENT
fname	varchar(255)	utf8mb4_general_ci		No	*None*		
lname	varchar(255)	utf8mb4_general_ci		No	*None*		
email	varchar(255)	utf8mb4_general_ci		No	*None*		
password	varchar(255)	utf8mb4_general_ci		No	*None*		
phone	varchar(15)	utf8mb4_general_ci		Yes	*NULL*		
age	int(11)			Yes	*NULL*		
gender	varchar(10)	utf8mb4_general_ci		Yes	*NULL*		
weight	decimal(5,2)			Yes	*NULL*		
height	decimal(5,2)			Yes	*NULL*		
additional_note	text	utf8mb4_general_ci		Yes	*NULL*		
group_id 🔑	int(11)			Yes	*NULL*		

Data Entity: GROUP



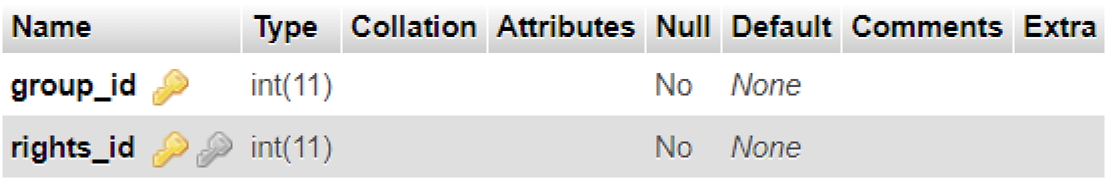
Name	Type	Collation	Attributes	Null	Default	Comments	Extra
group_id 🔑	int(11)			No	*None*		AUTO_INCREMENT
group_name	varchar(255)	utf8mb4_general_ci		No	*None*		

Data Entity: RIGHTS



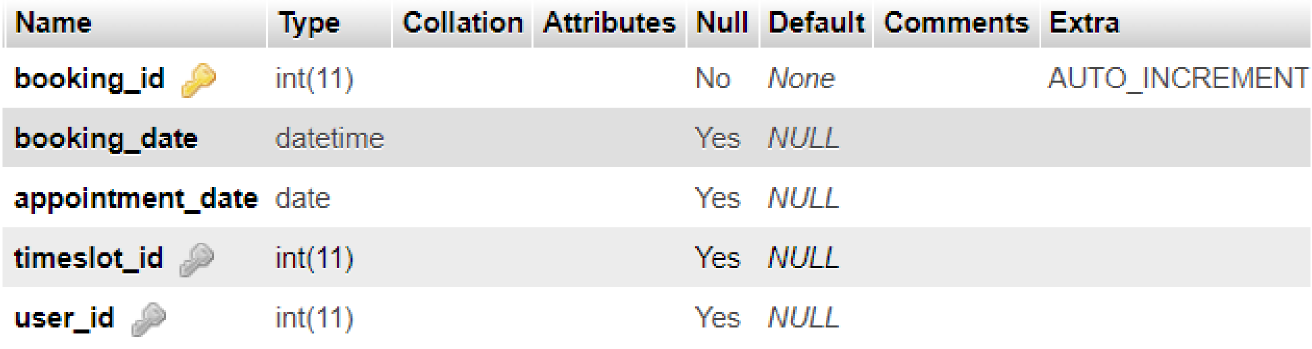
Name	Type	Collation	Attributes	Null	Default	Comments	Extra
rights_id 🔑	int(11)			No	*None*		AUTO_INCREMENT
action_name	varchar(255)	utf8mb4_general_ci		No	*None*		
class_name	varchar(255)	utf8mb4_general_ci		No	*None*		

Data Entity: GROUP_RIGHTS



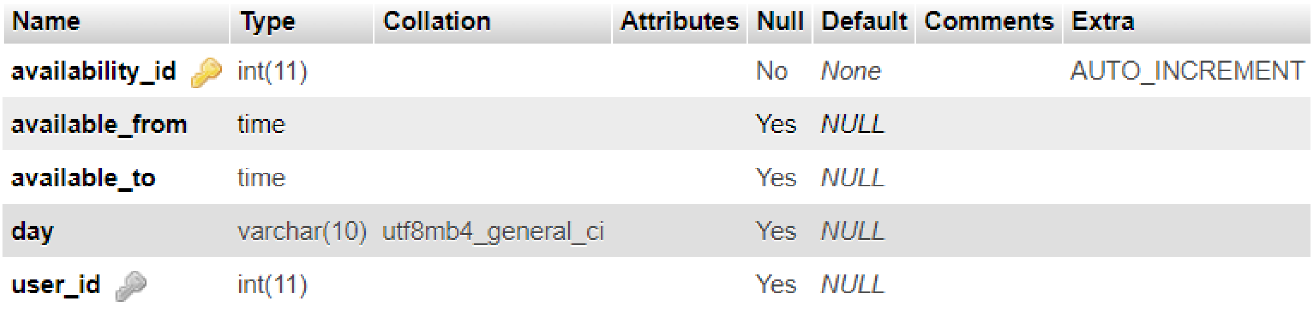
Name	Type	Collation	Attributes	Null	Default	Comments	Extra
group_id 🔑	int(11)			No	*None*		
rights_id 🔑 🗝	int(11)			No	*None*		

Data Entity: BOOKING



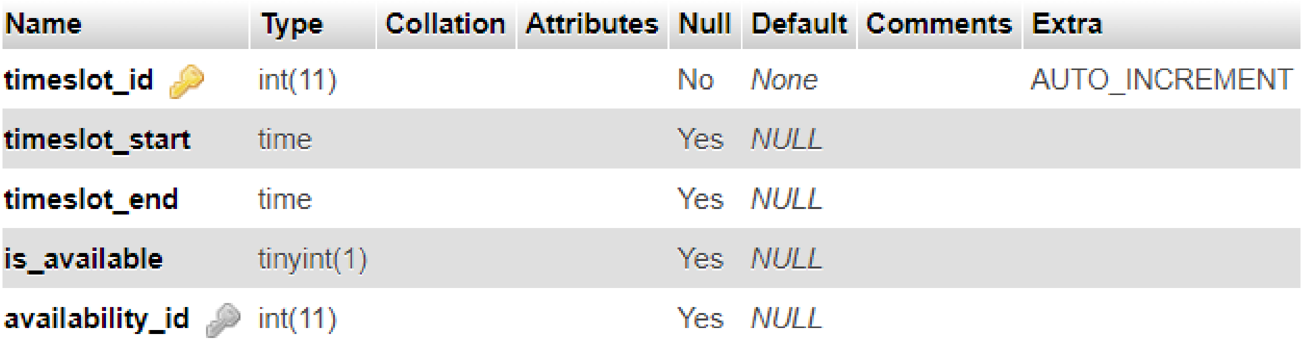
Name	Type	Collation	Attributes	Null	Default	Comments	Extra
booking_id 🔑	int(11)			No	*None*		AUTO_INCREMENT
booking_date	datetime			Yes	*NULL*		
appointment_date	date			Yes	*NULL*		
timeslot_id 🗝	int(11)			Yes	*NULL*		
user_id 🗝	int(11)			Yes	*NULL*		

Data Entity: AVAILABILITY



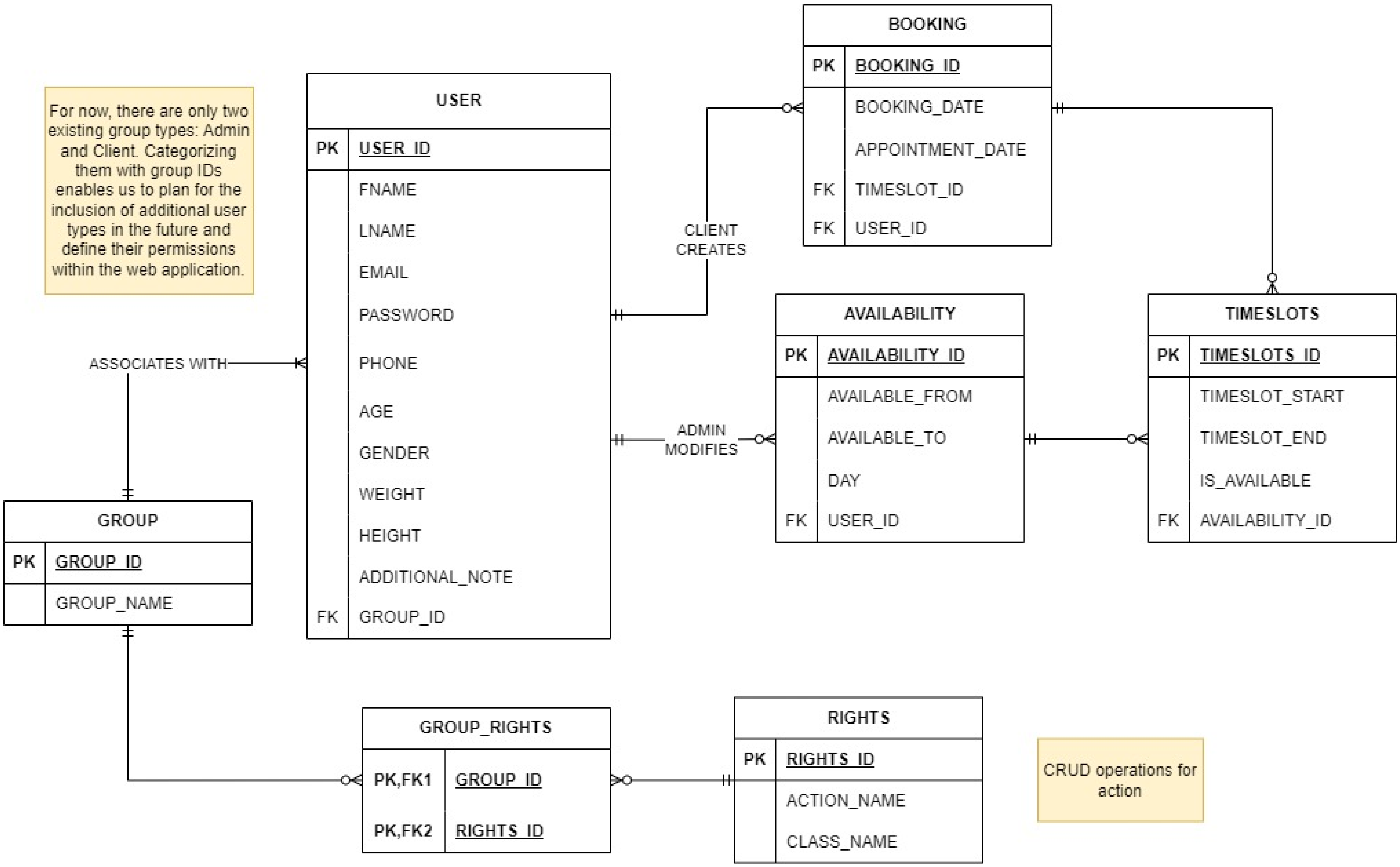
Name	Type	Collation	Attributes	Null	Default	Comments	Extra
availability_id 🔑	int(11)			No	*None*		AUTO_INCREMENT
available_from	time			Yes	*NULL*		
available_to	time			Yes	*NULL*		
day	varchar(10)	utf8mb4_general_ci		Yes	*NULL*		
user_id 🗝	int(11)			Yes	*NULL*		

Data Entity: TIMESLOTS



Name	Type	Collation	Attributes	Null	Default	Comments	Extra
timeslot_id 🔑	int(11)			No	*None*		AUTO_INCREMENT
timeslot_start	time			Yes	*NULL*		
timeslot_end	time			Yes	*NULL*		
is_available	tinyint(1)			Yes	*NULL*		
availability_id 🗝	int(11)			Yes	*NULL*		

Appendix 2: Entity-Relationship Diagram



For now, there are only two existing group types: Admin and Client. Categorizing them with group IDs enables us to plan for the inclusion of additional user types in the future and define their permissions within the web application.

USER	
PK	USER_ID
	FNAME
	LNAME
	EMAIL
	PASSWORD
	PHONE
	AGE
	GENDER
	WEIGHT
	HEIGHT
	ADDITIONAL_NOTE
FK	GROUP_ID

BOOKING	
PK	BOOKING_ID
	BOOKING_DATE
	APPOINTMENT_DATE
FK	TIMESLOT_ID
FK	USER_ID

AVAILABILITY	
PK	AVAILABILITY_ID
	AVAILABLE_FROM
	AVAILABLE_TO
	DAY
FK	USER_ID

TIMESLOTS	
PK	TIMESLOTS_ID
	TIMESLOT_START
	TIMESLOT_END
	IS_AVAILABLE
FK	AVAILABILITY_ID

GROUP	
PK	GROUP_ID
	GROUP_NAME

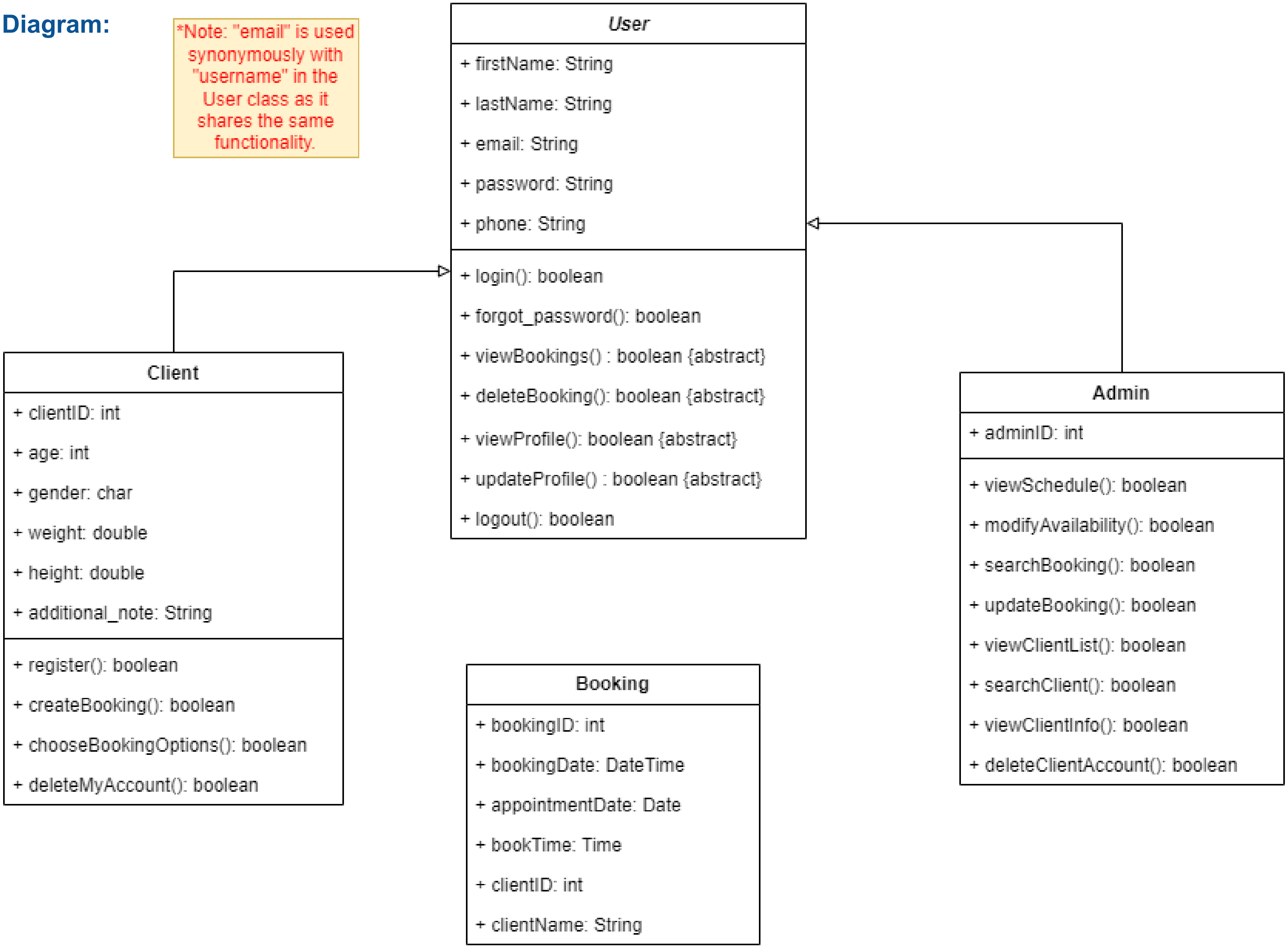
GROUP_RIGHTS	
PK,FK1	GROUP_ID
PK,FK2	RIGHTS_ID

RIGHTS	
PK	RIGHTS_ID
	ACTION_NAME
	CLASS_NAME

CRUD operations for action

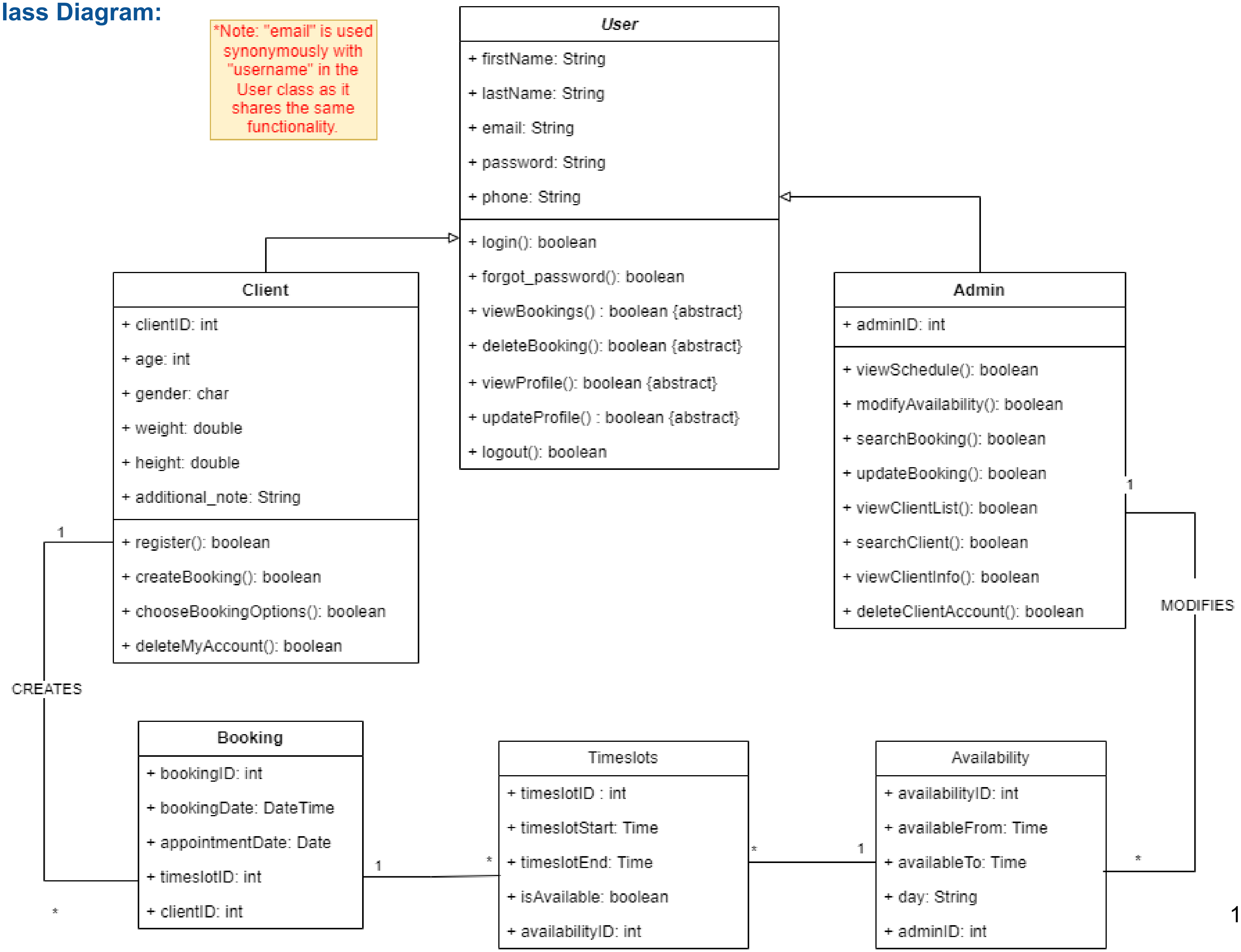
The original class diagram from the third deliverable has been updated with minor adjustments, including new variables and modified functions.

Class Diagram:


*Note: "email" is used synonymously with "username" in the User class as it shares the same functionality.
User
+ firstName: String
+ lastName: String
+ email: String
+ password: String
+ phone: String
+ login(): boolean
+ forgot_password(): boolean
+ viewBookings() : boolean {abstract}
+ deleteBooking(): boolean {abstract}
+ viewProfile(): boolean {abstract}
+ updateProfile() : boolean {abstract}
+ logout(): boolean
Client
+ clientID: int
+ age: int
+ gender: char
+ weight: double
+ height: double
+ additional_note: String
+ register(): boolean
+ createBooking(): boolean
+ chooseBookingOptions(): boolean
+ deleteMyAccount(): boolean
Admin
+ adminID: int
+ viewSchedule(): boolean
+ modifyAvailability(): boolean
+ searchBooking(): boolean
+ updateBooking(): boolean
+ viewClientList(): boolean
+ searchClient(): boolean
+ viewClientInfo(): boolean
+ deleteClientAccount(): boolean
Booking
+ bookingID: int
+ bookingDate: DateTime
+ appointmentDate: Date
+ bookTime: Time
+ clientID: int
+ clientName: String

The revised class diagram has been modified to be similar to our UML while maintaining the same classes as the original class diagram.

Revised Class Diagram:


*Note: "email" is used synonymously with "username" in the User class as it shares the same functionality.
User
+ firstName: String
+ lastName: String
+ email: String
+ password: String
+ phone: String
+ login(): boolean
+ forgot_password(): boolean
+ viewBookings() : boolean {abstract}
+ deleteBooking(): boolean {abstract}
+ viewProfile(): boolean {abstract}
+ updateProfile() : boolean {abstract}
+ logout(): boolean
Client
+ clientID: int
+ age: int
+ gender: char
+ weight: double
+ height: double
+ additional_note: String
+ register(): boolean
+ createBooking(): boolean
+ chooseBookingOptions(): boolean
+ deleteMyAccount(): boolean
Admin
+ adminID: int
+ viewSchedule(): boolean
+ modifyAvailability(): boolean
+ searchBooking(): boolean
+ updateBooking(): boolean
+ viewClientList(): boolean
+ searchClient(): boolean
+ viewClientInfo(): boolean
+ deleteClientAccount(): boolean
1
CREATES
*
1
MODIFIES
*
Booking
+ bookingID: int
+ bookingDate: DateTime
+ appointmentDate: Date
+ timeslotID: int
+ clientID: int
1
*
Timeslots
+ timeslotID : int
+ timeslotStart: Time
+ timeslotEnd: Time
+ isAvailable: boolean
+ availabilityID: int
*
1
Availability
+ availabilityID: int
+ availableFrom: Time
+ availableTo: Time
+ day: String
+ adminID: int

Since we developed the original database early on, we had many uncertainties regarding the database entities and their contents due to the lack of a fully developed plan. Following the development of the mockup of the web application, we gained a better understanding of what needed to be completed for the current database.

Similarities

- The admin and client classes share the same data, now consolidated under one entity; the "USER" table in the UML diagram.
- The "BOOKING" table and data structure are still in place, and we have defined relationships between entities to provide a clearer view of how various databases are interconnected.

Differences

- The absence of an inheritance structure in the UML diagram.
- To manage user roles and permissions more effectively, we have chosen to consolidate all admin and client data into a single entity called "USER," rather than maintaining separate entities for "ADMIN" and "CLIENT."
- To differentiate between user types, we've created a "GROUP" entity to define the user roles.
- The "USER_GROUP" relationship is defined using the "GROUP_ID" foreign key. This enables us to determine the roles and permissions assigned to users. Currently, we have two primary group types: Admin and Client. The inclusion of group IDs allocates the future addition of more user types, each with predefined permissions within the web application. In other words, this framework allows for future expansion.
- The specification of permissions, specifically for CRUD operations (Create, Read, Update, Delete) on various classes of the database, is established through the "RIGHTS" table, which is interconnected with the "GROUP_RIGHTS" table.
- The enhancement involves designing a more efficient database for managing bookings and schedule changes. This includes the introduction of a "BOOKING" table, along with an "AVAILABILITY" or "TIMESLOTS" table, which will be valuable for future API implementation.

Appendix 3: Descriptions and Explanations of Database

We have meticulously optimized our database by implementing effective indexes, steering clear of excessive joins, and adopting a selective approach in our queries, favoring specific fields over a general 'Select *'. This optimization is geared towards enhancing access speed, ultimately contributing to an improved performance of our web application for our clients.

In The USER Table:

Index	From	Description
group_id	GROUP	The User table is linked to the Group table, allowing for the assignment of specific roles and permissions to individual users based on their group rights. This connection facilitates the restriction of access to data for users who do not possess the necessary permissions, preventing unnecessary access.

In The GROUP_RIGHTS Table:

Index	From	Description
group_id	GROUP	The Group_rights table is linked to the Group table. When a user creates an account, they transition from a generic user to a specific group (e.g., Client). This connection ensures that users are assigned appropriate actions and permissions based on their group rights.
rights_id	RIGHTS	The Group_Rights table is linked to the Rights table. When a user registers for an account, they are automatically enrolled in the Client group, defining the specific actions and permissions associated with their account.

In The Booking Table:

Index	From	Description
timeslots_id	TIMESLOTS	The Booking table is connected to the Timeslots table. This connection ensures that a booking can only be made for an existing timeslot.
user_id	USER	The Booking table is connected to the User table. This connection ensures that only logged-in clients can make bookings.

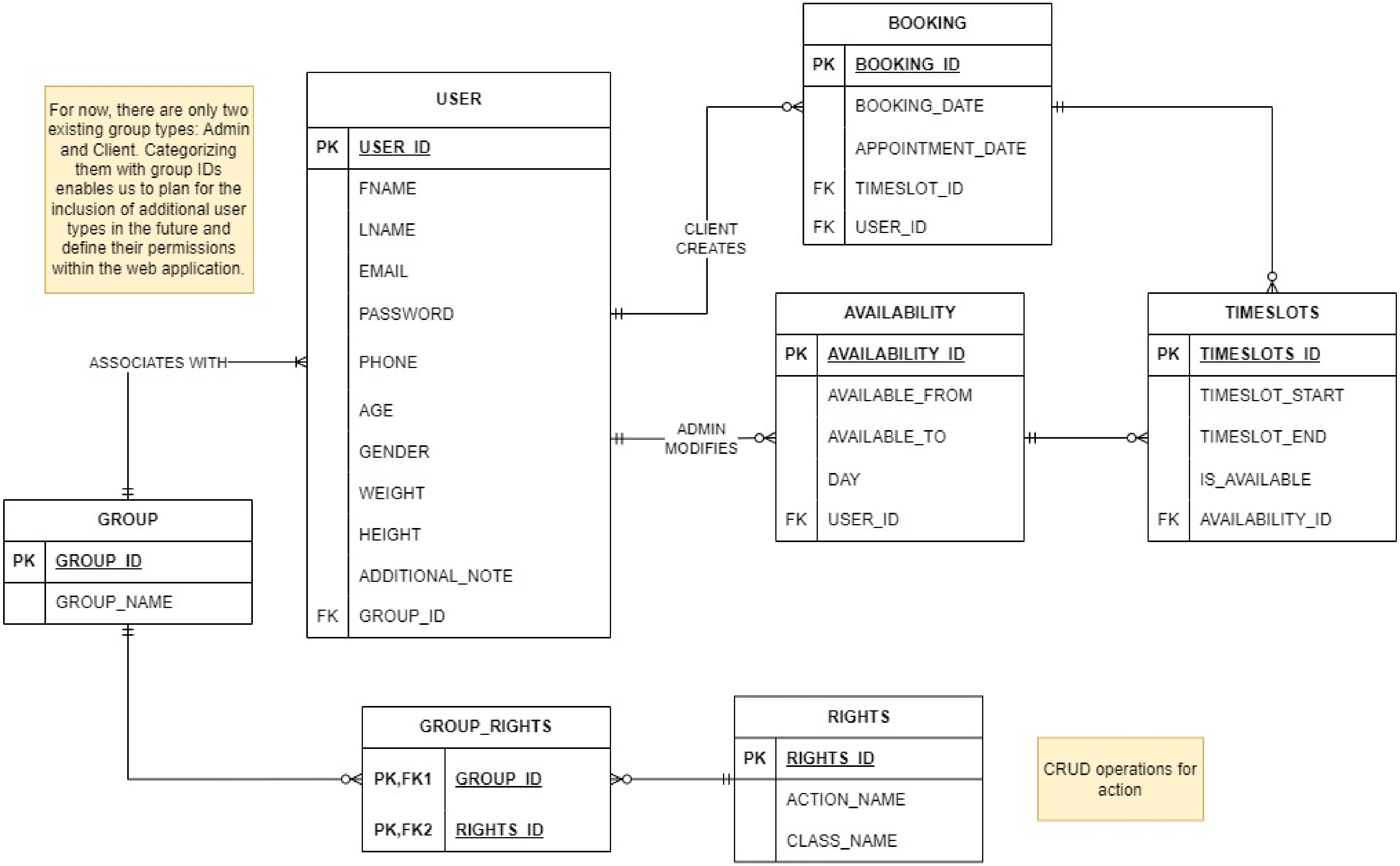
In The Timeslots Table:

Index	From	Description
availability_id	AVAILABILITY	The Timeslots table is linked to the Availability table. This connection ensures that timeslots are associated with specific availability periods, which are, in turn, associated with Admin's availability.

In The Availability Table:

Index	From	Description
user_id	USER	The Availability table is linked to the User table. This connection allows for updates to Admin availability and informs clients of any changes.

Normalization DB classes:



For now, there are only two existing group types: Admin and Client. Categorizing them with group IDs enables us to plan for the inclusion of additional user types in the future and define their permissions within the web application.

USER	
PK	USER_ID
	FNAME
	LNAME
	EMAIL
	PASSWORD
	PHONE
	AGE
	GENDER
	WEIGHT
	HEIGHT
	ADDITIONAL_NOTE
FK	GROUP_ID

BOOKING	
PK	BOOKING_ID
	BOOKING_DATE
	APPOINTMENT_DATE
FK	TIMESLOT_ID
FK	USER_ID

AVAILABILITY	
PK	AVAILABILITY_ID
	AVAILABLE_FROM
	AVAILABLE_TO
	DAY
FK	USER_ID

TIMESLOTS	
PK	TIMESLOTS_ID
	TIMESLOT_START
	TIMESLOT_END
	IS_AVAILABLE
FK	AVAILABILITY_ID

GROUP	
PK	GROUP_ID
	GROUP_NAME

GROUP_RIGHTS	
PK,FK1	GROUP_ID
PK,FK2	RIGHTS_ID

RIGHTS	
PK	RIGHTS_ID
	ACTION_NAME
	CLASS_NAME

CRUD operations for action

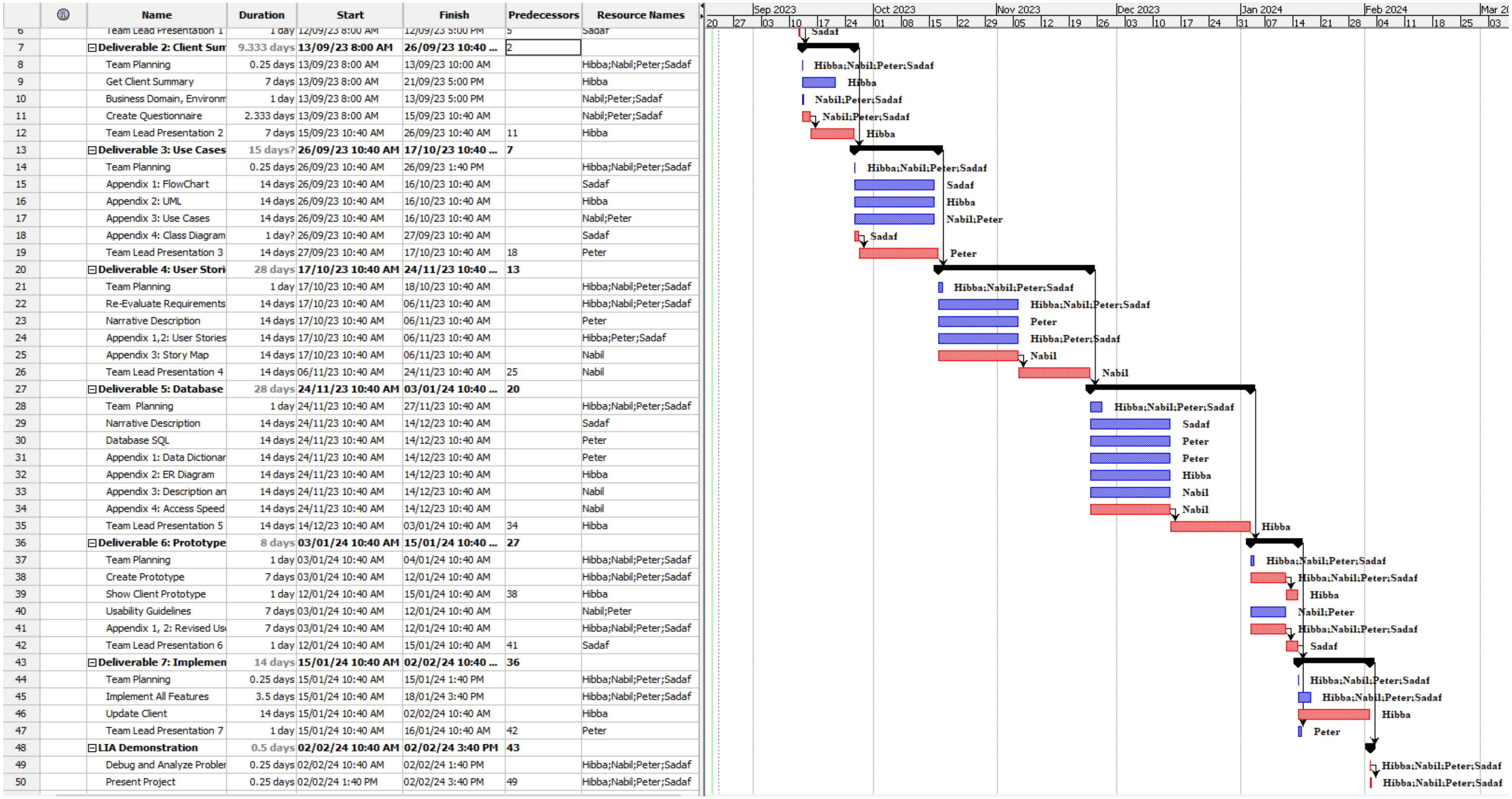
Appendix 4: The Access Speed Required

The required access speed for our database is critical to ensure seamless operations. When a customer books an appointment, the data must become quickly available to Justin for viewing. To meet this demand, our database design focuses on optimizing access speed. We have structured the database to provide Justin with rapid access to vital information in a lightweight format, aligning with his primary need to periodically view client data while efficiently managing his schedule and staying connected with clients.

Frequency of Database Access and Necessary Response Time:

The database will be accessed frequently due to the dynamic nature of client bookings. Justin will need to access the database regularly to stay informed and manage his clients effectively. Therefore, a fast response time is imperative to meet these operational needs. With our design approach, which includes proper indexing and query optimization, we aim to ensure that Justin's web application consistently delivers the best possible speed and responsiveness, aligning with his access frequency and performance requirements.

Project Plan



		Name	Duration	Start	Finish	Predecessors	Resource Names
6		Team Lead Presentation 1	1 day	12/09/23 8:00 AM	12/09/23 5:00 PM	5	Sadaf
7		⊟ **Deliverable 2: Client Sum**	9.333 days	**13/09/23 8:00 AM**	**26/09/23 10:40 ...**	2	
8		Team Planning	0.25 days	13/09/23 8:00 AM	13/09/23 10:00 AM		Hibba;Nabil;Peter;Sadaf
9		Get Client Summary	7 days	13/09/23 8:00 AM	21/09/23 5:00 PM		Hibba
10		Business Domain, Environm	1 day	13/09/23 8:00 AM	13/09/23 5:00 PM		Nabil;Peter;Sadaf
11		Create Questionnaire	2.333 days	13/09/23 8:00 AM	15/09/23 10:40 AM		Nabil;Peter;Sadaf
12		Team Lead Presentation 2	7 days	15/09/23 10:40 AM	26/09/23 10:40 AM	11	Hibba
13		⊟ **Deliverable 3: Use Cases**	15 days?	**26/09/23 10:40 AM**	**17/10/23 10:40 ...**	**7**	
14		Team Planning	0.25 days	26/09/23 10:40 AM	26/09/23 1:40 PM		Hibba;Nabil;Peter;Sadaf
15		Appendix 1: FlowChart	14 days	26/09/23 10:40 AM	16/10/23 10:40 AM		Sadaf
16		Appendix 2: UML	14 days	26/09/23 10:40 AM	16/10/23 10:40 AM		Hibba
17		Appendix 3: Use Cases	14 days	26/09/23 10:40 AM	16/10/23 10:40 AM		Nabil;Peter
18		Appendix 4: Class Diagram	1 day?	26/09/23 10:40 AM	27/09/23 10:40 AM		Sadaf
19		Team Lead Presentation 3	14 days	27/09/23 10:40 AM	17/10/23 10:40 AM	18	Peter
20		⊟ **Deliverable 4: User Stori**	28 days	**17/10/23 10:40 AM**	**24/11/23 10:40 ...**	**13**	
21		Team Planning	1 day	17/10/23 10:40 AM	18/10/23 10:40 AM		Hibba;Nabil;Peter;Sadaf
22		Re-Evaluate Requirements	14 days	17/10/23 10:40 AM	06/11/23 10:40 AM		Hibba;Nabil;Peter;Sadaf
23		Narrative Description	14 days	17/10/23 10:40 AM	06/11/23 10:40 AM		Peter
24		Appendix 1,2: User Stories	14 days	17/10/23 10:40 AM	06/11/23 10:40 AM		Hibba;Peter;Sadaf
25		Appendix 3: Story Map	14 days	17/10/23 10:40 AM	06/11/23 10:40 AM		Nabil
26		Team Lead Presentation 4	14 days	06/11/23 10:40 AM	24/11/23 10:40 AM	25	Nabil
27		⊟ **Deliverable 5: Database**	28 days	**24/11/23 10:40 AM**	**03/01/24 10:40 ...**	**20**	
28		Team Planning	1 day	24/11/23 10:40 AM	27/11/23 10:40 AM		Hibba;Nabil;Peter;Sadaf
29		Narrative Description	14 days	24/11/23 10:40 AM	14/12/23 10:40 AM		Sadaf
30		Database SQL	14 days	24/11/23 10:40 AM	14/12/23 10:40 AM		Peter
31		Appendix 1: Data Dictionar	14 days	24/11/23 10:40 AM	14/12/23 10:40 AM		Peter
32		Appendix 2: ER Diagram	14 days	24/11/23 10:40 AM	14/12/23 10:40 AM		Hibba
33		Appendix 3: Description an	14 days	24/11/23 10:40 AM	14/12/23 10:40 AM		Nabil
34		Appendix 4: Access Speed	14 days	24/11/23 10:40 AM	14/12/23 10:40 AM		Nabil
35		Team Lead Presentation 5	14 days	14/12/23 10:40 AM	03/01/24 10:40 AM	34	Hibba
36		⊟ **Deliverable 6: Prototype**	8 days	**03/01/24 10:40 AM**	**15/01/24 10:40 ...**	**27**	
37		Team Planning	1 day	03/01/24 10:40 AM	04/01/24 10:40 AM		Hibba;Nabil;Peter;Sadaf
38		Create Prototype	7 days	03/01/24 10:40 AM	12/01/24 10:40 AM		Hibba;Nabil;Peter;Sadaf
39		Show Client Prototype	1 day	12/01/24 10:40 AM	15/01/24 10:40 AM	38	Hibba
40		Usability Guidelines	7 days	03/01/24 10:40 AM	12/01/24 10:40 AM		Nabil;Peter
41		Appendix 1, 2: Revised Us	7 days	03/01/24 10:40 AM	12/01/24 10:40 AM		Hibba;Nabil;Peter;Sadaf
42		Team Lead Presentation 6	1 day	12/01/24 10:40 AM	15/01/24 10:40 AM	41	Sadaf
43		⊟ **Deliverable 7: Implemen**	14 days	**15/01/24 10:40 AM**	**02/02/24 10:40 ...**	**36**	
44		Team Planning	0.25 days	15/01/24 10:40 AM	15/01/24 1:40 PM		Hibba;Nabil;Peter;Sadaf
45		Implement All Features	3.5 days	15/01/24 10:40 AM	18/01/24 3:40 PM		Hibba;Nabil;Peter;Sadaf
46		Update Client	14 days	15/01/24 10:40 AM	02/02/24 10:40 AM		Hibba
47		Team Lead Presentation 7	1 day	15/01/24 10:40 AM	16/01/24 10:40 AM	42	Peter
48		⊟ **LIA Demonstration**	0.5 days	**02/02/24 10:40 AM**	**02/02/24 3:40 PM**	**43**	
49		Debug and Analyze Probler	0.25 days	02/02/24 10:40 AM	02/02/24 1:40 PM		Hibba;Nabil;Peter;Sadaf
50		Present Project	0.25 days	02/02/24 1:40 PM	02/02/24 3:40 PM	49	Hibba;Nabil;Peter;Sadaf

Link to WBS and Gantt Chart & PDF Version