Fall 2021

CSC 3326 - 02

After School Final Project

Team Members

Full Name	ID
Ait Soussi Khaoula	79155
Benabdallah Yassir	76865
Bounajma Nada	77807
Driouech Saad	77804

Supervised By:

Prof. Lamiae Bouanane

Table of Contents

I. Introduction

1. Project Description

Following the scrum agile methodology, we have been adopting since the beginning of this project, we had our final meeting with the client during which we received answers for the rest of the matters that were ambiguous. Afterwards, our backlog, therefore, only include technical issues to be fixed, such as the designs of the interfaces, the code's bugs, and functionalities to be added or updated.

Concerning the team spirit, we worked within an amazing atmosphere. Because this project overlapped with some academic concerns, we had to meet regularly to complete the items pending on our product backlog. Still, these exhausting conditions did not impact the interaction between the team members.

When it comes to the architecture of the team, all the members kept their positions: Yassir as a product owner, Khaoula and Nada as scrum masters and Saad as head developer. However, since the team includes a small number of members, all of us had tasks that belong to different roles.

2. Project Objectives

a. Learning Objectives

Within the context of Database Systems, this the first time the team members are exposed to such projects. In Software Engineering class, we did not have to implement a back-end for our system; therefore, interfaces for all the possible users were more than enough to be on the final deliverable. Now, it is time to upgrade our skills, by developing a web application with a back-end and a front-end, in addition to a database, which is the core of the entire project.

Thanks to the content of this course, we have succeeded in understanding the main challenge of building a database, that is its design. Throughout the whole semester, we have learned all the possible techniques that have allowed us to translate the excel sheets provided by the client into an Entity Relationship Diagram, and therefore build an SQL code accordingly.

Connecting our database to Java, the main programming language we used to implement our project, was not that easy task. Even though we went through such practice during one lab session, its content was not enough to build a web app from scratch. However, thanks to many internet resources in addition to the appreciated assistance of the tutor, we managed to overcome this problem.

The teammates are proud to add this project to the collection of projects each member has performed during their academic and professional careers. We are all considering this final deliverable as a micro internship, which would be a preparation to engage in the professional world.

b. Application objectives

The project is a booking system that serves After School, a private tutoring center in Meknes. Students, being users, will be able to book sessions according to their availabilities rather than keeping a fixed schedule. Also, students will be allowed to choose any tutor that offers the session. They can also keep track of their balance which will be deducted whenever they attend the session. Sessions can be private or in groups according to the preferences of the student.

Tutors will have their own interface in which they will be able to add their availability to be seen by the students in order to book sessions. Also, they can keep track of their balance that will be incremented whenever they teach a session.

Finally, an admin view will be reserved to the director/secretaries of the center. They will take care of adding, deleting and updating the information of students and tutors. They will be able to keep track of all the students and the tutors who are enrolled in the center. In fact, they will be responsible of increasing the balances of students once they pay by cash and increase the balances of tutors when they are done with a tutoring session.

3. Project Planning

The following Gantt Chart Summarizes a chronological planning for our project

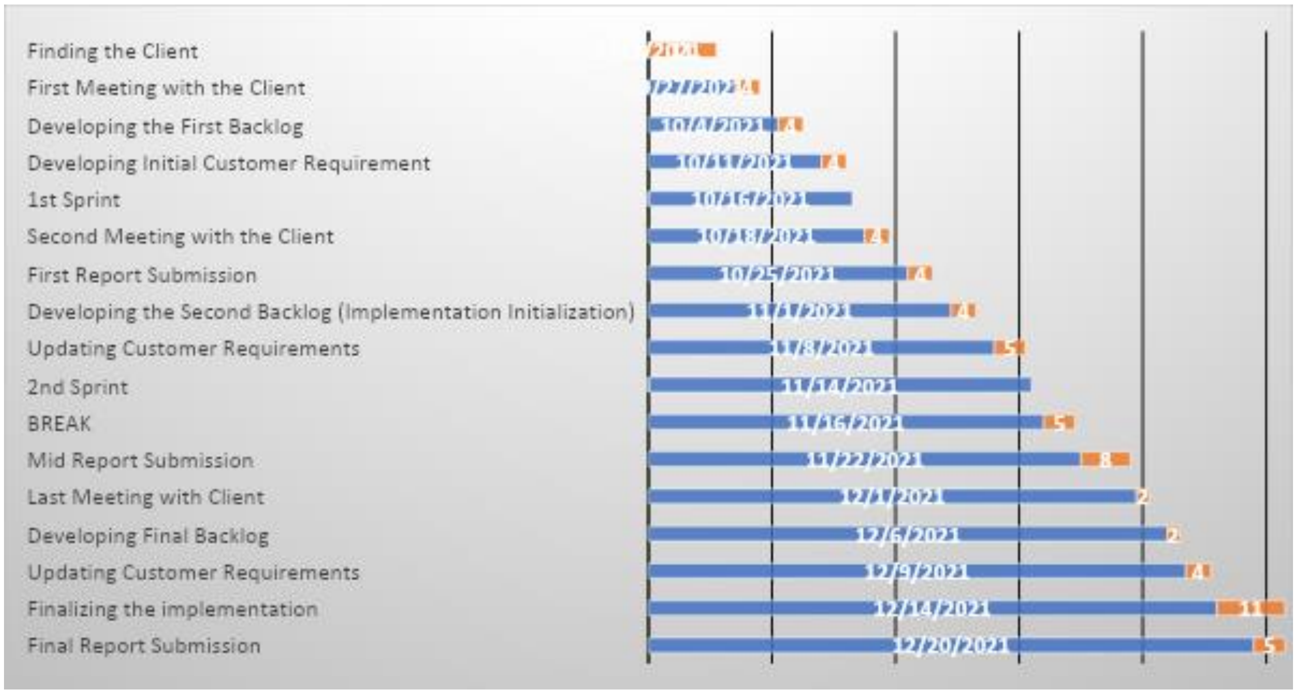


Figure 1: Gantt Chart – Tasks Distribution over Time

We invested a lot of time in finding the client. This step was the first one to complete since the team members needed some time to get familiar with each other, and then start the journey of finding a real-world client who would agree to provide us with their institution data.

Another step that took us some time was the implementation one. In fact, we needed to build a web app from scratch, especially that we do not have a heavy knowledge about coding such programs. It also collided with the final exams, so we could not completely focus on the implementation and build a full functionality system.

There are also some tasks that took place more than one time. These tasks were part of the scrum agile methodology that allows us to have different meetings with the client to better understand their requirements, and hence deliver the best possible product.

II. Design

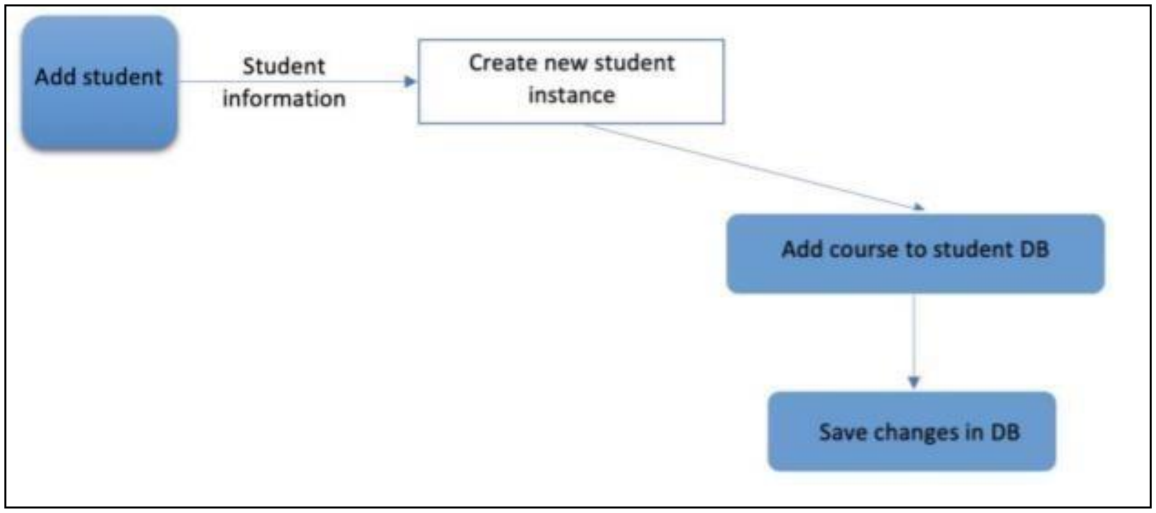
1. Conceptual Design

a. Requirements' Specifications: Main System Processes

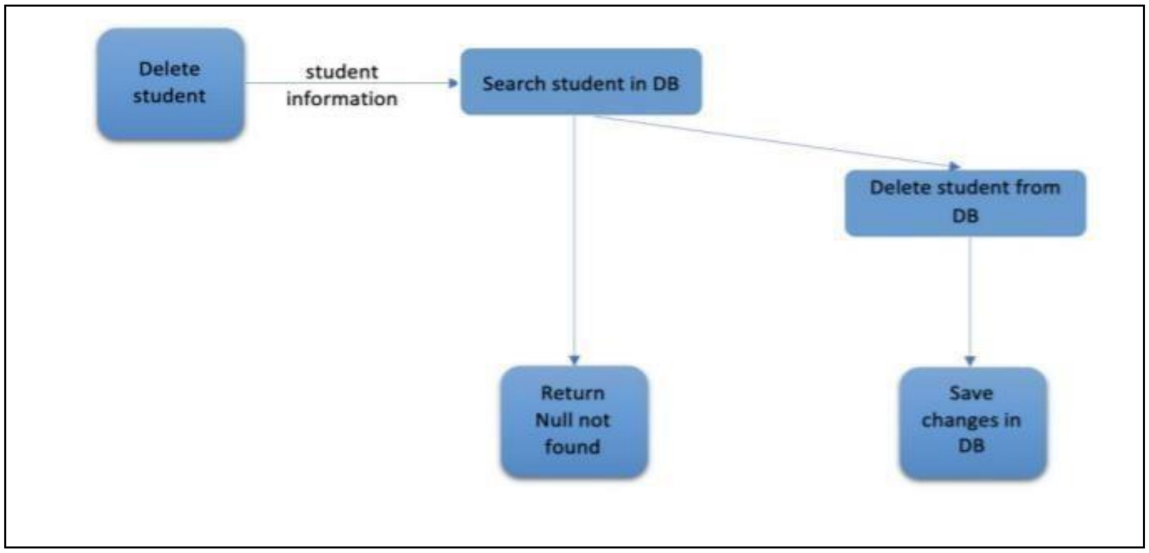
- *Functional Requirements*

Management of Students

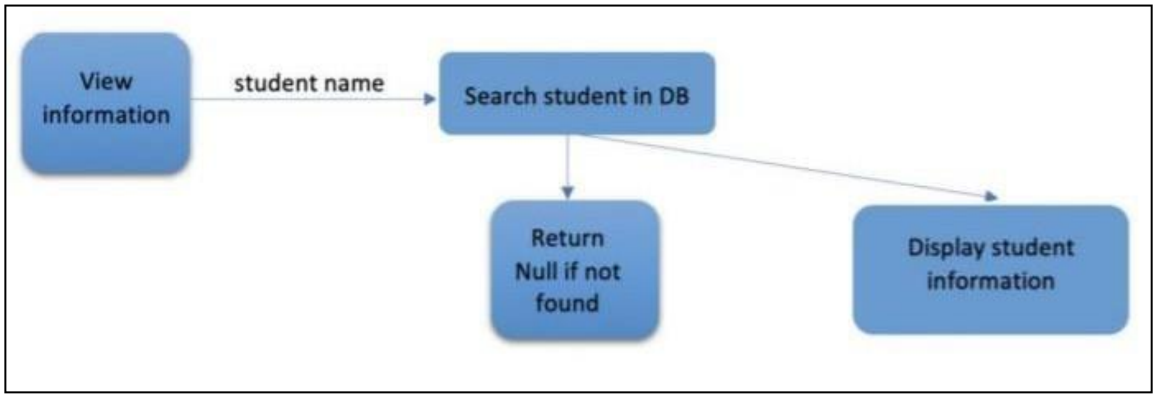
✔ The admin shall add a student to the database



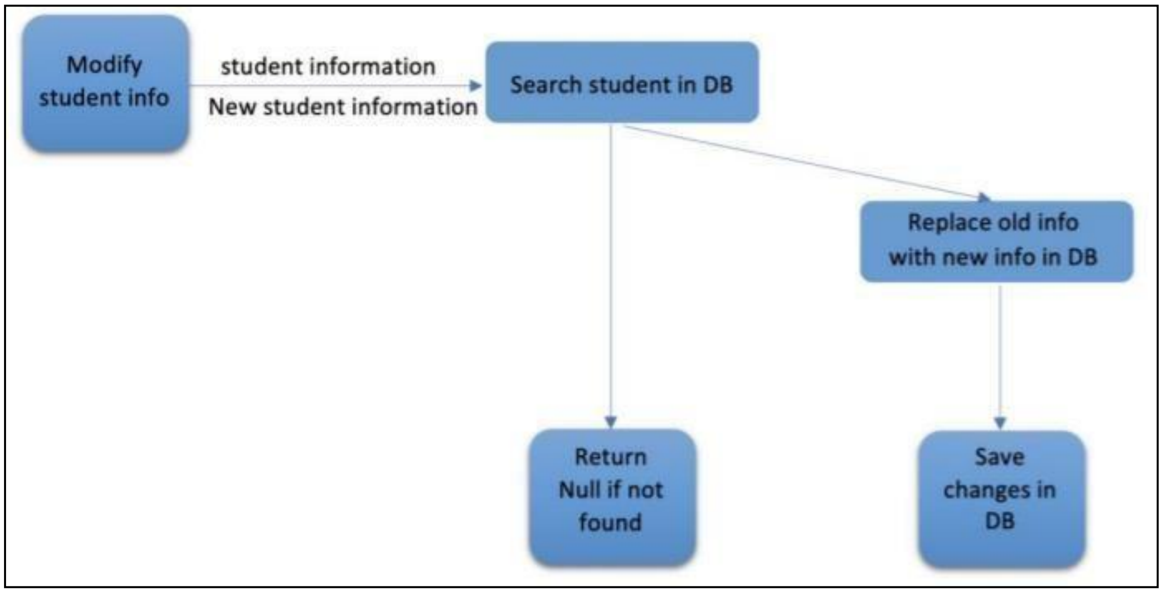
✔ The admin shall delete a student from the database



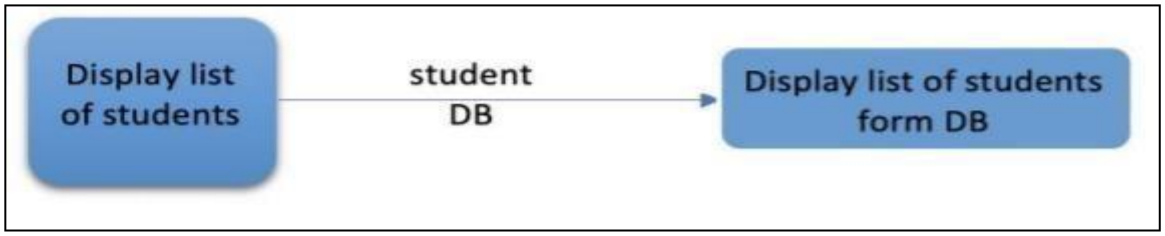
✔ The admin and student shall be able to view the student’s information



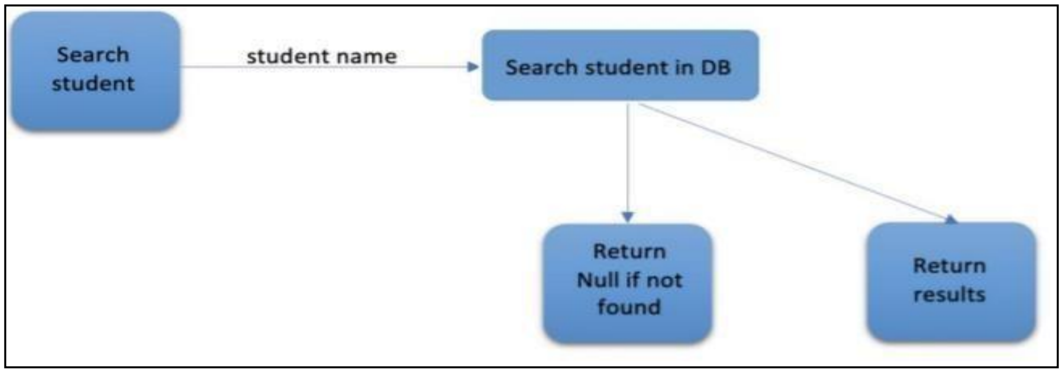
✔ The admin shall be able to modify student information



✔ Users should be able to display the list of students

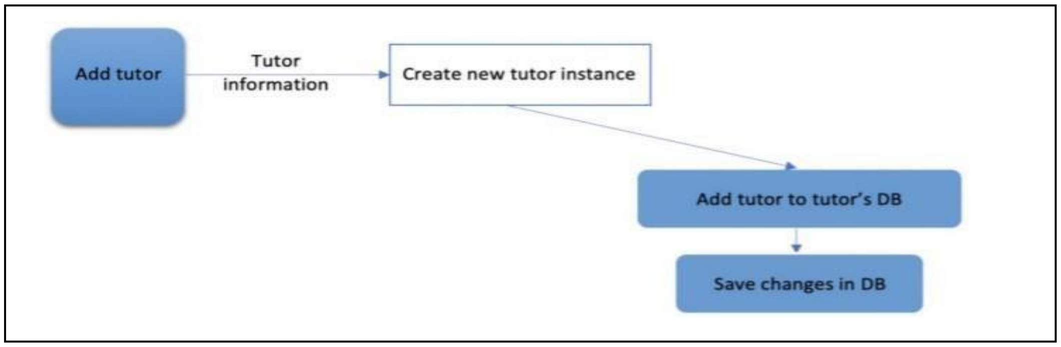


✔ Users should be able to search for a student

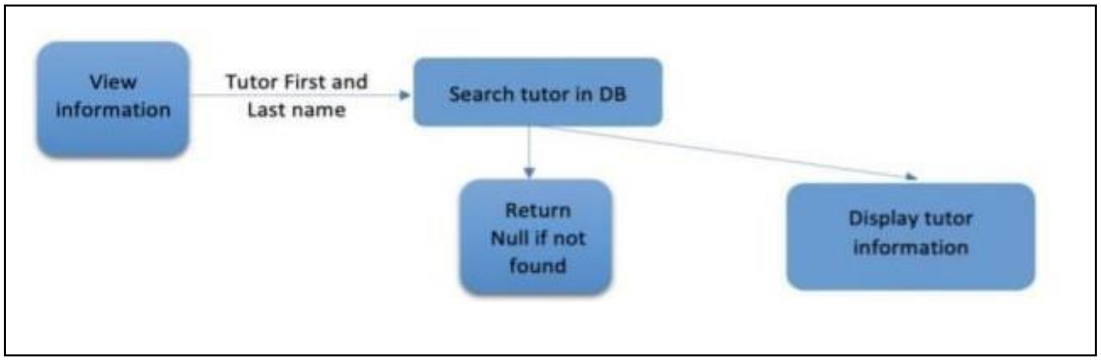


Management of Tutors

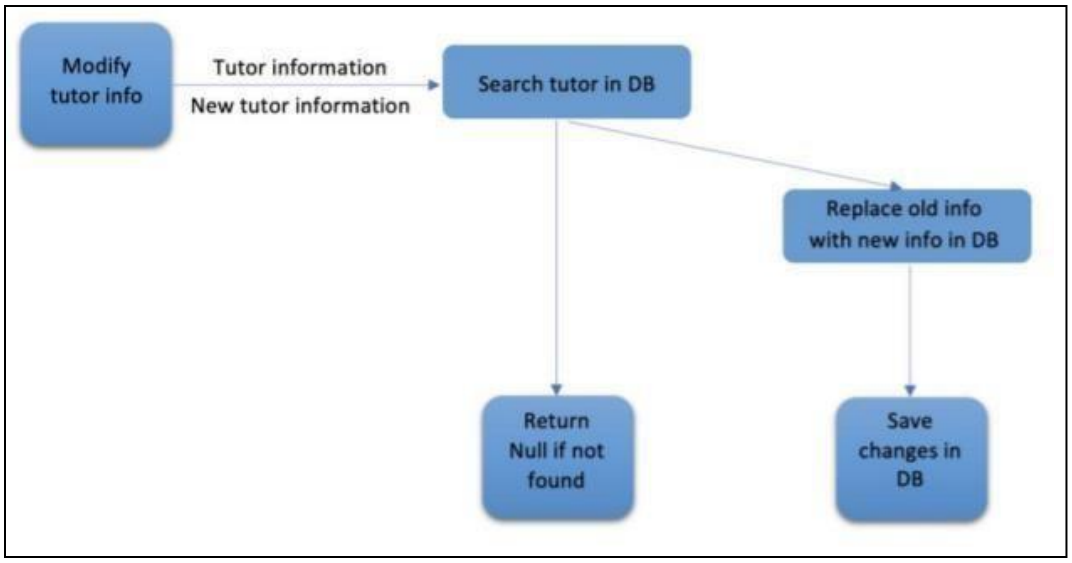
✔ The admin can add tutor to the database



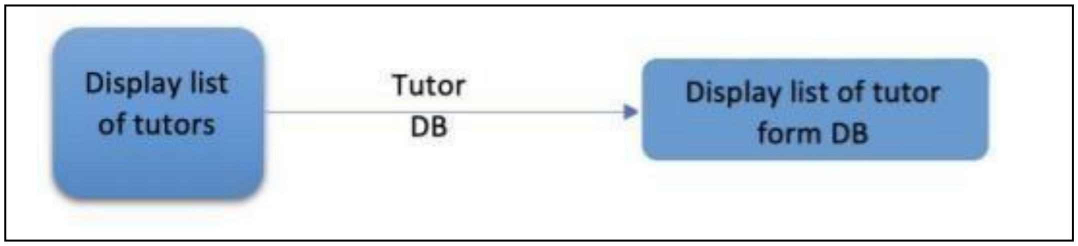
✔ The admin and tutor shall be able to view the tutor's information



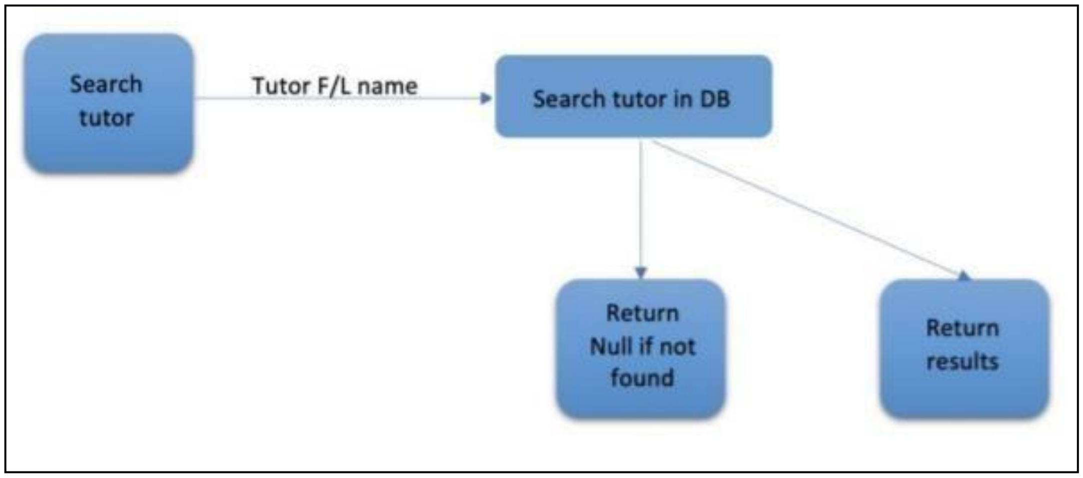
✔ The admin shall be able to modify tutor information



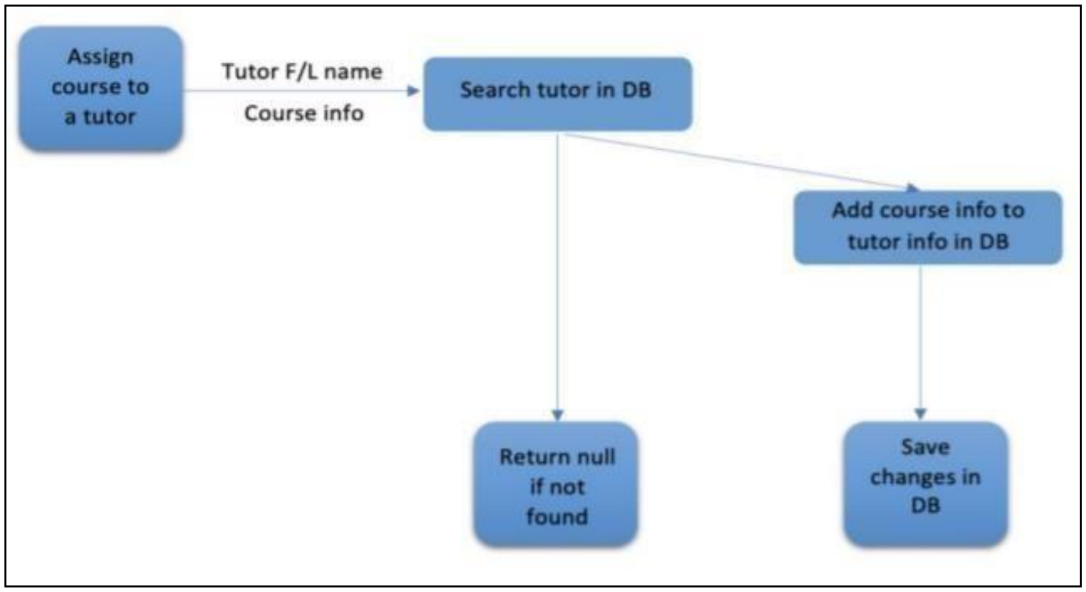
✔ Users shall be able to display the list of tutors



✔ Users shall be able to search a tutor

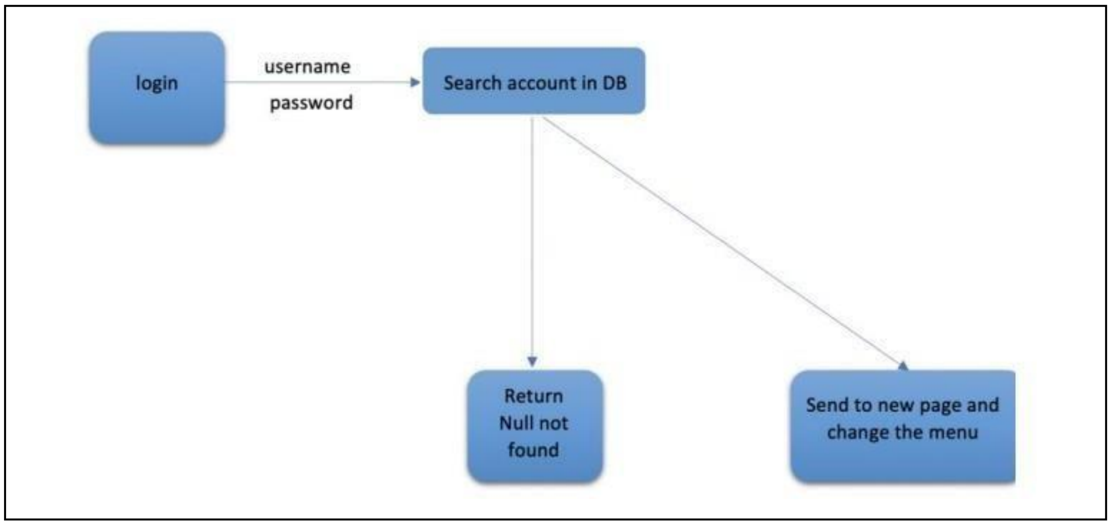


✔ The admin shall be able to assign a course to a tutor

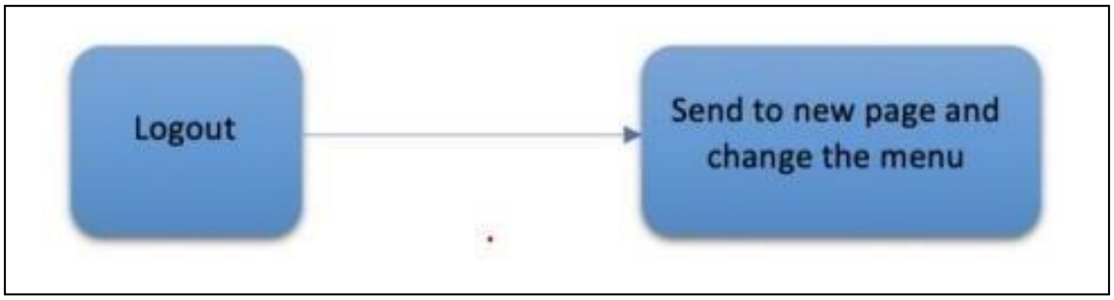


Management of Accounts

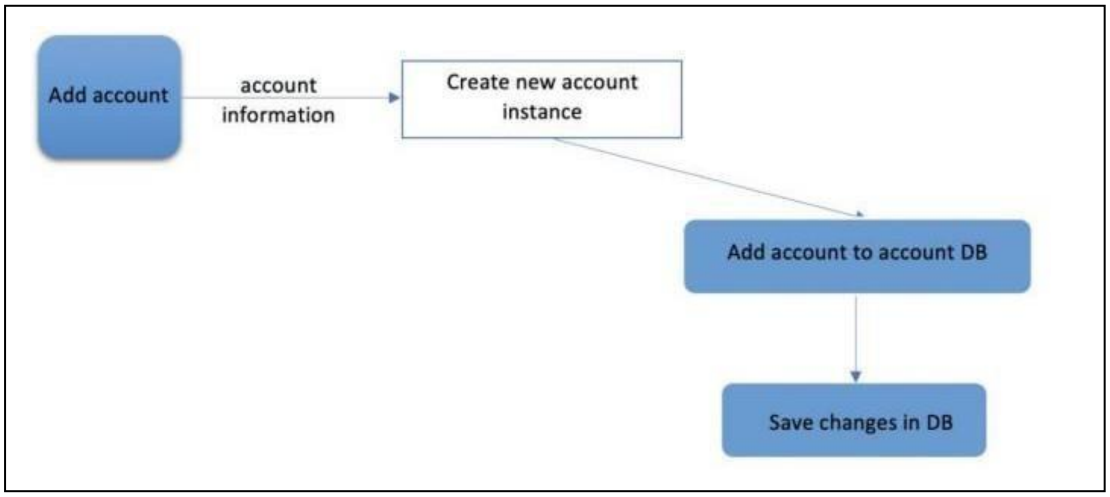
- ✓ The user shall log in using their credentials (username and password)



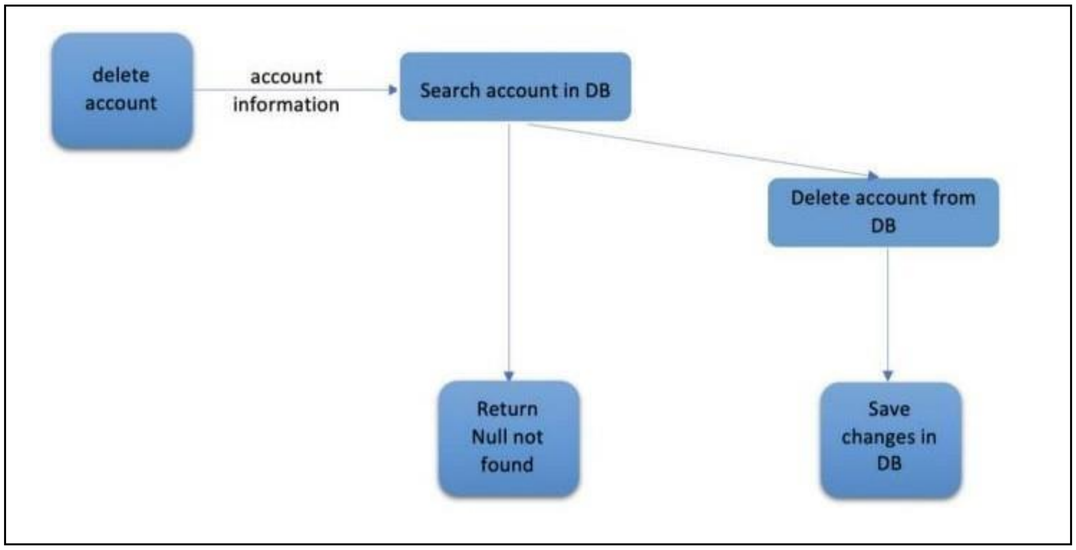
- ✓ The user shall log out using the "Log Out" button.



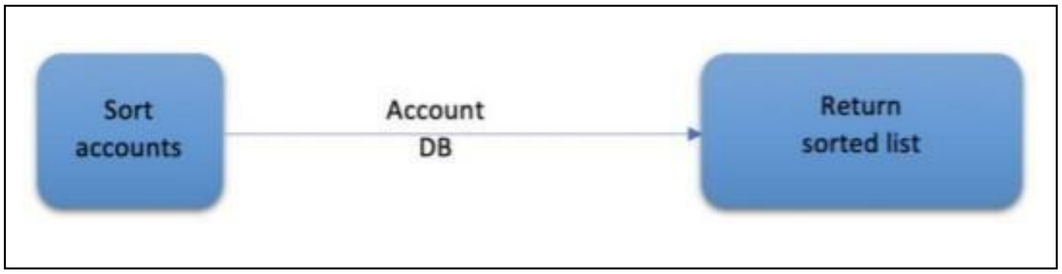
- ✓ The admin shall create new accounts for different users.


Add account
account information
Create new account instance
Add account to account DB
Save changes in DB

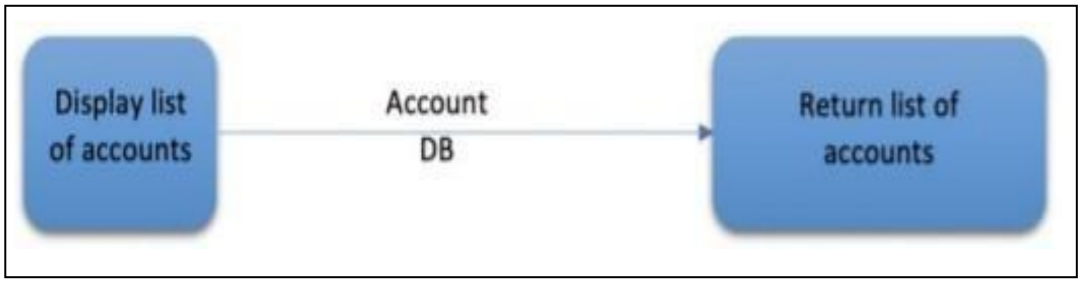
✔ The admin is responsible for deleting accounts.



✔ The admin shall be able to sort accounts

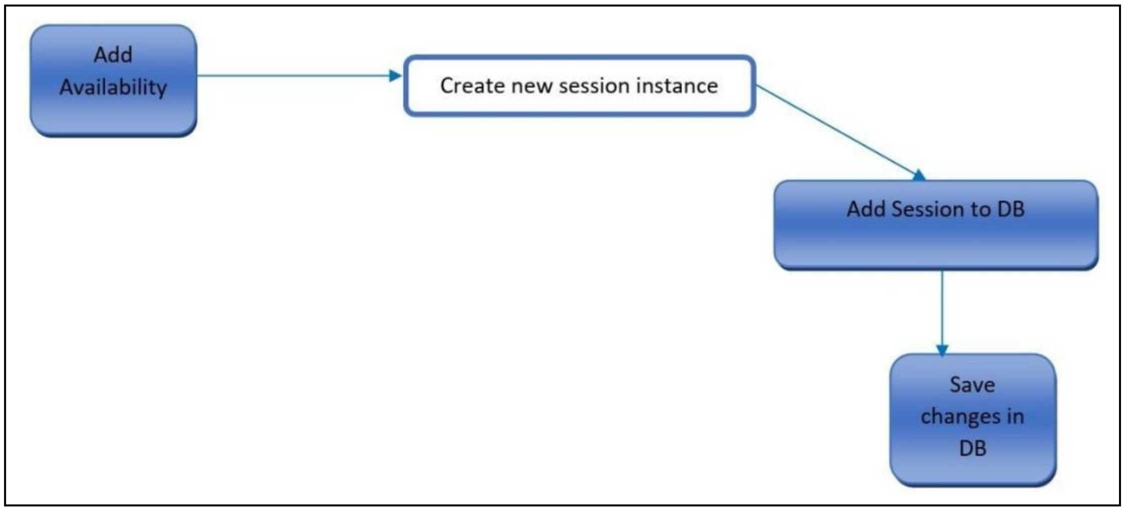


✔ The admin shall be able to display the list of accounts

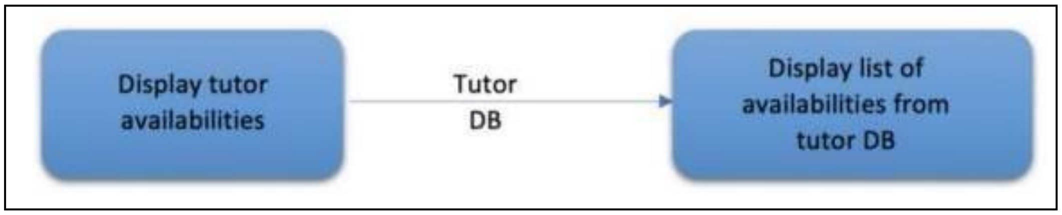


Management of bookings

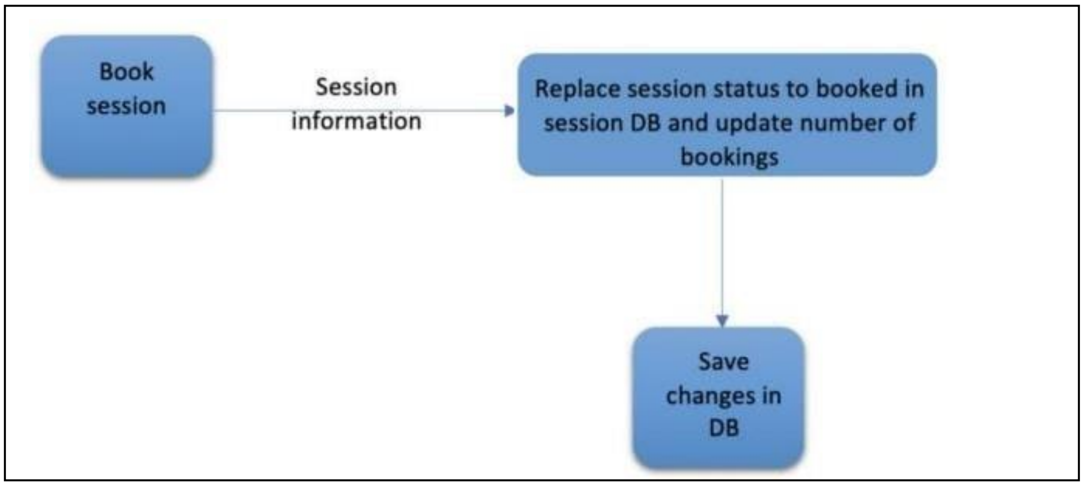
✔ Tutors shall add their availabilities.



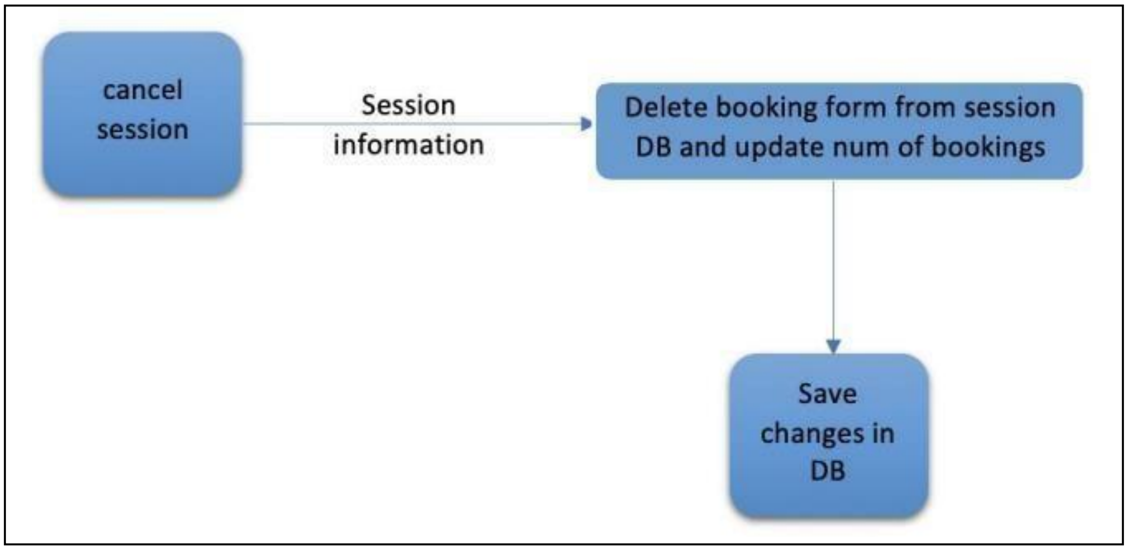
✔ Users shall look up tutors' availabilities by tutor.


Display tutor availabilities
Tutor DB
Display list of availabilities from tutor DB

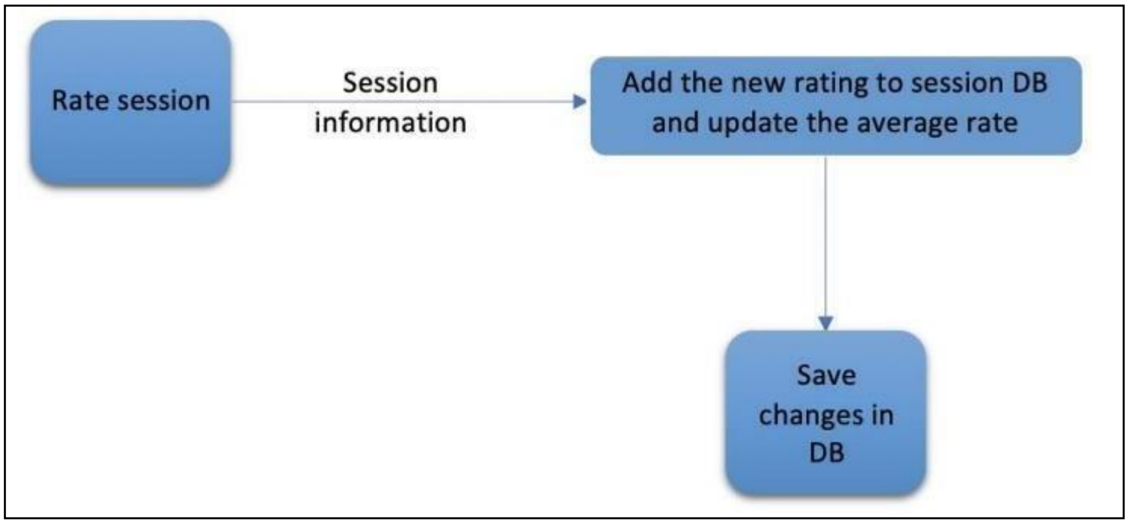
✔ Students shall book sessions by clicking on the wanted availability.


Book session
Session information
Replace session status to booked in session DB and update number of bookings
Save changes in DB

✔ Students shall cancel sessions at least two hours before the starting time of the session.

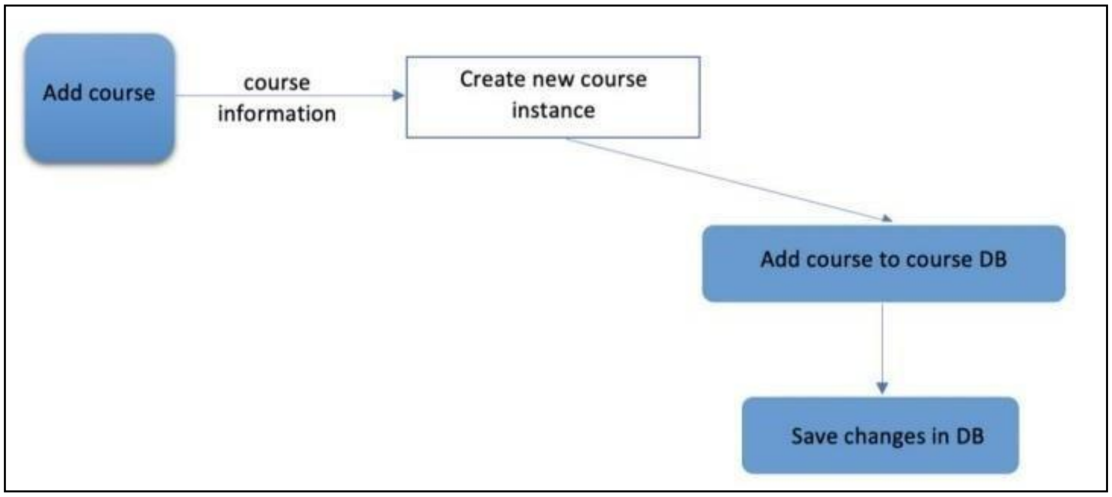

cancel session
Session information
Delete booking form from session DB and update num of bookings
Save changes in DB

✔ Students shall rate the sessions after its end time.

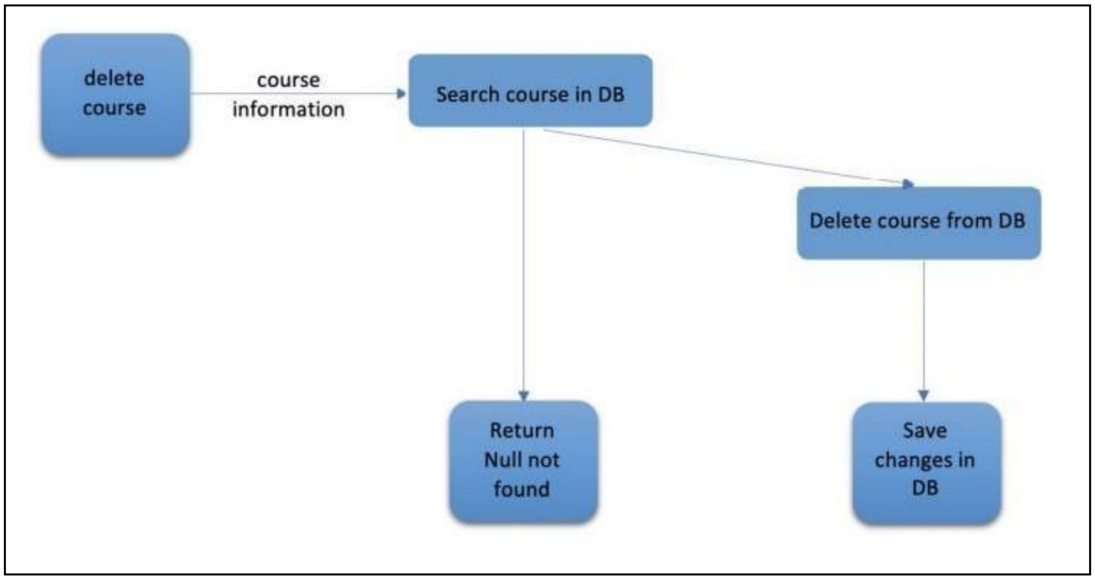

Rate session
Session information
Add the new rating to session DB and update the average rate
Save changes in DB

Management of courses

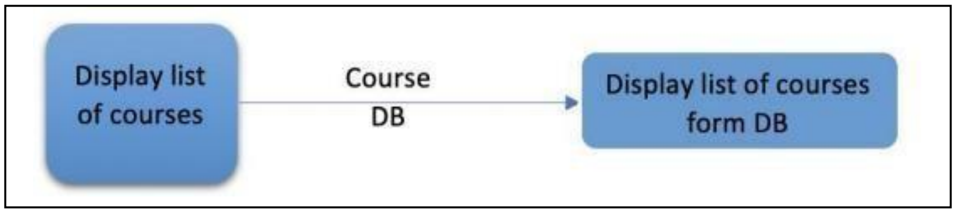
✔ The admin shall add a course to the system.



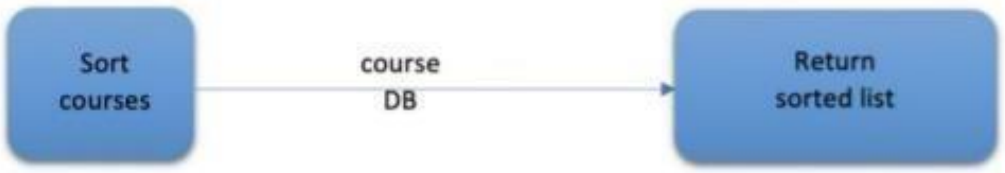
✔ The admin shall delete a course from the system.



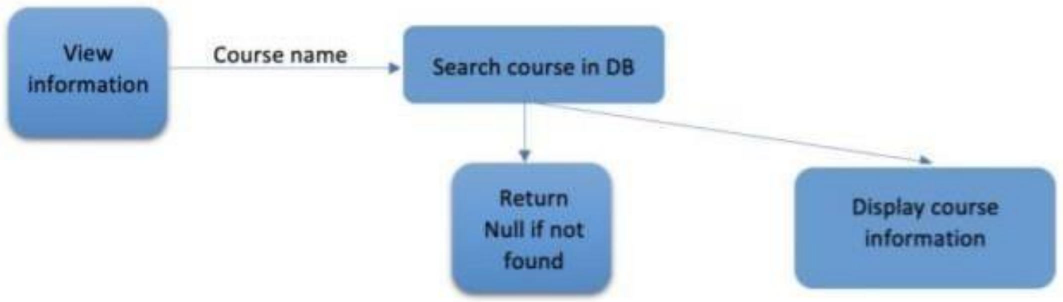
✔ Users shall be able to display the list of courses.



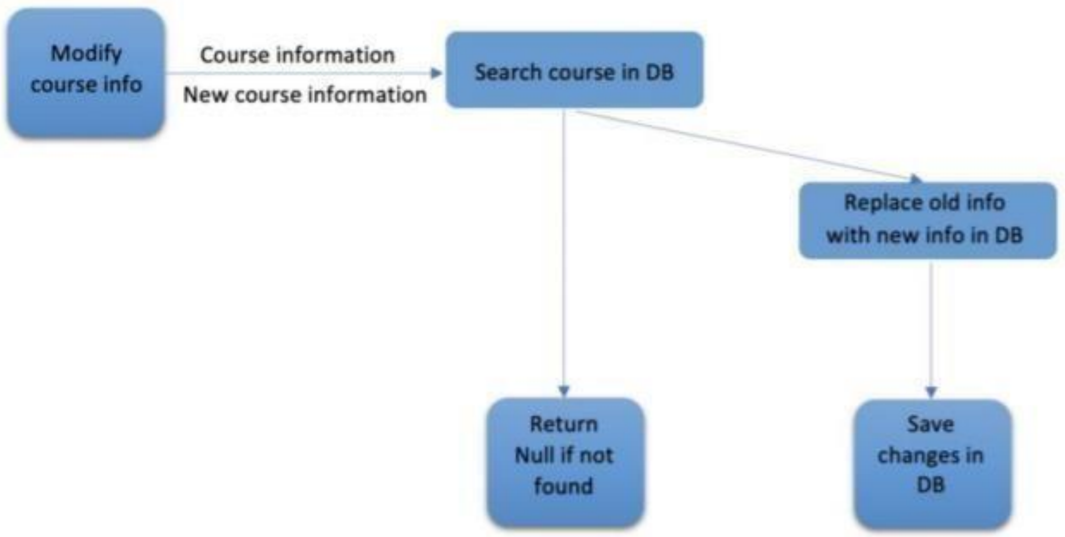
✔ Users shall be able to sort courses



✔ Users shall search for courses by typing its name in the search box.



✔ The admin shall modify a course information



Non-Functional Requirements

Operation Constraints	The system should allow for easy maintenance and maintainability. Crash downs should be delayed as much as possible.
	The system shall not be heavy in terms of size and usage of resources.

	The system should be robust. For instance, in the case of wrong inputs, the system should perform appropriately.
Performance Constraints	The system should be available most of the time, i.e., the downtime should be minimized as possible.
	The system must be able to accommodate for increasing number of users without lowering the performance standards.
Usability Constraints	The system should incorporate three interfaces: · Administrator's interface. · Tutor's interface. · Student's interface.
	Clear and intuitive graphical interfaces: Each interface should have only the necessary functionalities for each type of user. They should be easy to use such that no user faces any difficulties while using the application.
Security Constraints	Because of the nature of the system that requires proceeding with payments and dealing with users' data, data security should be enforced.
Other Constraints	The cost of development should be as cheap as possible.
	The time of finishing the system is limited.

b. Business Rules, Entities, and relationships

- There are three possible users that can use the system: admins, tutors, and students. All these users are uniquely identified by an ID. They can log in using their ID and password provided by the admin. The system records the first name, last name, day of birth and phone number.
- There are 3 admin roles that have the same characteristics: an owner, a secretary, and a manager.
- In addition to the attributes they inherit from user, the system records the balance (i.e., salary) of tutors. The salary is calculated depending on the course they teach. For example, one hour of mathematics is equivalent to MAD 150, while one hour of physics is equivalent to MAD 120.
- Tutors can add many availabilities, and each availability belongs to one tutor.
- Availabilities are uniquely identified by a code in addition to start and end times.

- A tutor can teach many courses and a course can be taught by many tutors.
- Each course is uniquely identified by a course code. The system also records the title of the course, its academic level, and its hourly price.
- Each session a tutor conducts has certain available seats that cannot be exceeded while booking that session by students.
- All sessions have: a date, a beginning and end time, a location, a status (open, done, canceled), and seats available.
- A session can be held in one location, while a location can have many sessions in different time slots.
- Students also inherit the attributes of user (ID, email, password, etc.) Moreover, the balance and number of hours attended by a student are also recorded, as the tenth session can be attended for free.
- Students can attend many sessions and a session can host up to 15 students. Students can rate the session after they finish Business Rules, Entities and Relationships.

c. Initial ER Model

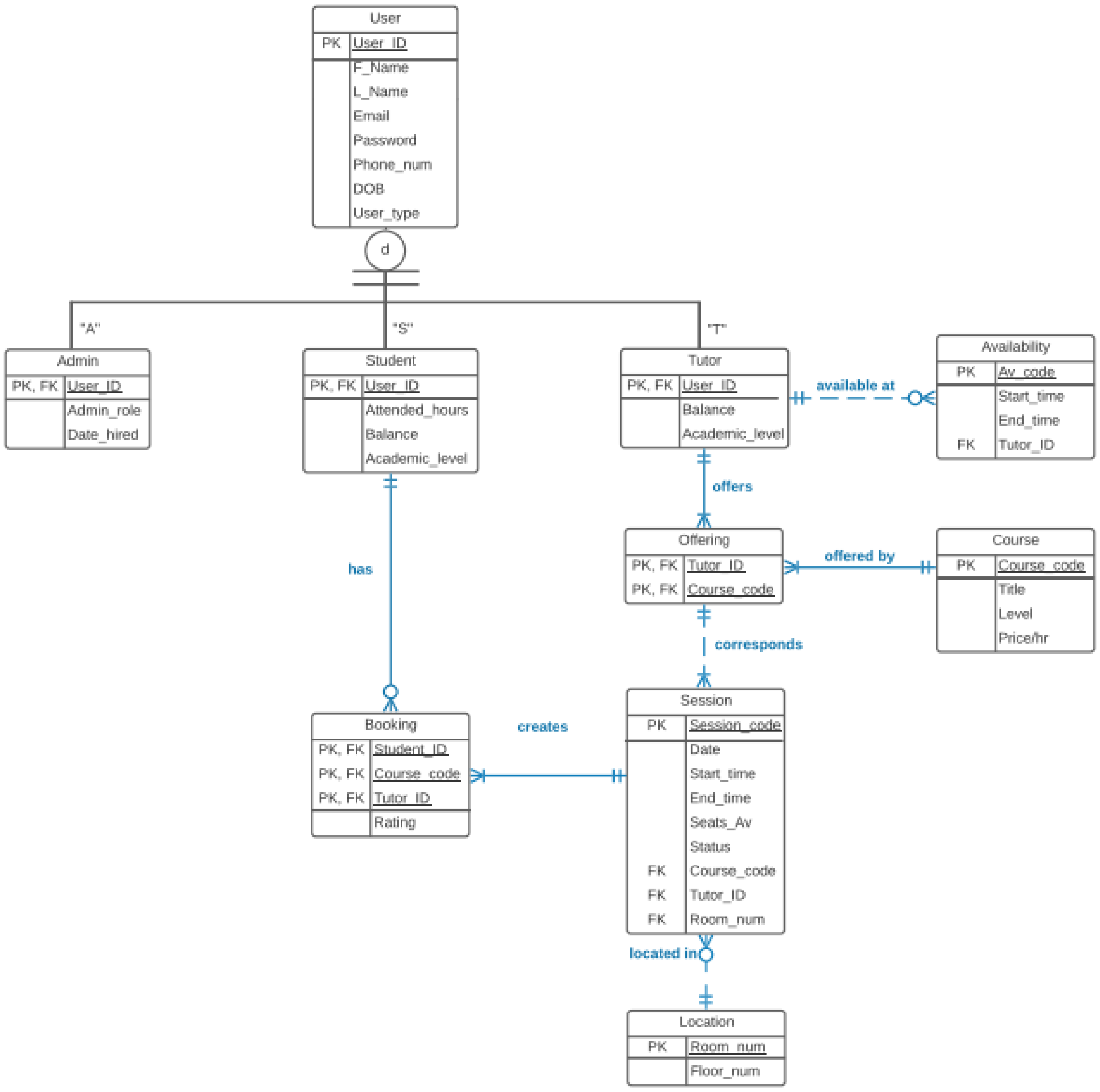


Figure 2: Initial ER Model

d. Table Normalization

1NF Check :

CSC 3326 Database Systems Fall 2021
Final Report

User Table

User_ID	User_FName	User_LName	User_PhoneNum	User_DOB	User_Type

Account Table

User_ID	password

Admin Table

User_ID	Admin_role	Admin_hring_date

Student Table

User_ID	Attended_hours	Stud_Balance

Tutor Table

User_ID	Tutor_balance	Tutor_level

Availability Table

Av_code	Av_Date	Av_Start_time	Av_End_time	Tutor_ID

Offering Table

Tutor_ID	Course_code

Course Table

Course_code	Course_title	Course_level	Course_PriceHr

Booking Table

Student_ID	Session_Code	Rating

Session Table

Session_code	Session_date	Session_start_time	Session_end_time	Session_seats_av	Session_status	Session_course_code	Session_tutor_ID	Session_room_num

Location Table

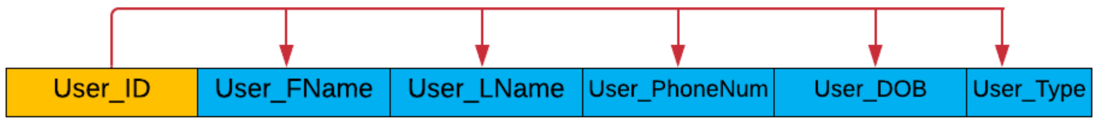
Room_num	Floor_num	Room_seats_av

All the tables are already in 1NF because there is no repeating group, and each cell is supposed to have a single value.

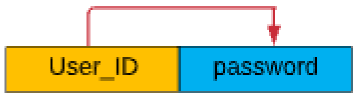
The primary keys are shaded in **orange**, while the other attributes are shaded in **blue**.

- 2NF Check

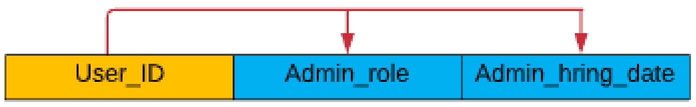
User Table



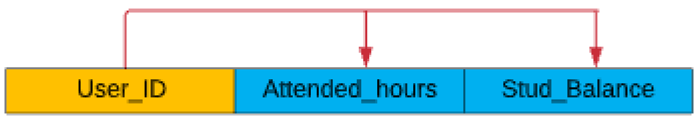
Account Table



Admin Table

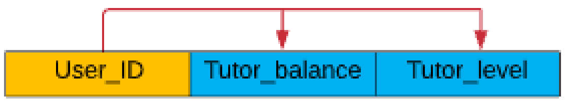


Student Table

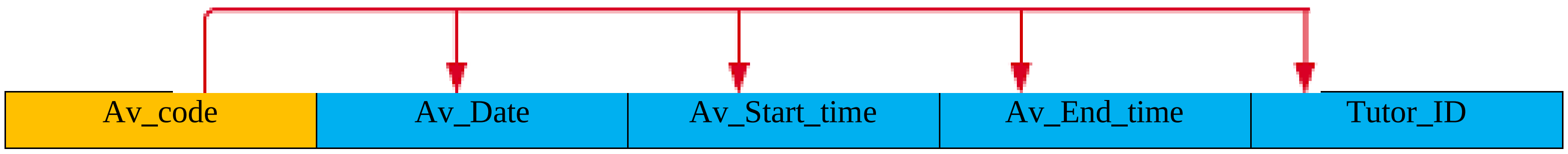


CSC 3326 Database Systems Fall 2021
Final Report

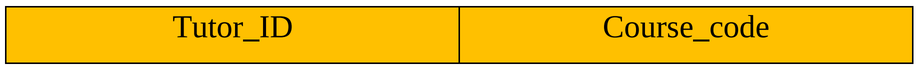
Tutor Table



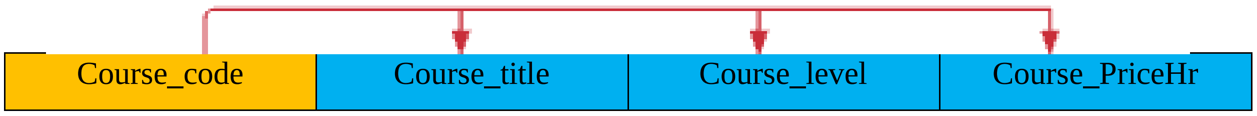
Availability Table



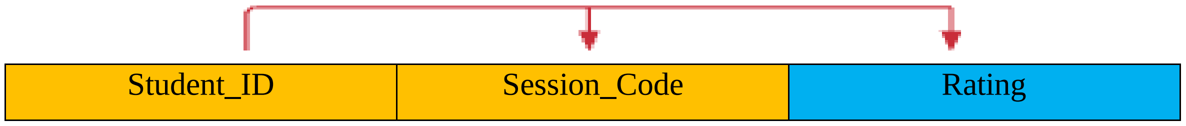
Offering Table



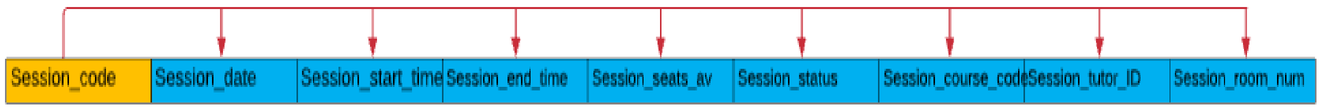
Course Table



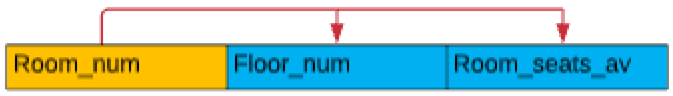
Booking Table



Session Table



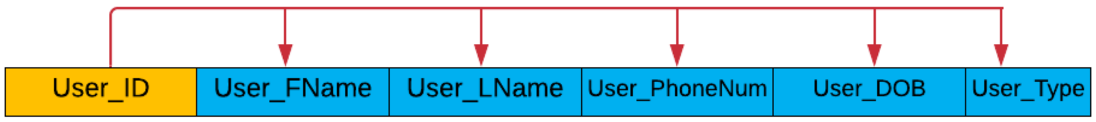
Location Table



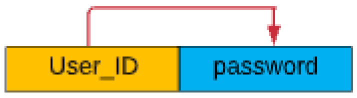
We could not find any partial dependency in all the tables, as there is no non-prime attribute that is functionally dependent on part of a candidate key. Therefore, all the tables are already in 2NF.

- **3NF Check**

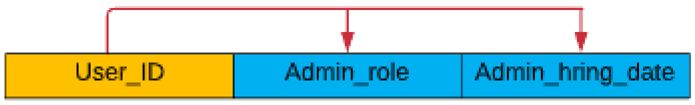
User Table



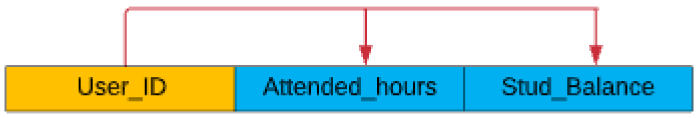
Account Table



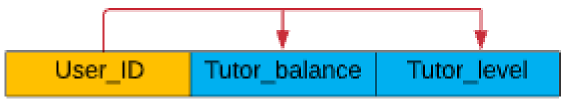
Admin Table



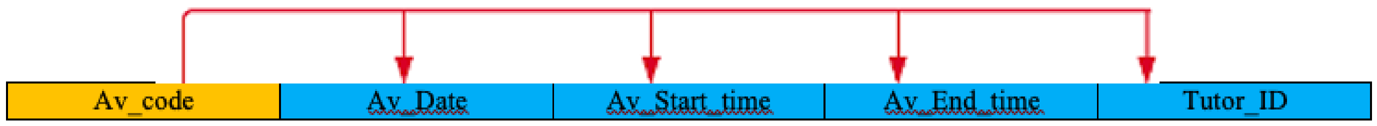
Student Table



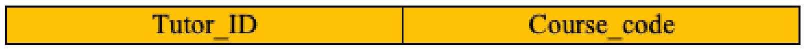
Tutor Table



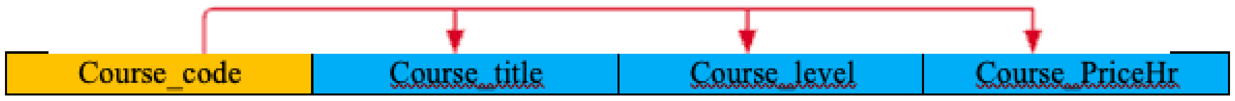
Availability Table



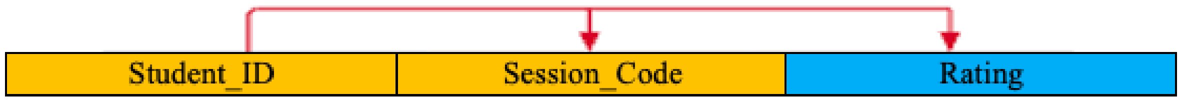
Offering Table



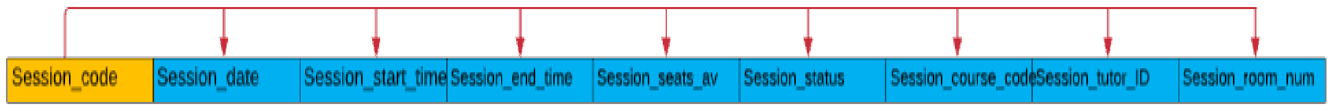
Course Table



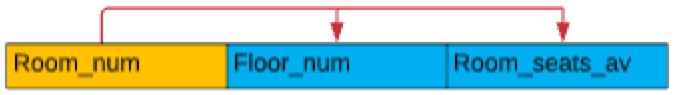
Booking Table



Session Table



Location Table



Again, we could not find any transitive dependency in all the tables, and since the satisfy 2NF conditions, then the tables are in 3NF.

- **BCNF Check**

All the tables are in 3NF, and no determinant in the table is a candidate key; therefore, all the tables are in BCNF.

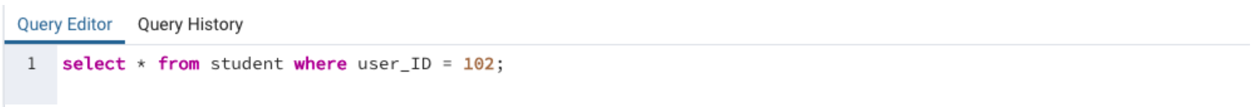
- **4NF Check**

The tables still satisfy the conditions of 4NF because there is no multivalued dependency.

e. Model Verification: Data Manipulation operations and queries

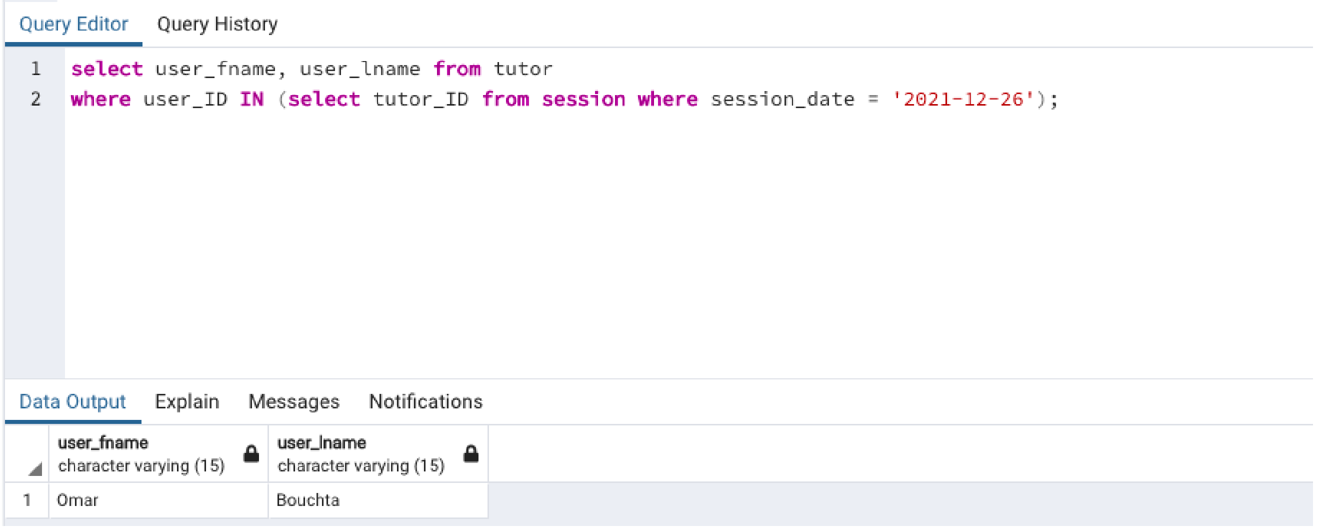
Please refer to **DBInit.sql** that is attached to the submission file!

- *Get student with ID 102:*



Query Editor Query History

```sql
select * from student where user_ID = 102;
```



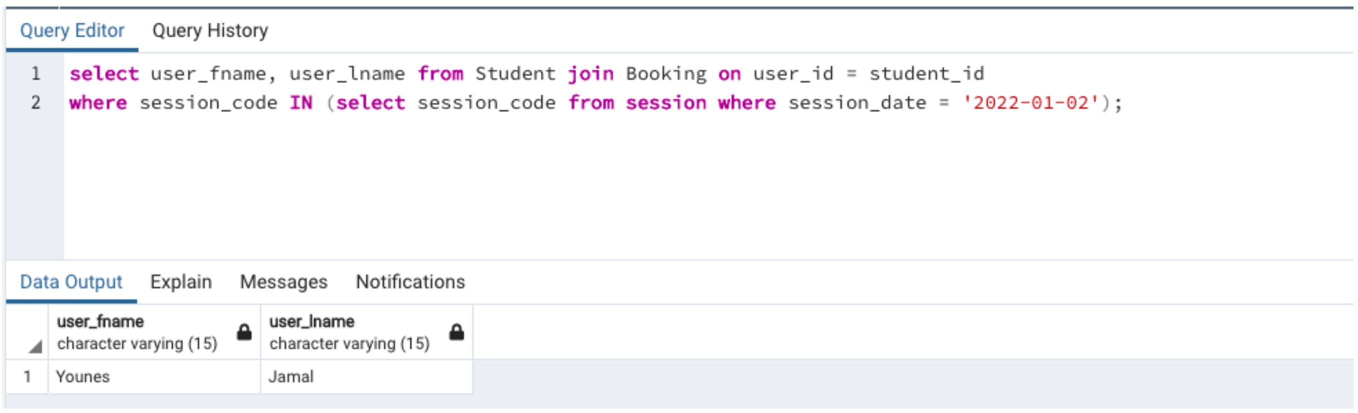
Query Editor Query History

```sql
select user_fname, user_lname from tutor
where user_ID IN (select tutor_ID from session where session_date = '2021-12-26');
```

Data Output Explain Messages Notifications

	user_fname character varying (15)	user_lname character varying (15)
1	Omar	Bouchta

- *Get names of all tutors that will be tutoring during the date mentioned in the following query:*
- *Get names of students who have a session on the date mentioned in the following query:*



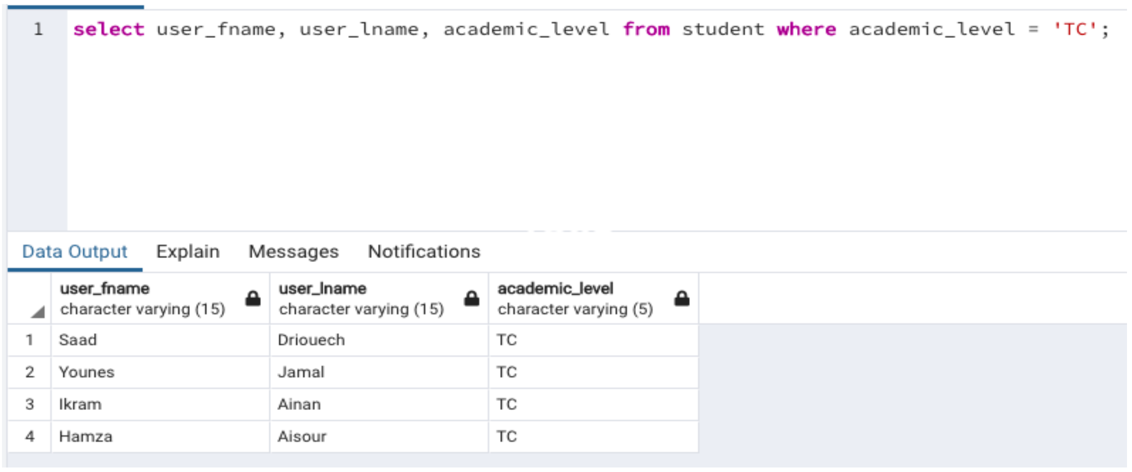
Query Editor Query History

```sql
select user_fname, user_lname from Student join Booking on user_id = student_id
where session_code IN (select session_code from session where session_date = '2022-01-02');
```

Data Output Explain Messages Notifications

	user_fname character varying (15)	user_lname character varying (15)
1	Younes	Jamal

- *Display students that are in academic level 'TC':*

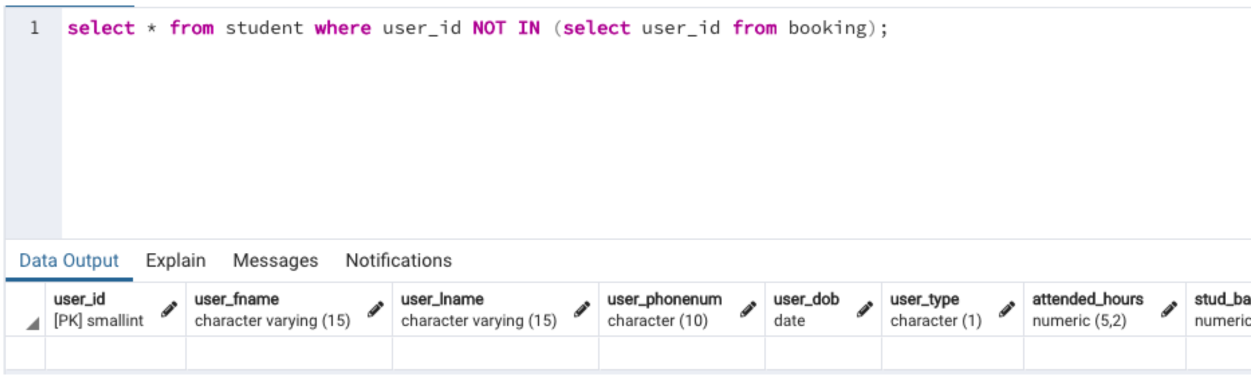


```
select user_fname, user_lname, academic_level from student where academic_level = 'TC';
```

Data Output | Explain | Messages | Notifications

	user_fname character varying (15)	user_lname character varying (15)	academic_level character varying (5)
1	Saad	Driouech	TC
2	Younes	Jamal	TC
3	Ikram	Ainan	TC
4	Hamza	Aisour	TC

- *Get names of students who did not book a session yet:*

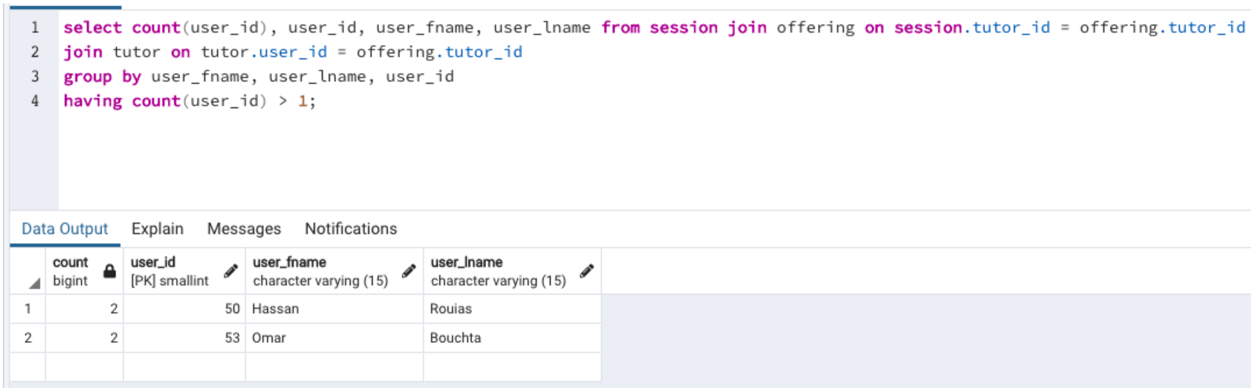


```
select * from student where user_id NOT IN (select user_id from booking);
```

Data Output | Explain | Messages | Notifications

	user_id [PK] smallint	user_fname character varying (15)	user_lname character varying (15)	user_phonenum character (10)	user_dob date	user_type character (1)	attended_hours numeric (5,2)	stud_ba numeric

- *Get student ID and names of students who have booked more than 1 session:*

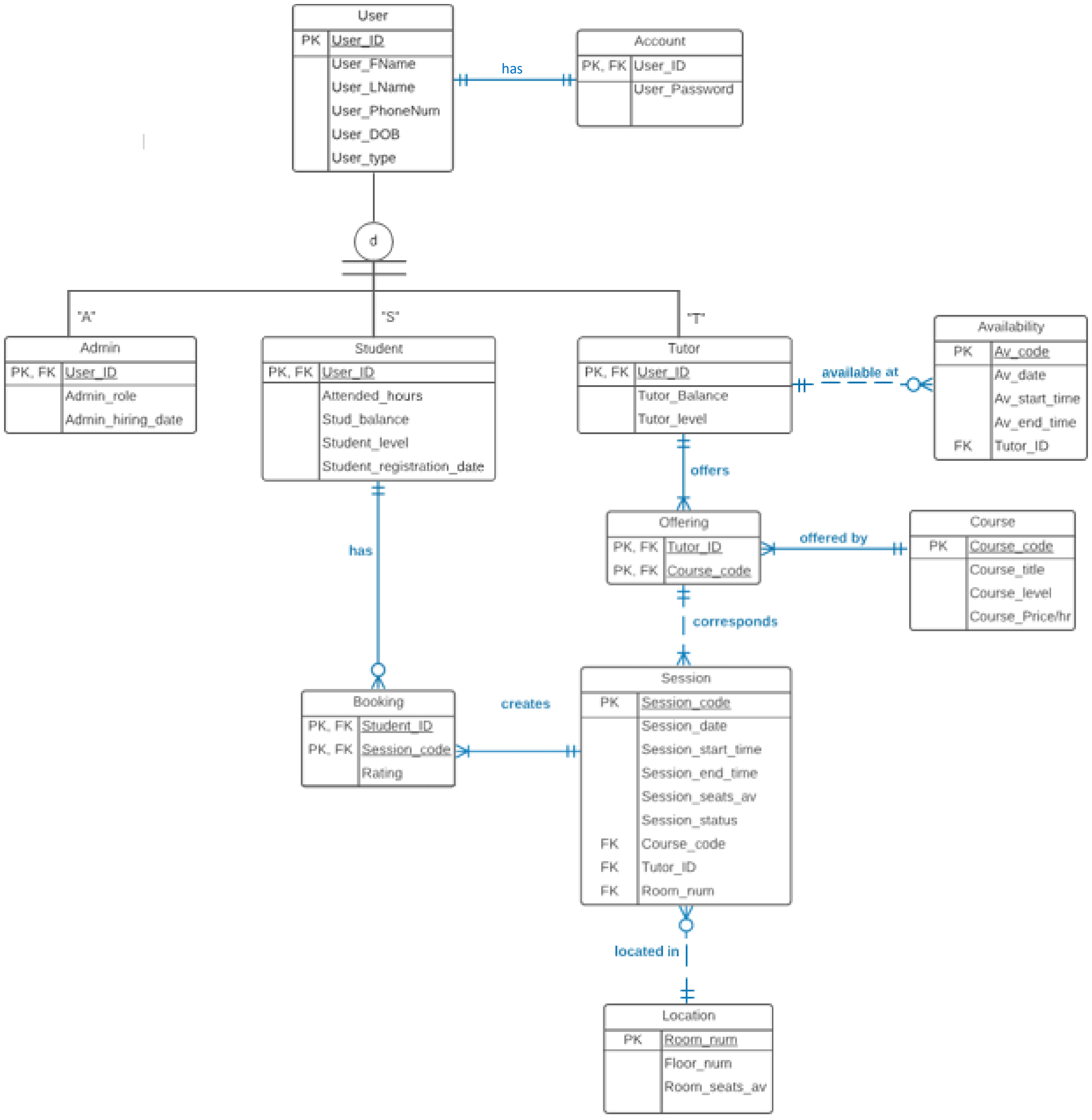


```
select count(user_id), user_id, user_fname, user_lname from session join offering on session.tutor_id = offering.tutor_id
join tutor on tutor.user_id = offering.tutor_id
group by user_fname, user_lname, user_id
having count(user_id) > 1;
```

Data Output | Explain | Messages | Notifications

	count bigint	user_id [PK] smallint	user_fname character varying (15)	user_lname character varying (15)
1	2	50	Hassan	Rouias
2	2	53	Omar	Bouchta

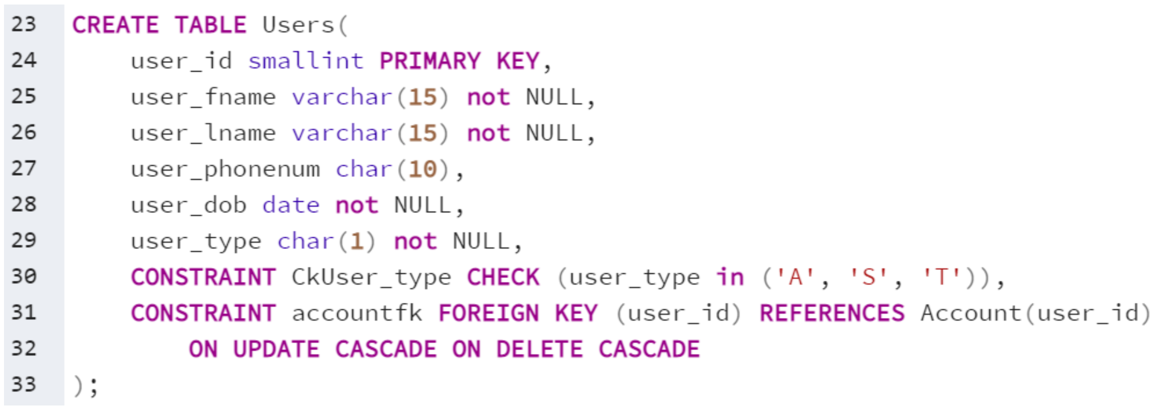
f. Final ER Model



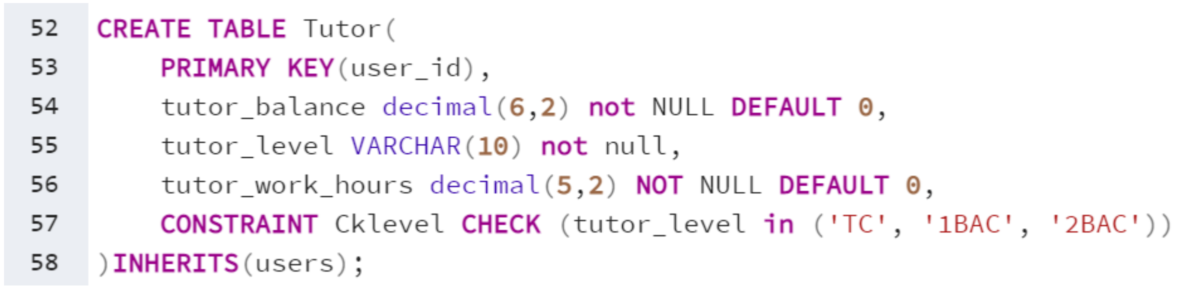
2. Logical Design

All the tables, attributes' data types, constraints, views, and indexes are included in the DBInit.sql file.

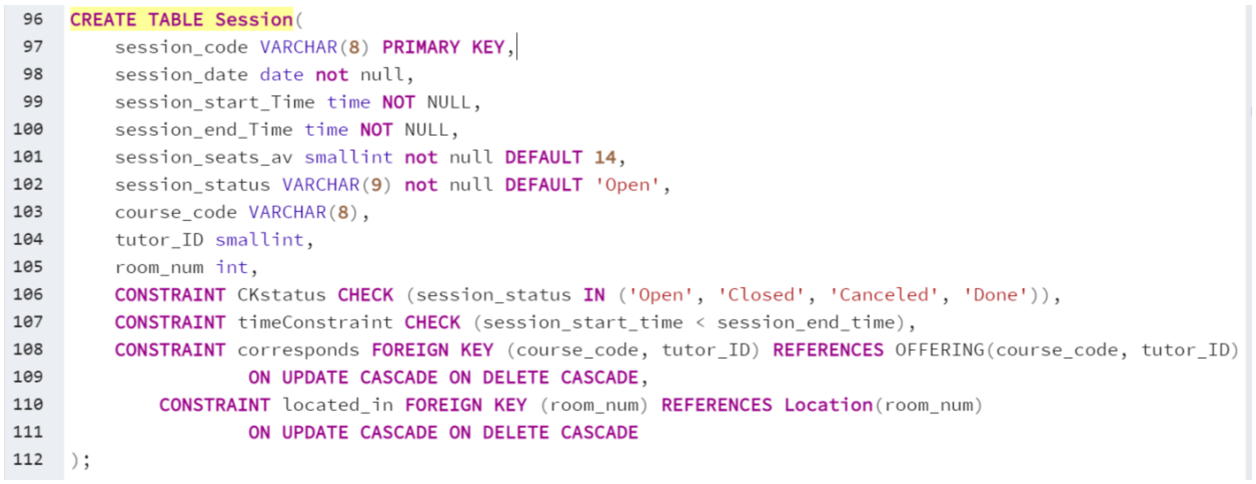
a. Tables



```sql
CREATE TABLE Users(
    user_id smallint PRIMARY KEY,
    user_fname varchar(15) not NULL,
    user_lname varchar(15) not NULL,
    user_phonenum char(10),
    user_dob date not NULL,
    user_type char(1) not NULL,
    CONSTRAINT CkUser_type CHECK (user_type in ('A', 'S', 'T')),
    CONSTRAINT accountfk FOREIGN KEY (user_id) REFERENCES Account(user_id)
        ON UPDATE CASCADE ON DELETE CASCADE
);
```

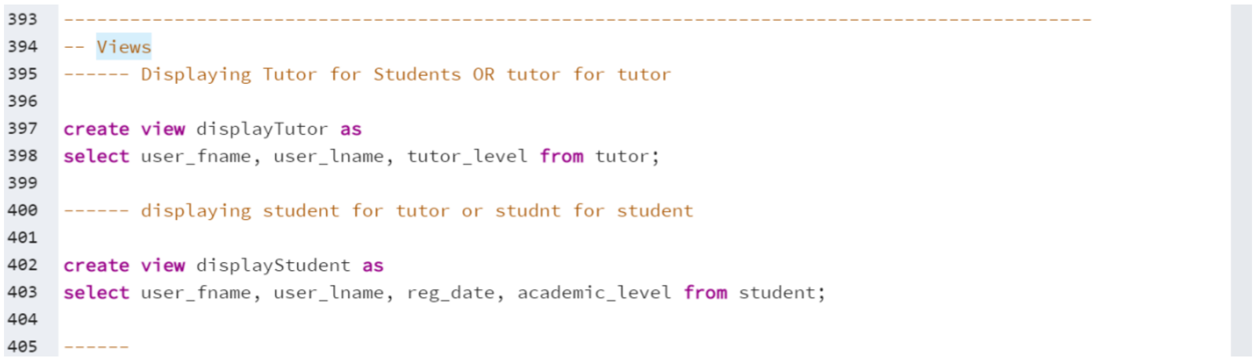


```sql
CREATE TABLE Tutor(
    PRIMARY KEY(user_id),
    tutor_balance decimal(6,2) not NULL DEFAULT 0,
    tutor_level VARCHAR(10) not null,
    tutor_work_hours decimal(5,2) NOT NULL DEFAULT 0,
    CONSTRAINT Cklevel CHECK (tutor_level in ('TC', '1BAC', '2BAC'))
)INHERITS(users);
```



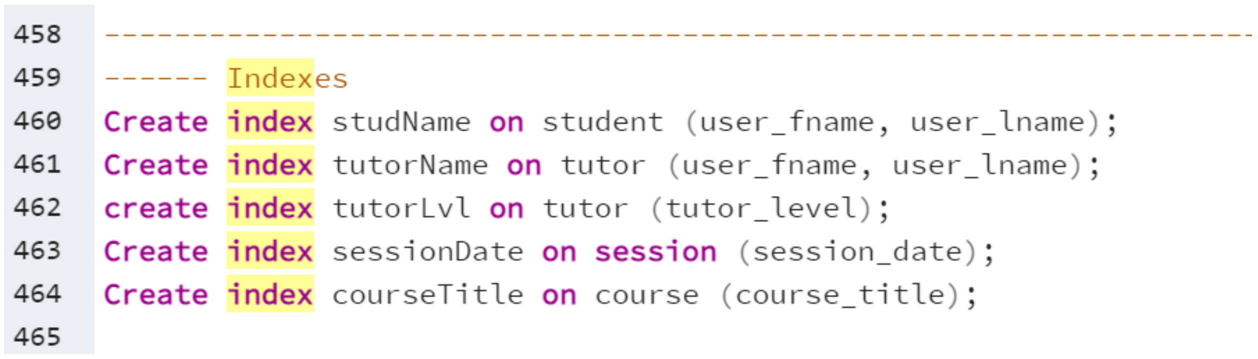
```sql
CREATE TABLE Session(
    session_code VARCHAR(8) PRIMARY KEY,
    session_date date not null,
    session_start_Time time NOT NULL,
    session_end_Time time NOT NULL,
    session_seats_av smallint not null DEFAULT 14,
    session_status VARCHAR(9) not null DEFAULT 'Open',
    course_code VARCHAR(8),
    tutor_ID smallint,
    room_num int,
    CONSTRAINT CKstatus CHECK (session_status IN ('Open', 'Closed', 'Canceled', 'Done')),
    CONSTRAINT timeConstraint CHECK (session_start_time < session_end_time),
    CONSTRAINT corresponds FOREIGN KEY (course_code, tutor_ID) REFERENCES OFFERING(course_code, tutor_ID)
                ON UPDATE CASCADE ON DELETE CASCADE,
        CONSTRAINT located_in FOREIGN KEY (room_num) REFERENCES Location(room_num)
                ON UPDATE CASCADE ON DELETE CASCADE
);
```

b. Views



```
-----------------------------------------------------------------------------------------------
-- Views
------ Displaying Tutor for Students OR tutor for tutor

create view displayTutor as
select user_fname, user_lname, tutor_level from tutor;

------ displaying student for tutor or studnt for student

create view displayStudent as
select user_fname, user_lname, reg_date, academic_level from student;

------
```

c. Indexes



```
------------------------------------------------------------------
------ Indexes
Create index studName on student (user_fname, user_lname);
Create index tutorName on tutor (user_fname, user_lname);
create index tutorLvl on tutor (tutor_level);
Create index sessionDate on session (session_date);
Create index courseTitle on course (course_title);

```

III. Implementation

1. DB Creation and Population

Tables creation is included in the ***DBInit.sql*** file.

2. Application Architecture

- **Jakarta Server Pages (JSP)**

Following the use of a web-based application, JSP plays the role of enabling the creation of dynamic and platform-independent methods. We used it for many purposes, such as allowing dynamic elements in HTML to be embedded in HTML pages instead of having multiple and separate text files. Also, JSP completes the duties of Java Servlet, which takes care of handling the business logic. Additionally, it has access to many Java APIs since it is built on top of Java Servlets.

- **jQuery**

jQuery is a Javascript library. We implemented it in order to fetch data from Java Servlet and display it in an HTML list. In fact, it facilitates the process of retrieving information about many elements of our pages, in addition to binding the events of those elements.

- **Java Servlet**

Java servlets are technologies that are used to build web applications, such that it is located at the server side to generate dynamic web-based applications. Besides what we mentioned in the previous sections in terms of how Java Servlet is linked to both jQuery and JSP, servlets can respond to incoming requests from the user at the server level.

- **JavaScript**

JavaScript is a programming language that primarily deals with web-based applications. Traditionally, it is used on the client side for web page behavior. JS is usually associated with HTML and CSS which go hand in hand to respectively define the content web pages and specify their layouts.

- **PostgreSQL**

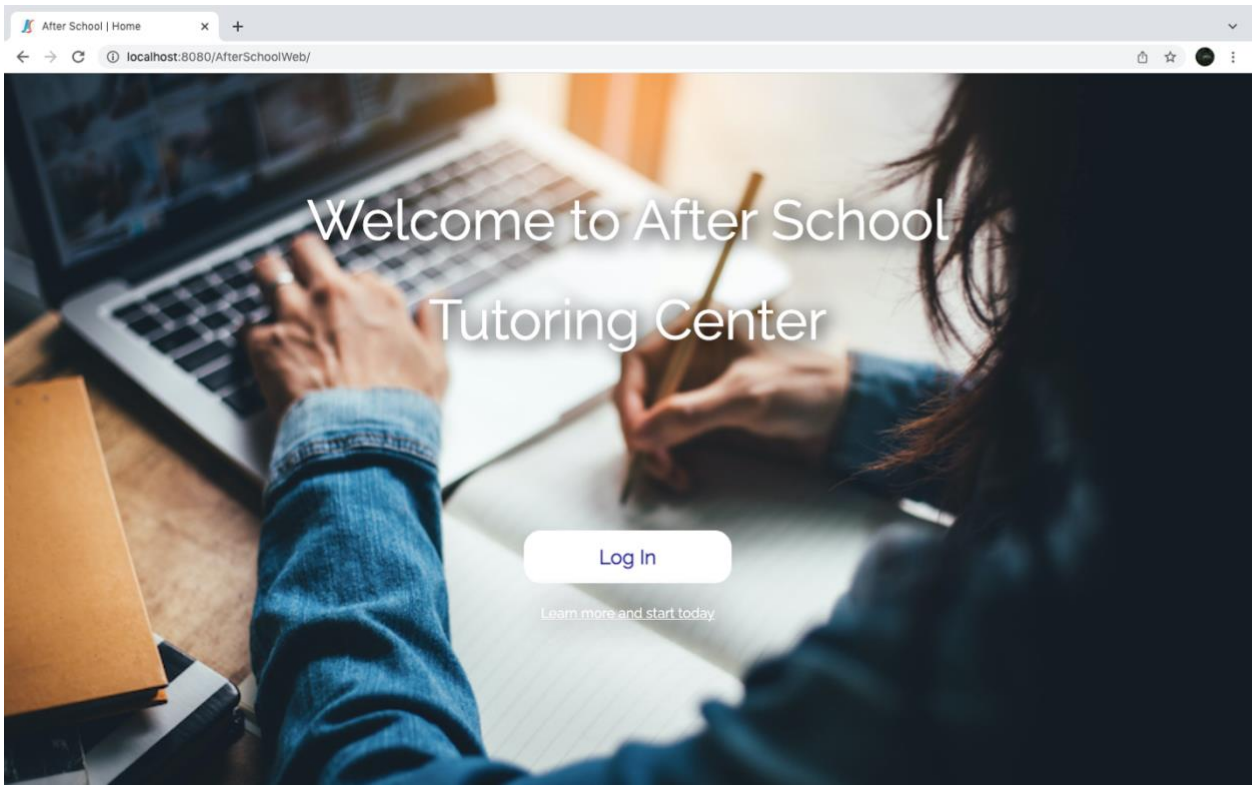
The outcomes of the Database System Course arise with this technology. PostgreSQL is a relational database management system (DBMS). It allows storage of large and sophisticated objects safely. It also helps to build complex applications in terms of storage point of view. It can be linked to the other technologies, such as Java, JavaScript, Python, CSS, HTML, etc. in order to fulfill that part of database management.

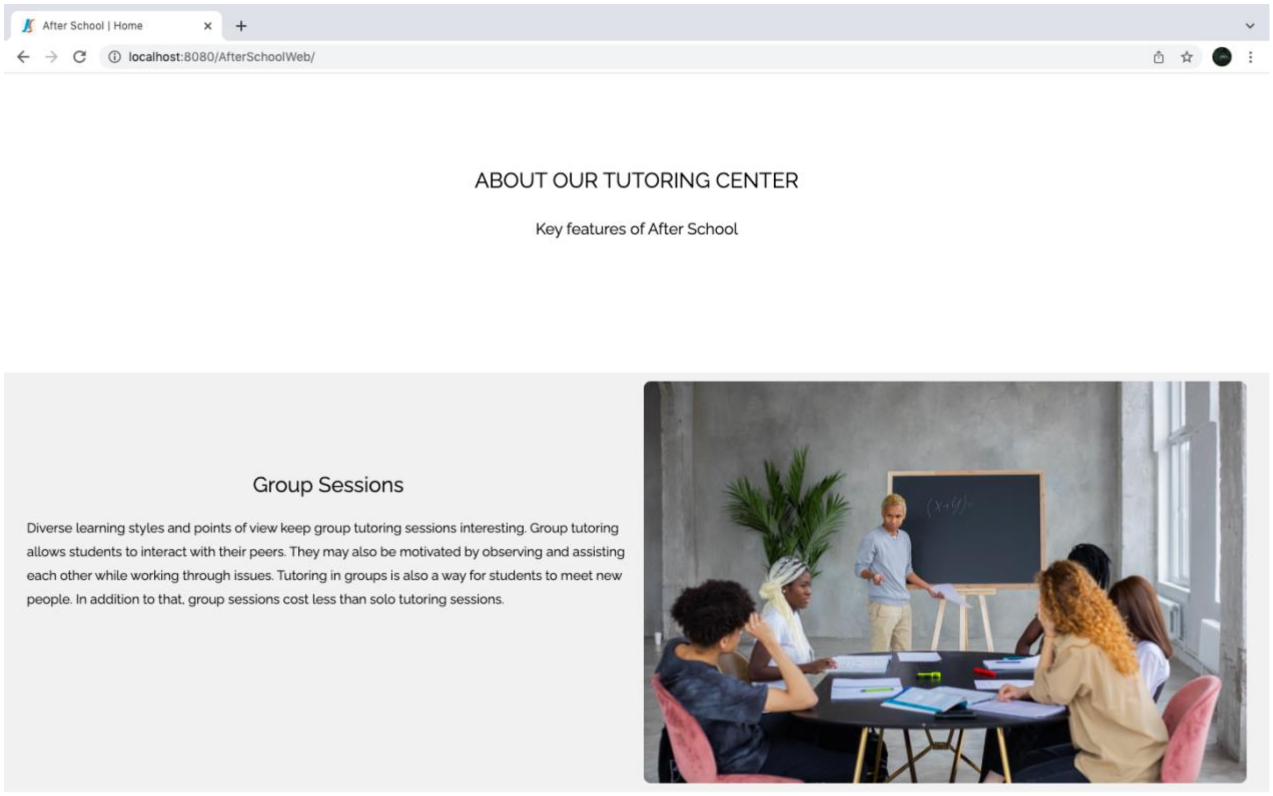
- **Bootstrap**

For design and aesthetic purposes, we have used Bootstrap. It is a CSS framework that serves responsive from-end web development. It also completes the tasks of other programming languages, such as styling HTML pages and enforcing JavaScript Plugins.

IV. Testing and fine-tuning & User Manual

Homepage - click on Log in or scroll down to see more detailed information about After School.



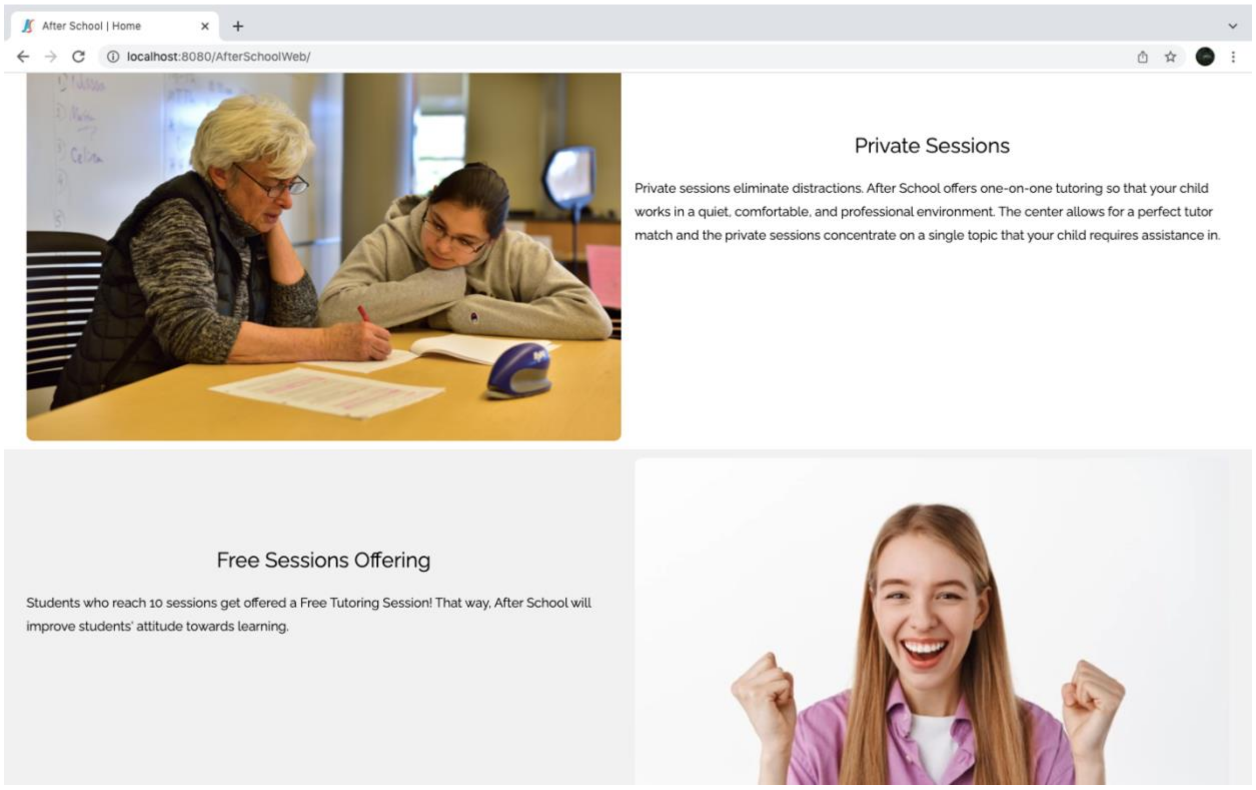


ABOUT OUR TUTORING CENTER

Key features of After School

Group Sessions

Diverse learning styles and points of view keep group tutoring sessions interesting. Group tutoring allows students to interact with their peers. They may also be motivated by observing and assisting each other while working through issues. Tutoring in groups is also a way for students to meet new people. In addition to that, group sessions cost less than solo tutoring sessions.

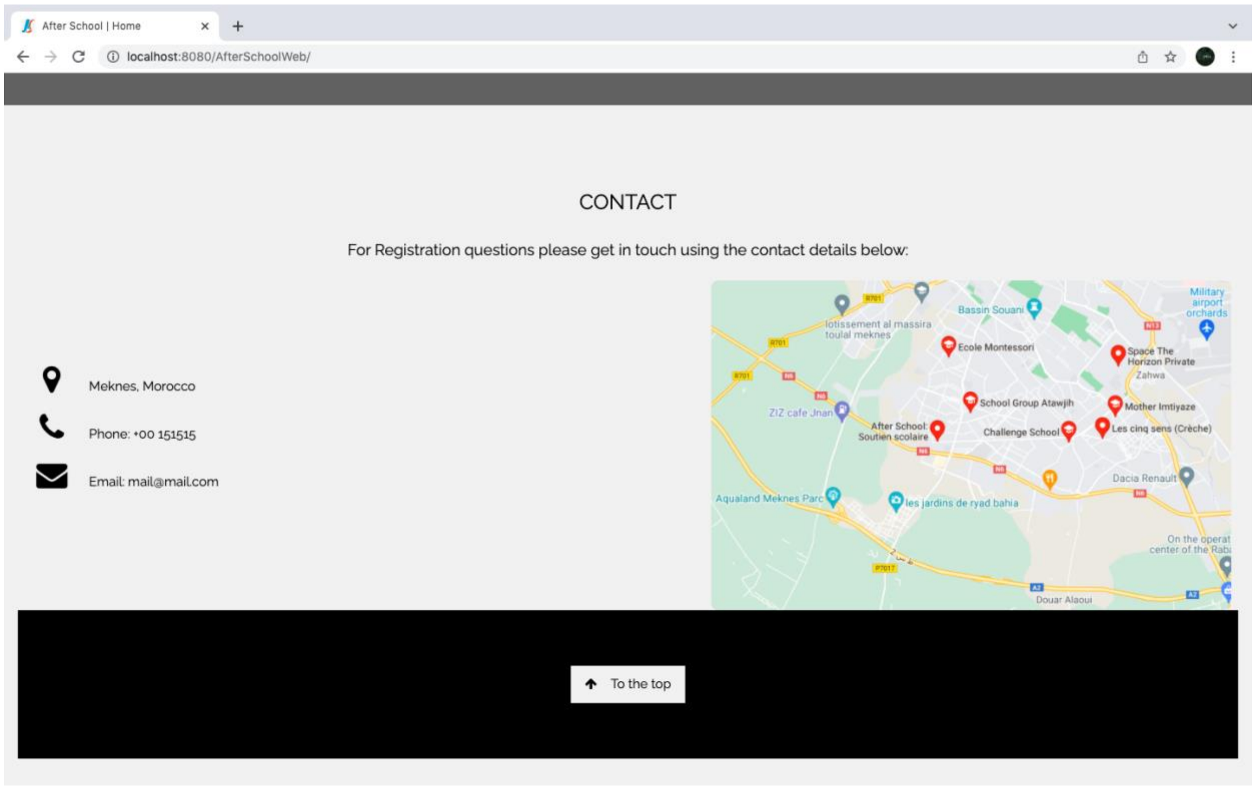


Private Sessions

Private sessions eliminate distractions. After School offers one-on-one tutoring so that your child works in a quiet, comfortable, and professional environment. The center allows for a perfect tutor match and the private sessions concentrate on a single topic that your child requires assistance in.

Free Sessions Offering

Students who reach 10 sessions get offered a Free Tutoring Session! That way, After School will improve students' attitude towards learning.



CONTACT

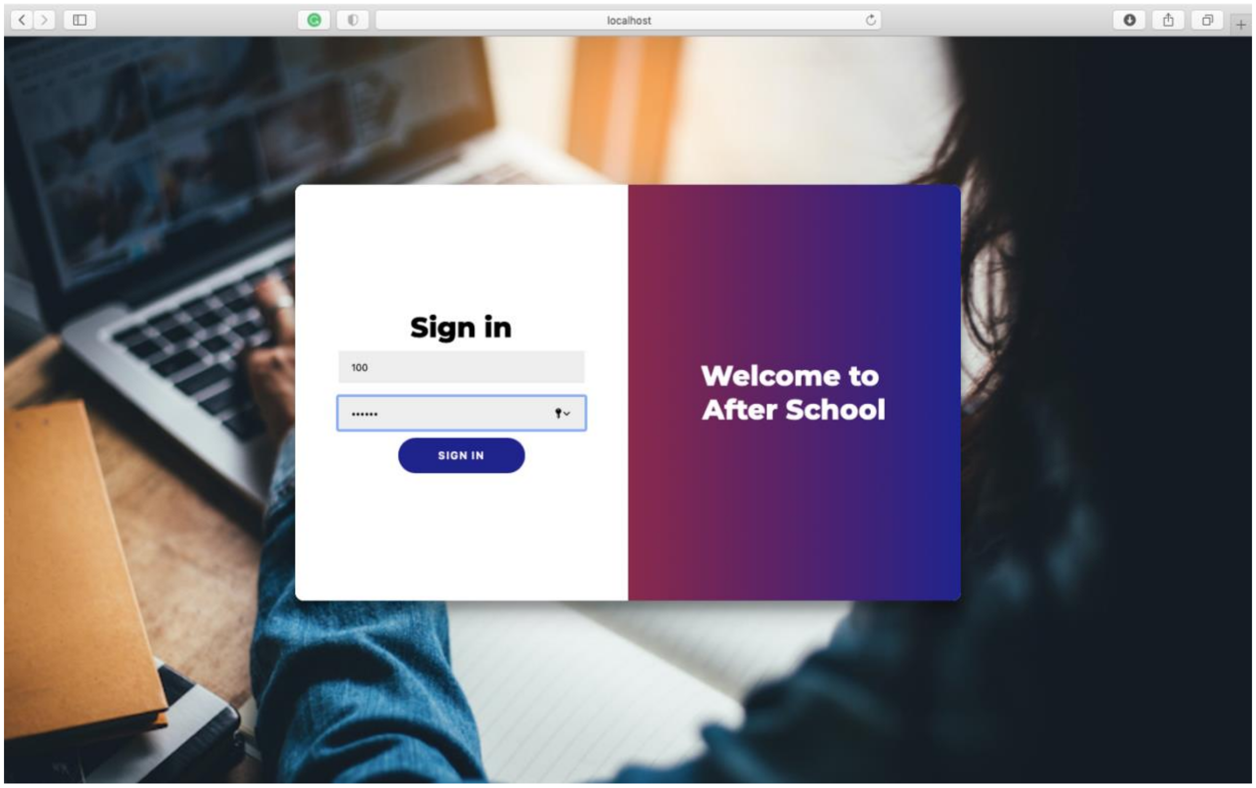
For Registration questions please get in touch using the contact details below:

Meknes, Morocco

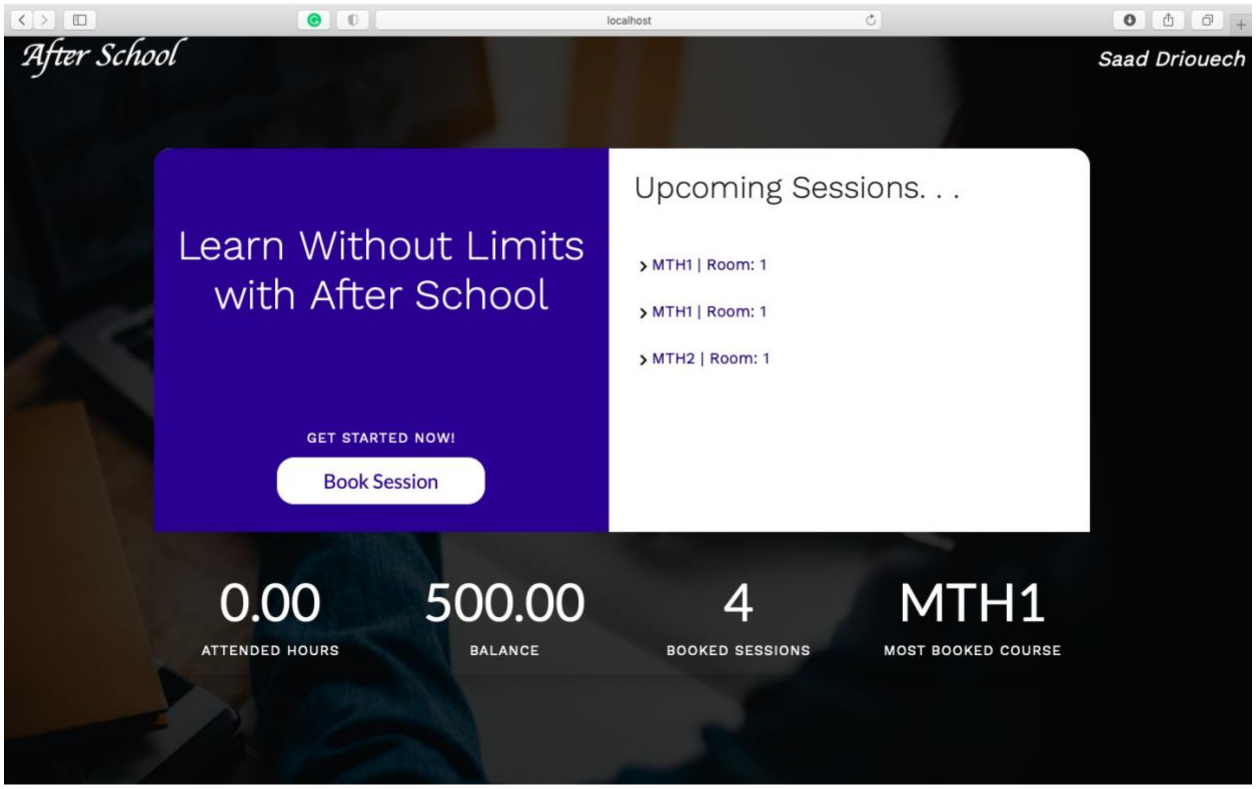
Phone: +00 151515

Email: mail@mail.com

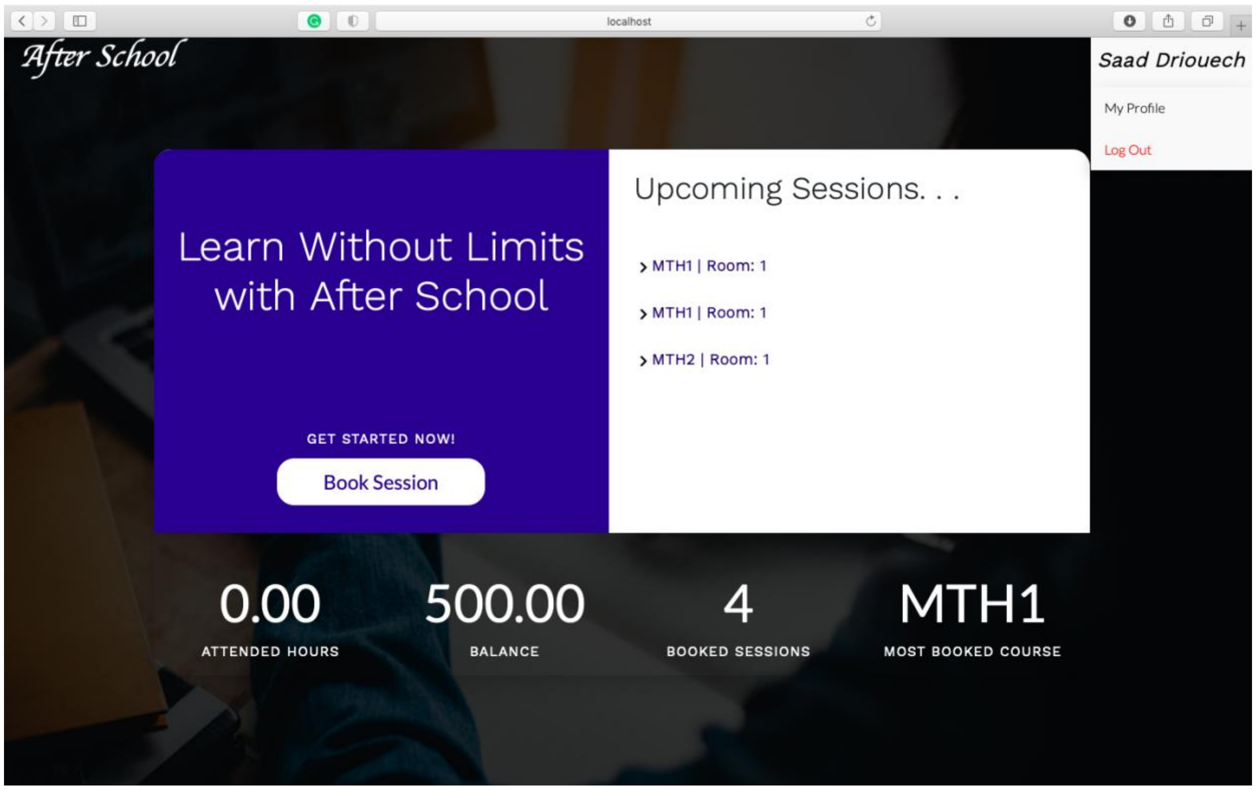
Sign In Interface - Input ID and Password!



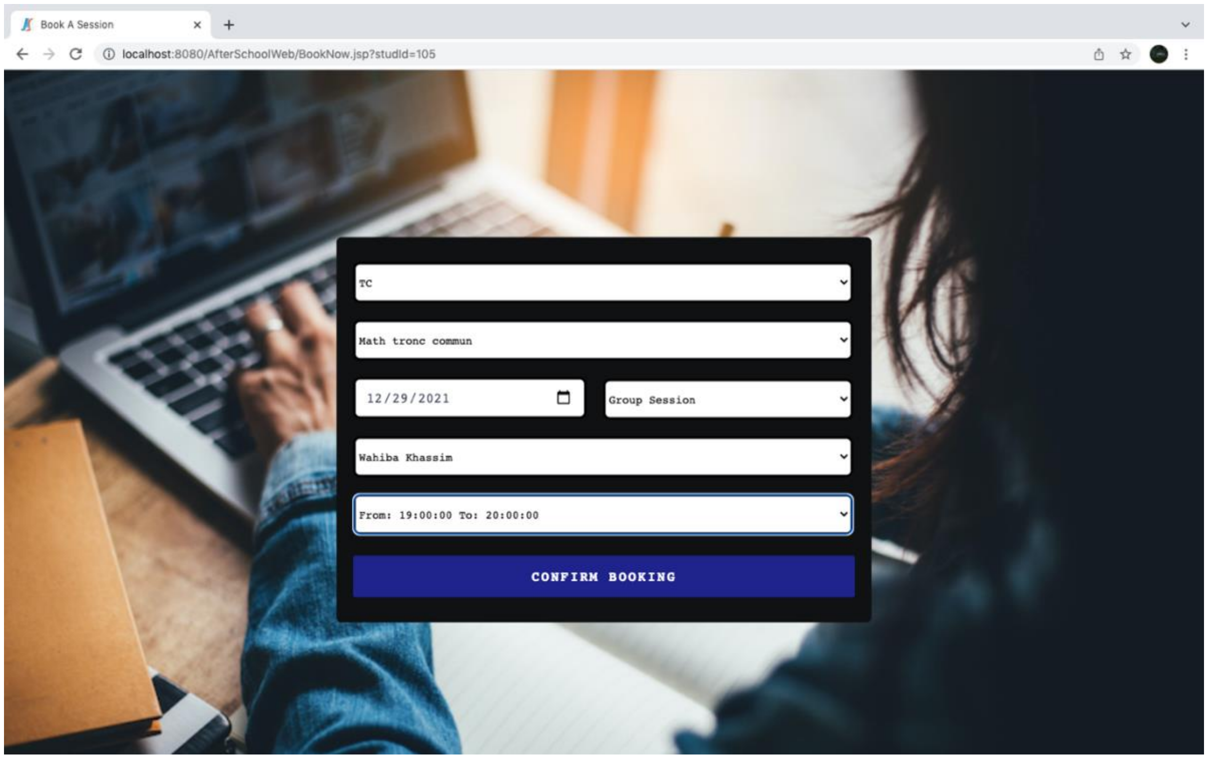
Student Interface

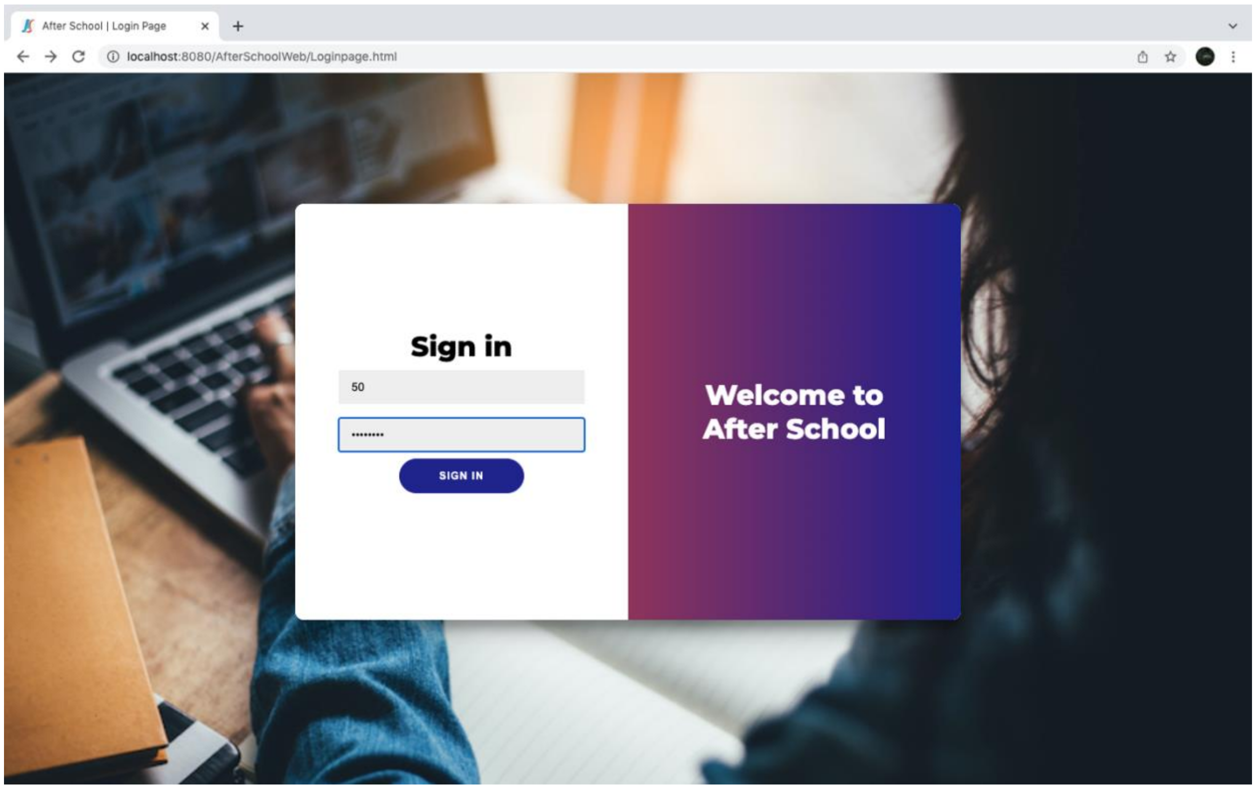


Drop Down Menu from the Right

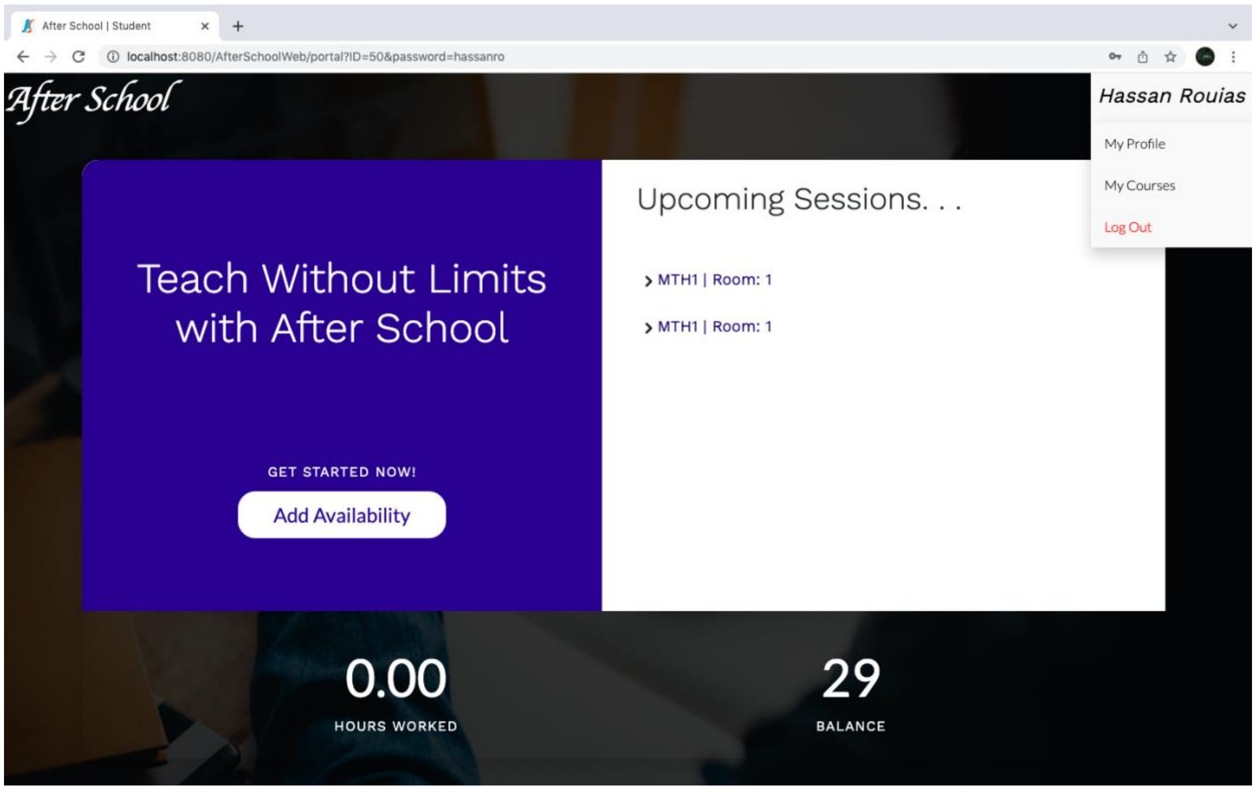


This form is obtained after pressing on the Book Session button in Student's Interface

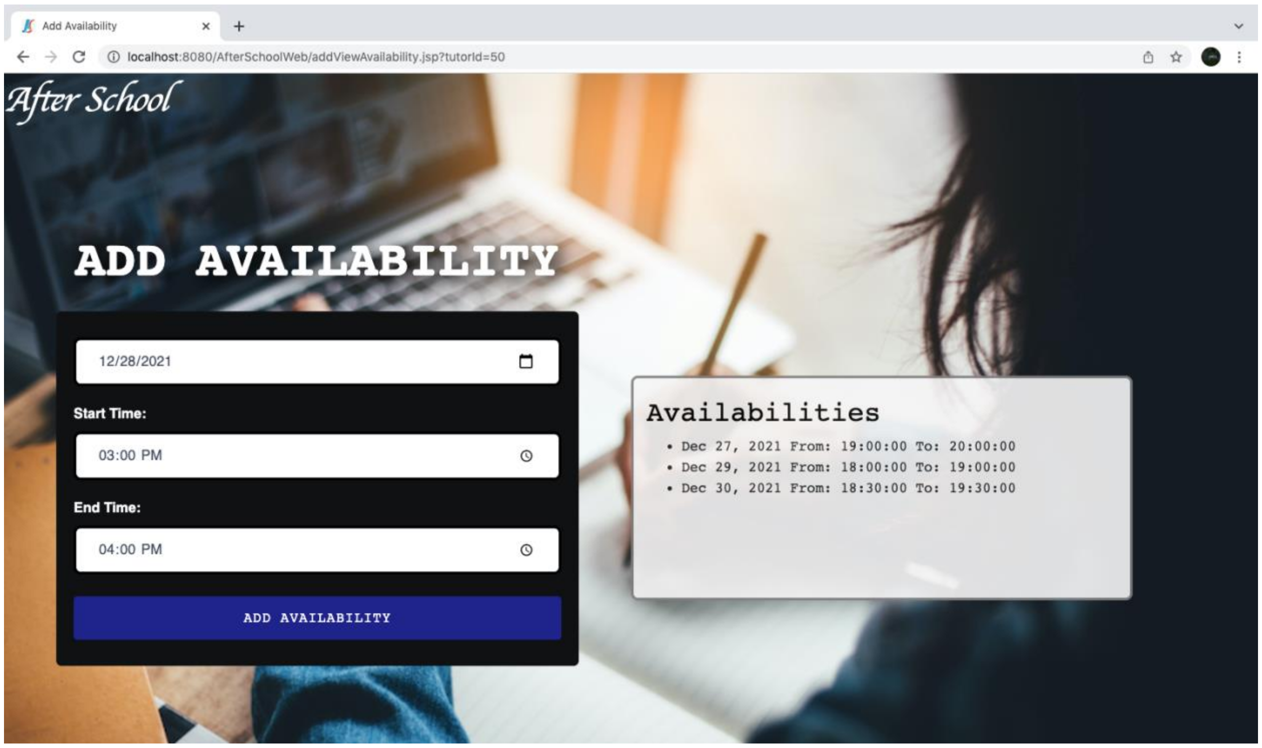



After School | Login Page
localhost:8080/AfterSchoolWeb/Loginpage.html
Sign in
50
••••••••
SIGN IN
Welcome to After School

Trying Professor Interface -



This form is obtained after pressing on the Add Availability button


Add Availability
localhost:8080/AfterSchoolWeb/addViewAvailability.jsp?tutorId=50
After School
ADD AVAILABILITY
ADD AVAILABILITY

12/28/2021

Start Time:

03:00 PM

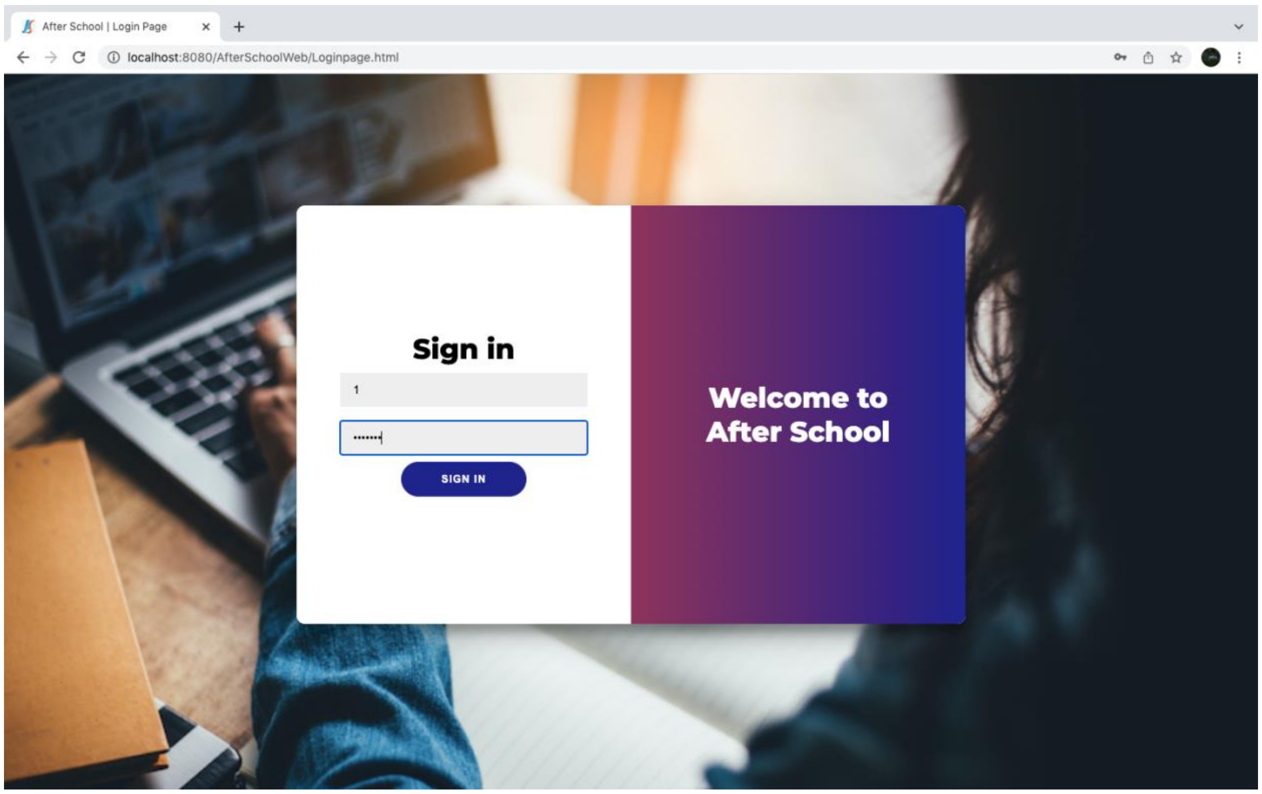
End Time:

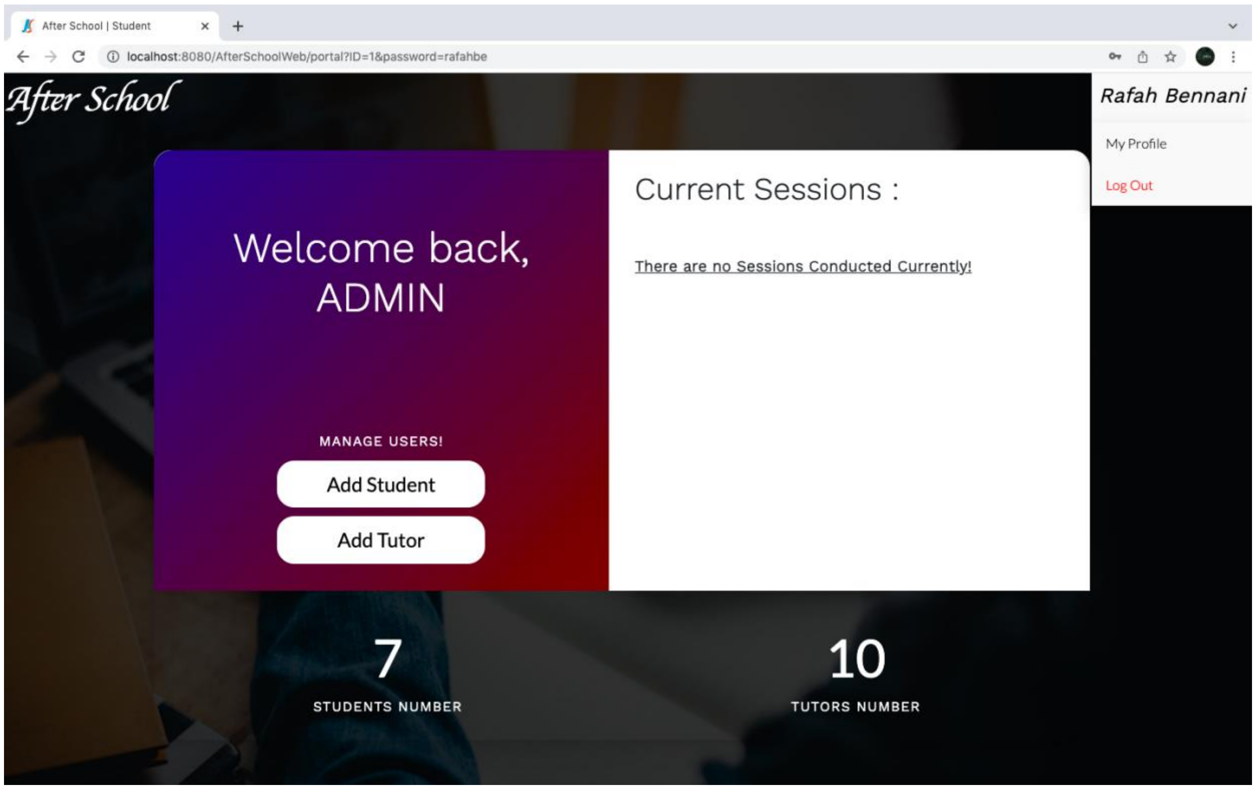
04:00 PM

Availabilities

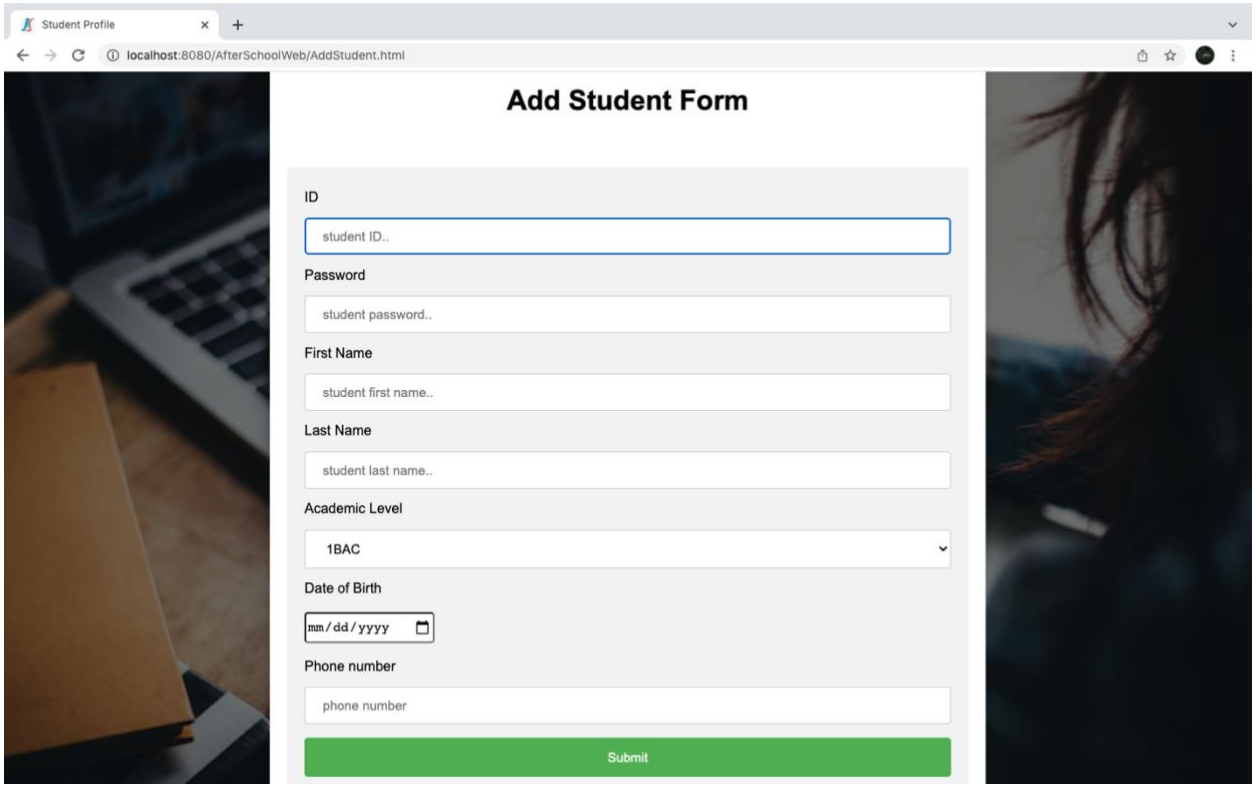
- Dec 27, 2021 From: 19:00:00 To: 20:00:00
- Dec 29, 2021 From: 18:00:00 To: 19:00:00
- Dec 30, 2021 From: 18:30:00 To: 19:30:00

Trying Admin Interface -


After School | Login Page
localhost:8080/AfterSchoolWeb/Loginpage.html
Sign in
1
•••••••
SIGN IN
Welcome to After School



We get this form after pressing on Add Student Button



V. Conclusion & Future Work

Throughout the implementation of this project, we cared a lot about developing a user-friendly interface. This took us too much time and given that the final exams period collided with the implementation phase of our project, we were not able to implement all the functionalities of the system. This is for sure something that we will be working on in the future since we are highly motivated to achieve a full functional system.

However, we have succeeded in achieving the big image of the outcomes of this project. The physical efforts of the manager are now reduced since the majority of the human interactions are now automated and stored safely.

Many features can be added to our product to improve its performance in terms of populating its functionalities. For instance, we could allow the students to pay their fees through credit cards instead of stopping by the director's office to pay in cash. Another example would be transferring money to the bank balances of tutors. Moreover, because of time constraints, we could not implement some minor functionalities, namely allowing a student to search for tutors and vice-versa, recording the attendance, displaying graphs for admins that show the visual data about the center, etc.

It is the first time all team members are implementing the technologies we mentioned earlier, which made this project a good and enjoyable experience. In addition to the knowledge we have acquired from other courses, working on this project has set the basis for learning database systems and applying them in a real-world project. It is very crucial to get exposed to such techniques during the academic career before engaging in the professional field.
In the future, it would be amazing to keep working on this project until it is fully functional with the use of some agile and DevOps technology enablers, such as Jira for planning and monitoring, Bitbucket as a version control repository, Jenkins for continuous integration, Selenium for testing, Kubernetes for orchestration, Docker for containerization and AWS Amazon Cloud for hosting.