Hotel Management System

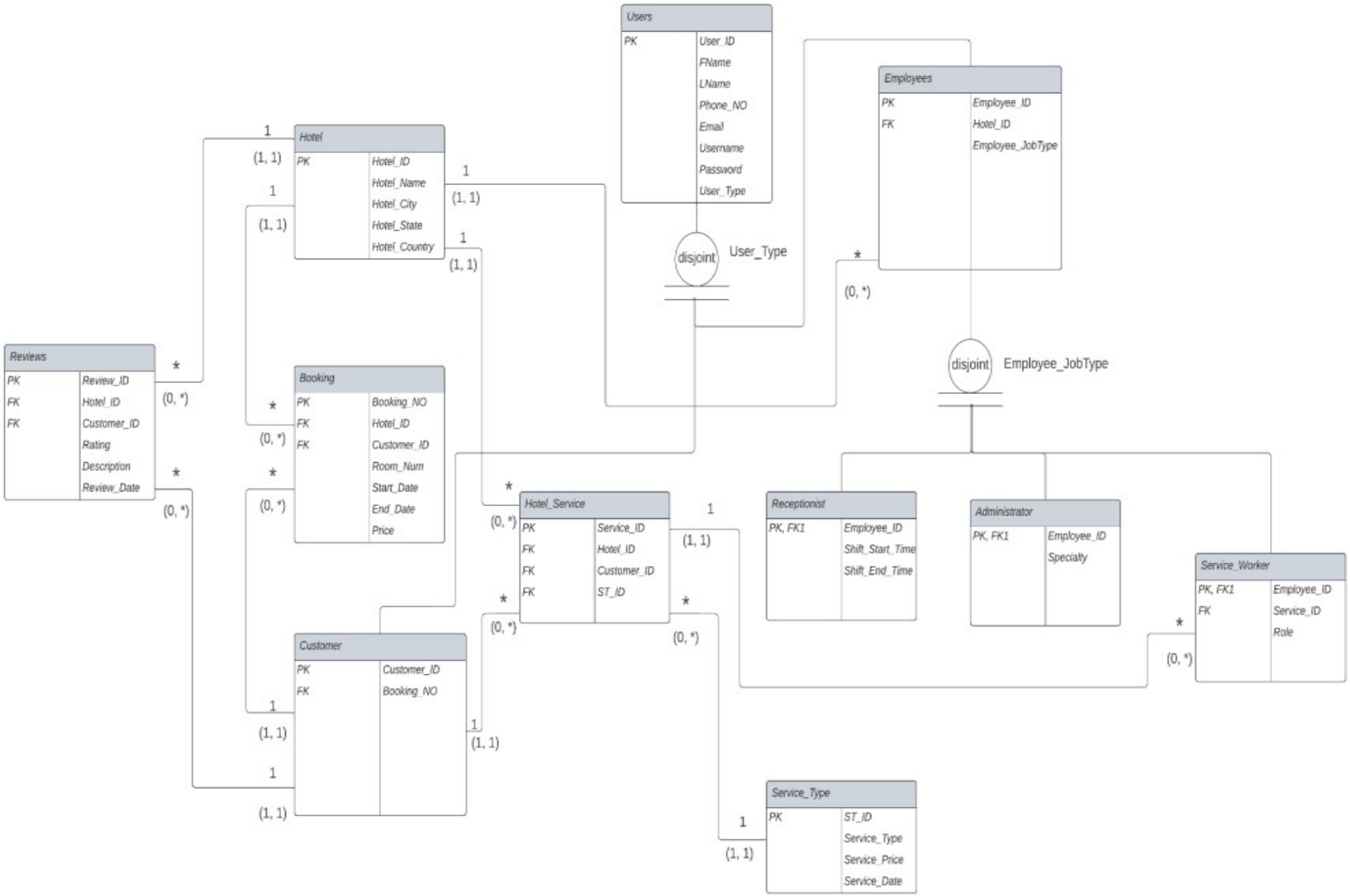
CSCE 310 Database Systems

Jacob Enerio
Yaroub Hussein
Uchenna Akahara
Krish Chhabra

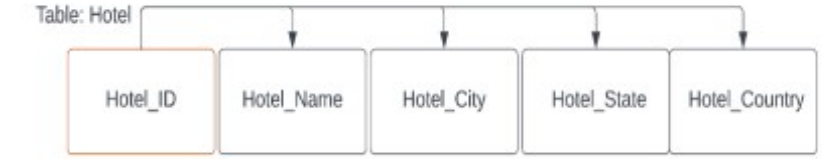
PROJECT DESIGN

Previous Project Design

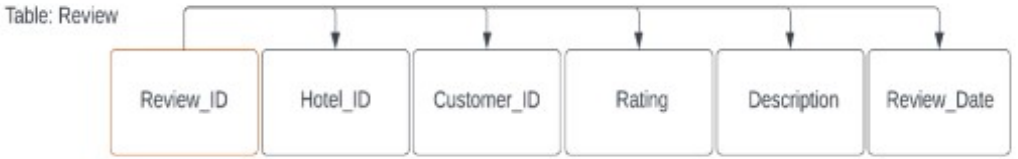
ERD:



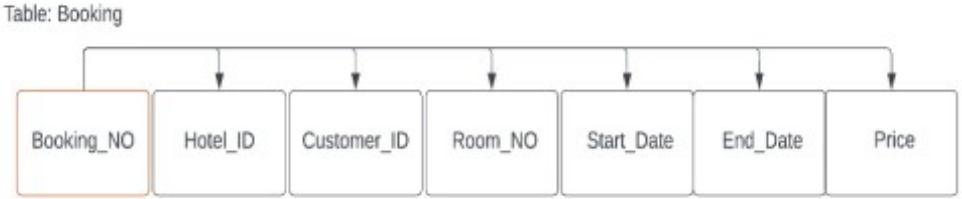
Dependency Diagram:



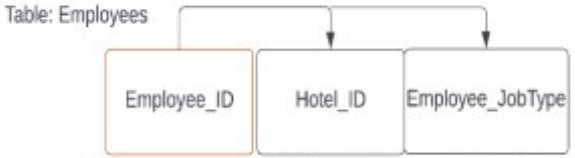
Hotel (Hotel_ID, Hotel_Name, Hotel_City, Hotel_State, Hotel_Country)



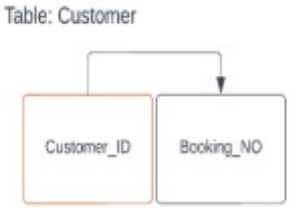
Review (Review_ID, Hotel_ID, Customer_ID, Rating, Description, Review_Date)



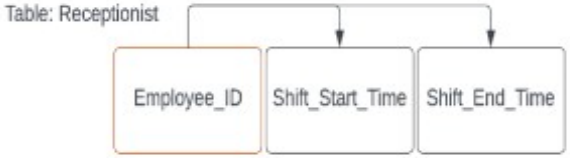
Booking (Booking_NO, Hotel_ID, Customer_ID, Room_NO, Start_Date, End_Date, Price)



Employee (Employee_ID, Hotel_ID, Employee_JobType)



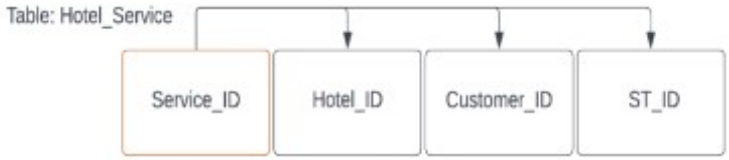
Customer (Customer_ID, Booking_NO)



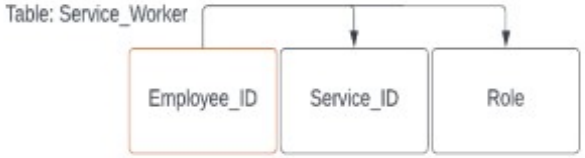
Receptionist (Employee_ID, Shift_Start_Time, Shift_End_Time)

Administrator (Employee_ID, Specialty)

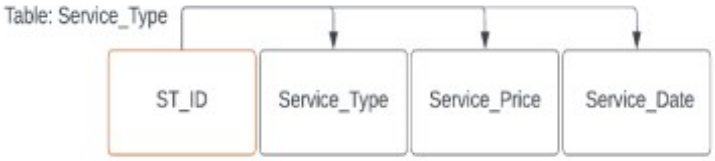
All Tables are 3NF



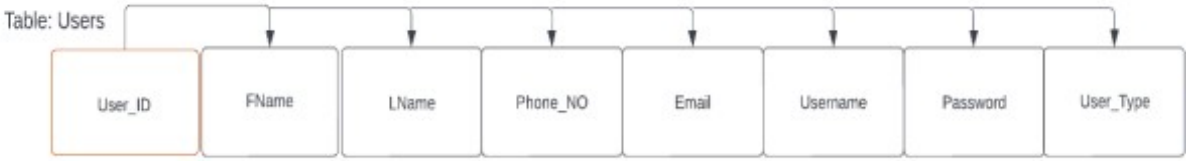
Hotel_Service (Service_ID, Hotel_ID, Customer_ID, ST_ID)



Service_Worker (Employee_ID, Service_ID, Role)



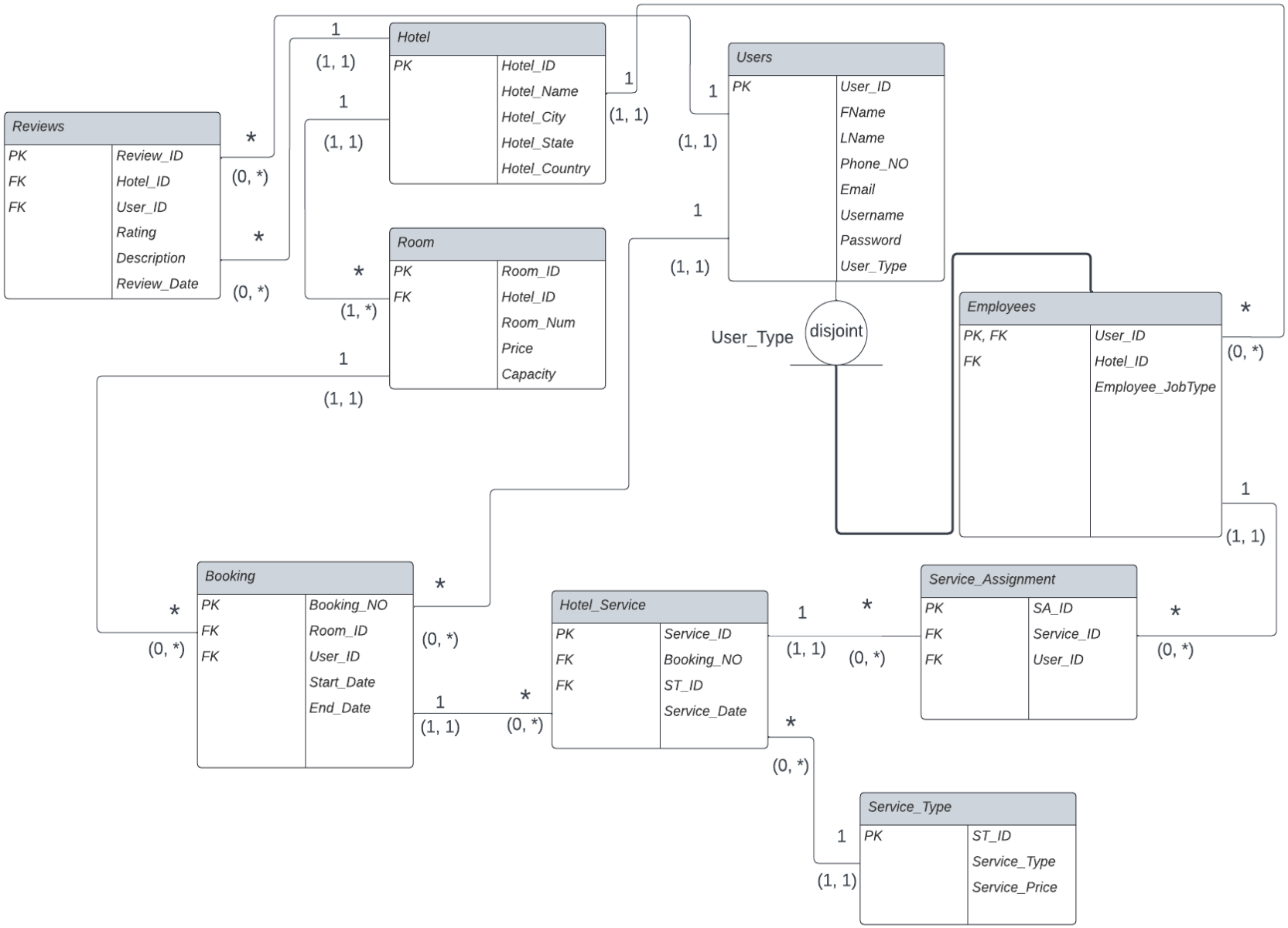
Service_Type (ST_ID, Service_Type, Srervice_Price, Service_Date)



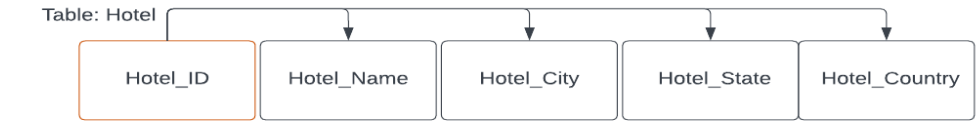
User (User_ID, FName, LName, Phone_NO, Email, Username, Password, User_Type)

Current Project Design:

ERD:

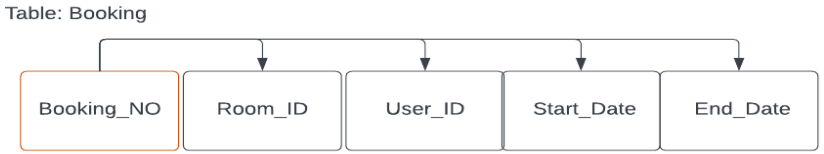


Dependency Diagram:

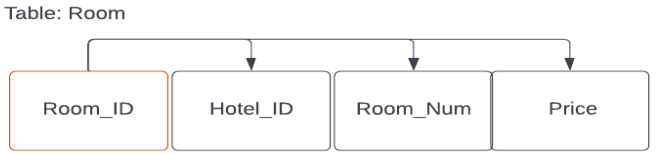


Hotel (Hotel_ID, Hotel_Name, Hotel_City, Hotel_State, Hotel_Country)

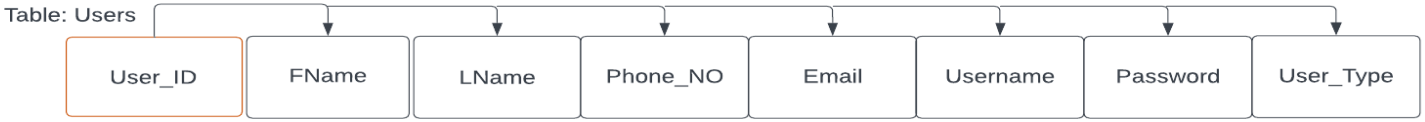
All Tables are 3NF



Booking (Booking_NO, Room_ID, Customer_ID, Start_Date, End_Date)



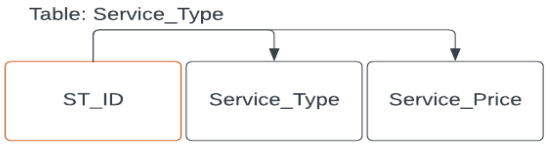
Room (Room_ID, Hotel_ID, Room_Num, Price)



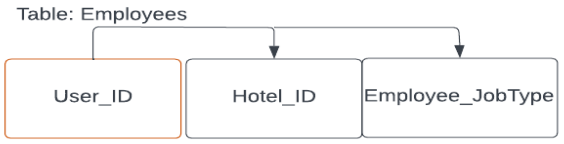
User (User_ID, FName, LName, Phone_NO, Email, Username, Password, User_Type)

Hotel_Service (Service_ID, Booking_NO, ST_ID, Service_Date)

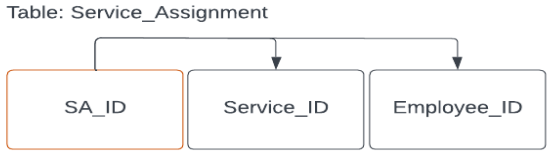
Review (Review_ID, Hotel_ID, Customer_ID, Rating, Description, Review_Date)



Service_Type (ST_ID, Service_Type, Service_Price)



Employee (Employee_ID, Hotel_ID, Employee_JobType)



Service_Assignment (SA_ID, Service_ID, Employee_ID)

Explanation for Modifications

We changed our project because some of the original entities and relationships didn't make sense in the original project. For example, there were no room entities in the original ERD, which meant each Hotel had to have a fixed amount of rooms, which is unreasonable. Therefore, a Room entity was added, so each Hotel has their own set of rooms. There is a 1 to many relationship between the hotels and the rooms, and a one to many relationship between the rooms and the bookings. This is because a hotel has many rooms and a room can have many bookings.

Next, We got rid of the employee subtype tables because we didn't really do anything with the information in the receptionist, administrator, and service workers tables. So, we just removed them and we can just access the employee subtype via the Employee_JobType attribute in the Employees table. Additionally, we removed the customers table because there was no point in keeping the Booking_NO as a foreign key when the Bookings automatically holds a foreign key to the User_ID of the customer and there is a 1 to many relationship between the customers and the bookings. We also changed the completeness constraint for the Users supertype to be partial completeness instead of total completeness. This is because there is no longer a Customers table.

Finally, we transferred the service_date from Service_Type to Hotel_Service. This is because the Service_Type is just the type of the service while the Hotel_Service is actually each service instance. Therefore, the date should belong to each service instance, not the type. Also, the Booking entity now has a 1 to many relationship with the Hotel_Service entity to simplify the relationships (originally there was a relationship involving both the Hotel_Service, Hotel, and Customers entities. Even more so, there is now a bridge entity between Employees and Hotel_Service called Service_Assignment. This was added because it may be useful to keep track of what employees were assigned to a service, which was previously not possible before Service_Assignment was added.

PROJECT FILES & RESPONSIBILITIES

Jacob Enerio

Files:

test.sql

/website_files

database_intializer.php
table_dropper.php
table_intializer.php *(shared)*
table_maker.php *(shared)*

/website_files/bookings

bookings.php
bookings_admin.php
bookings_receptionist.php

/website_files/res

connect.php
data_filter.php
data_table.php
head.php
head_no_nav.php
main.css
query_handler.php
query_overrides.php
table_editor.js
table_editor.php
table_generator.php
toc.js

/website_files/review

create_review_admin.php
delete_review_admin.php
edit_review_admin.php (shared)
edit_review.php (shared)
hotel_reviews_admin.php (shared)

/website_files/nav

navbar.php *(shared)*

Functionalities:

Jacob is responsible for the functionality set 2, which deals with scheduling. The Customer, Receptionist, and Administrator users are integrated into this functionality. The video shows the Customer and Receptionist users utilizing functionality set 2 (admin was scrapped for time, but the code for it is still there). This functionality set includes:

- Create appointments (Bookings for rooms)
    - Customers can create bookings only for themselves.
    - Receptionists and Administrators can create bookings for any user.
- View appointments (Bookings for rooms)
    - Customers see all bookings but not who booked them.
    - Receptionists & administrators can see all users and their associated bookings.
- Update appointments (Bookings for rooms)
    - Customers can update their own bookings.
    - Receptionists & administrators can update all bookings.
- Delete appointments (Bookings for rooms)
    - Customers can delete their own bookings.
    - Receptionists & administrators have the ability to delete any bookings.

Used `'CREATE INDEX IF NOT EXISTS idx_room_room_num ON Room (Room_Num)'`

Essentially creates an index on the room number for the room entity. This was useful if a user wanted to select a room, but its room number existed in multiple hotels.

Used `'CREATE VIEW IF NOT EXISTS Hotel_View AS SELECT Hotel_Name, Hotel_City, Hotel_State, Hotel_Country FROM Hotel'`

Essentially creates a view of the hotel without the hotel id. This is useful since customers don't need to know the hotel id, so this is what is displayed on the customer side.

Yaroub Hussein

Files:

/website_files

table_maker.php *(shared)*

/website_files/login

admin_accounts.php
authenticate.php
delete_account.php
home.css
home.php
login.css
login.php
logout.php
profile.php
register.css
registerC.php
registerCustomer.php
registerE.php
registerEmployee.php
style.css
test.css
update_account_cust.php
update_account_emp.php
update_account_handler_c.php
update_account_handler_e.php

/website_files/nav

navbar.php *(shared)*

Functionalities:

Yaroub is responsible for functionality set **1**. This functionality set includes:

- Registration methods for both user types of a customer or employee (Admin included).
- The ability to login into the user's account and access proper functionality.
- The ability to update relevant profiles.
    - Admin has access to update all registered profiles.
- The ability to delete the account and all related appropriate data.
    - Admin has the ability to delete any user account and all related appropriate data.

Used `'CREATE INDEX IF NOT EXISTS idx_users_username ON Users (Username)'`

Essentially creates an index on the username for a user entity. This was useful for updating a user profile since it makes updating faster as there is now an index.

Used `'CREATE VIEW IF NOT EXISTS User_View AS SELECT FName, LName, Phone_No, Email, Username, Password, User_Type FROM Users'`

Essentially creates an view of the users table that doesn't contain the user id. This is useful for updating the profile since we can use username instead of user_id as the unique identifier of the user entity. This means, user_id does not need to be used when displaying the data to the user.

Uchenna Akahara

Files:

/website_files

table_maker.php *(shared)*
table_initializer.php *(shared)*

/website_files/review

create_review.php
delete_review.php
edit_review.php (shared)
hotel_reviews.php
hotel_reviews_admin.php (shared)
style.css

/website_files/nav

navbar.php *(shared)*

Functionalities:

Uchenna was originally assigned responsibility for the functionality set **3**. He ended up being responsible for the functionality set **4** due to a confusion between him and Krish on what functionalities they were originally assigned. The functionality set Uchenna completed for this project includes:

- Create Review : The user can create a review with a rating, description, and date. Upon creating a review, the database is updated and the "**Reviews**" table is populated with the new data.
  - User_type: Customer
  - User_type: Employee
- Delete Review : Users, except for Admins are able to delete reviews they have made. Provided that the password associated with the user account is provided for verification. Admins have the ability to delete customer and employee reviews without the need for a password.
  - User_type: Customer
  - User_type: Employee
  - User_type: Administrator
- Edit Review : Users are able to edit the rating and description of their review. Provided that the password associated with the user account is provided for verification
  - User_type: Customer
  - User_type: Employee
- View Review : Users are able to view their reviews
  - User_type: Customer
  - User_type: Administrator

Used: `"CREATE VIEW Review_View AS SELECT Reviews.Review_ID, Users.User_ID, Users.username, Hotel.Hotel_ID, Hotel.Hotel_Name, Reviews.Rating, Reviews.Description, Reviews.Review_Date FROM Reviews INNER JOIN Users on`

```
Users.User_ID = Reviews.User_ID INNER JOIN Hotel on Hotel.Hotel_ID =
Reviews.Hotel_ID"
```

Essentially creates a view that features attributes from Reviews, Hotels, and Users. This allows for the reviews on the admin side to show the username, user id, hotel name, hotel id, and all the other review information.

Used: `'CREATE INDEX IF NOT EXISTS idx_hotel_hotel_name ON Hotel (Hotel_Name)'`

Essentially creates an index on the Hotel Name for the Hotel entity. This is useful for speeding up queries involving hotels, such as the Review_View since it uses hotel name.

Used: `'CREATE INDEX IF NOT EXISTS idx_users_username ON Users (Username)'`

Essentially creates an index on the username for a user entity. This was useful for speedup up queries such as the Review_View since it uses username.

Krish Chhabra

Files:

/website_files

table_maker.php *(shared)*
table_initializer.php *(shared)*

/website_files/services

services_cust.php
services_serv.php
delete_service.php
update_service_cust.php
update_service_serv.php

website_files/nav

navbar.php *(shared)*

Functionalities:

Krish was originally assigned responsibility for the functionality set **4**. He ended up being responsible for the functionality set **3** due to a confusion between him and Uchenna on what functionalities they were originally assigned. The "experiences" that are in our project are referred to as services. These represent services such as room services, delivery, repairs, etc. The functionality set Krish completed for this project includes:

- View Service: Allows the user to view services that are applicable to them.
    - User Type Customer: If the user is a customer, they can view the services that are associated with their active bookings.
    - User Type Service Worker or Admin: If the user is a service worker or an administrator, they can view the services that have been assigned to them or that belong to the hotel they work at.
- Delete Service: Allows the user to delete a service associated with them
    - User Type Customer: If the user is a customer, they can delete the services that are associated with them. This means that they can only delete services which apply to their bookings.
    - User Type Service Worker: If the user is a service worker, they can delete the services that have been assigned to them.
    - User Type Administrator: If the user is an administrator, they can delete any valid service.
- Update Service: Allows the user to update important aspects of their service.
    - User Type Customer: If the user is a customer, they can update the date and time at which one of their services will occur.
    - User Type Service Worker: If the user is a service worker, they can update the assigned employees of services that are assigned to them or unassigned (transfer service to another employee).
- Add Service: Allows the user to add/schedule a service. This functionality was planned, but implementation of this functionality was not completed.
    - User Type Customer: If the user is a customer, they can add a new service to an active booking.
    - User Type Service Worker: If the user is a service worker, they can add a new service to any active booking in their hotel.

Used: `"CREATE VIEW IF NOT EXISTS Service_View AS SELECT Hotel_Service.Service_ID AS Service_ID, Hotel_Service.Service_Date AS Service_Date, Service_Type.ST_ID AS ST_ID, Service_Type.Service_Type AS Service_Type, Service_Type.Price AS Price, Service_Assignment.SA_ID AS SA_ID, Service_Assignment.User_ID AS Emp_ID, Users.FName AS Emp_FName, Users.LName AS Emp_LName, Users.Email AS Emp_Email, Booking.Booking_NO AS Booking_NO, Booking.User_ID AS Cust_ID, Booking.Start_Date AS Start_Date, Booking.End_Date AS End_Date, Room.Room_ID AS Room_ID, Room.Room_Num AS Room_Num, Hotel.Hotel_ID AS Hotel_ID, Hotel.Hotel_Name AS Hotel_Name, Hotel.Hotel_City AS Hotel_City, Hotel.Hotel_State AS Hotel_State, Hotel.Hotel_Country AS Hotel_Country FROM ((((((Hotel_Service LEFT JOIN Service_Type ON Hotel_Service.ST_ID = Service_Type.ST_ID) LEFT JOIN Service_Assignment ON Hotel_Service.Service_ID = Service_Assignment.Service_ID) LEFT JOIN Users ON Service_Assignment.User_ID = Users.User_ID) LEFT JOIN Booking ON Hotel_Service.Booking_NO = Booking.Booking_NO) LEFT JOIN Room ON Booking.Room_ID = Room.Room_ID) LEFT JOIN Hotel ON Room.Hotel_ID = Hotel.Hotel_ID)"`

Essentially creates a service view of the services which consolidates all of the important information pertaining to both the user and employee sides of a service into one virtual table for easy data retrieval.

Used: `'CREATE INDEX IF NOT EXISTS idx_service_service_id ON Hotel_Service (Service_ID)'`

Essentially creates an index on the Service_ID for the Hotel_Service entity. This is useful for speeding up select queries involving services, as this table is the bridge entity which connects the customer and employee sides of the services.