Mihir Jain

Parth Desai

Group 35

CS166

CS166 Project Phase 3 Briefing

Languages Used:

- Java
- PostgreSQL

Starting the Database:

1. Enter the "Project" directory on the terminal taken from the project zip.
2. Use the "source ./startPostgreSQL.sh" command and then "source ./"createPostgreDB.sh" command to start and create the database.
3. Use the "source ./sql/scripts/create_db.sh" command to load the sql scripts.
4. Use the "source ./java/scripts/compile.sh" command to load Hotel.java file and start the Hotel Database.

High Level Implementation

1. viewHotels: We asked the user to input their current latitude and longitude and saved this as a double variable. We then made a string query that returned all the hotel names with the corresponding euclidean distance, which we included in the SQL query, as we were accessing the latitude and longitude of every hotel. We did a for loop afterwards that iterated through the list<list<string> created and checked if the hotel was within 30 units. If so, we printed it and incremented the hotel counter by 1.
2. viewRooms: We asked the user to input the hotelID and date to display the available rooms at that time. Our query then retrieved the hotel room information from the database based on the condition that the hotelID and bookingDate was the same as the user input. This was saved as a 2 dimensional list, with the number of rooms in the list being returned. We displayed this on the console after the query was executed with the hotelName in a separate for loop afterwards.

3. bookRooms: We asked the user to input the hotelId, room number, and booking date. We then performed error checking for the room number and booking date. If the query searching for the room number returned nothing, an error message was displayed and asked the user to enter a new room number. This same concept applied to the booking Date, except the error message was displayed if the booking Date returned a result, indicating that the date was booked. We then executed queries that found the unavailable rooms and the max booking id to assign to our new booking. At this point, everything has been checked and the query inserting a new booking is executed, otherwise an error message is printed.
4. viewRecentBookingsFromCustomer: For this function, we were supposed to display the customer and their most recent 5 bookings. We first took an input for the customer id to use for our query. Our query then drew the various elements needed for this function, such as room number, booking date, price, hotel number, based on various conditions, such as the id on the booking was the same as the input id. We then sorted the results by booking date and then set a limit to 5 to display the 5 most recent orders.
5. updateRoomInfo: For this function, we asked for various inputs, such as userid, price, room number, hotel number, and image url. We then executed a query that created a new entry in the Rooms database with the input information that was asked earlier. This query would execute and update if the user id was the same as the stored manager id.
6. viewRecentUpdates: This function lets the manager see updates to their hotel, and asked for the Manager ID, and then a query would be executed that would select all the contents from the RoomUpdatesLog relevant to that Manager ID and return the data ordered by the update number, and only showing the 5 most recent in descending order.
7. viewBookingHistoryofHotel: This function allows the Manager to view the booking history of the hotel and asked for the Hotel ID, and then a query would be executed that would select all the content from the RoomBooking table, where the HotelID given by the user was matched to the hotelID in the table, and all relevant information from Room Bookings was outputted.
8. viewRegularCustomers: This function allows the Manager to view the top 5 most bookings by customers, and asks for the Hotel ID. The query selects the customerID, and

then Counts from the RoomBookings table all information/bookings where the hotelID is equal to the input, and returns the top 5 customers for that hotel in a descending order.

9. placeRoomRepairRequests: This function allows the Manager to submit a room repair request, and asks for HotelID, Room Number, Maintenance Company ID, and the Repair Date. A query then inserts all this data into the Room Repairs table, and outputs a confirmation message if successful.
10. viewRoomRepairHistory: This function allows the manager to view all their previous room repair requests for a specific Hotel, and asks for them to input the HotelID. A query then selects all information relevant to that Hotel ID from the Room Repairs table and outputs it.

Contributions

1. Mihir: did the first 5 member functions
2. Parth: did the last 5 member functions, helped Mihir with connecting vsCode to ssh to work more efficiently, helped rectify viewRecentBookings function

Assumptions:

- The User accessing the database is truthful about their status and does not try and access data not meant for them.
- The Manager knows the company ID they wish to send the Room Repair Request to beforehand.

Issues we Faced:

1. Creating the correct SQL queries was tricky, and for some functions (bookRooms, updateRoomInfo), we were unable to successfully execute our queries without getting errors, and for others we were able to only get partial information from what was requested in the instructions.
2. (specific to Mihir) had issues with Vim. Vim was giving me a lot of issues, as I was working on the functions in VsCode and was pasting it in Vim, which often led to strange formatting errors. Parth sent me a file that had directions to connect vscode to the ssh, allowing me to code and test more efficiently.

3. How to update the database after inserting the correct information was also something we had trouble with aswell.
4. Manager authentication was a big issue that faced us throughout our project. We weren't sure initially how to confirm that the user was a manager in order to have proper access to the appropriate features.
5. Checking the availability of bookings by Dates was also an issue we were unable to resolve, as our queries would not match and output the correct data.
6. We were unable to authenticate the user or match availability in our database, but all of our other functions work and we have attempted all required parts of the project to the best of our abilities, as well as seeking help from the TA's during office hours.

Extra Credit:

- We added indexes for all relevant tables and attributes based on usage throughout the DataBase, and only implemented indexes for Data that was frequently called upon or updated to help increase the speed of the query time.
- We added certain phrases and crash checks throughout our Java code to ensure the user understood the input instructions to the best of their ability.
- We do not have any errors regarding the Java code.