Train Station System

#	Name	ID
1	Abdel-Rahman mohamed	20211061
2	Mohamed Ashraf	20210327
3	Mohamed Ahmed Amin	20210323
4	Abdalla Ahmed	20200878
5	Mohmed safaa	20210537

This report provides an overview of the Train Station System, which is designed to manage train bookings and provide various functionalities for users. The system is implemented using the Python programming language and relies on the pypyodbc library for database connectivity.

1. User Management:

The system supports two types of users: admin and customer. The following functionalities are available for user management:

- **Signing up new users:** The **signup** function allows a new user to register by providing their ID, name, age, gender, and password. The user's details are stored in the database, and a confirmation message is displayed upon successful registration.
- **Updating user details:** The **update_user_details** function enables users to modify their personal information such as name, age, and gender. Users can input their ID and provide the updated details, which are then stored in the database. A success message is displayed upon successful update.

2. Train Management:

The Train Management functionality is primarily focused on administrators. The key features are as follows:

- **Adding a train:** Administrators can use the **add_train** function to add a new train to the system. They need to provide details such as the train ID, name, source station, destination station, and total number of seats. The train information is stored in the database, and a confirmation message is displayed upon successful addition.
- **Updating train details:** The **update_train** function allows administrators to modify train information such as the train name, source station, destination station, and total number of seats. By providing the train ID and updated details, administrators can ensure that the train details are up to date in the system. A success message is displayed upon successful update.

3. Trip Management:

The Trip Management functionality focuses on creating and updating trip details. The key functionalities are as follows:

- **Adding a trip:** Administrators can use the **add_trip** function to create a new trip. They need to provide details such as the trip ID, train ID, departure time, and arrival time. The trip details are stored in the database, and a confirmation message is displayed upon successful addition.
- **Updating trip details:** The **update_trip** function enables administrators to modify trip information, including the train ID, departure time, and arrival time. By providing the trip ID and updated details, administrators can ensure that the trip information is accurate. A success message is displayed upon successful update.

4. Seat Availability:

The Seat Availability functionality allows users to search for available seats based on specific criteria. The key features are as follows:

- **Criteria for searching available seats:** Users can search for available seats by specifying criteria such as the date, time, source station, destination station, and the required number of seats. The system considers these factors when retrieving seat information.
- **Displaying available seats:** The **show_available_seats** function retrieves and presents a list of available seats that meet the specified criteria. The retrieved information includes seat numbers, train names, source stations, destination stations, departure times, and arrival times. The user interface is designed to provide a clear and easy-to-navigate view of the available seat options.

5. Trip Operations: Booking and Canceling:

The system allows users to perform operations related to trip bookings. The key functionalities are as follows:

- **Booking a trip:** The **book trip** function enables users to book a trip by providing their passenger ID and the trip ID. The system checks for seat availability and assigns the first available seat. The booking details are stored in the database, and a confirmation message with the assigned seat number is displayed.
- **Canceling a booking:** Users can cancel their bookings using the **cancel booking** function. By providing their passenger ID and booking ID, the system retrieves the associated seat number and updates its availability. The booking details are then removed from the database, and a confirmation message is displayed.

6. System Interface:

The Train Station System provides a user-friendly interface for users to interact with the functionalities. The main menu presents options for signing up, logging in, or exiting the system. Once logged in, users can access various functionalities based on their role (admin or customer). Clear prompts and input requests guide users through the process, ensuring ease of use and clarity.

7. Database Connectivity:

The system establishes a connection to the database using the pypyodbc library. The connection string specifies the SQL Server driver, server name, and database name. The connection is closed when the system is exited.

8. tools :

conceptual ERD using power designer.

physical ERD using power designer

overview of the code in GUI:

The code imports modules such as tkinter for GUI, and pypyodbc for connecting to the SQL Server database.

- **The code defines functions for updating trip details (GUI_Update_Trip), updating train details (GUI_Update_Train), canceling tickets (cancel), passenger booking (PassengerBooking), and adding a train (addingTrainGui).**

- **The GUI_Update_Trip function creates a GUI window for updating trip details. It includes form fields for trip ID, train ID, departure time, and arrival time. The user can enter the details and click the "Update" button to update the trip details in the database.**

- **The addingTrainGui function creates a GUI window for adding a train. It includes form fields for train ID, train name, source station, destination station, and total seats. The user can enter the details and click the "Add Train" button to add a new train to the database**

- **The GUI_Update_Train function creates a GUI window for updating train details. It includes form fields for train ID, train name, source station, destination station, and total seats. The user can enter the details and click the "Update" button to update the train details in the database.**

- **The PassengerBooking function creates a GUI window for passenger booking. It allows the user to enter a destination, displays available trains, and allows the user to choose a seat and book it. The ticket information is printed to the console.**

- **. The cancel function creates a GUI window for canceling tickets. It includes fields for ticket ID and booking ID. When the user clicks the "Cancel Ticket" button, the corresponding ticket, booking, and seat in the database are updated.**

Train Station Passenger Booking Report

Date: [Current Date]

1. Total Number of Bookings: [Count of bookings in the Booking table]
2. List of Bookings:

Booking ID	Passenger ID	Trip ID	Train ID	Seat Number	Date	Time
[Booking ID]	[Passenger ID]	[Trip ID]	[Train ID]	[Seat Number]	[Date]	[Time]
[Booking ID]	[Passenger ID]	[Trip ID]	[Train ID]	[Seat Number]	[Date]	[Time]
...	...	...	...	...	...	...

3. Total Number of Passengers: [Count of distinct passenger IDs in the Booking table]
4. Total Number of Trips: [Count of distinct trip IDs in the Booking table]
5. List of Trips:

Trip ID	Train ID	Departure Time	Arrival Time
[Trip ID]	[Train ID]	[Departure Time]	[Arrival Time]

Trip ID	Train ID	Departure Time	Arrival Time
[Trip ID]	[Train ID]	[Departure Time]	[Arrival Time]
	...	...	...

some additional aggregate functions the Train Station Passenger Booking Report:

1. Total Number of Bookings:

SELECT COUNT(*) AS total_bookings FROM Booking.

2.List of Bookings:

```
SELECT b.booking_id, b.passenger_id, b.trip_id, t.train_id, b.seat_number,
tck.date, tck.time
FROM Booking b
JOIN Ticket tck ON b.booking_id = tck.ticket_id;
```

3. Total Number of Passengers:

```
SELECT COUNT(DISTINCT passenger_id) AS total_passengers FROM Booking.
```

4. Total Number of Trips:

```
SELECT COUNT(DISTINCT trip_id) AS total_trips FROM Booking.
```

5. List of Trips:

```
SELECT t.trip_id, t.train_id, t.departure_time, t.arrival_time.

FROM Trip t

JOIN Booking b ON t.trip_id = b.trip_id.
```

6. (Highest Number of Bookings):

```
SELECT trip_id, COUNT(*) AS total_bookings FROM Booking GROUP BY
trip_id ORDER BY total bookings DESC
```

Please note that these queries assume that the necessary data exists in the tables and the table relationships are properly defined.