**SNIoE Campus Pool System**

**Monsoon 2023**

**Submitted by**

**Sagar Kaistha**

**Under supervision of**

**Associate Professor, Dept. of Computer Science Engg.**

**Shiv Nadar University, Greater Noida, India**

****

**Department of Computer Science Engineering**

**School of Engineering**

**Shiv Nadar University**

**Objective and key features**

1. **User Authentication:**  
   **→ Objective:** Implement a secure user authentication system allowing users to either log in or sign up with their credentials.  
   **→ Key Deliverables:**

* User login functionality.
* User registration functionality with appropriate validation.
* Secure storage of user credentials.

1. **Ride Booking Feature:**  
   **→ Objective:** Develop a ride booking feature to allow users to find suitable rides based on their preferences.  
   **→ Key Deliverables:**

* User interface for entering ride details (number of people, date, contact details and destination).
* Backend logic to process and store ride booking information.
* Algorithm to match user preferences with available rides.

1. **Ride Registration Feature:**  
   **→ Objective:** Implement a streamlined process for users to register their rides.  
   **→ Key Deliverables:**

* User interface for entering ride details (number of available seats, date, contact details and destination).
* Backend logic to process and store ride registration information.
* Seamless user experience for registering rides.

1. **Listing Available Rides:**  
   **→ Objective:** Display a list of available rides matching user preferences.  
   **→ Key Deliverables:**

* Algorithm to filter and display available rides based on user input.
* User interface to present a clear and organized list of matching rides.
* Real-time or near-real-time updating of available rides.

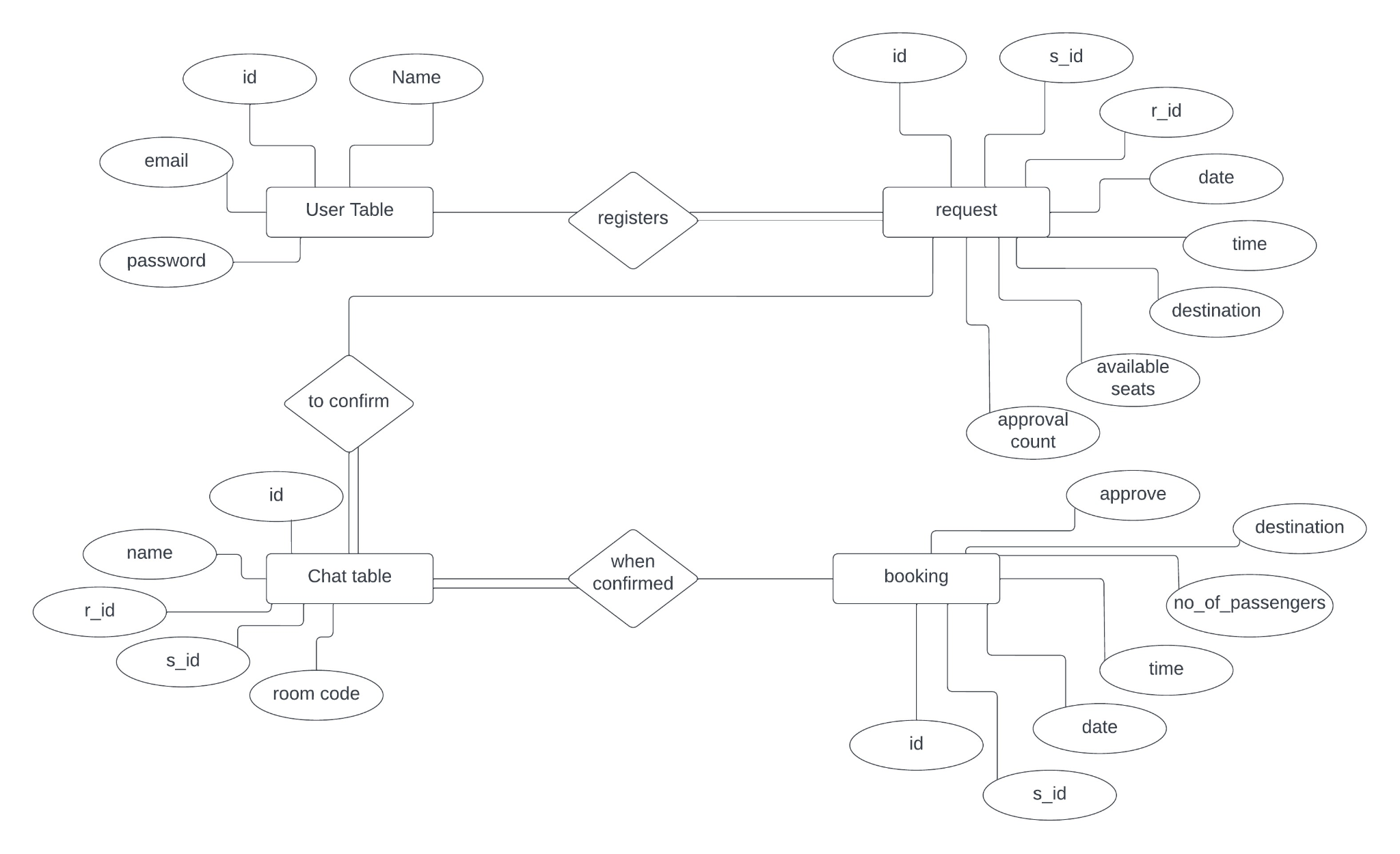
1. **User Experience Enhancement:**  
   **→ Objective:** Enhance the overall user experience of the transportation service.  
   **→ Key Deliverables:**

* User-friendly interfaces for both ride booking and registration.
* Clear and concise instructions throughout the user journey.
* Feedback mechanisms to gather user input for continuous improvement.

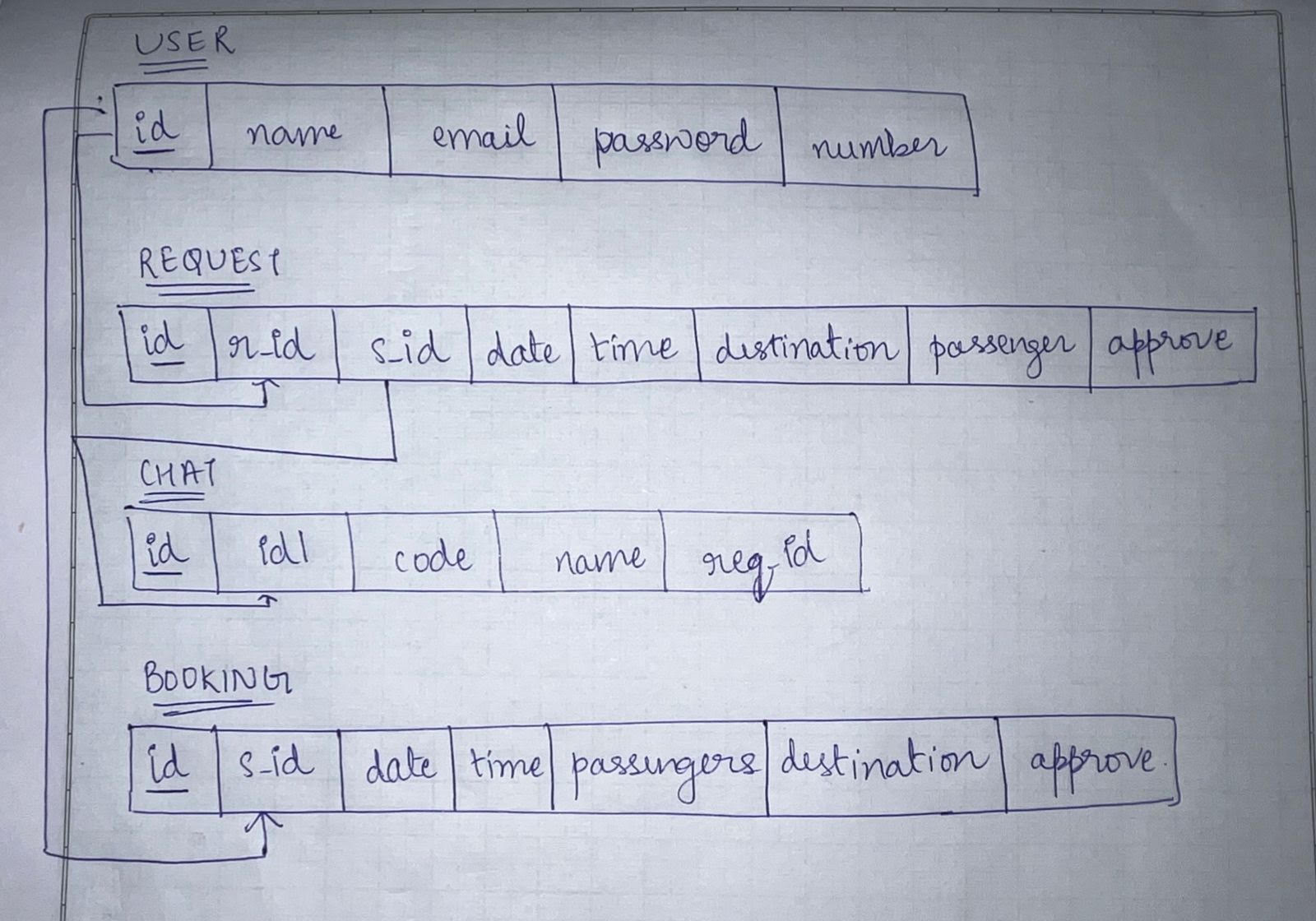
**Technology Stack Used**

* Frontend - HTML & CSS
* Backend - FLASK & MySQL

**ER DIAGRAM**

****

**RELATIONAL MODEL**

****

**TABLES:-**

1. CREATE TABLE user (

id int NOT NULL AUTO\_INCREMENT,

name varchar(100) NOT NULL,

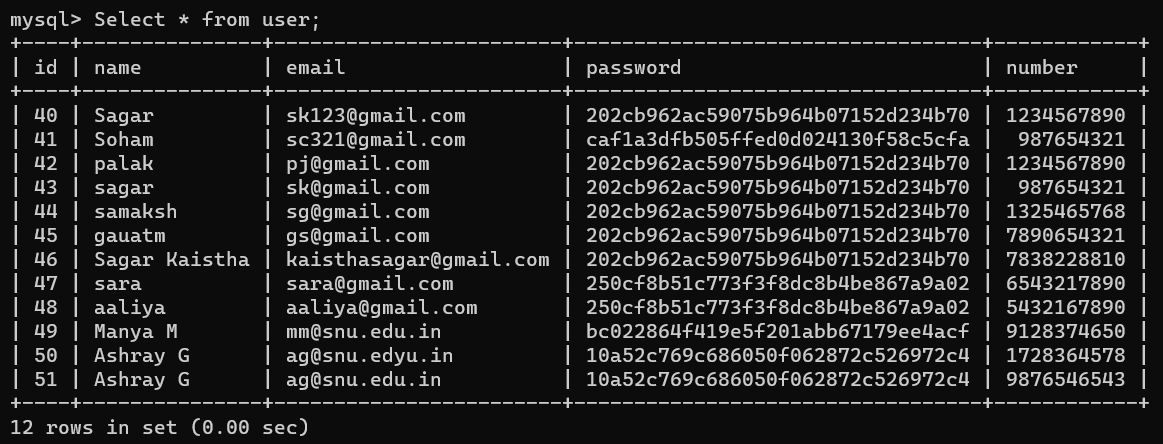
email varchar(100) NOT NULL,

password varchar(255) NOT NULL,

number bigint DEFAULT NULL,

PRIMARY KEY (id)

);



1. CREATE TABLE request (

id int NOT NULL AUTO\_INCREMENT,

r\_id int NOT NULL,

s\_id int NOT NULL,

date date NOT NULL,

time time NOT NULL,

destination varchar(100) NOT NULL,

passenger int NOT NULL,

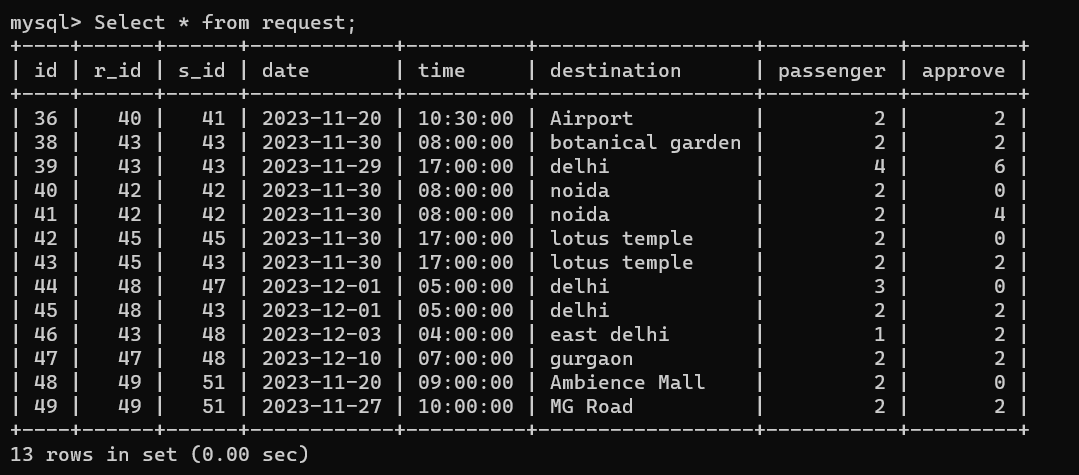
approve int NOT NULL DEFAULT '0',

PRIMARY KEY (id)

);

ALTER TABLE request

ALTER approve SET DEFAULT 0;



1. CREATE TABLE chat (

id int NOT NULL AUTO\_INCREMENT,

id1 varchar(100) NOT NULL,

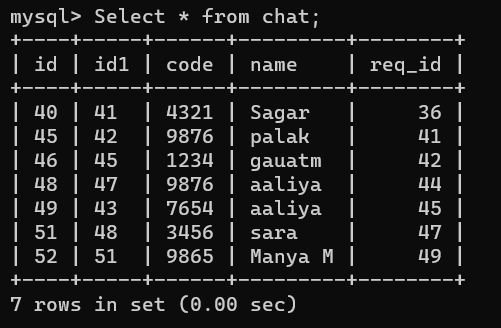
code int NOT NULL,

name varchar(100) NOT NULL,

req\_id int NOT NULL,

PRIMARY KEY (id)

);



1. CREATE TABLE booking (

id int NOT NULL AUTO\_INCREMENT,

s\_id int NOT NULL,

date date NOT NULL,

time time NOT NULL,

passengers int NOT NULL,

destination varchar(100) NOT NULL,

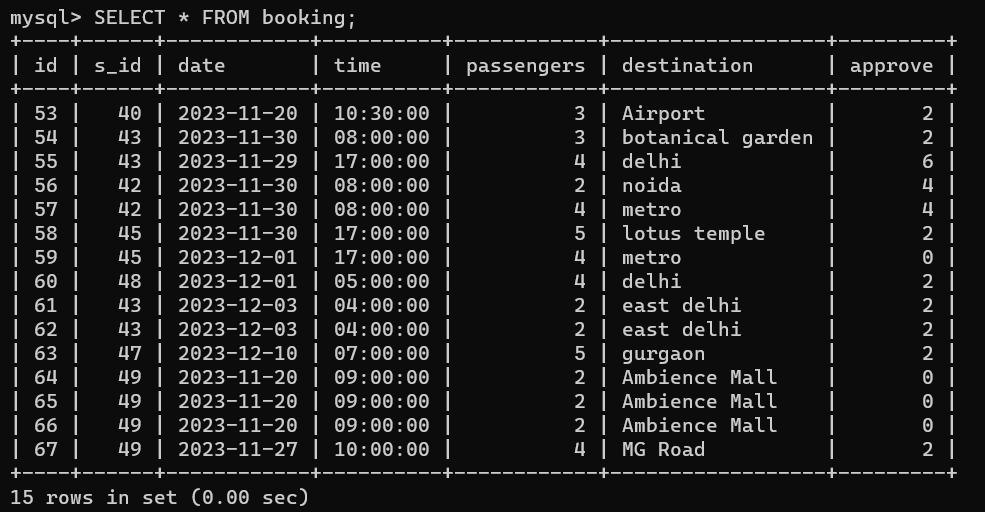
approve int NOT NULL DEFAULT '0',

PRIMARY KEY (id)

);

ALTER TABLE booking

ALTER approve SET DEFAULT 0;



**QUERIES**

1. HOME PAGE  
   SELECT \* FROM user
2. SIGNUP PAGE  
   **→** 'SELECT \* FROM user WHERE email = %s', (email, )  
   **→** "INSERT INTO user (name,email,number,password) VALUES (%s,%s,%s,MD5(%s))",(name,email,number,password)
3. LOGIN PAGE  
   → 'SELECT \* FROM user WHERE email = %s AND password = MD5(%s)', (email, password)
4. BOOK PAGE  
   →'SELECT \* FROM booking WHERE date = %s and passengers >=%s and time =%s and approve<2', (date,passenger,time)
5. REGISTER PAGE  
   → "INSERT INTO booking (s\_id,date,time,passengers,destination) VALUES (%s,%s,%s,%s,%s)",(s\_id,date,time,passengers,destination)
6. RESULT PAGE  
   → 'SELECT \* FROM user WHERE id = %s',(id,)  
   → 'SELECT \* FROM booking WHERE s\_id = %s AND time =%s and approve<2',(id,time)
7. APPROVE PAGE  
   → "INSERT INTO request (r\_id,s\_id,date,time,destination,passenger) VALUES (%s,%s,%s,%s,%s,%s)",(id,id1,date,time,destination,passenger)
8. REQUESTS PAGE  
   → 'SELECT \* FROM request WHERE r\_id=%s',(id,)
9. CONFIRM PAGE  
   → 'SELECT \* FROM user WHERE id=%s',(id,)  
   → 'SELECT \* FROM request WHERE id=%s and approve=0',(ids,)
10. CHATREQ PAGE  
    → 'SELECT \* FROM chat WHERE id1=%s', (id,)
11. CHATHOME PAGE  
    → 'SELECT \* FROM request WHERE id=%s',(id,)  
    → "DELETE FROM chat WHERE id1=%s",(ids,)  
    → "INSERT INTO chat (id1,code,name,req\_id) VALUES (%s,%s,%s,%s)",(ids,room,nam,idz)
12. ROOM PAGE  
    → 'SELECT \* FROM chat WHERE code=%s',(room,)
13. CONFIRMED PAGE  
    → 'SELECT \* FROM request WHERE r\_id=%s or s\_id=%s and approve>1', (id,id)  
    → 'SELECT \* FROM user'
14. REGISTERED PAGE  
    → 'SELECT \* FROM booking WHERE s\_id=%s', (id,)
15. YES PAGE  
    → 'SELECT \* FROM request WHERE id=%s',(ids,)  
    → "UPDATE request SET approve=approve+1 where id=%s", [[ids]]  
    → 'SELECT \* FROM booking WHERE s\_id=%s AND date=%s AND time=%s', [[id],[date],[time]]  
    → "UPDATE booking SET approve=approve+1 where s\_id=%s AND date=%s AND time=%s", [[id],[date],[time]]
16. NO PAGE  
    → 'DELETE FROM request WHERE id=%s', (id,)  
    → 'DELETE FROM chat WHERE id=%s',([cid])
17. EDIT PAGE  
    → 'SELECT \* FROM booking WHERE id=%s ', (keys,)  
    → "UPDATE BOOKING SET date=%s, time=%s, passengers=%s, destination=%s WHERE id=%s, (date, time, passengers, destination, keys)

**LIMITATIONS**

1. Exclusive Booking Policy

→ The system currently supports a single user per registered request, that means multiple users cannot book the same request simultaneously.

2. Restricted Viewing of Requests

→ The system currently displays requests exclusively for the specified date and time entered by the person who made the request.

3. Contact Information Visibility

→ Presently, the system restricts the visibility of contact details, allowing the person who booked the ride to view the information of the person who made the request but not vice-versa.

**REFERENCES**

1. <https://kris-litman.medium.com/connecting-flask-to-a-mysql-database-6f4d71b85d4e>
2. <https://medium.com/@AlexanderObregon/building-a-web-application-from-scratch-with-flask-and-python-f25f1f638aec>