Big Data Storage

Submitted by:

Khushi Chauhan

SAP ID: 500102244

Enrolment no: R2142221203

Semester IV

B. Tech CSE Big Data B2 (Non-Hons)

Submitted To:

Mr Deepak Kumar Sharma

DATA SCIENCE CLUSTER

UNIVERSTIY OF PETROLEUM & ENERGY STUDIES

DEHRADUN

Domain-Hotel Booking System

A Hotel Booking System Domain refers to the specific industry or sector where hotel booking systems are used to facilitate the reservation, management, and tracking of hotel accommodations. It involves a range of entities, processes, and interactions related to hotels, guests, and their bookings.
The Hotel Booking System Domain is an integral part of the broader hospitality and tourism industry. It encompasses various elements and stakeholders, including hotels, customers, and third-party booking platforms. The primary objective is to streamline the booking process, enhance guest experiences, and optimize hotel management.
Here is a short note on the Hotel Booking System domain:

The Hotel Booking System allows customers to book rooms and make reservations at hotels.

Key entities in the domain of the hotel booking system include:

Hotel: Represents the physical accommodation establishment where rooms are available for booking. It has attributes such as name, location, contact details, and features.

Room: Each room within the hotel, categorized by types such as Suite, Deluxe, Luxury, and Standard. Rooms have attributes like room number, type, price per night, and amenities.

User: Individuals who interact with the system, either as administrators or regular users. Users have attributes such as username, password, name, age, and email.

Booking: Represents a reservation made by a user for a particular room within a specific time frame. Booking entities include details such as the user making the booking, room number, check-in and check-out dates, total price, and timestamp of the booking.

Admin: Special type of user with elevated privileges, capable of performing administrative tasks such as adding, updating, or deleting rooms, managing bookings, and editing hotel details.

Facilities/Amenities: Additional features or services provided by the hotel, such as Wi-Fi, swimming pool, gym, etc., which may be associated with individual rooms or the hotel as a whole.

Flow of the code:

1. MongoDB Connection Setup:

Initialize a connection to the MongoDB server.

Access the database (hotel_booking_system) and relevant collections (users, rooms, booked_rooms).

2. Main Function (main ()):

Display a welcome message and a menu of options.

Prompt the user to choose an action.

Based on the user's choice, call corresponding functions such as about_hotel(), register(), login(), admin_menu(), user_menu(), or exit.

3. User Registration (register()):

Prompt the user to enter registration details (username, password, name, age, email).

Validate input data.

Check if the username already exists in the database.

Insert a new user document into the users collection if registration is successful.

4. About Hotel Function (about_hotel()):

Display information about the hotel, including its name, location, and contact details.

5. Login Function (login()):

Prompt the user to enter their username and password.

Authenticate the user against the data in the users collection.

Return user information if authentication is successful.

6. Admin Menu (admin_menu()):

Present a menu of options for admin users.

Based on the admin's choice, call corresponding functions to manage rooms, bookings, or hotel details.

7. User Menu (user_menu()):

Present a menu of options for regular users.

Based on the user's choice, allow them to book a room, view their bookings, cancel bookings, or logout.

8. Booking Functions (book_room(), cancel_booking(), view_booking()):

Allow users to book a room, view their bookings, and cancel bookings.

Validate input data and perform necessary database operations to manage bookings.

Room Management Functions (add_room(), change_room_price(), delete_room(), add_facilities_or_amenities(), view_rooms()):

Allow admin users to add, update, and delete room details.

Implement functionalities to change room prices, add amenities, and view room information.

9. Utility Functions:

Implement utility functions for date validation and formatting.

10. Looping and Exiting:

Utilize while loops to maintain the program's interactivity until the user chooses to exit.

Exit the program when the user selects the exit option.

Source Code

Code:

```
import pymongo
from datetime import datetime

# MongoDB connection
client= pymongo.MongoClient('mongodb://localhost:27017/')
db= client['hotel_booking_system']
users_collection= db['users']
rooms_collection= db['rooms']
booked_rooms_collection= db['booked_rooms']
def register():
    while True:
        name= input("Enter your name: ")
        age= input("Enter your age: ")
        email= input("Enter your email: ")
        username= input("Create username: ")
        password= input("Enter password: ")
        confirm_password= input("Confirm password: ")
        if password != confirm_password:
            print("Passwords do not match. Please try again.")
            continue
        # for admin login the username and password per-define
        admin_username= "admin"
        admin_password= "adminpass"
        if username== admin_username and password == admin_password:
            print("Username already exists. Please choose another one.")
            continue
        if users_collection.find_one({"username": username}):
            print("Username already exists. Please choose another one.")
            continue
        users_collection.insert_one({"username": username, "password":
password, "type": "user", "name": name, "age": age, "email": email})
        print("Registration successful.")
        break
# Information about the hotel
def about_hotel():
    print("Hotel Name: Oceanview Resort")
    print("Location: ")
    print("123 Oceanview Road")
    print("Coastal Town, Seaside District")
    print("State of Goa, India")
    print("Features: Every room facing towards the sea, Top-ranked hotel")
    print("Contact Details: Phone - 123-456-7890, Email -
info@oceanviewresort.com")

def login():
    username= input("Enter username: ")
    password= input("Enter password: ")
    if username== "admin" and password == "adminpass":
        return {"username": username, "type": "admin"}
```

```
    else:
        user= users_collection.find_one({"username": username, "password":
password})
        if user:
            return {"username": username, "type": "user"}
        else:
            print("Invalid username or password.")
            return None

# Admin menu
def admin_menu():
    while True:
        print("Choose your operation ")
        print("1. View all rooms details")
        print("2. Add a new room details")
        print("3. Change room price")
        print("4. Add facilities or amenities")
        print("5. View booked rooms")
        print("6. Delete a room")
        print("7. About Hotel")
        print("8. Edit Hotel Details")
        print("9. Logout")
        admin_choice= input("Enter your choice: ")
        if admin_choice== '1':
            view_rooms()
        elif admin_choice== '2':
            add_room()
        elif admin_choice== '3':
            change_room_price()
        elif admin_choice== '4':
            add_facilities_or_amenities()
        elif admin_choice== '5':
            view_booked_rooms()
        elif admin_choice== '6':
            delete_room()
        elif admin_choice== '7':
            about_hotel()
        elif admin_choice== '8':
            edit_hotel_details()
        elif admin_choice== '9':
            print("Logged out successfully")
            break
        else:
            print("Invalid choice.")
# Features available for user
# Define the function to edit hotel details
def edit_hotel_details():
    print("Editing Hotel Details:")
    hotel_name= "Oceanview Resort"
    print("Enter Hotel Location:")
    print("123 Oceanview Road")
    print("Coastal Town, Seaside District")
    print("State of Goa, India")
```

```
    features= input("Enter Hotel Features (comma-separated): ").split(',')
    contact_phone= input("Enter Contact Phone: ")
    contact_email= input("Enter Contact Email: ")
    print("Hotel details updated successfully.")

def change_room_price():
    room_type= input("Enter room type to change price: ")
    new_price= float(input("Enter new price per night: "))
    result= rooms_collection.update_one({"type": room_type}, {"$set":
{"price_per_night": new_price}})
    if result.modified_count > 0:
        print("Room price updated successfully.")
    else:
        print("Room type not found. Please enter a valid room type.")

def delete_room():
    while True:
        room_number= input("Enter room number to delete: ")
        result= rooms_collection.delete_one({"room_number": room_number})
        if result.deleted_count > 0:
            print("Room deleted successfully.")
            break  # Exit the loop if deletion is successful
        else:
            print("Room not found. Please enter a valid room number.")

def add_facilities_or_amenities():
    room_type= input("Enter room type to add facilities or amenities: ")
    facilities= input("Enter facilities or amenities separated by commas:
").split(',')
    result= rooms_collection.update_one({"type": room_type}, {"$push":
{"amenities": {"$each": facilities}}})
    if result.modified_count > 0:
        print("Facilities or amenities added successfully.")
    else:
        print("Room type not found. Please enter a valid room type.")

def view_booked_rooms():
    # Sort by checkin_date in descending order
    pipeline= [
        {"$sort": {"checkin_date": pymongo.DESCENDING}}
    ]
    booked_rooms= list(booked_rooms_collection.aggregate(pipeline))

    if booked_rooms:
        print("Booked Rooms:")
        for idx, room in enumerate(booked_rooms, start=1):
            print(f"Slot {idx} --> Room Number: {room['room_number']}, Check-
in Date: {room['checkin_date'].strftime('%d-%m-%Y')}, Check-out Date:
{room['checkout_date'].strftime('%d-%m-%Y')}, Booked By: {room['username']},
Total price: {room['total_price']}")
    else:
        print("No rooms are currently booked.")
```

```
    # Count total number of rooms for each room type
    pipeline_total_rooms= [
        {"$group": {"_id": "$type", "total_rooms": {"$sum": 1}}}
    ]
    total_rooms_per_type= {doc['_id']: doc['total_rooms'] for doc in
rooms_collection.aggregate(pipeline_total_rooms)}

    # Count number of booked rooms for each room type
    pipeline_booked_rooms= [
        {"$group": {"_id": "$type", "booked_rooms": {"$sum": 1}}}
    ]
    booked_rooms_per_type= {doc['_id']: doc['booked_rooms'] for doc in
booked_rooms_collection.aggregate(pipeline_booked_rooms)}

    # Calculate number of available rooms for each room type
    available_rooms_per_type= {room_type: total_rooms_per_type.get(room_type,
0) - booked_rooms_per_type.get(room_type, 0) for room_type in
total_rooms_per_type}

    print("\nNumber of Available Rooms per Room Type:")
    for room_type, available_rooms in available_rooms_per_type.items():
        print(f"{room_type}: {available_rooms}")

def add_room():
    room_type= input("Enter room type (S for Suite, D for Deluxe, L for
Luxury, ST for Standard): ").upper()


    room_type_map= {"S": "Suite", "D": "Deluxe", "L": "Luxury", "ST":
"Standard"}

    if room_type not in room_type_map:
        print("Invalid room type prefix. Please enter a valid room type
prefix.")
        return

    room_number_prefix= room_type

    while True:
        room_number_suffix= input(f"Enter room number (starting from 1): ")
        room_number= room_number_prefix + room_number_suffix


        existing_room= rooms_collection.find_one({"room_number": room_number})
        if existing_room:
            print(f"Room number {room_number} already exists for
{existing_room['type']} room type. Please choose another one.")
        else:
            break

    price= float(input("Enter price per night: "))
    rooms_collection.insert_one({"room_number": room_number, "type":
room_type_map[room_type], "price_per_night": price})
```

```
    print("Room added successfully.")

def view_rooms():
    print("Available Room Types and Prices Per Night:")
    pipeline= [
        {"$group": {"_id": "$type", "price_per_night": {"$first":
"$price_per_night"}, "amenities": {"$first": "$amenities"}}}
    ]
    distinct_room_types= list(rooms_collection.aggregate(pipeline))

    if distinct_room_types:
        for room_type in distinct_room_types:
            print(f"Room Type: {room_type['_id']}, Price Per Night:
{room_type['price_per_night']}")
            print("Amenities:", ", ".join(room_type['amenities']))
    else:
        print("No room types found.")
# user Menue
def user_menu(user):
    print(f"Welcome {user['username']}!to Oceanview Resort")
    while True:
        print("User Menu:")
        print("1. Book a room ")
        print("2. ViewBooking details")
        print("3. Cancel Booking")
        print("4. Logout")
        user_choice= input("Enter your choice: ")
        if user_choice== '1':
            book_room(user)
        elif user_choice== '2':
            view_booking(user)
        elif user_choice== '3':
            cancel_booking(user)
        elif user_choice== '4':
            print("Logged out successfully")
            break
        else:
            print("Invalid choice.")
# Features provied to user
def view_booking(user):
    username= user["username"]
    bookings= list(booked_rooms_collection.find({"username": username}))
    if bookings:
        print("Your Bookings:")
        for idx, booking in enumerate(bookings, start=1):
            print(f"Slot {idx} --> Room Number: {booking['room_number']},
Check-in Date: {booking['checkin_date'].strftime('%d-%m-%Y')}, Check-out Date:
{booking['checkout_date'].strftime('%d-%m-%Y')}, Price Per Night:
{booking['price_per_night']}")
    else:
        print("No bookings found.")

def is_valid_date(date_str):
```

```
    try:
        datetime.strptime(date_str, "%d-%m-%Y")
        return True
    except ValueError:
        return False

def book_room(user):
    while True:
        checkin_date_str= input("Enter check-in date (DD-MM-YYYY): ")
        checkout_date_str= input("Enter check-out date (DD-MM-YYYY): ")

        if not is_valid_date(checkin_date_str) or not
is_valid_date(checkout_date_str):
            print("Invalid date format. Please enter dates in DD-MM-YYYY
format.")
            continue  # Ask for dates again

        checkin_date= datetime.strptime(checkin_date_str, "%d-%m-%Y")
        checkout_date= datetime.strptime(checkout_date_str, "%d-%m-%Y")

        if checkin_date>= checkout_date:
            print("Check-out date must be after check-in date.")
            continue

        night_count= (checkout_date - checkin_date).days

        print(f"\nSearching for available rooms for {night_count} nights from
{checkin_date.strftime('%d-%m-%Y')} to {checkout_date.strftime('%d-%m-
%Y')}...\n")


        distinct_room_types = rooms_collection.distinct("type")
        room_prices= {room['type']: room['price_per_night'] for room in
rooms_collection.find()}

        if distinct_room_types:
            print("Available Room Types and Prices Per Night:")
            for idx, room_type in enumerate(distinct_room_types, start=1):
                print(f"{idx}. {room_type} - Price Per Night:
{room_prices[room_type]}")

            room_type_choice= input("Enter the room type number to book: ")
            selected_room_type= distinct_room_types[int(room_type_choice) - 1]


            available_rooms= rooms_collection.find({
                "type": selected_room_type,
                "$or": [
                    {"bookings.checkin_date": {"$gte": checkout_date}},
                    {"bookings.checkout_date": {"$lte": checkin_date}},
                    {"bookings": {"$exists": False}}
                ]
            })
```

```
            available_rooms_list = list(available_rooms)

            if available_rooms_list:
                num_rooms= int(input(f"How many {selected_room_type} rooms do
you want to book? "))

                if num_rooms > len(available_rooms_list):
                    print(f"Sorry, there are only {len(available_rooms_list)}
{selected_room_type} rooms available.")
                    return


                selected_rooms= available_rooms_list[:num_rooms]

                total_price= sum(room_prices[selected_room_type] for _ in
range(num_rooms)) * night_count

                print("Selected Room Details:")
                for idx, room in enumerate(selected_rooms, start=1):
                    print(f"Room Number {idx}: {room['room_number']}")

                print(f"Check-in Date: {checkin_date.strftime('%d-%m-%Y')}")
                print(f"Check-out Date: {checkout_date.strftime('%d-%m-%Y')}")
                print(f"Total Price for {num_rooms} {selected_room_type}
rooms: {total_price}")
                confirm_booking= input("Do you want to proceed with booking?
(1 for yes/2 for no): ")

                if confirm_booking.lower()== '1':
                    current_timestamp= datetime.now()
                    for room in selected_rooms:
                        booking_data= {
                            "username": user["username"],
                            "checkin_date": checkin_date,
                            "checkout_date": checkout_date,
                            "room_number": room["room_number"],
                            "price_per_night":
room_prices[selected_room_type],
                            "total_price": total_price / num_rooms,
                            "booking_timestamp": current_timestamp
                        }
                        booked_rooms_collection.insert_one(booking_data)
                    print("Rooms booked successfully!")
                    print("Enjoy your stay!")
                    break

                else:
                    print("Booking canceled.")
            else:
                print(f"No available {selected_room_type} rooms for the
selected dates.")
        else:
```

```
            print("No available room types.")

def cancel_booking(user):
    username= user["username"]
    bookings= list(booked_rooms_collection.find({"username": username}))
    if bookings:
        print("Your Bookings:")
        for idx, booking in enumerate(bookings, start=1):
            print(f"Slot  {idx} --> Room Number: {booking['room_number']},
Check-in Date: {booking['checkin_date'].strftime('%d-%m-%Y')}, Check-out Date:
{booking['checkout_date'].strftime('%d-%m-%Y')}, Price Per Night:
{booking['price_per_night']}")

        choice= int(input("Enter the slot number to cancel booking: ")) - 1

        if 0 <= choice < len(bookings):
            booking_to_cancel= bookings[choice]
            cancellation_fee= 0.25 * (booking_to_cancel["checkout_date"] -
booking_to_cancel["checkin_date"]).days * booking_to_cancel["price_per_night"]
            print(f"Cancellation Fee: {cancellation_fee}")
            confirm_cancel= input("Do you want to proceed with cancellation?
(1 for yes/2 for no): ")
            if confirm_cancel.lower()== '1':
                booked_rooms_collection.delete_one({"_id":
booking_to_cancel["_id"]})
                print("Booking canceled successfully.")
            else:
                print("Cancellation canceled.")
        else:
            print("Invalid slot number. Please enter a valid slot.")
    else:
        print("No bookings found for cancellation.")

def main():
    while True:
        print("\nWelcome to Oceanview Resort ")
        print("1. About Hotel")
        print("2. Register")
        print("3. Login")
        print("4. Exit")
        choice= input("Enter your choice: ")
        if choice== '1':
            about_hotel()
        elif choice== '2':
            register()
        elif choice== '3':
            user= login()
            if user:
                if user["type"]== "admin":
                    admin_menu()
                    continue
                elif user["type"]== "user":
                    user_menu(user)
```

```
        elif choice== '4':
            print("Thank you for visting Oceanview Resort.")
            break
        else:
            print("Invalid choice. Please enter a valid option.")
main()
```

Sample output

1.About the Hotel:

```
Welcome to Oceanview Resort
1. About Hotel
2. Register
3. Login
4. Exit
Enter your choice: 1
Hotel Name: Oceanview Resort
Location:
123 Oceanview Road
Coastal Town, Seaside District
State of Goa, India
Features: Every room facing towards the sea, Top-ranked hotel
Contact Details: Phone - 123-456-7890, Email - info@oceanviewresort.com
```

2.User registration:

```
Welcome to Oceanview Resort
1. About Hotel
2. Register
3. Login
4. Exit
Enter your choice: 2
Enter your name: Khushi Chauhan
Enter your age: 20
Enter your email: Khushi@gmail.com
Create username: KhushiChauhan
Enter password: 123@34
Confirm password: 123@34
Registration successful.
```

2. User login:

```
Welcome to Oceanview Resort
1. About Hotel
2. Register
3. Login
4. Exit
Enter your choice: 3
Enter username: KhushiChauhan
Enter password: 123@34
Welcome KhushiChauhan!to Oceanview Resort
User Menu:
1. Book a room
2. ViewBooking details
3. Cancel Booking
4. Logout
```

3.User Menu:

3.1 Book a room:

```
User Menu:
1. Book a room
2. View Booking details
3. Cancel Booking
4. Logout
Enter your choice: 1
Enter check-in date (DD-MM-YYYY): 23-05-2024
Enter check-out date (DD-MM-YYYY): 26-05-2024

Searching for available rooms for 3 nights from 23-05-2024 to 26-05-2024...

Available Room Types and Prices Per Night:
1. Deluxe - Price Per Night: 8000
2. Luxury - Price Per Night: 12000
3. Standard - Price Per Night: 5000
4. Suite - Price Per Night: 20000.0
Enter the room type number to book: 4
How many Suite rooms do you want to book? 2
Selected Room Details:
Room Number 1: S1
Room Number 2: S2
Check-in Date: 23-05-2024
Check-out Date: 26-05-2024
Total Price for 2 Suite rooms: 120000.0
Do you want to proceed with booking? (1 for yes/2 for no): 1
Rooms booked successfully!
Enjoy your stay!
```

3.2 View Booking Details

```
User Menu:
1. Book a room
2. View Booking details
3. Cancel Booking
4. Logout
Enter your choice: 2
Your Bookings:
Slot 1 --> Room Number: S1, Check-in Date: 23-05-2024, Check-out Date: 26-05-2024, Price Per Night: 20000.0
Slot 2 --> Room Number: S2, Check-in Date: 23-05-2024, Check-out Date: 26-05-2024, Price Per Night: 20000.0
```

3.3 Cancel Booking

```
User Menu:
1. Book a room
2. View Booking details
3. Cancel Booking
4. Logout
Enter your choice: 3
Your Bookings:
Slot  1 --> Room Number: S1, Check-in Date: 23-05-2024, Check-out Date: 26-05-2024, Price Per Night: 20000.0
Slot  2 --> Room Number: S2, Check-in Date: 23-05-2024, Check-out Date: 26-05-2024, Price Per Night: 20000.0
Enter the slot number to cancel booking: 2
Cancellation Fee: 15000.0
Do you want to proceed with cancellation? (1 for yes/2 for no): 1
Booking canceled successfully.
```

3.4 Logout

```
User Menu:
1. Book a room
2. View Booking details
3. Cancel Booking
4. Logout
Enter your choice: 4
Logged out successfully
```

4. Admin login

```
Welcome to Oceanview Resort
1. About Hotel
2. Register
3. Login
4. Exit
Enter your choice: 3
Enter username: admin
Enter password: adminpass
```

5. Admin Menu

```
Welcome to Oceanview Resort
1. About Hotel
2. Register
3. Login
4. Exit
Enter your choice: 3
Enter username: admin
Enter password: adminpass
Choose your operation
1. View all rooms details
2. Add a new room details
3. Change room price
4. Add facilities or amenities
5. View booked rooms
6. Delete a room
7. About Hotel
8. Edit Hotel Details
9. Logout
```

5.1 View Room Details

```
Choose your operation
1. View all rooms details
2. Add a new room details
3. Change room price
4. Add facilities or amenities
5. View booked rooms
6. Delete a room
7. About Hotel
8. Edit Hotel Details
9. Logout
Enter your choice: 1
Available Room Types and Prices Per Night:
Room Type: Deluxe, Price Per Night: 8000
Amenities: WiFi, TV
Room Type: Suite, Price Per Night: 20000
Amenities: WiFi, Swimming Pool, Jacuzzi
Room Type: Luxury, Price Per Night: 12000
Amenities: WiFi, Swimming Pool, Spa
Room Type: Standard, Price Per Night: 4999.0
Amenities: TV, Balcony
```

5.2 Add new room details

```
Choose your operation
1. View all rooms details
2. Add a new room details
3. Change room price
4. Add facilities or amenities
5. View booked rooms
6. Delete a room
7. About Hotel
8. Edit Hotel Details
9. Logout
Enter your choice: 1
Available Room Types and Prices Per Night:
Room Type: Deluxe, Price Per Night: 8000
Amenities: WiFi, TV
Room Type: Suite, Price Per Night: 20000
Amenities: WiFi, Swimming Pool, Jacuzzi
Room Type: Luxury, Price Per Night: 12000
Amenities: WiFi, Swimming Pool, Spa
Room Type: Standard, Price Per Night: 4999.0
Amenities: TV, Balcony
```

5.3 Change price

```
Choose your operation
1. View all rooms details
2. Add a new room details
3. Change room price
4. Add facilities or amenities
5. View booked rooms
6. Delete a room
7. About Hotel
8. Edit Hotel Details
9. Logout
Enter your choice: 3
Enter room type to change price: Deluxe
Enter new price per night: 100000
Room price updated successfully.
```

5.4 Add Facilities

```
Choose your operation
1. View all rooms details
2. Add a new room details
3. Change room price
4. Add facilities or amenities
5. View booked rooms
6. Delete a room
7. About Hotel
8. Edit Hotel Details
9. Logout
Enter your choice: 4
Enter room type to add facilities or amenities: Standard
Enter facilities or amenities separated by commas: Balcony
Facilities or amenities added successfully.
```

5.5 View Booking

```
Choose your operation
1. View all rooms details
2. Add a new room details
3. Change room price
4. Add facilities or amenities
5. View booked rooms
6. Delete a room
7. About Hotel
8. Edit Hotel Details
9. Logout
Enter your choice: 5
Booked Rooms:
Slot 1 --> Room Number: S20, Check-in Date: 21-12-2024, Check-out Date: 25-12-2024, Booked By: a, Total price: 80000.0
Slot 2 --> Room Number: D1, Check-in Date: 20-09-2024, Check-out Date: 30-09-2024, Booked By: Khushi Chauhan, Total price: 80000
Slot 3 --> Room Number: L1, Check-in Date: 23-05-2024, Check-out Date: 28-05-2024, Booked By: Vamika Chauhan, Total price: 60000.0
Slot 4 --> Room Number: L2, Check-in Date: 23-05-2024, Check-out Date: 28-05-2024, Booked By: Vamika Chauhan, Total price: 60000.0
Slot 5 --> Room Number: L3, Check-in Date: 23-05-2024, Check-out Date: 28-05-2024, Booked By: Vamika Chauhan, Total price: 60000.0
Slot 6 --> Room Number: S1, Check-in Date: 23-05-2024, Check-out Date: 26-05-2024, Booked By: KhushiChauhan, Total price: 60000.0
Slot 7 --> Room Number: ST38, Check-in Date: 24-04-2024, Check-out Date: 30-04-2024, Booked By: Khushi Chauhan, Total price: 30000
Slot 8 --> Room Number: S1, Check-in Date: 20-04-2024, Check-out Date: 15-05-2024, Booked By: Dhuruv Kumar, Total price: 500000.0
Slot 9 --> Room Number: D1, Check-in Date: 12-01-2024, Check-out Date: 16-01-2024, Booked By: David Roy, Total price: 32000.0
Slot 10 --> Room Number: D2, Check-in Date: 12-01-2024, Check-out Date: 16-01-2024, Booked By: David Roy, Total price: 32000.0
Slot 11 --> Room Number: D3, Check-in Date: 12-01-2024, Check-out Date: 16-01-2024, Booked By: David Roy, Total price: 32000.0
Slot 12 --> Room Number: D4, Check-in Date: 12-01-2024, Check-out Date: 16-01-2024, Booked By: David Roy, Total price: 32000.0
```

5.6 Delete a room Details

```
Choose your operation
1. View all rooms details
2. Add a new room details
3. Change room price
4. Add facilities or amenities
5. View booked rooms
6. Delete a room
7. About Hotel
8. Edit Hotel Details
9. Logout
Enter your choice: 6
Enter room number to delete: s19
Room not found. Please enter a valid room number.
Enter room number to delete: S19
Room deleted successfully.
```

5.7 View about Hotel

```
Choose your operation
1. View all rooms details
2. Add a new room details
3. Change room price
4. Add facilities or amenities
5. View booked rooms
6. Delete a room
7. About Hotel
8. Edit Hotel Details
9. Logout
Enter your choice: 7
Hotel Name: Oceanview Resort
Location:
123 Oceanview Road
Coastal Town, Seaside District
State of Goa, India
Features: Every room facing towards the sea, Top-ranked hotel
Contact Details: Phone - 123-456-7890, Email - info@oceanviewresort.com
```

5.8 Change Hotel Details

```
Choose your operation
1. View all rooms details
2. Add a new room details
3. Change room price
4. Add facilities or amenities
5. View booked rooms
6. Delete a room
7. About Hotel
8. Edit Hotel Details
9. Logout
Enter your choice: 8
Editing Hotel Details:
Enter Hotel Location:
123 Oceanview Road
Coastal Town, Seaside District
State of Goa, India
Enter Hotel Features (comma-separated): Ranked 5 star by NIYE Ranking
Enter Contact Phone: 123-456-7899
Enter Contact Email: info@Ocenviewresort.com
Hotel details updated successfully.
```

5.9 Logout

```
Choose your operation
1. View all rooms details
2. Add a new room details
3. Change room price
4. Add facilities or amenities
5. View booked rooms
6. Delete a room
7. About Hotel
8. Edit Hotel Details
9. Logout
Enter your choice: 9
Logged out successfully
```

6. Exit

```
Welcome to Oceanview Resort
1. About Hotel
2. Register
3. Login
4. Exit
Enter your choice: 4
Thank you for visting Oceanview Resort.
```

MongoDB

MongoDB is a powerful NoSQL database system that stores data in flexible, JSON-like documents known as BSON. It's known for its scalability, performance, and ability to handle semi-structured and unstructured data efficiently. MongoDB's document-oriented architecture allows for dynamic schemas, enabling easy adaptation to changing data requirements. The database supports horizontal scaling through sharding and ensures high availability with replica sets, making it suitable for large-scale applications.

MongoDB has been adopted as backend software by a number of major websites and services including EA, Cisco, Shutterfly, Adobe, Ericsson, Craigslist, eBay, and Foursquare.

MongoDB Concepts and usage in the code:

1.Database Connection(MongoDB Connection with python):

The code establishes a connection to a MongoDB database using the pymongo library, specifying the database URL ('mongodb://localhost:27017/') and selecting the appropriate database ('hotel_booking_system').

1.Client Connection:

- pymongo.MongoClient('mongodb://localhost:27017/'): This command establishes a connection to the MongoDB server running on the local machine at the default port 27017. The MongoClient object represents the client that interacts with the MongoDB server**.**

```
import pymongo
from datetime import datetime

# MongoDB connection
client= pymongo.MongoClient('mongodb://localhost:27017/')
```

2. Creating Database

- db = client['hotel_booking_system']: This line accesses the database named 'hotel_booking_system' within the MongoDB server. If the specified database does not exist, MongoDB will create it when data is first written.

```
db= client['hotel_booking_system']
```

3. Creating Collection in Database

In Hotel booking system we have 3 Collection to store the Document.

- users_collection = db['users']: This line accesses the collection named 'users' within the 'hotel_booking_system' database. Collections in MongoDB are analogous to tables in relational databases. Is Created to stored all the data regarding the User in database Example the username , password, booking details etc.
- rooms_collection= db['rooms']:This Line accesses the collection named ‘rooms’ within the 'hotel_booking_system' database. Collections in MongoDB are analogous to tables in relational databases. Is created to stored all the data regarding the room in database Example types of room ,no.of room, facilities each type of room provide etc
- booked_rooms_collection= db['booked_rooms'] :This line accesses the collection named 'booked_rooms' within the 'hotel_booking_system' database. Collections in MongoDB are analogous to tables in relational databases. It is created so that all the details regarding booking can be stored in the databases so that their in no overbooking and data could be used get profitable values.

```
users_collection= db['users']
rooms_collection= db['rooms']
booked_rooms_collection= db['booked_rooms']
```

MongoDB Commands:

1.Inserting Document:

- **Syntax:** Collection_name.insert_one({"element1": value1, "element2": value2,…})
- **Explanation:** Inserts a single document representing a user into the users_collection with fields like username, password, type, name, age, and email.
- Command used in code:
  Inserting username and password during registeration:

```
users_collection.insert_one({"username": username, "password": password, "type": "user", "name": name, "age": age, "email": email})
print("Registration successful.")
break
```

2.Finding Document :

- **Syntax:** user = Collection_name.insert_one({"element1": value1, "element2": value2,…})
- **Explanation:** Retrieves a single document from the collection where the given parameters match match the provided credentials.
- Command used in code:
  Finding if user already exist in registration:

```
if users_collection.find_one({"username": username}):
    print("Username already exists. Please choose another one.")
    continue
```

Finding if the username and password are correct during login:

```
user= users_collection.find_one({"username": username, "password": password})
if user:
    return {"username": username, "type": "user"}
else:
    print("Invalid username or password.")
    return None
```

```
existing_room= rooms_collection.find_one({"room_number": room_number})
if existing_room:
    print(f"Room number {room_number} already exists for {existing_room['type']} room type. Please c
else:
    break
```

3.Adding Room Document:

- **Syntax:** rooms_collection.insert_one({"room_number": room_number, "type": room_type_map[room_type], "price_per_night": price})
- **Explanation:** Inserts a single document representing a room into the rooms_collection with fields like room_number, type, and price_per_night.

```
price= float(input("Enter price per night: "))
rooms_collection.insert_one({"room_number": room_number, "type": room_type_map[room_type], "price_per_night": price})
print("Room added successfully.")
```

4.Deleting Room Document:

- **Syntax:** rooms_collection.delete_one({"room_number": room_number})
- **Explanation:** Deletes a single document from the rooms_collection where the room_number matches the specified room to be deleted.

```
if confirm_cancel.lower()== '1':
    booked_rooms_collection.delete_one({"_id": booking_to_cancel["_id"]})
    print("Booking canceled successfully.")
```

5.Updating Room Price:

- **Syntax:** rooms_collection.update_one({"type": room_type}, {"$set": {"price_per_night": new_price}})
- **Explanation:** Updates the price_per_night field of a single document in the rooms_collection where the room type matches the specified type.

```
result= rooms_collection.update_one({"type": room_type}, {"$push": {"amenities": {"$each": facilities}}})
  if result.modified_count > 0:
    print("Facilities or amenities added successfully.")
  else:
    print("Room type not found. Please enter a valid room type.")
```

6.Aggregating Booked Room Data:

Explanation: Aggregates data from the booked_rooms_collection using the specified pipeline, allowing for processing and analysis of booked room data.

```
# Sort by checkin_date in descending order
  pipeline= [
    {"$sort": {"checkin_date": pymongo.DESCENDING}}
  ]
  booked_rooms= list(booked_rooms_collection.aggregate(pipeline))
```

7.Finding Booked Rooms for User:

Explanation: Retrieves documents from the booked_rooms_collection where the username matches the specified user, representing bookings made by that user.

```
# Count total number of rooms for each room type
  pipeline_total_rooms= [
    {"$group": {"_id": "$type", "total_rooms": {"$sum": 1}}}
  ]
            total_rooms_per_type= {doc['_id']: doc['total_rooms'] for doc in
rooms_collection.aggregate(pipeline_total_rooms)}
```

8.Counting Available Rooms per Type:

Syntax: total_rooms_per_type= {doc['_id']: doc['total_rooms'] for doc in rooms_collection.aggregate(pipeline_total_rooms)}

Explanation: Counts the total number of rooms for each room type by aggregating data from the rooms_collection.

```
# Count number of booked rooms for each room type
   pipeline_booked_rooms= [
      {"$group": {"_id": "$type", "booked_rooms": {"$sum": 1}}}
   ]
            booked_rooms_per_type=      {doc['_id']:      doc['booked_rooms']      for      doc      in
booked_rooms_collection.aggregate(pipeline_booked_rooms)}
```

9.Distinct Room Types:

Syntax: distinct_room_types= list(rooms_collection.distinct("type"))

Explanation: Retrieves distinct room types from the rooms_collection, allowing for differentiation between different types of rooms.

```
distinct_room_types = rooms_collection.distinct("type")
      room_prices= {room['type']: room['price_per_night'] for room in rooms_collection.find()}
```

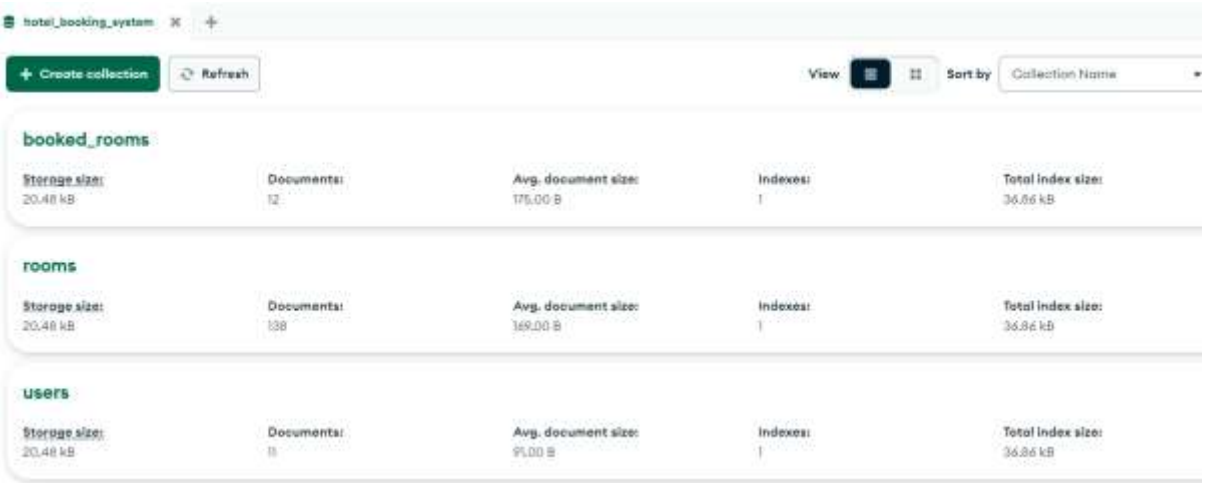
10.Adding Facilities or Amenities to Room:

Syntax: rooms_collection.update_one({"type": room_type}, {"$push": {"amenities": {"$each": facilities}}})

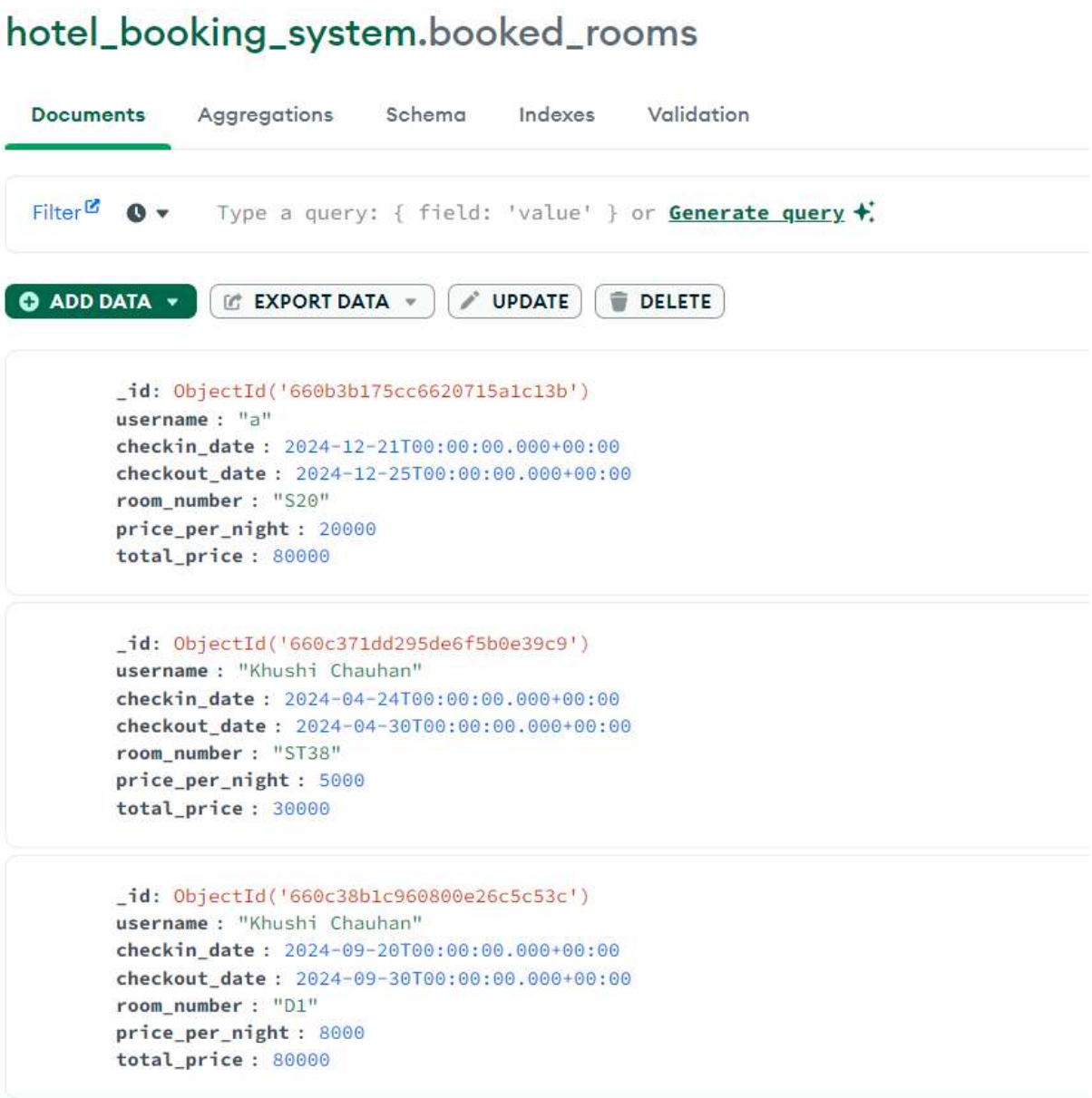
Explanation: Updates a single document in the rooms_collection by adding facilities or amenities to the specified room type.

```
room_type= input("Enter room type to change price: ")
   new_price= float(input("Enter new price per night: "))
   result= rooms_collection.update_one({"type": room_type}, {"$set": {"price_per_night": new_price}})
```

MongoDB



Booked_rooms Collection




```
_id: ObjectId('660b3b175cc6620715a1c13b')
username : "a"
checkin_date : 2024-12-21T00:00:00.000+00:00
checkout_date : 2024-12-25T00:00:00.000+00:00
room_number : "S20"
price_per_night : 20000
total_price : 80000
```

```
_id: ObjectId('660c371dd295de6f5b0e39c9')
username : "Khushi Chauhan"
checkin_date : 2024-04-24T00:00:00.000+00:00
checkout_date : 2024-04-30T00:00:00.000+00:00
room_number : "ST38"
price_per_night : 5000
total_price : 30000
```

```
_id: ObjectId('660c38b1c960800e26c5c53c')
username : "Khushi Chauhan"
checkin_date : 2024-09-20T00:00:00.000+00:00
checkout_date : 2024-09-30T00:00:00.000+00:00
room_number : "D1"
price_per_night : 8000
total_price : 80000
```

```
_id: ObjectId('660c4640807f0969dd96c483')
username : "Vamika Chauhan"
checkin_date : 2024-05-23T00:00:00.000+00:00
checkout_date : 2024-05-28T00:00:00.000+00:00
room_number : "L1"
price_per_night : 12000
total_price : 60000
booking_timestamp : 2024-04-02T23:24:08.353+00:00
```

```
_id: ObjectId('660c4640807f0969dd96c484')
username : "Vamika Chauhan"
checkin_date : 2024-05-23T00:00:00.000+00:00
checkout_date : 2024-05-28T00:00:00.000+00:00
room_number : "L2"
price_per_night : 12000
total_price : 60000
booking_timestamp : 2024-04-02T23:24:08.353+00:00
```

```
_id: ObjectId('660c4640807f0969dd96c485')
username : "Vamika Chauhan"
checkin_date : 2024-05-23T00:00:00.000+00:00
checkout_date : 2024-05-28T00:00:00.000+00:00
room_number : "L3"
```

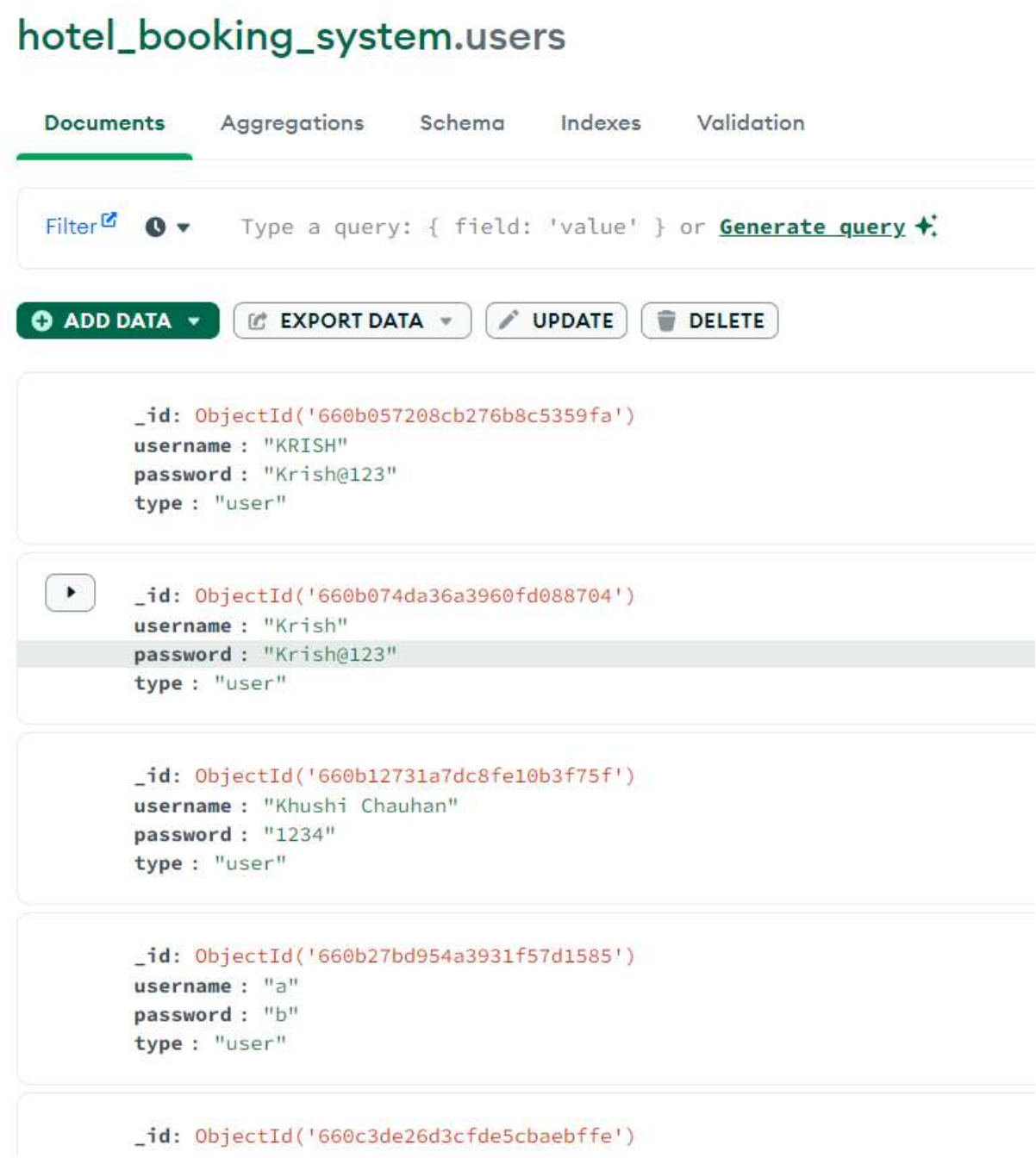
```
_id: ObjectId('660c4da07063dfca1b9e67df')
username : "David Roy"
checkin_date : 2024-01-12T00:00:00.000+00:00
checkout_date : 2024-01-16T00:00:00.000+00:00
room_number : "D1"
price_per_night : 8000
total_price : 32000
booking_timestamp : 2024-04-02T23:55:36.632+00:00
```

```
_id: ObjectId('660c4da07063dfca1b9e67e0')
username : "David Roy"
checkin_date : 2024-01-12T00:00:00.000+00:00
checkout_date : 2024-01-16T00:00:00.000+00:00
room_number : "D2"
price_per_night : 8000
total_price : 32000
booking_timestamp : 2024-04-02T23:55:36.632+00:00
```

```
_id: ObjectId('660c4da07063dfca1b9e67e1')
username : "David Roy"
checkin_date : 2024-01-12T00:00:00.000+00:00
checkout_date : 2024-01-16T00:00:00.000+00:00
room_number : "D3"
```

Rooms_Collection

Users_collection



users

	_id ObjectId	username String	password String	type String	name String	age String
1	ObjectId('660b057208cb276…	"KRISH"	"Krish@123"	"user"	No field	No field
2	ObjectId('660b074da36a396…	"Krish"	"Krish@123"	"user"	No field	No field
3	ObjectId('660b12731a7dc8f…	"Khushi Chauhan"	"1234"	"user"	No field	No field
4	ObjectId('660b27bd954a393…	"a"	"b"	"user"	No field	No field
5	ObjectId('660c3de26d3cfde…	"Ishan"	"3455"	"user"	No field	No field
6	ObjectId('660c43d8807f095…	"Vanika Chauhan"	"103"	"user"	No field	No field
7	ObjectId('660c4b0caf02134…	"Dhuruv Kumar"	"1234"	"user"	No field	No field
8	ObjectId('660c4ce87003dfc…	"David Roy"	"234506"	"user"	No field	No field
9	ObjectId('660c4f5de7e2bb3…	"admin "	"adminpass"	"user"	No field	No field
10	ObjectId('660fed1c3b2a909…	"Ac"	"1233"	"user"	"Khushi"	"12345"
11	ObjectId('660fed463b3a909…	"KhushiChauhan"	"123@34"	"user"	"Khushi Chauhan"	"20"