Homestay Booking System Using MongoDB

1. Collections Setup

Here are the MongoDB collections with the corresponding sample data.

1.1 RoomTypes Collection

```
db.roomTypes.insertOne({
  _id: ObjectId("6803ac07e87402dd1a86cb26"),
  typeName: "Suite",
  description: "Luxury suite with sea view, king bed, and jacuzzi.",
  maxGuests: 4,
  bedCount: 2,
  hasPrivateBathroom: true
});
```

Fields:

- _id: Unique ID for the room type.
- typeName: The name of the room type (e.g., Suite).
- description: A description of the room.
- maxGuests: Maximum number of guests for this room.
- bedCount: Number of beds in the room.
- hasPrivateBathroom: Boolean indicating if the room has a private bathroom.

1.2 Amenities Collection

```
db.amenities.insertOne({
  _id: ObjectId("6803ac076724780da3cfd021"),
  name: "WiFi",
  description: "High-speed wireless internet access",
  icon: "wifi"
```

});

Fields:

- _id: Unique ID for the amenity.
- name: The name of the amenity (e.g., WiFi).
- description: A description of the amenity.
- icon: The icon associated with the amenity.

1.3 Permissions Collection

```
db.permissions.insertOne({
  _id: ObjectId("6803ac079123982c8591c158"),
  permissionName: "ManageBookings",
  description: "Allows admin to create, modify, and cancel bookings."
});
```

Fields:

- _id: Unique ID for the permission.
- permissionName: Name of the permission (e.g., ManageBookings).
- description: Description of what the permission allows.

1.4 Users Collection

```
db.users.insertOne({
  _id: ObjectId("6803ac07427aca8a0f34a30a"),
  name: "Emily Johnson",
  email: "emily.johnson@gmail.com",
  phone: "+14155552678",
  address: "742 Evergreen Terrace, Springfield, IL",
  bookings: [ObjectId("6803ac07e0f5416a7cb56e0f")],
  preferredRoomType: "Suite",
  status: "active",
  dateJoined: new Date("2024-12-10"),
```

```
  lastLogin: new Date("2025-04-18T14:33:00Z")
});
```

Fields:

- _id: Unique ID for the user.
- name: Name of the user.
- email: Email address of the user.
- phone: Phone number of the user.
- address: Address of the user.
- bookings: Array of bookingId that the user has made.
- preferredRoomType: Preferred room type by the user.
- status: The status of the user's account (e.g., active).
- dateJoined: Date when the user joined.
- lastLogin: Date and time of the last login.

1.5 Hosts Collection

```
db.hosts.insertOne({
  _id: ObjectId("6803ac07e0ac5c7207b3bb41"),
  name: "Michael Brown",
  email: "michael.hosting@yahoo.com",
  phone: "+16175551234",
  location: "Miami Beach, FL",
  roomsAvailable: 3,
  rating: 4.8,
  pricePerNight: 150,
  roomTypes: [ObjectId("6803ac07e87402dd1a86cb26")],
  amenities: [ObjectId("6803ac076724780da3cfd021")],
  description: "Modern beachfront villa with pool and WiFi included.",
  status: "active",
  joinDate: new Date("2023-08-21"),
```

```
  reviews: [ObjectId("6803ac0712b747e7543c103a")]
});
```

Fields:

- _id: Unique ID for the host.
- name: Name of the host.
- email: Email address of the host.
- phone: Phone number of the host.
- location: Location of the host.
- roomsAvailable: Number of rooms available for booking.
- rating: Host's rating (e.g., 4.8).
- pricePerNight: Price per night for staying at the host's property.
- roomTypes: Array of ObjectId referencing the room types offered.
- amenities: Array of ObjectId referencing amenities available at the property.
- description: Host's property description.
- status: Host status (e.g., active).
- joinDate: Date the host joined the platform.
- reviews: Array of ObjectId referencing reviews given to the host.

1.6 Admins Collection

```
db.admins.insertOne({
  _id: ObjectId("6803ac0720b72f7b2b2f82f5"),
  username: "admin_jessica",
  email: "admin.jessica@homestay.com",
  role: "Admin",
  password: "hashed_pw_123456abcdef",
  permissions: [ObjectId("6803ac079123982c8591c158")],
  dateJoined: new Date("2022-05-10")
});
```

Fields:

- _id: Unique ID for the admin.
- username: Username of the admin.
- email: Email address of the admin.
- role: Role of the admin (e.g., Admin).
- password: Hashed password of the admin.
- permissions: Array of ObjectId referencing permissions granted to the admin.
- dateJoined: Date the admin joined the platform.

1.7 Bookings Collection

```
db.bookings.insertOne({
  _id: ObjectId("6803ac07e0f5416a7cb56e0f"),
  customerId: ObjectId("6803ac07427aca8a0f34a30a"),
  hostId: ObjectId("6803ac07e0ac5c7207b3bb41"),
  roomType: ObjectId("6803ac07e87402dd1a86cb26"),
  startDate: new Date("2025-04-25"),
  endDate: new Date("2025-04-29"),
  status: "booked",
  durationDays: 4,
  totalPrice: 600,
  specialRequests: "Need early check-in",
  paymentStatus: "paid",
  createdAt: new Date()
});
```

Fields:

- _id: Unique ID for the booking.
- customerId: Reference to the user who made the booking.
- hostId: Reference to the host offering the accommodation.
- roomType: Reference to the type of room booked.

- startDate: Start date of the booking.
- endDate: End date of the booking.
- status: Status of the booking (e.g., booked).
- durationDays: Number of days the booking lasts.
- totalPrice: Total price of the booking.
- specialRequests: Any special requests from the user.
- paymentStatus: Payment status of the booking.
- createdAt: Date and time when the booking was created.

1.8 Reviews Collection

```
db.reviews.insertOne({
  _id: ObjectId("6803ac0712b747e7543c103a"),
  userId: ObjectId("6803ac07427aca8a0f34a30a"),
  hostId: ObjectId("6803ac07e0ac5c7207b3bb41"),
  rating: 5,
  reviewText: "Absolutely stunning place! Host was very friendly and helpful.",
  reviewDate: new Date()
});
```

Fields:

- _id: Unique ID for the review.
- userId: Reference to the user who left the review.
- hostId: Reference to the host being reviewed.
- rating: Rating given by the user (e.g., 5).
- reviewText: Text of the review left by the user.
- reviewDate: Date when the review was submitted.

1.9 Transactions Collection

```
db.transactions.insertOne({
  _id: ObjectId("6803ac075b9837ad2ab60d91"),
```

```
  bookingId: ObjectId("6803ac07e0f5416a7cb56e0f"),
  customerId: ObjectId("6803ac07427aca8a0f34a30a"),
  amount: 600,
  paymentMethod: "Credit Card",
  paymentStatus: "successful",
  transactionDate: new Date()
});
```

Fields:

- _id: Unique ID for the transaction.
- bookingId: Reference to the booking related to the transaction.
- customerId: Reference to the customer making the payment.
- amount: Amount paid for the booking.
- paymentMethod: Method of payment (e.g., Credit Card).
- paymentStatus: Payment status (e.g., successful).
- transactionDate: Date and time when the transaction occurred.

1.10 Rooms Collection

```
db.rooms.insertOne({
  _id: ObjectId("6803ac072ee86d98f215edbe"),
  hostId: ObjectId("6803ac07e0ac5c7207b3bb41"),
  roomType: ObjectId("6803ac07e87402dd1a86cb26"),
  pricePerNight: 150,
  amenities: [ObjectId("6803ac076724780da3cfd021")],
  availability: "available",
  description: "King suite with ocean view, free breakfast, and smart TV."
});
```

Fields:

- _id: Unique ID for the room.
- hostId: Reference to the host offering the room.

- roomType: Reference to the room type.
- pricePerNight: Price per night for the room.
- amenities: Array of amenities available in the room.
- availability: Availability status of the room (e.g., available).
- description: Description of the room.

---

2. Utility Functions

Here are the utility functions in MongoDB for managing the homestay booking system.

2.1 Book a Room

```
function bookRoom(customerId, hostId, roomType, startDate, endDate, totalPrice) {
  var booking = db.bookings.insertOne({
    customerId: ObjectId(customerId),
    hostId: ObjectId(hostId),
    roomType: ObjectId(roomType),
    startDate: new Date(startDate),
    endDate: new Date(endDate),
    status: "booked",
    durationDays: (new Date(endDate) - new Date(startDate)) / (1000 * 60 * 60 * 24),
    totalPrice: totalPrice,
    specialRequests: "",
    paymentStatus: "pending",
    createdAt: new Date()
  });

  db.users.updateOne(
    { _id: ObjectId(customerId) },
    { $push: { bookings: booking.insertedId } }
```

```
  );

  db.hosts.updateOne(
    { _id: ObjectId(hostId) },
    { $inc: { roomsAvailable: -1 } }
  );
}
```

2.2 Cancel a Booking

```
function cancelBooking(bookingId) {
  var booking = db.bookings.findOne({ _id: ObjectId(bookingId) });

  db.bookings.updateOne(
    { _id: ObjectId(bookingId) },
    { $set: { status: "cancelled" } }
  );

  db.hosts.updateOne(
    { _id: ObjectId(booking.hostId) },
    { $inc: { roomsAvailable: 1 } }
  );

  db.users.updateOne(
    { _id: ObjectId(booking.customerId) },
    { $pull: { bookings: ObjectId(bookingId) } }
  );
}
```

2.3 Complete a Booking

```
function completeBooking(bookingId) {
  db.bookings.updateOne(
    { _id: ObjectId(bookingId) },
    { $set: { status: "completed" } }
  );
}
```

2.4 Vacate Room

```
function vacateRoom(bookingId) {
  var booking = db.bookings.findOne({ _id: ObjectId(bookingId) });

  db.bookings.updateOne(
    { _id: ObjectId(bookingId) },
    { $set: { status: "vacated" } }
  );

  db.hosts.updateOne(
    { _id: ObjectId(booking.hostId) },
    { $inc: { roomsAvailable: 1 } }
  );
}
```

2.5 Leave a Review

```
function leaveReview(userId, hostId, rating, reviewText) {
  var review = db.reviews.insertOne({
    userId: ObjectId(userId),
    hostId: ObjectId(hostId),
    rating: rating,
```

```
    reviewText: reviewText,
    reviewDate: new Date()
  });

  db.hosts.updateOne(
    { _id: ObjectId(hostId) },
    { $push: { reviews: review.insertedId } }
  );
}
```

---

3. Conclusion

This document includes the **MongoDB setup**, **sample data**, and **utility functions** for the homestay booking platform. You can use the provided MongoDB shell commands to create the necessary collections and insert the sample data into your MongoDB instance. The utility functions will help you manage bookings, users, hosts, reviews, and transactions.