Distributed House Rental System

PROJECT REPORT

Birgül Yüksel ER1922 | Big Data and Distributed Processing | 09.06.2025

Ingredients

1. Project Overview 2

2. System Architecture 2

3. Data Model (ER Diagram) 3

4. Functions and Screenshots 4

4.1 List all houses 4

4.2 LIST AVAILABLE HOUSES 5

4.3 Make a reservation 5

4.4 Cancel a reservation 6

4.5 *Search by keyword + price* 6

4.6 VIEW MY RESERVATIONS 7

4.7 VIEW CITY STATISTICS 7

4.8 UPDATE AN EXISTING RESERVATION 8

5. Stress Testing 8

6. Evaluation & Challenges 10

Strengths: 10

Challenges Encountered: 11

7. Conclusion 11

8. References 11

1. Project Overview

This project simulates a **distributed database system** for house rentals.
Each node represents a city (Istanbul, Ankara, Izmir), operating independently with local data.
A central client connects users to these nodes and performs all key reservation operations.

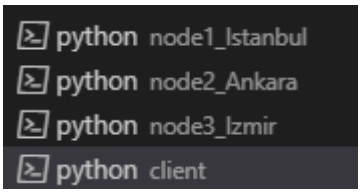
Main Functionalities:

- List and search available houses
- Reserve houses for specific dates
- Cancel or update existing reservations
- View reservation history
- View per-city statistics
- Stress test handling for concurrency

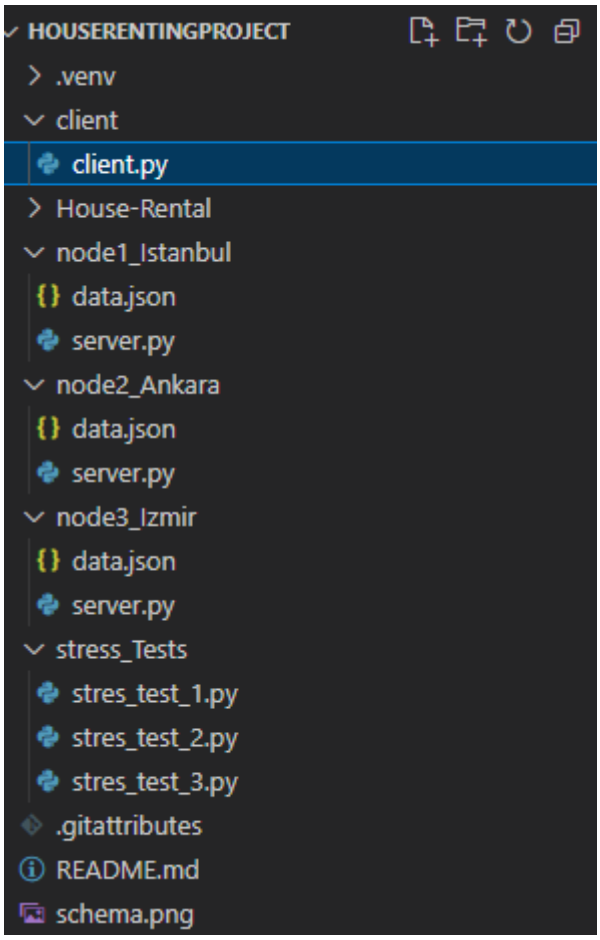
2. System Architecture

This system is composed of:

- **Three nodes (Flask servers)**:
  Each node stores and handles reservations for a specific city using its own data.json file.
- **Client application (CLI)**:
  A console-based interface connects to any of the nodes using REST API.
- **Stress tests**:
  Python scripts simulate concurrent access to test system resilience.



2 Terminals



1 Architecture

3. Data Model (ER Diagram)

Below is the Entity-Relationship diagram used for modeling the distributed system.

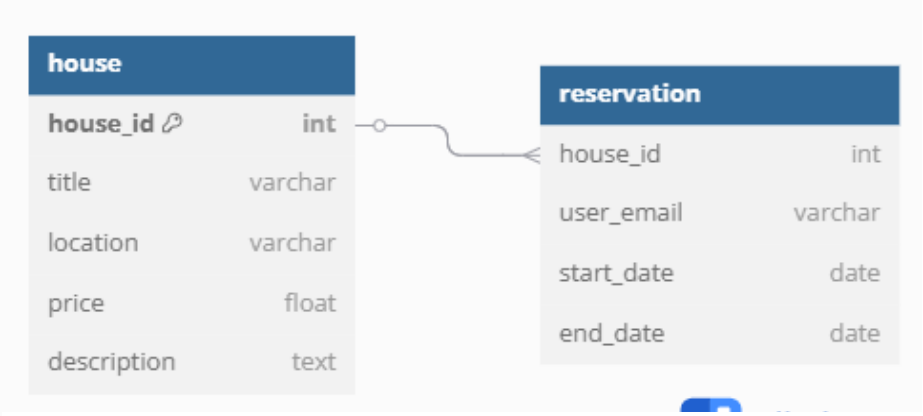
Tables:

House:

Field	Type	Description
house_id	int	Unique identifier
title	string	House title
location	string	City name
price	float	Monthly price
description	string	Text description

Reservation

Field	Type	Description
house_id	int	Linked to house (foreign key)
user_email	string	Who made the reservation
start_date	date	Reservation start date
end_date	date	Reservation end date



3 *Schema*

4. Functions and Screenshots

Please find below various screenshots of the client application in action.

4.1 LIST ALL HOUSES

```
====== Distributed House Rental System ======
1. List all houses
2. List available houses
3. Make a reservation
4. Cancel a reservation
5. Search for a house
6. View my reservations
7. View city statistics
8. Update an existing reservation
0. Exit
Choose an option: 1

Available Cities:
1. Istanbul
2. Ankara
3. Izmir
Choose a city: 1

🏠 Houses in Istanbul:
ID: 1 | 2 Bedroom Apartment | 5000₺ | Status: Reserved
    A cozy 2-bedroom apartment in a quiet neighborhood
    🔒 2025-06-10 → 2025-06-15 (by user1@example.com)
------------------------------------------------------------
ID: 2 | Studio Flat | 3500₺ | Status: Reserved
    Fully furnished studio, ideal for single tenants
    🔒 2025-07-10 → 2025-07-11 (by fastuser@example.com)
    🔒 2025-07-12 → 2025-07-13 (by fastuser@example.com)
    🔒 2025-07-14 → 2025-07-15 (by fastuser@example.com)
------------------------------------------------------------
ID: 3 | 3 Bedroom Duplex | 7500₺ | Status: Available
    Spacious duplex with balcony and sea view
------------------------------------------------------------
```

4 List all houses

4.2 LIST AVAILABLE HOUSES

```
====== Distributed House Rental System ======
1. List all houses
2. List available houses
3. Make a reservation
4. Cancel a reservation
5. Search for a house
6. View my reservations
7. View city statistics
8. Update an existing reservation
0. Exit
Choose an option: 2

Available Cities:
1. Istanbul
2. Ankara
3. Izmir
Choose a city: 1

● Available Houses in Istanbul:
ID: 3 | 3 Bedroom Duplex | 7500₺
    Spacious duplex with balcony and sea view
------------------------------------------------------------
ID: 4 | Furnished Loft | 6200₺
    Modern loft near the city center, fully equipped
------------------------------------------------------------
ID: 5 | 1 Bedroom Garden Flat | 4000₺
    Ground floor flat with access to private garden
------------------------------------------------------------
ID: 6 | Penthouse Suite | 11000₺
    Luxury penthouse with rooftop terrace and smart home features
------------------------------------------------------------
```

5 List Available Houses

4.3 MAKE A RESERVATION

```
====== Distributed House Rental System ======
1. List all houses
2. List available houses
3. Make a reservation
4. Cancel a reservation
5. Search for a house
6. View my reservations
7. View city statistics
8. Update an existing reservation
0. Exit
Choose an option: 3

Available Cities:
1. Istanbul
2. Ankara
3. Izmir
Choose a city: 3
Enter the house ID to reserve: 4
Enter your email: birgul@gmail.comi
Start date (YYYY-MM-DD): 2025-07-15
End date (YYYY-MM-DD): 2025-07-22
Reservation successful!
```

6 Make Reservation

4.4 CANCEL A RESERVATION

```
====== Distributed House Rental System ======
1. List all houses
2. List available houses
3. Make a reservation
4. Cancel a reservation
5. Search for a house
6. View my reservations
7. View city statistics
8. Update an existing reservation
0. Exit
Choose an option: 4

Available Cities:
1. Istanbul
2. Ankara
3. Izmir
Choose a city: 3
Enter the house ID to cancel reservation: 4
Enter your email: birgul@gmail.comi
Start date of the reservation (YYYY-MM-DD): 2025-07-15
End date of the reservation (YYYY-MM-DD): 2025-07-22
Reservation cancelled.
```

7 Cancel Reservation

4.5 SEARCH BY KEYWORD + PRICE

```
====== Distributed House Rental System ======
1. List all houses
2. List available houses
3. Make a reservation
4. Cancel a reservation
5. Search for a house
6. View my reservations
7. View city statistics
8. Update an existing reservation
0. Exit
Choose an option: 5

Available Cities:
1. Istanbul
2. Ankara
3. Izmir
Choose a city: 2
Enter keyword (e.g., 'studio', 'villa'): apartment
Max price (or leave empty): 7000

🔍 Search Results in Ankara:
ID: 1 | 1+1 Apartment in City Center | 4200₺ | Status: Available
    Modern apartment with great transportation access
------------------------------------------------------------
ID: 4 | Luxury 2+1 Apartment | 7000₺ | Status: Available
    Fully furnished with smart home features
------------------------------------------------------------
```

8 Search for a house

4.6 VIEW MY RESERVATIONS

```
====== Distributed House Rental System ======
1. List all houses
2. List available houses
3. Make a reservation
4. Cancel a reservation
5. Search for a house
6. View my reservations
7. View city statistics
8. Update an existing reservation
0. Exit
Choose an option: 6

Available Cities:
1. Istanbul
2. Ankara
3. Izmir
Choose a city: 1
Enter your email: user1@example.com

📋 Reservations for user1@example.com in Istanbul:
House ID: 1 | 2 Bedroom Apartment | 5000₺
From 2025-06-10 to 2025-06-15
------------------------------------------------------------
```

9 View Reservations

4.7 VIEW CITY STATISTICS

```
====== Distributed House Rental System ======
1. List all houses
2. List available houses
3. Make a reservation
4. Cancel a reservation
5. Search for a house
6. View my reservations
7. View city statistics
8. Update an existing reservation
0. Exit
Choose an option: 7

Available Cities:
1. Istanbul
2. Ankara
3. Izmir
Choose a city: 1

📊 Statistics for Istanbul:
Total houses: 6
Reserved: 2
Available: 4
```

10 View Statistics

4.8 UPDATE AN EXISTING RESERVATION

```
====== Distributed House Rental System ======
1. List all houses
2. List available houses
3. Make a reservation
4. Cancel a reservation
5. Search for a house
6. View my reservations
7. View city statistics
8. Update an existing reservation
0. Exit
Choose an option: 8

Available Cities:
1. Istanbul
2. Ankara
3. Izmir
Choose a city: 1
House ID of the reservation: 1
Your email: user1@example.com
Old start date (YYYY-MM-DD): 2025-06-10
Old end date (YYYY-MM-DD): 2025-06-15
New start date (YYYY-MM-DD): 2025-06-28
New end date (YYYY-MM-DD): 2025-07-04
Reservation updated successfully
```

11 Update Reservation

5. Stress Testing

☑ Test 1: Concurrent Reservations

Two users attempt to book the same house at the exact same time.
Result: One succeeds, the other is blocked (as expected). It is secured with “threading.Lock()”.

```
node1_Istanbul > {} data.json > {} 0
2       {
4           "title": "2 Bedroom Apartment",
5           "location": "Istanbul",
6           "price": 5000,
7           "description": "A cozy 2-bedroom apartment in a quiet neighborhood",
8           "reservations": [
9               {
10                  "user_email": "user1@example.com",
11                  "start_date": "2025-06-10",
12                  "end_date": "2025-06-15"
13              }
14          ]
15      },

PROBLEMS   OUTPUT   TERMINAL   PORTS   DEBUG CONSOLE

PS C:\Users\birgu\Desktop\HouseRentingProject\stress_Tests> python stres_test_1.py
[user1@example.com] → Reservation successful!
[user2@example.com] → House is already booked for selected dates.
PS C:\Users\birgu\Desktop\HouseRentingProject\stress_Tests>
```

12 Stress Test 1 Result

☑ Test 2: Rapid-fire bookings

A user sends multiple booking requests rapidly for different dates.
Result: System accepts them without collision or failure.

```
node1_Istanbul > {} data.json > {} 1
16        {
18            "title": "Studio Flat",
19            "location": "Istanbul",
20            "price": 3500,
21            "description": "Fully furnished studio, ideal for single tenants",
22            "reservations": [
23                {
24                    "user_email": "fastuser@example.com",
25                    "start_date": "2025-07-10",
26                    "end_date": "2025-07-11"
27                },
28                {
29                    "user_email": "fastuser@example.com",
30                    "start_date": "2025-07-12",
31                    "end_date": "2025-07-13"
```

```
PROBLEMS    OUTPUT    TERMINAL    PORTS    DEBUG CONSOLE

        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
KeyboardInterrupt
PS C:\Users\birgu\Desktop\HouseRentingProject\stress_Tests> python stres_test_2.py
Attempt 1: Reservation successful!
Attempt 2: Reservation successful!
Attempt 3: Reservation successful!
Attempt 4: Reservation successful!
Attempt 5: Reservation successful!
PS C:\Users\birgu\Desktop\HouseRentingProject\stress_Tests>
```

13 Stress Test 2 Result

☑ Test 3: Repeated reserve-cancel cycle

A user continuously reserves and cancels the same house.
Result: No data corruption or crash observed.

```
PS C:\Users\birgu\Desktop\HouseRentingProject\stress_Tests> python stres_test_3.py

🔁 Loop 1: Reserving...
→ Reservation successful!
🔁 Cancelling...
→ Reservation cancelled.

🔁 Loop 2: Reserving...
→ Reservation successful!
🔁 Cancelling...
→ Reservation cancelled.

🔁 Loop 3: Reserving...
→ Reservation successful!
🔁 Cancelling...
→ Reservation cancelled.

🔁 Loop 4: Reserving...
→ Reservation successful!
🔁 Cancelling...
→ Reservation cancelled.

🔁 Loop 5: Reserving...
→ Reservation successful!
🔁 Cancelling...
→ Reservation cancelled.
```

14 Stress Test 3 Result

☑ Test 4: Fair Distribution of Reservations Across Clients

Two users attempt to reserve all available houses at the exact same time.
Each house is targeted by both clients in parallel, simulating a competitive booking scenario.
Result: Reservations were distributed between the two clients.
No user was able to take all houses, and each client successfully secured at least one reservation.
This confirms that the system ensures **fair access** and avoids monopolization, even under simultaneous load.

```
PS C:\Users\birgu\Desktop\HouseRentingProject\stress_Tests> python stres_test_4.py
[client1@example.com] tried house 2: Reservation successful!
[client2@example.com] tried house 1: Reservation successful!
[client2@example.com] tried house 3: Reservation successful!
[client2@example.com] tried house 2: House is already booked for selected dates.
[client1@example.com] tried house 1: House is already booked for selected dates.
[client1@example.com] tried house 3: House is already booked for selected dates.
```

15 Stres Test 4 Result

6. Evaluation & Challenges

STRENGTHS:

- Fully distributed structure (each node operates independently)
- Robust conflict handling for reservations
- User-friendly CLI client
- Clear modular architecture

Main Strength: Distributed, Date-Aware Reservation System

The main strength of this project is its realistic and scalable reservation system, which combines date-aware logic with a distributed architecture.
Instead of relying on a simple "reserved/available" status, each house maintains a list of reservations with start_date and end_date values. This enables:

- Accurate availability tracking over time
- Multiple non-overlapping reservations per house
- Cancellation and updates without affecting others

Each node (Istanbul, Ankara, Izmir) runs independently, storing local data and handling all operations via its own Flask server. The client interacts with any of them through a shared interface, making the system feel unified while maintaining full decentralization.

With robust conflict detection, thread-safe reservation handling, and stress-tested concurrency, the system ensures both data integrity and real-world usability — making it a strong example of distributed application design.

CHALLENGES ENCOUNTERED:

- Implementing date-based conflict detection
- Updating reservation structure from simple status to reservation arrays
- Synchronizing client features with new data model

Main Challenge: Race Conditions in Concurrent Reservations

One of the key challenges encountered was handling **race conditions** during concurrent reservations.

Initially, when two users attempted to reserve the same house for the same date at the same time (as simulated in stres_test_1.py), both were able to succeed.
This happened because both threads read the JSON file **before either had written**, so the house appeared "available" to both.

Solution: Thread Locking (Critical Section)

To solve this, a **thread-level lock** (threading.Lock) was implemented in the /reserve endpoint.
The entire reservation block was enclosed in a with lock: statement to ensure **atomicity** — meaning only one thread can reserve a house at a time.

✅ Result

After adding the lock, one reservation succeeds while the other receives a 409 Conflict, as expected.

7. Conclusion

The project meets all the expected outcomes for a distributed database system:

- Multiple nodes with local data
- Full CRUD operations across nodes
- Robustness under load and concurrent access

8. References

- Flask documentation
- Python requests & threading
- dbdiagram.io
- Course lectures and notes