

Luna Urban Apartment

Rebecca Zhang
Yu Lu
Yvette Zhu



Description of Luna

Luna Apartment, located in Montreal, offers an elevated rental experience. Just steps away from McGill University and downtown Montreal, it combines the best of campus and city life. With a choice of stylish suites ranging from one to three bedrooms, Luna offers plenty of space for study and relaxation. With a wide array of facilities, a 24-hour security, on-site professional management, and diverse events, Luna redefines the concept of campus living.

Develop a brand new chatbot to enhance user experience and elevate management efficiency

Interact with the Chatbot–Hello! How can I help you today?

01.

**Scheduling for
apartment tour**

02.

**Property facility
maintenance**

03.

**Housekeeping
appointment**

04.

Delivery management

Mock Data

- **DataFrame Columns:** Unit number, floor plan, square feet, rent, tenant name, DOB, job, lease start date.
- **Building Structure:** 16 floors, 20 rooms/floor.
- **Floor Plans:**
 - 1b1b, 550 sq.ft, \$1600 base rent.
 - 1b1b, 750 sq.ft, \$1750 base rent.
 - 2b1b, 775 sq.ft, \$1900 base rent.
 - 2b2b, 950 sq.ft, \$2200 base rent.
- **Rent Increment:** 1% for each floor up.
- **Room Distribution:** Each floor has 5 rooms of each floor plan.
- **Availability:** 20 rooms currently vacant.

Faker: a Python package that generates fake data

```
from faker import Faker
faker = Faker()
Faker.seed(331)
faker.name()
faker.job()
```

Scenario 1: Scheduling for apartment tour



User Preference Collection:

- Inquiring about occupants and budget.
- Obtaining the list of vacant units from existing data.

Displaying Options:

- Filtering and showcasing suitable apartments in detail.
- Users select their preferred type.

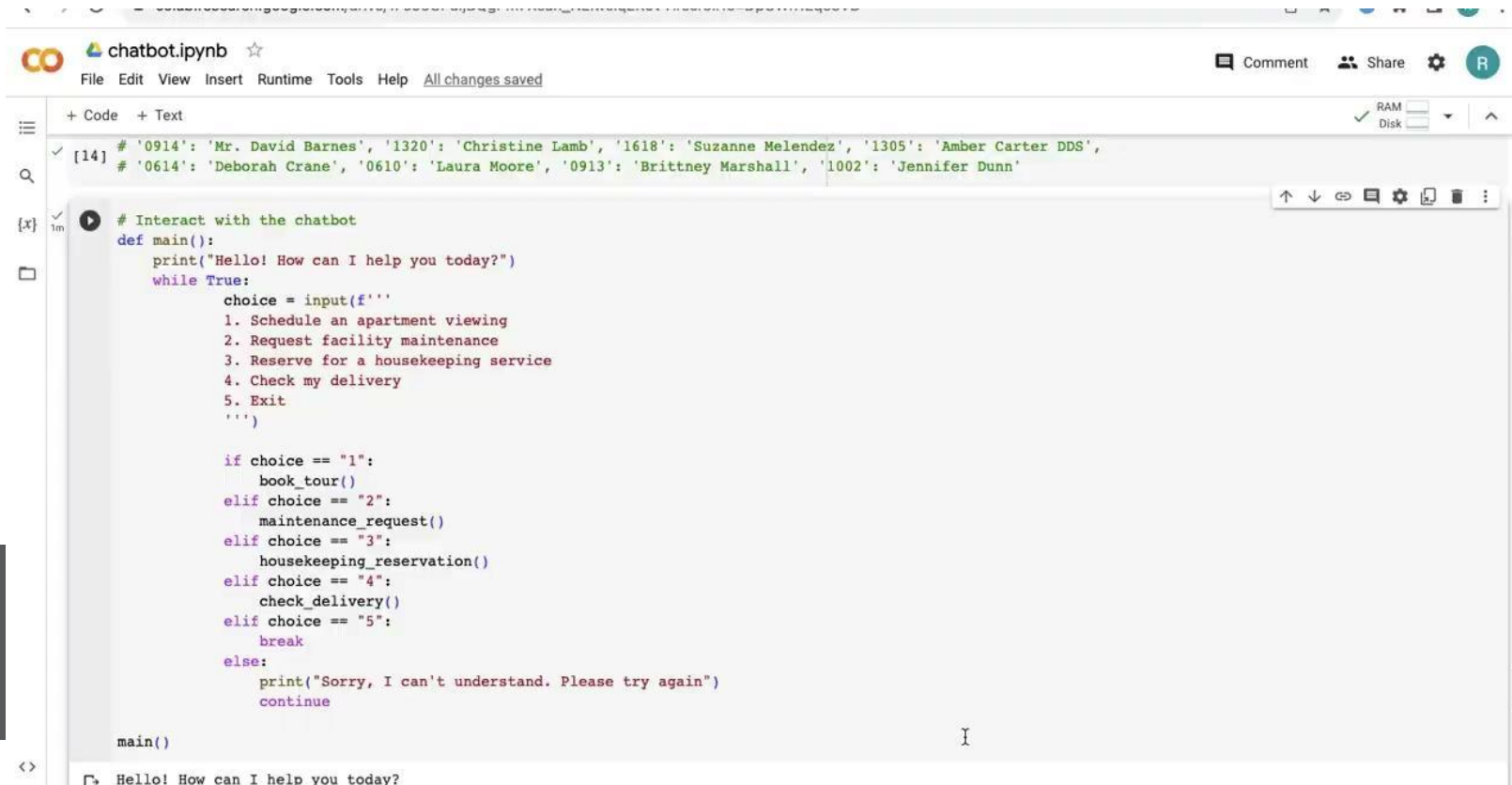
Tour Scheduling:

- Listing available dates and times.
- Confirming the scheduled tour with details.

Data Collection & Analysis:

- Recording all bookings.
- Providing insights into renter preferences and popular tour times.

Video of scenario 1



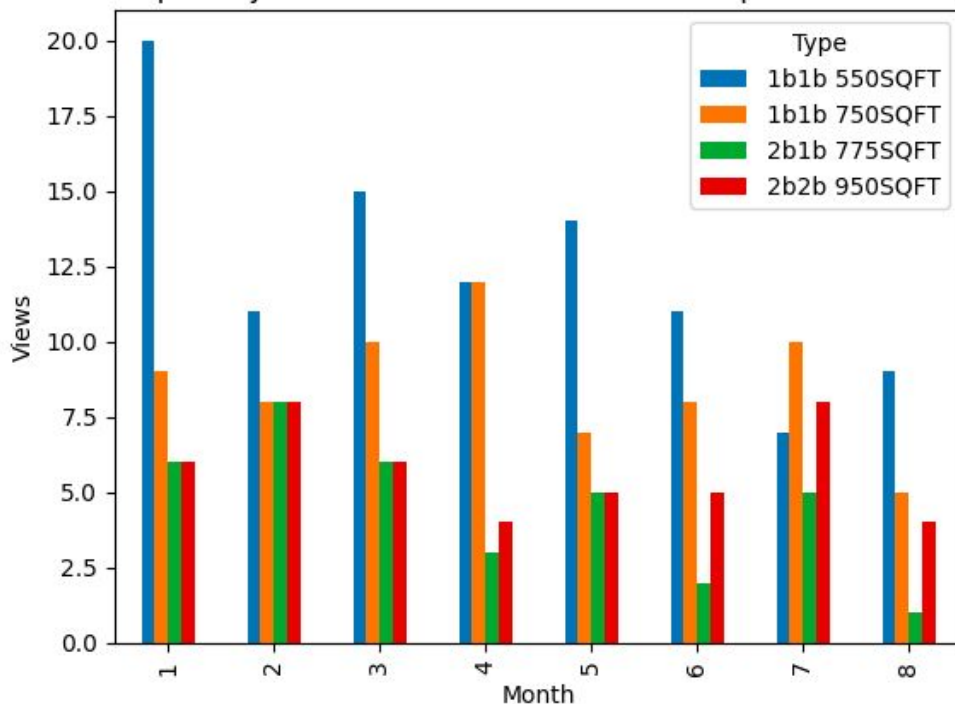
The screenshot shows a Jupyter Notebook interface with the title 'chatbot.ipynb'. The top bar includes a menu (File, Edit, View, Insert, Runtime, Tools, Help) and a status indicator 'All changes saved'. On the right, there are buttons for 'Comment', 'Share', and a settings icon. Below the menu, the notebook is in 'Code' mode. The code is as follows:

```
[14] # '0914': 'Mr. David Barnes', '1320': 'Christine Lamb', '1618': 'Suzanne Melendez', '1305': 'Amber Carter DDS',  
# '0614': 'Deborah Crane', '0610': 'Laura Moore', '0913': 'Brittney Marshall', '1002': 'Jennifer Dunn'  
  
# Interact with the chatbot  
def main():  
    print("Hello! How can I help you today?")  
    while True:  
        choice = input(f'''  
        1. Schedule an apartment viewing  
        2. Request facility maintenance  
        3. Reserve for a housekeeping service  
        4. Check my delivery  
        5. Exit  
        ''')  
  
        if choice == "1":  
            book_tour()  
        elif choice == "2":  
            maintenance_request()  
        elif choice == "3":  
            housekeeping_reservation()  
        elif choice == "4":  
            check_delivery()  
        elif choice == "5":  
            break  
        else:  
            print("Sorry, I can't understand. Please try again")  
            continue  
  
    main()
```

At the bottom of the notebook, the output of the first cell is visible: 'Hello! How can I help you today?'

Popularity Analysis

Popularity of Different Floor Plans in Luna Apartment 2023



Data Collection

- Chatbot recorded bookings

Data Preprocessing

- Date -> Month
- Merge with apartment data
- Group by unit type

Data Analysis

- Matplotlib - visualization

Insights Sharing

- 1b1b 550 SQFT - most popular
- 2b1b - least popular
- Predict # tours - time series

Scenario 2: Property facility maintenance



User Issue Reporting

- Residents describe their maintenance issue.
- Chatbot captures the problem details.



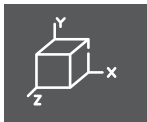
Permission Inquiry

- Asks if staff can enter without the resident.
- Residents specify their preference.



Maintenance Scheduling

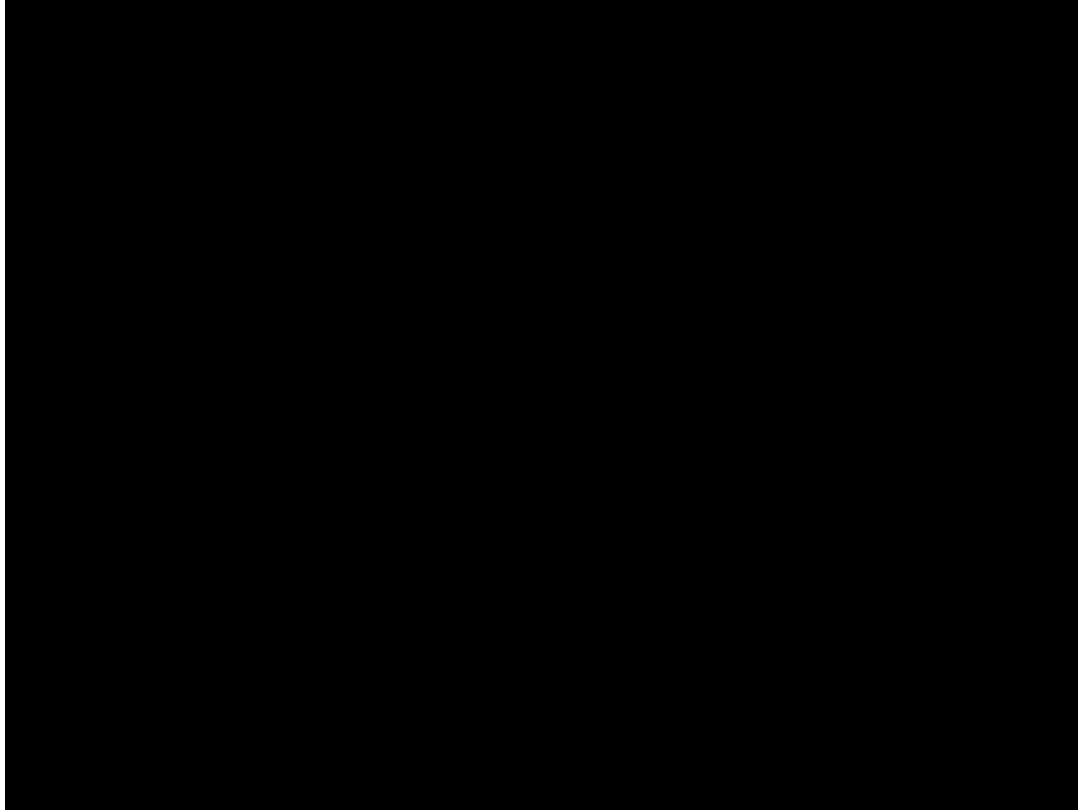
- If resident wants to be present:
 - a. Chatbot lists available dates and times.
 - b. Confirms the scheduled maintenance with details.



Efficiency & Privacy Assurance

- Streamlines maintenance scheduling.
- Ensures respect for resident's privacy and preferences.

Video of scenario 2



Scenario 3: Housekeeping appointment



User Information Collection:

- Identity the unit number which determines the price range

Displaying Options:

- Choose specific services from carpet cleaning, window cleaning and bathroom cleaning
- User select their preferred time slot

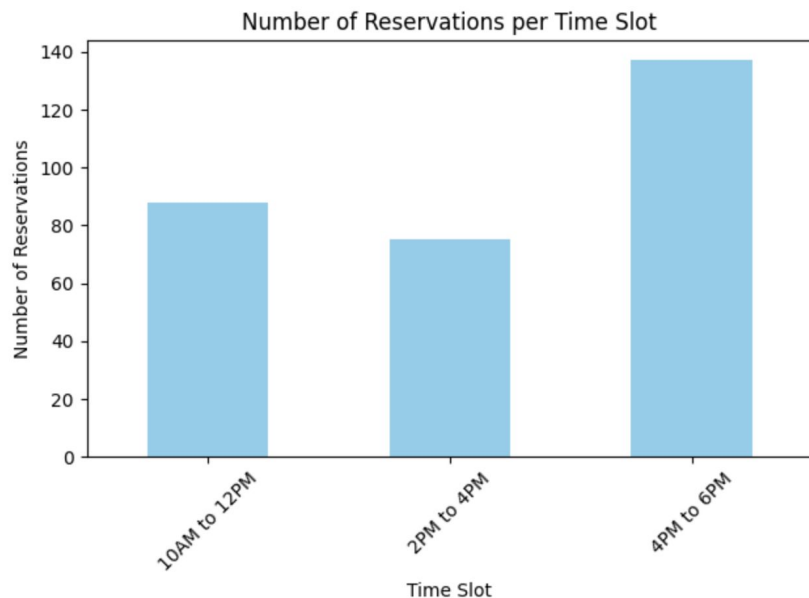
Availability Check & Reservation:

- Checks service availability in real-time
- Confirms the reservation and displays the price if available

Data Analysis & visualization:

- Conduct a bar chart showing reservation across time slots
- Understand the distribution to achieve better resource allocation.

Housekeeping Analysis



Rationale for library: matplotlib

- Create interactive visualizations by making plots and charts

Observation

- High demand during the 4PM and 6PM slot around 140 reservations
- Lowest demand from 2PM to 4PM around 70 observations

Suggestions:

- Hire part-time staff for peak hours
- Introduce promotion offers for the 2PM to 4PM slot

Video of scenario 3



The screenshot shows a code editor with a Python script for a chatbot. The script is titled "# Interact with the chatbot" and defines a `main()` function. The function prints a greeting and enters a `while True` loop. Inside the loop, it prompts the user for a choice from a list of options: 1. Schedule an apartment viewing, 2. Request facility maintenance, 3. Reserve for a housekeeping service, 4. Check my delivery, and 5. Exit. The script uses `if` and `elif` statements to handle each choice, calling corresponding functions like `book_tour()`, `maintenance_request()`, `housekeeping_reservation()`, and `check_delivery()`. If the user enters an invalid choice, it prints "Sorry, I can't understand. Please try again" and continues the loop. The script ends with a call to `main()`.

```
# Interact with the chatbot
def main():
    print("Hello! How can I help you today?")
    while True:
        choice = input(f'''
        1. Schedule an apartment viewing
        2. Request facility maintenance
        3. Reserve for a housekeeping service
        4. Check my delivery
        5. Exit
        ''')

        if choice == "1":
            book_tour()
        elif choice == "2":
            maintenance_request()
        elif choice == "3":
            housekeeping_reservation()
        elif choice == "4":
            check_delivery()
        elif choice == "5":
            break
        else:
            print("Sorry, I can't understand. Please try again")
            continue

main()
```

Below the code editor, the output of the program is shown in a terminal window. It displays the greeting "Hello! How can I help you today?" followed by a list of options: 1. Schedule an apartment viewing, 2. Request facility maintenance, 3. Reserve for a housekeeping service, 4. Check my delivery, and 5. Exit.

```
Hello! How can I help you today?
1. Schedule an apartment viewing
2. Request facility maintenance
3. Reserve for a housekeeping service
4. Check my delivery
5. Exit
```

Scenario 4: Delivery management



Tenant Verification:

- Asks for the tenant's unit number.
- Requests name for identity confirmation if a package is available.



Package Delivery Options:

- Offers choice to place the package at the door.
- Provides option for secure pickup at the doorman.



Secure Pickup Code Generation:

- Generates a unique code for pickup at doorman.
- Ensures added security for package retrieval.



Error Handling & Feedback:

- Informs tenant of identification mismatches.
- Notifies if no package is available for the given unit.

Video of scenario 4



```
+ Code + Text

43s # Interact with the chatbot
def main():
    print("Hello! How can I help you today?")
    while True:
        choice = input(f'''
            1. Schedule an apartment viewing
            2. Request facility maintenance
            3. Reserve for a housekeeping service
            4. Check my delivery
            5. Exit
            ''')

        if choice == "1":
            book_tour()
        elif choice == "2":
            maintenance_request()
        elif choice == "3":
            housekeeping_reservation()
        elif choice == "4":
            check_delivery()
        elif choice == "5":
            break
        else:
            print("Sorry, I can't understand. Please try again")
            continue

    main()

Hello! How can I help you today?

1. Schedule an apartment viewing
2. Request facility maintenance
3. Reserve for a housekeeping service
```

OUR CHATBOT STRENGTHS



Database Management

- Created dataframes to keep track of apartment, tenants, bookings, delivery information
- Easier to find, retrieve, update and analyze relevant data.



User-centric Experience

- Streamline the process of booking apartment tours
- enhance the convenience of requesting service
- ensure additional security for package pickups



Efficient Management

- Better resource allocation
- Enable quicker response times, 24/7 availability
- Reduce workload on management



Valuable Insights

- Identify tenant preference
- Adjust pricing strategy
- Forecast demand
- Identify peak demand time for service

Extending Our Work – Tenant Clustering for Enhanced Insights

- **Data Collection & Preprocessing:**

Gather tenant data: number of people, budget, annual income, pet ownership, and work location distance.

Standardize data to ensure consistent and uniform format of each feature.

- **Tenant Segmentation:**

Employ K-Means algorithm to segment tenants into best K distinct clusters.

Examine cluster centroids to understand average tenant characteristics.

- **Visualization & Analysis:**

Showcase tenant clusters using a 2D scatter plot.

Highlight cluster centroids for a clear representation of average profiles.

- **Potential Insights:**

Understand diverse tenant profiles for better service tailoring.

Identify patterns and preferences among tenant groups.

Develop different promotional and marketing strategies for different tenant clusters.

Appendix – Improvements after Discussion

Feedback

- In scenario 1, when there's no appropriate units for viewing (for example because the budget is too low), the chatbot still show available time and asks for the user to schedule a showing.
- there wasn't much data analysis part.

Improvements

- Stop execution of scenario 1 if there's no appropriate units for viewing
- Add analysis about popularity of different floor plans over time and housekeeping time slot analysis.
- Add a main function to integrate all the scenarios.



Thank You