BLUEBIRD HOTEL MANAGEMENT SYSTEM

CSE SECTION B – BATCH 2 - PROJECT SUBMISSION

Team Members:

1. Shivang Gulati – 220905264 – 38
2. Devansh Desai - 220905368 – 43
3. Tejeswar Pokuri - 220905236 – 33
4. Gaurav Nayak H - 220905388 - 44

Bluebird Hotel Booking Software Development Lifecycle

Introduction

The Bluebird Hotel Booking Software is a comprehensive platform designed to facilitate hotel booking management. It offers a range of features for both users and staff, including room booking, payment management, data analytics, and a chatbot for user assistance. The software was developed using the Scrum methodology, allowing for incremental progress and iterative improvements. Weekly sprints were conducted, and updates were pushed to GitHub for version control.

Project Overview

The Bluebird software supports two primary user types:
1. Users - who can book rooms, make payments, and communicate via the chatbot.
2. Staff - who can manage rooms, view bookings, and handle payment records.
Additionally, data analytics and security features enhance the software's utility and reliability.

Scrum Model Development Process

The Scrum model was used for the development of the Bluebird Hotel Booking Software. The project was divided into several sprints, each lasting one week. Each sprint included the phases of planning, designing, development, and testing.

1. Requirements Gathering

In the initial phase, requirements for the software were gathered. Features like room booking, payment processing, data analytics, staff management, and an AI chatbot were outlined. User stories were created to represent functionalities from the perspectives of different types of users.

2. Sprint Planning

During each sprint planning session, tasks were broken down into manageable parts. Each team member was assigned tasks based on their expertise, ensuring a balanced workload. Key tasks were prioritized to ensure essential functionalities were developed first.

3. Design

The design phase included creating wireframes and prototypes for the user interface. Key design elements included a dashboard, room management, payment section, and an analytics view. The AI chatbot interface was also designed to provide user assistance.

4. Development

In the development phase, coding was done using an iterative approach. Features such as room booking, payment processing, staff management, and chatbot integration were implemented. GitHub was used for version control, and code was reviewed before merging.

5. Testing

Each sprint included a testing phase to ensure that newly developed features functioned correctly. Unit tests, integration tests, and user acceptance tests were conducted. Bugs were identified and resolved promptly to maintain software quality.

6. Deployment

After successful testing, the software was deployed to a staging environment for final testing. Once all tests were passed, the software was deployed to the production environment, allowing users to access its features.

7. Maintenance

Post-deployment, the software entered the maintenance phase. Regular updates and bug fixes are conducted as needed. The team continues to monitor performance and gather user feedback to make further improvements.

Feature Overview

1. User and Staff Logins - Separate login interfaces for users and staff.
2. Room Booking - Users can browse rooms, check availability, and make bookings.
3. Payment Processing - Secure payment options integrated with booking.
4. Data Analytics - Staff can view booking statistics and revenue reports.
5. AI Chatbot - Provides assistance with booking and general inquiries.
6. Security Features - Ensures user data protection and access control.

Conclusion

The Bluebird Hotel Booking Software was developed using the Scrum model to ensure a flexible and collaborative approach. Through weekly sprints, the development team implemented and tested features, adapting to changes as required. The result is a user-friendly, efficient, and secure hotel management platform that meets the needs of both users and hotel staff.

Software Requirements Specification (SRS) for Hotel Management Software

1. Introduction

1.1 Purpose

The purpose of this document is to outline the requirements and specifications for a hotel management software. The software aims to streamline hotel operations for both administrators and users by managing room bookings, staff, payments, data analytics, and security.

1.2 Scope

The software provides functionalities for two user roles: admin and user. Admins can manage rooms, staff, bookings, payments, and analytics, while users can view facilities, book rooms, and access a chatbot for assistance.

1.3 Definitions, Acronyms, and Abbreviations

- Admin: Authorized hotel staff with access to management functions.
- User: Hotel customers who can browse facilities and book rooms.
- YOLO: "You Only Look Once" - a real-time object detection model used for security surveillance.

2. System Overview

The hotel management software is a web-based application designed to handle various aspects of hotel operations and guest services, providing efficient, real-time management of rooms, bookings, payments, and security.

3. Functional Requirements

3.1 User Management

3.1.1 User Login
Description: Allows users to log in with valid credentials.
Input: Username, password.
Output: User access to user-specific features.

3.1.2 Admin Login
Description: Allows admins to log in with secure credentials.
Input: Username, password.
Output: Admin access to management dashboard.

3.2 Admin Functionalities

3.2.1 Dashboard
Description: Provides an overview of hotel status, including current bookings, occupancy rate, and financial summary.
Input: None (post-login).
Output: Display of dashboard data for quick insights.

3.2.2 Room Management

3.2.2.1 Add Room:
Description: Allows admin to add new rooms to the system.
Input: Room details (number, type, price, status).
Output: Room added to the available list.

3.2.2.2 Delete Room:
Description: Enables admin to remove rooms from the system.
Input: Room ID or number.
Output: Room removed from the list.

3.2.3 Staff Management

3.2.3.1 Add Staff:
Description: Allows admin to add new staff members.
Input: Staff details (name, role, contact information).
Output: New staff added to the database.

3.2.3.2 Delete Staff:
Description: Enables admin to remove staff members.
Input: Staff ID or name.
Output: Staff member deleted from the database.

3.2.4 Booking Management

3.2.4.1 View Bookings:
Description: Displays all current and past bookings.
Input: None.
Output: List of all bookings.

3.2.4.2 Confirm Booking:
Description: Allows admin to confirm or reject a booking.
Input: Booking ID.
Output: Booking status updated.

3.2.4.3 Download Bookings (CSV):
Description: Exports booking data into a CSV file.

Input: None.
Output: CSV file with booking data.

3.2.5 Payment Management

3.2.5.1 Download Invoice:
Description: Enables admin to download an invoice for any booking.
Input: Booking ID.
Output: PDF invoice file.

3.2.6 Data Analytics
Description: Provides data insights on bookings, occupancy rate, revenue, etc.
Input: Selected analysis type.
Output: Data visualization or report.

3.2.7 Security Surveillance
Description: Uses YOLO for real-time security monitoring of hotel premises.
Input: Security camera feed.
Output: Detection of objects or people for monitoring purposes.

3.3 User Functionalities

3.3.1 Facility Viewing
Description: Allows users to view available facilities in the hotel.
Input: None.
Output: Display of facility information.

3.3.2 Chatbot
Description: A chatbot to assist users with queries.
Input: User queries.
Output: Chatbot responses.

3.3.3 Room Booking
Description: Allows users to book available rooms.
Input: Room choice, booking dates, personal information.
Output: Booking confirmation.

4. Non-Functional Requirements

4.1 Usability
Description: The software should be user-friendly, with intuitive navigation and clear labeling of features.

4.2 Reliability
Description: The software should ensure data integrity and be capable of handling multiple simultaneous bookings and admin actions without errors.

4.3 Performance
Description: The software must respond to user actions within 2 seconds, except for analytics and reporting, which may take up to 5 seconds.

4.4 Security
Description: All admin and user data must be secured with encryption. Admin functionalities should be restricted by role-based access controls.

4.5 Availability
Description: The system should maintain 99.9% uptime to ensure availability for booking and management at all times.

4.6 Scalability
Description: The system should support the addition of new rooms, staff, and facilities with minimal configuration adjustments.

4.7 Maintainability
Description: The system should be modular to allow updates and maintenance on specific modules (e.g., booking, analytics) without impacting the entire system.

5. System Interface Requirements

5.1 User Interface

Login Pages for both user and admin.
Admin Dashboard: with tabs for rooms, staff, bookings, payments, analytics, and surveillance.
User Interface: Simple navigation with easy access to facilities, booking, and chatbot.

5.2 Hardware Interfaces

Security Surveillance: Requires connection to security cameras capable of integrating with the YOLO model for object detection.

5.3 Software Interfaces

Database: MySQL or PostgreSQL database to store all data, including bookings, room details, and staff.
Payment Gateway: Integration with a secure payment processor for handling transactions.
Analytics: Connection to a data visualization library for generating insights.

Hotel Management System Structure

The Hotel Management System comprises the following main modules, each responsible for a specific set of operations:

1. User Management (B)

Description: Handles all operations related to staff and guest management, including authentication and user data management.
Subcomponents:

Staff Management (B1)

B11. Add Staff: Adds new staff members.

B12. Update Staff: Modifies details of existing staff members.

B13. Delete Staff: Removes staff members from the system.

Guest Management (B2)

B21. Register Guest: Registers a new guest in the system.

B22. Update Guest Info: Updates existing guest information.

Authentication (B3)

Manages user login, logout, and access control.

2. Room Management (C)

Description: Manages room operations, configuration, and status.
Subcomponents:

Room Operations (C1)

C11. Add Room: Adds a new room to the inventory.

C12. Update Room: Updates room information.

C13. Delete Room: Removes a room from the inventory.

Room Configuration (C2)

C21. Set Room Type: Defines the type of room (e.g., single, double, suite).

3. Reservation System (D)

Description: Oversees the reservation lifecycle, including check-in and check-out.
Subcomponents:

Check-in (D1)

D11. Verify Booking: Verifies reservation details at check-in.

D12. Generate Token: Generates unique tokens for guest identification.

Check-out (D2)

Manages guest check-out and updates room availability.

Reservation Operations (D3)

D31. Make Reservation: Creates a new reservation.

D32. Cancel Reservation: Cancels an existing reservation.

D33. Modify Reservation: Allows changes to existing reservations.

4. Billing System (E)

Description: Processes all financial transactions, including room and additional services charges.
Subcomponents:

Room Charges (E1)

Calculates charges based on room type and stay duration.

Food Charges (E2)

Manages charges for ordered food.

Additional Services (E3)

Manages charges for other services (e.g., spa, laundry).

5. Analytics & Reporting (G)

Description: Provides insights and reports on occupancy, revenue, and guest feedback.
Subcomponents:

Occupancy Reports (G1)

G11. Daily Analysis: Analyzes daily occupancy trends.

G12. Monthly Analysis: Generates monthly occupancy reports.

G13. Seasonal Trends: Analyzes seasonal occupancy patterns.

Revenue Reports (G2)

G21. Room Revenue: Reports on revenue from room bookings.

G22. Food Revenue: Reports on revenue from food orders.

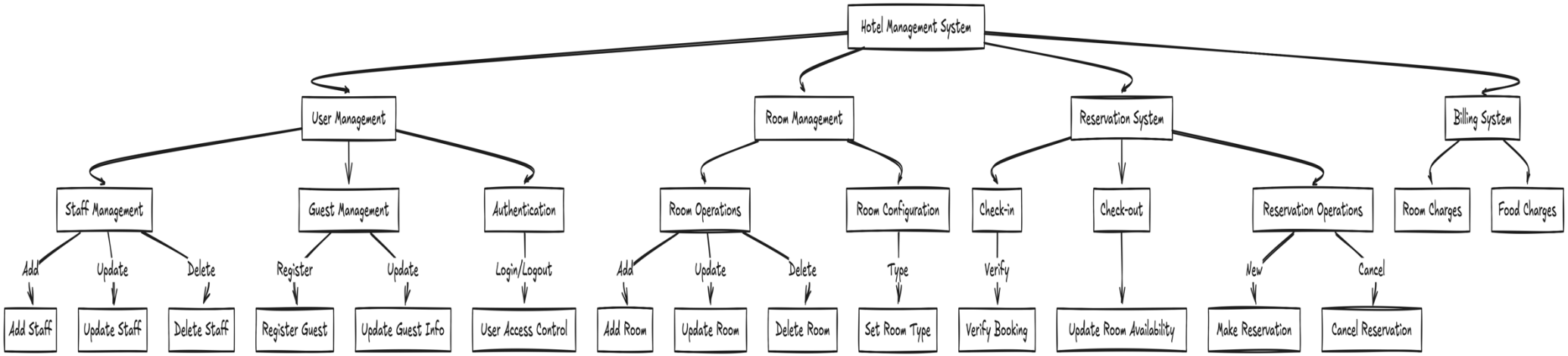
G23. Total Revenue: Aggregates total revenue from all sources.

Guest Feedback (G3)

Collects and analyzes feedback for service improvement.

Summary

Each module is designed to handle a specific function within the Hotel Management System, ensuring efficient management and reporting of hotel operations. This structured chart helps in understanding the hierarchy and relationships among components, facilitating smoother operations and improved guest services.


Hotel Management System
User Management
Room Management
Reservation System
Billing System
Staff Management
Guest Management
Authentication
Room Operations
Room Configuration
Check-in
Check-out
Reservation Operations
Room Charges
Food Charges
Add
Update
Delete
Register
Update
Login/Logout
Add
Update
Delete
Type
Verify
New
Cancel
Add Staff
Update Staff
Delete Staff
Register Guest
Update Guest Info
User Access Control
Add Room
Update Room
Delete Room
Set Room Type
Verify Booking
Update Room Availability
Make Reservation
Cancel Reservation

Object-Oriented Design for Bluebird Hotel Booking Software

Introduction

This document outlines the Object-Oriented Design (OOD) of the Bluebird Hotel Booking Software. The design is based on key object-oriented principles, including encapsulation, inheritance, and polymorphism. The system is structured with classes representing different components and functionalities, allowing for efficient code management and feature extension.

Class Diagram Overview

The main classes in the Bluebird software design are User, Staff, Room, Booking, Payment, Analytics, and Chatbot. Each class is designed with specific attributes and methods to encapsulate the responsibilities of the components they represent. The relationships between these classes follow the principles of inheritance and composition to ensure a well-organized system structure.

Class: User

Represents a general user of the system, including guests booking rooms.

Attributes:

- - user_id: Unique identifier for the user
  - name: Name of the user
  - email: Email address
  - phone: Contact number
  - country: Country of residence

Methods:

- - login(): Allows the user to log in
  - view_rooms(): Enables the user to browse available rooms
  - make_booking(): Allows the user to book a room

Class: Staff (Inherits User)

Represents staff members who have additional privileges over general users.

Attributes (Inherited from User):

- - role: The role of the staff member (e.g., manager, cleaner, cook)
  - staff_id: Unique identifier specific to staff

Methods:

- - manage_rooms(): Allows staff to add, edit, and delete rooms
  - view_analytics(): Enables staff to view system analytics

Class: Room

Represents a hotel room available for booking.

Attributes:

- - room_id: Unique identifier for the room
  - type: Type of room (e.g., Superior, Deluxe, Single)
  - bed_type: Type of bed (e.g., Single, Double, Triple)
  - status: Availability status (e.g., available, booked)

Methods:

- - check_availability(): Checks if the room is available
  - update_status(): Updates the room's availability status

Class: Booking

Handles the booking details for a user.

Attributes:

- - booking_id: Unique identifier for the booking
  - user_id: Identifier linking to the User who made the booking
  - room_id: Identifier linking to the Room booked
  - check_in_date: Date of check-in
  - check_out_date: Date of check-out
  - meal_plan: Chosen meal plan

Methods:

- - confirm_booking(): Confirms and saves booking details
  - cancel_booking(): Cancels an existing booking

Class: Analytics

Provides data analysis and reporting features for staff.

Attributes:

- - total_bookings: Total number of bookings
  - revenue: Total revenue generated
  - occupancy_rate: Room occupancy rate

Methods:

- - generate_report(): Generates a summary report
  - view_metrics(): Displays analytics metrics for staff

Class: Chatbot

Provides assistance and interacts with users to handle requests.

Attributes:

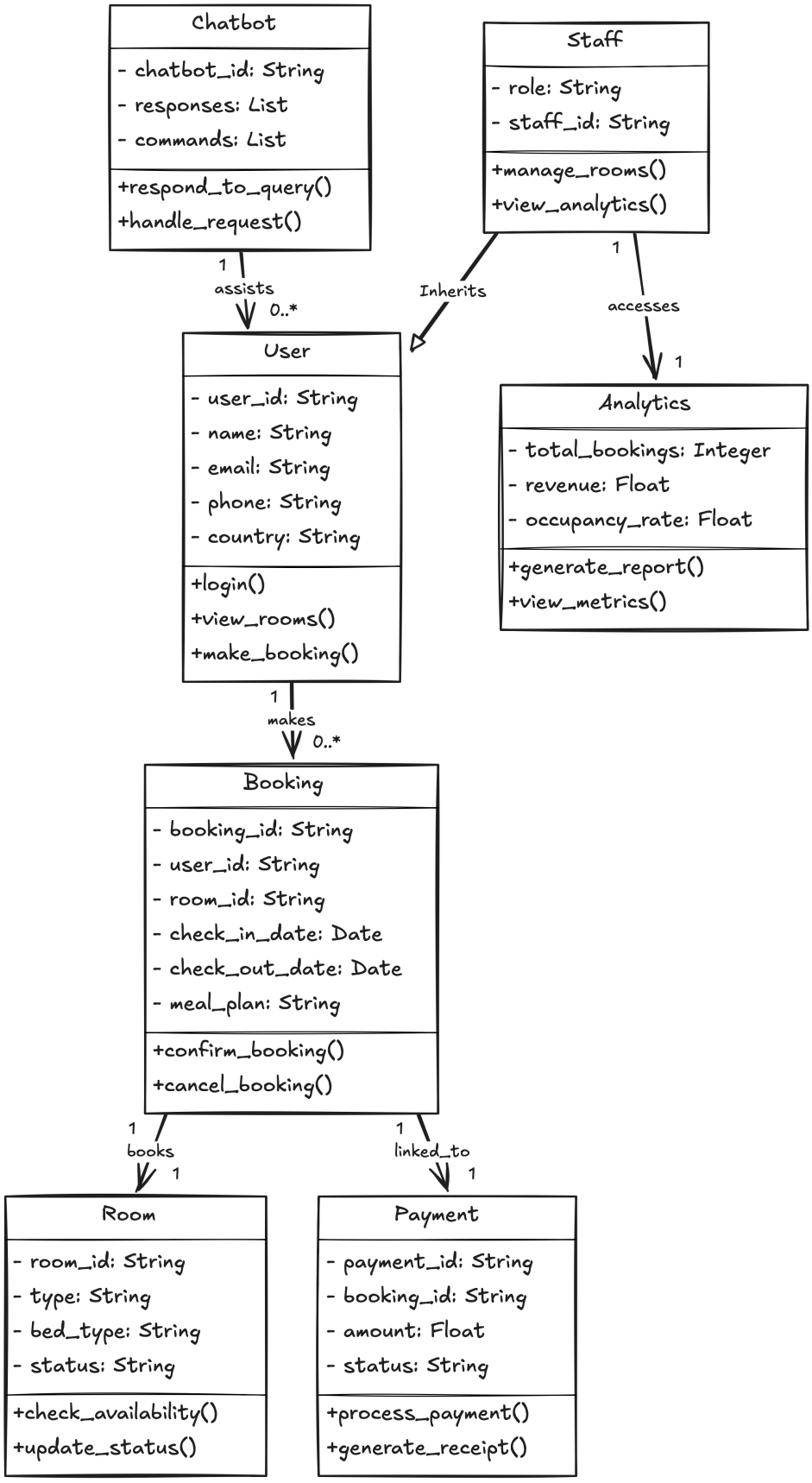
- - chatbot_id: Unique identifier for the chatbot session
  - responses: List of predefined responses
  - commands: Supported commands for user requests

Methods:

- - respond_to_query(): Generates a response based on user input
  - handle_request(): Processes specific user requests (e.g., room service)

Conclusion

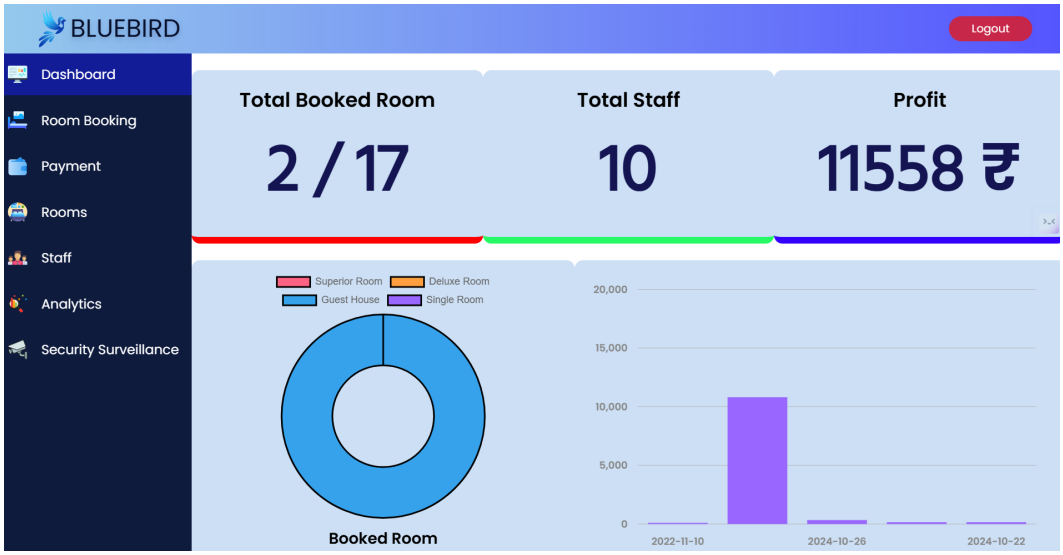
The Object-Oriented Design for Bluebird Hotel Booking Software ensures modularity and reusability of code. By organizing functionalities into distinct classes, the design facilitates code maintenance and future feature extensions. This OOD structure provides a clear and organized representation of the software components, making it easier to manage and scale the application.


Chatbot
- chatbot_id: String
- responses: List
- commands: List
+respond_to_query()
+handle_request()
Staff
- role: String
- staff_id: String
+manage_rooms()
+view_analytics()
1
assists
0..*
Inherits
1
accesses
1
User
- user_id: String
- name: String
- email: String
- phone: String
- country: String
+login()
+view_rooms()
+make_booking()
Analytics
- total_bookings: Integer
- revenue: Float
- occupancy_rate: Float
+generate_report()
+view_metrics()
1
makes
0..*
Booking
- booking_id: String
- user_id: String
- room_id: String
- check_in_date: Date
- check_out_date: Date
- meal_plan: String
+confirm_booking()
+cancel_booking()
1
books
1
1
linked_to
1
Room
- room_id: String
- type: String
- bed_type: String
- status: String
+check_availability()
+update_status()
Payment
- payment_id: String
- booking_id: String
- amount: Float
- status: String
+process_payment()
+generate_receipt()

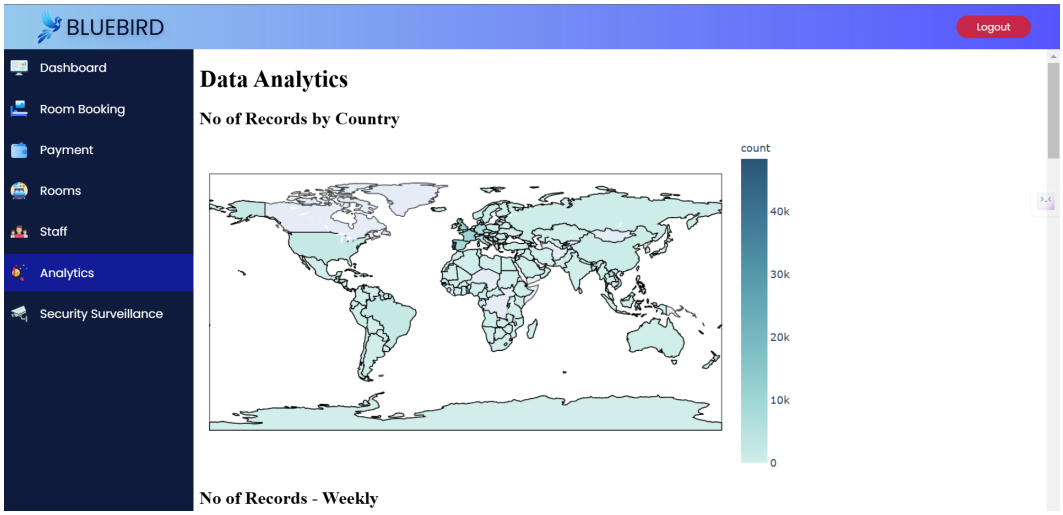
Hotel Management System Website Screenshots

Staff Login Section

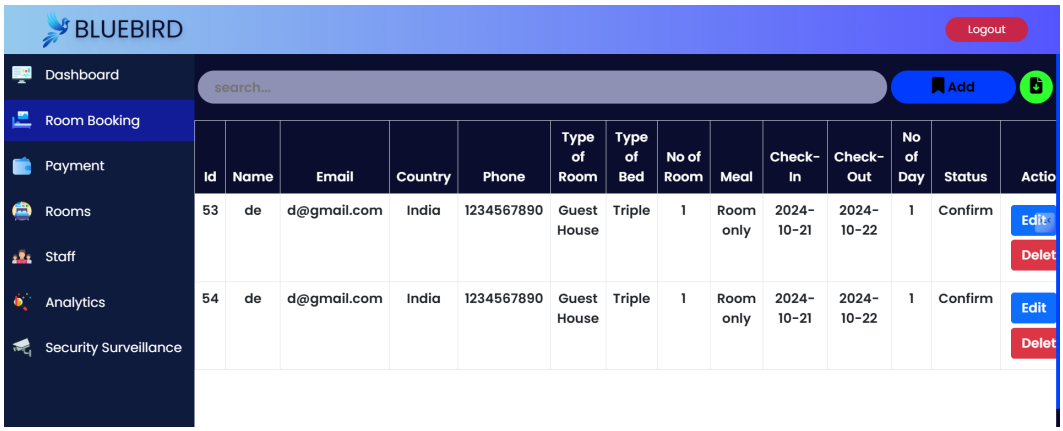
Admin Dashboard



Admin Analytics

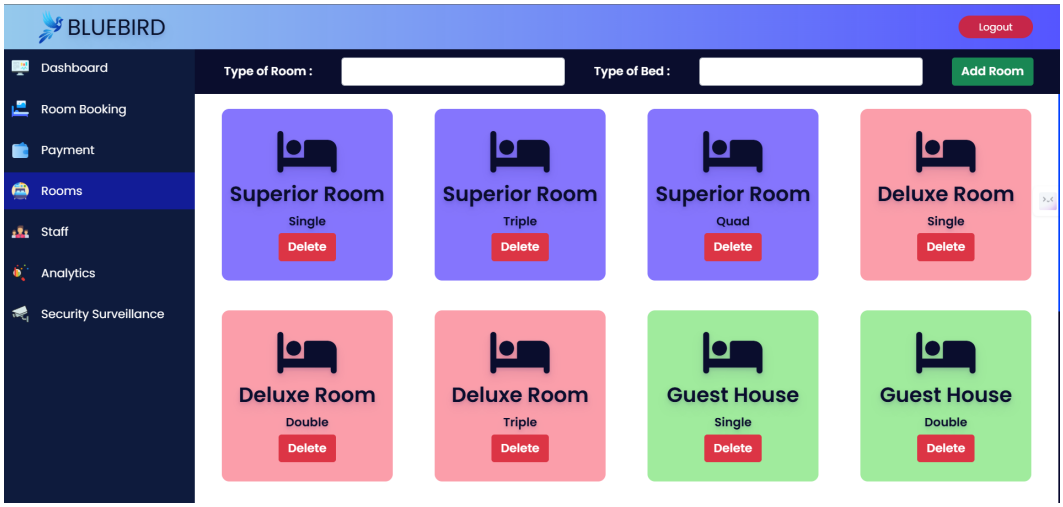


Admin Bookings

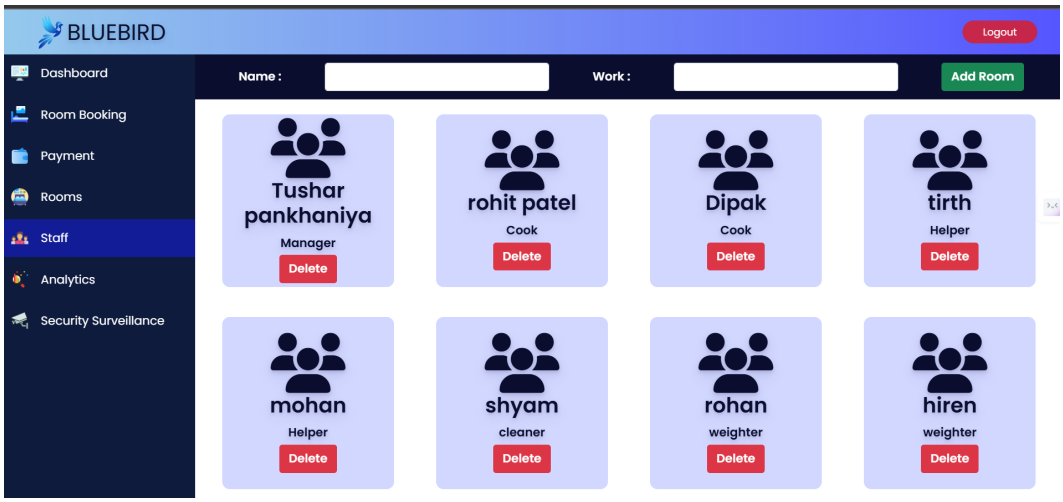


Admin Payments

Admin Rooms

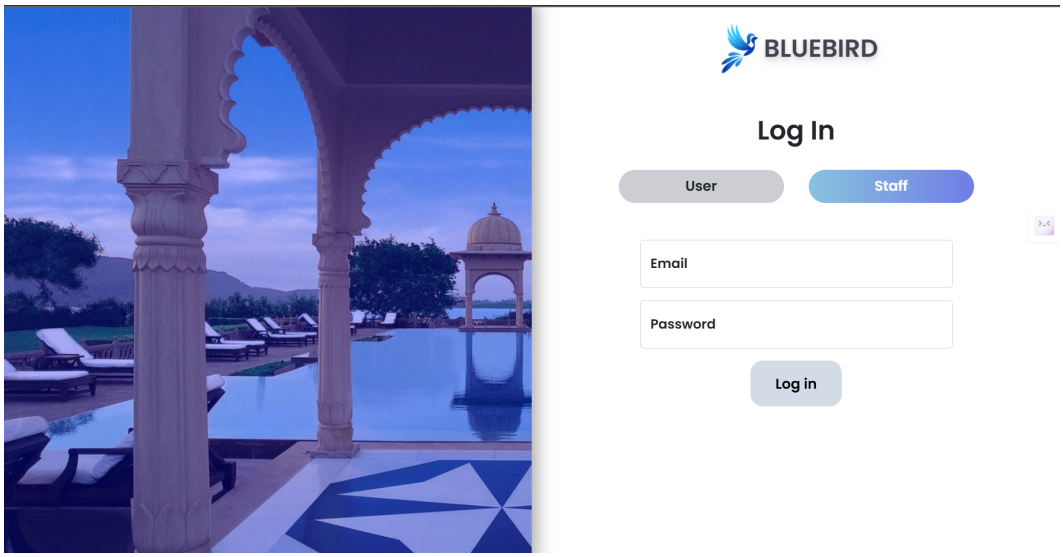


Admin Staff



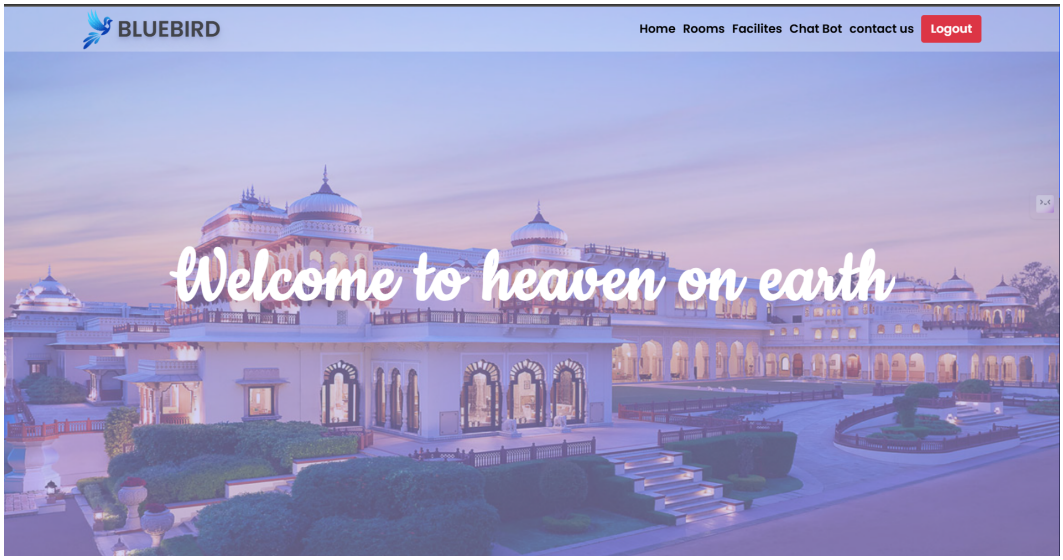
Admin Security

Staff Login

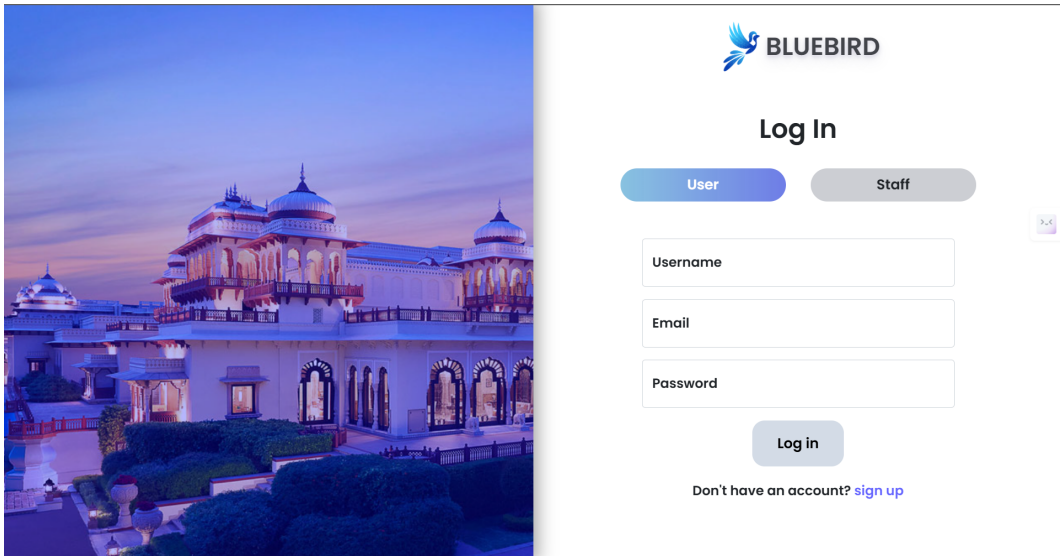


User Login Section

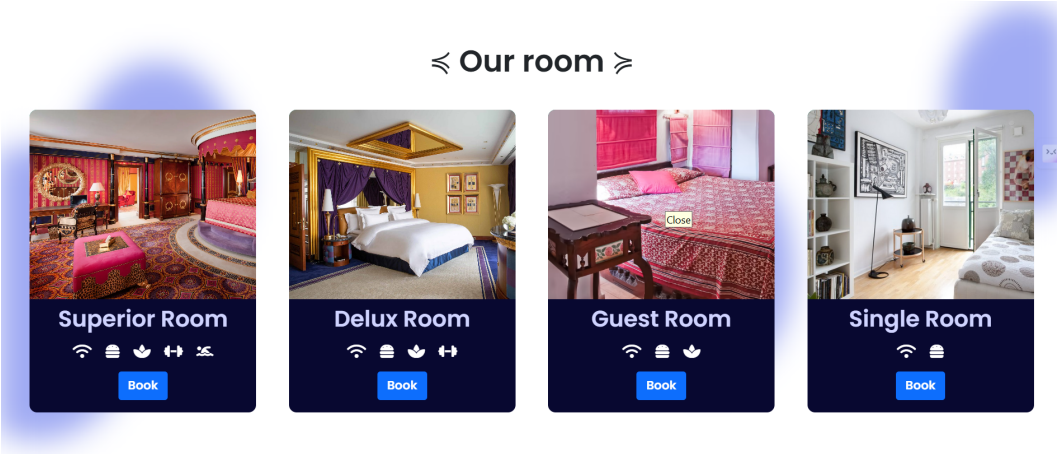
User Home



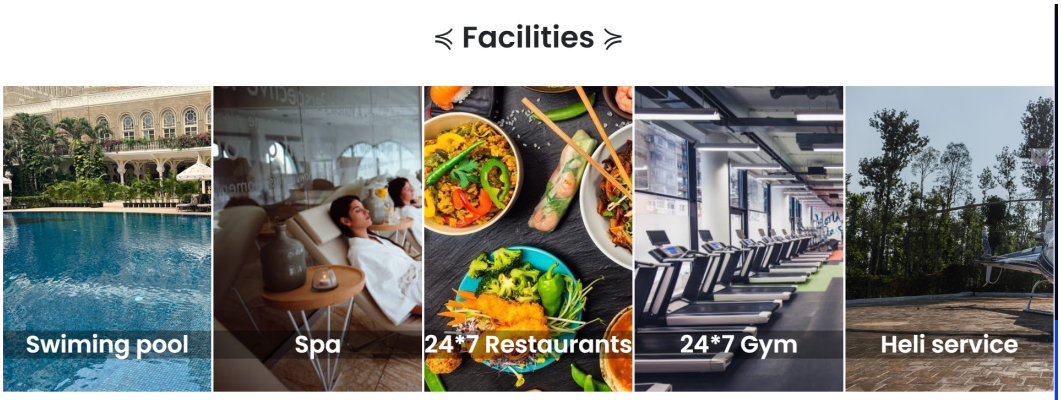
User Login



User Rooms

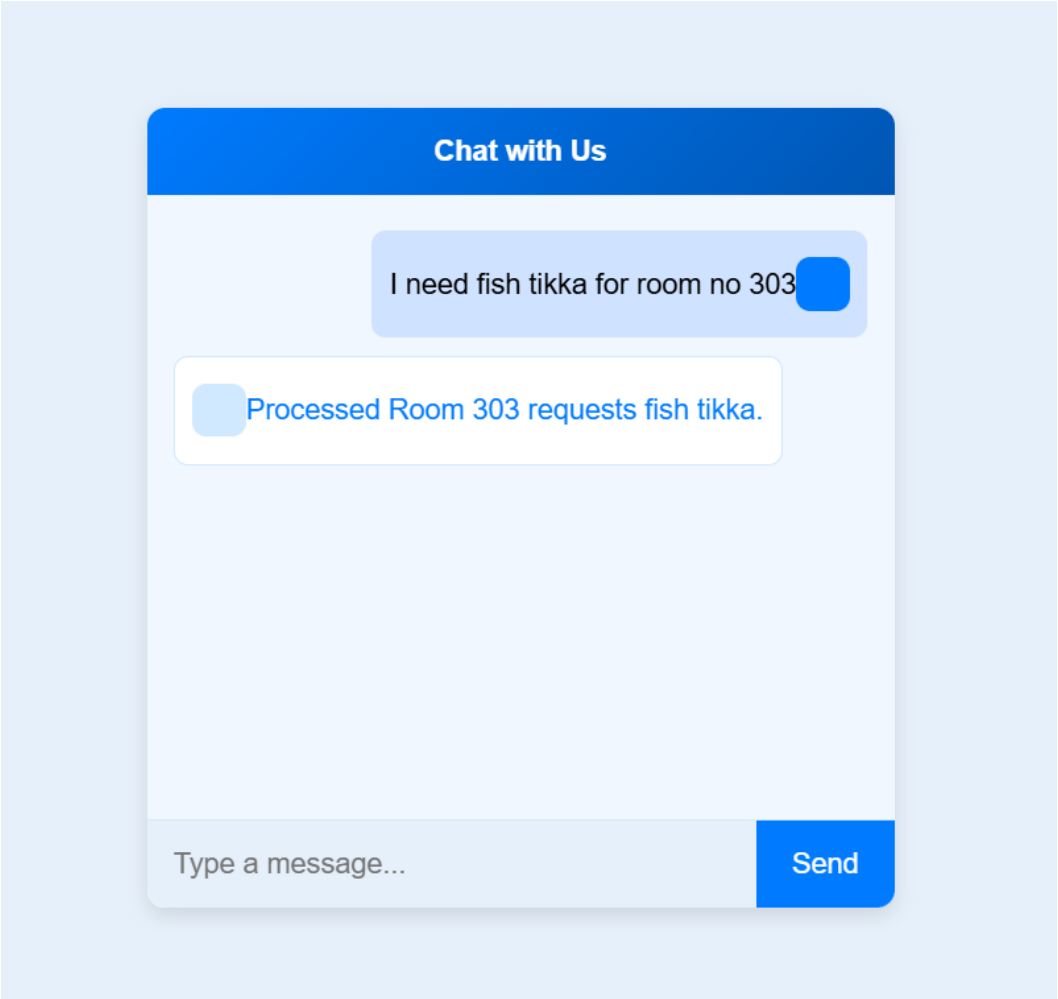


User Facilities

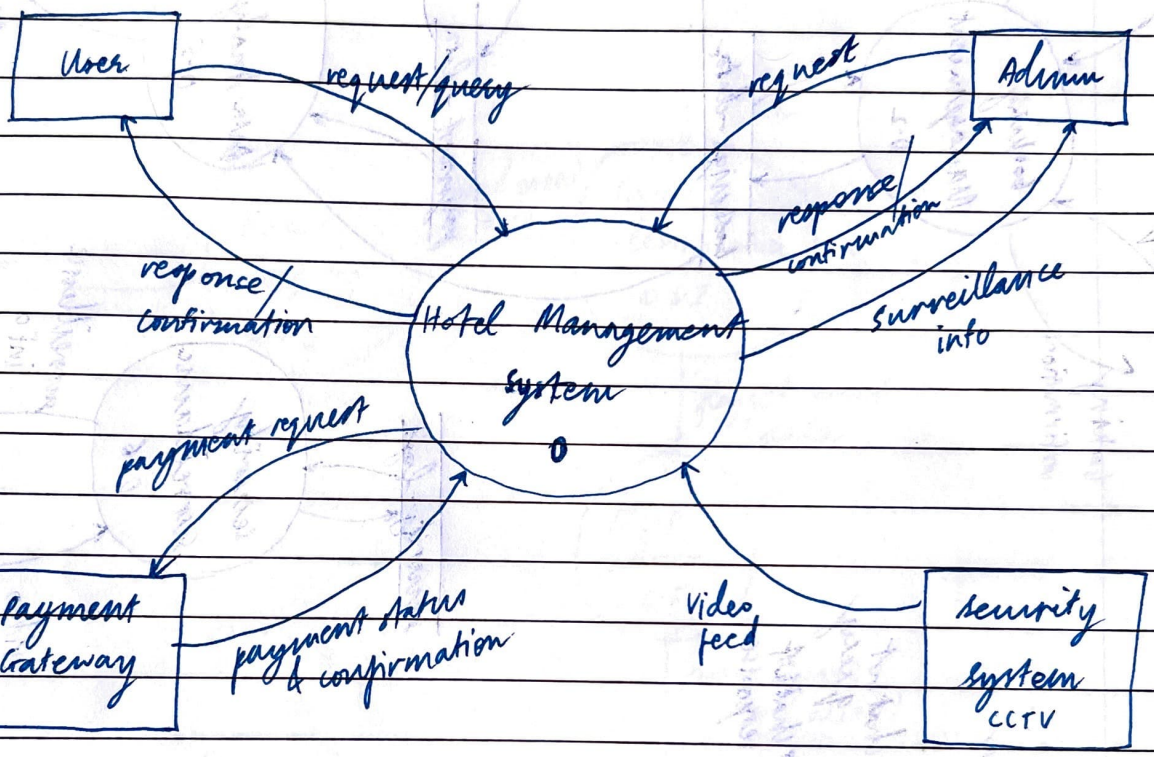


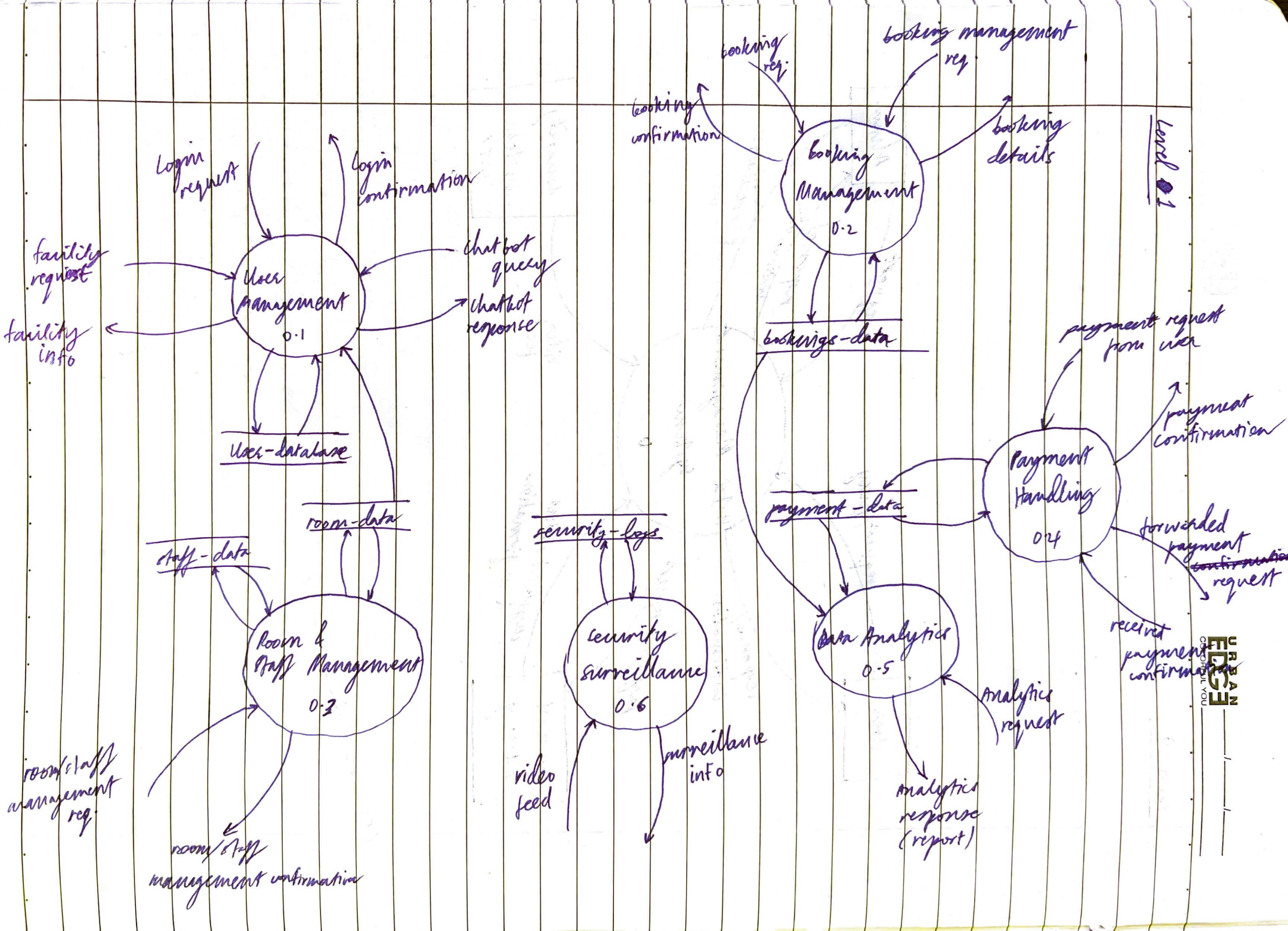
User Reservation

User Chatbot

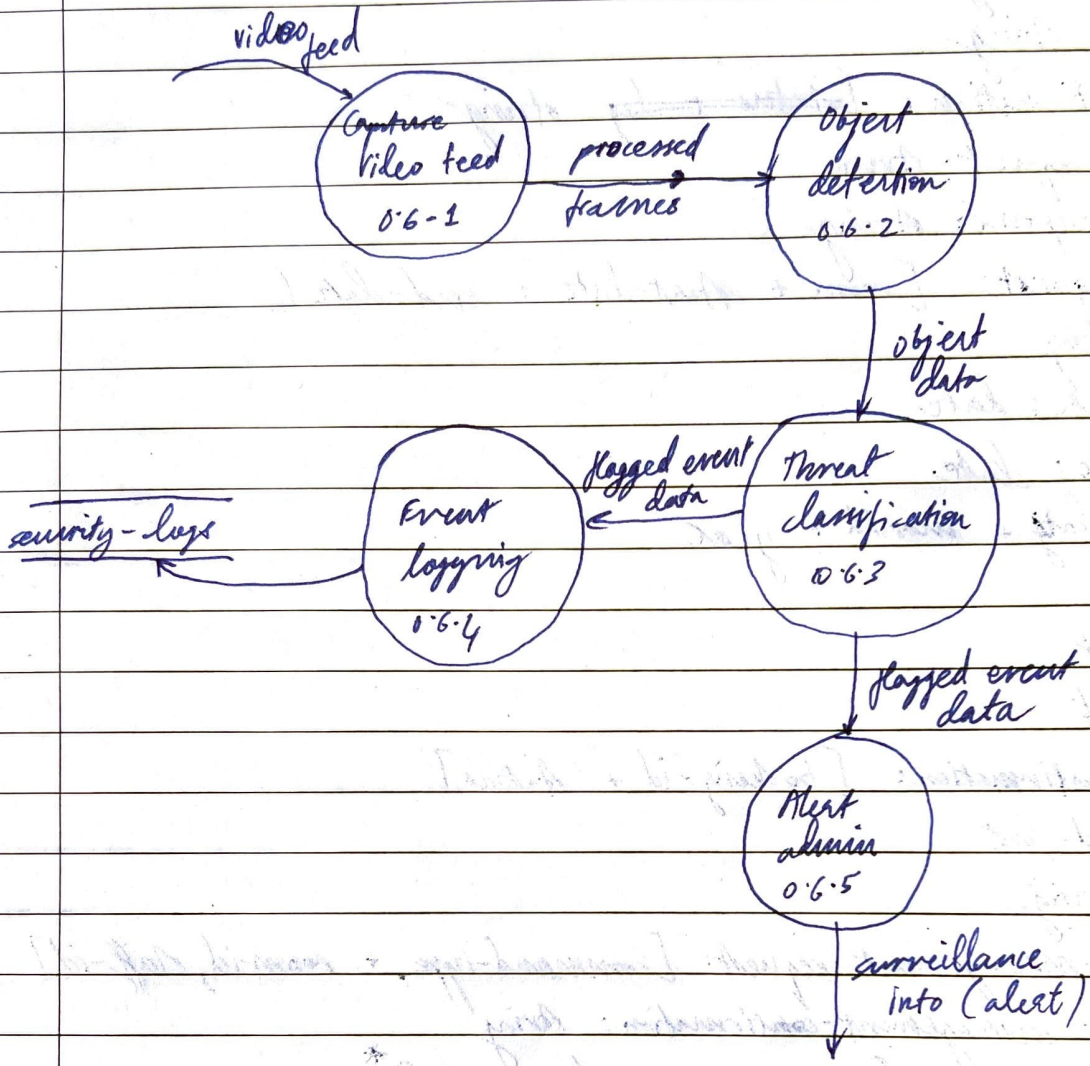


Level 0


User
request/query
response/
confirmation
Hotel Management
System
0
request
Admin
response/
confirmation
surveillance
info
payment request
payment status
& confirmation
payment
Gateway
Video
feed
security
System
CCTV


booking req
booking management req
booking confirmation
booking details
Booking Management 0.2
Level 0.1
Login request
Login confirmation
facility request
facility info
User management 0.1
chatbot query
chatbot response
User-database
bookings-data
payment request from user
payment confirmation
Payment Handling 0.4
payment-data
forwarded payment confirmation request
room-data
staff-data
security-logs
Room & Staff Management 0.3
Security Surveillance 0.6
Data Analytics 0.5
received payment confirmation
analytics request
analytics response (report)
room/staff management req
room/staff management confirmation
video feed
surveillance info

Level 2 : for process 0.6 security surveillance



URBAN EDGE COLORFUL YOU

Data dictionary

login-request : [username + password]
username : string
password : string
login-confirmation : [~~status + log~~ string
chatbot_request : string
chatbot-response : string
booking-request : [room + start-date + end-date]
room : string
start-date : date
end-date : date
date : day + month + year
day : int
month : int
year : int
booking-confirmation : [booking-id + status]
booking-id : int
status : string
room/staff-management-request : [command-type, + room-id, staff-id]
room/staff-management-confirmation : string
surveillance-info : {threat-type + description}*
threat-type : int
description : string
analytics-request : metric
analytics-response : image
payment-request :
payment-confirmation : [payment-id + status]
payment-id : int
status : string

URBAN EDGE COLORFUL YOU

Use Case Diagram