Software Design Specifications

[HOTEL MANAGEMENT SYSTEM]

Project Code	CL1004/CS1004
Supervisor	Mrs Iqra Fahad
Co Supervisor	-------------------------------
Project Team	Hamza Yousuf
Submission Date	19th April 2023

Document History

Version	Name of Person	Date	Description of change
1	Hamza Yousuf	11 feb 2023	Document Created for hotel management system
2	Hamza Yousuf	18 feb 2023	Added initial draft, outlining key functionalities → Including room booking, guest management, and payment processing.
3	Hamza Yousuf	22 feb 2023	Updated Document to include user roles and permissions for system access control, and detailed descriptions of the check-in and check-out processes.
4	Hamza Yousuf	5 march 2023	Added non-functional requirements including security standards, and backup procedures.
5	Hamza Yousuf	9 march 2023	Revised entire document to incorporate feedback updated use cases for room service ordering and emergency protocols.
6	Hamza Yousuf	16 march 2023	Final review added a change log Prepared document for approval and implementation phase.

Distribution List

Name	Role
Mrs Iqra Fahad	Supervisor
Hamza Yousuf	Lead developer
Hamza Yousuf	secondary/assistant developer

Document Sign-Off

Version	Sign-off Authority	Project Role	Signature	Sign-off Date
1	Mrs Iqra Fahad	Supervisor	Iqra Fahad	12 Feb 2023
2	Mrs Iqra Fahad	Supervisor	Iqra Fahad	23 Feb 2023
5	Mrs Iqra Fahad	Supervisor	Iqra Fahad	10 March 2023
6	Mrs Iqra Fahad	Supervisor	Iqra Fahad	18 March 2023

Document Information

Category	Information
Customer	FAST-NU
Project	Hotel Management System
Document	Software Design Specification
Document Version	1.0
Status	Finish
Author(s)	Hamza Yousuf
Approver(s)	Mrs. Iqra Fahad
Issue Date	19th April 2023
Document Location	-----------------------------------
Distribution	Advisor Project Coordinator's Office (through Advisor)

Definition of Terms, Acronyms and Abbreviations

Term	Description
DD	Design Specification - A document that outlines the design and architecture of a system, including its components, functionality, and interactions. The design specification provides detailed descriptions of the system's requirements, user interface design, database structure, algorithms, and other technical specifications. It serves as a blueprint for the development team to implement the system accurately and efficiently.
F&B	Food and Beverage - Refers to the food and drink offerings within a hotel, including restaurants, room service, and catering services.
GRC	Guest Reservation Center - A centralized system or department within a hotel responsible for managing guest reservations, inquiries, and bookings.

Table of Contents

1. Introduction

1.1 Purpose of Document

The purpose of this Software Design Specification (SDS) document is to provide a comprehensive guide to the design aspects of the Hotel Management System (HMS) project. It outlines the architectural design, components, and functionalities of the system, facilitating a clear understanding for the development team. Additionally, it specifies the intended audience, document conventions, and the design methodology employed for the project.

1.2 Intended Audience

This document is intended for:

Members of the project development team

Project stakeholders, including customers, managers, and receptionists

Individuals involved in the design, development, or evaluation of the HMS

1.3 Document Convention

The document follows standard font conventions for readability and consistency. The font used is Arial, with a font size of 12 for the main body text and 14 for section headings.

1.4 Project Overview

The Hotel Management System (HMS) project aims to streamline hotel management processes through automation. The system will encompass functionalities such as room booking, guest management, billing, reporting, and maintenance management. To achieve this, the project will adopt an Object-Oriented Design Methodology, facilitating the organization of system components and the modeling of real-world entities and interactions.

1.5 Scope

The scope of the HMS project, as defined in the Software Requirements Specification (SRS) document, includes:

Developing a comprehensive HMS solution to automate hotel management tasks

Implementing functionalities for room booking, guest management, billing, reporting, and maintenance management

Utilizing Object-Oriented Design Methodology to ensure scalability, maintainability, and extensibility of the system architecture

However, it's important to note that the project will not include the following functionalities in this phase:

Online booking

Online payment processing

Advanced revenue management algorithms

2. Design Considerations

This section serves to formally set the groundwork for the system design of the Hotel Management System (HMS) project.

2.1 Assumptions and Dependencies

This subsection addresses new issues relevant to the design phase, beyond those already captured in the Software Requirements Specification (SRS) document. These may include:

Technology Stack Compatibility: Ensuring compatibility and integration of selected technologies and frameworks with the overall system design.

Scalability Requirements: Considering potential future scalability requirements and designing the system architecture accordingly.

Third-Party Integrations: Identifying any dependencies on external systems or APIs and ensuring seamless integration into the HMS architecture.

Regulatory Compliance: Addressing any regulatory or compliance requirements specific to the design phase, such as data privacy regulations or industry standards.

2.2 Risks and Volatile Areas

This subsection identifies the most likely sources of change and risks that could impact the design of the system. Potential risks include:

Emerging Technologies: Rapid advancements in technology may introduce new design paradigms or require updates to existing components.

Changing Requirements: Evolving stakeholder needs or market demands may necessitate adjustments to the system design.

Security Vulnerabilities: Emerging threats or vulnerabilities in the system architecture may require proactive design measures to mitigate risks.

Resource Constraints: Limitations in budget, time, or available expertise may pose challenges to the design process.

Additionally, this subsection outlines how the system will be designed to accommodate changes and mitigate risks effectively. Contingency plans may include:

Modular Design: Adopting a modular architecture to facilitate flexibility and ease of modification.

Iterative Development: Embracing an iterative design approach to incorporate feedback and adapt to changing requirements.

Continuous Evaluation: Implementing mechanisms for continuous evaluation and improvement of the system design to address evolving needs and mitigate risks proactively.

System Architecture

1.1 System Level Architecture

Let's simplify the description based on the provided guidelines:

System Architecture Description:

System Decomposition into Elements:
The system architecture consists of key elements such as Users (including Managers, Receptionists, and Guests), Rooms, and Food Orders.

Relationships between Elements:
- Users interact with the system through different roles, each with specific functionalities and responsibilities.
- Managers oversee administrative tasks, while Receptionists handle guest-related operations.
- Guests interact with the system to book rooms, order food, and manage reservations.
- Rooms are managed by Receptionists and are associated with food orders placed by Guests.

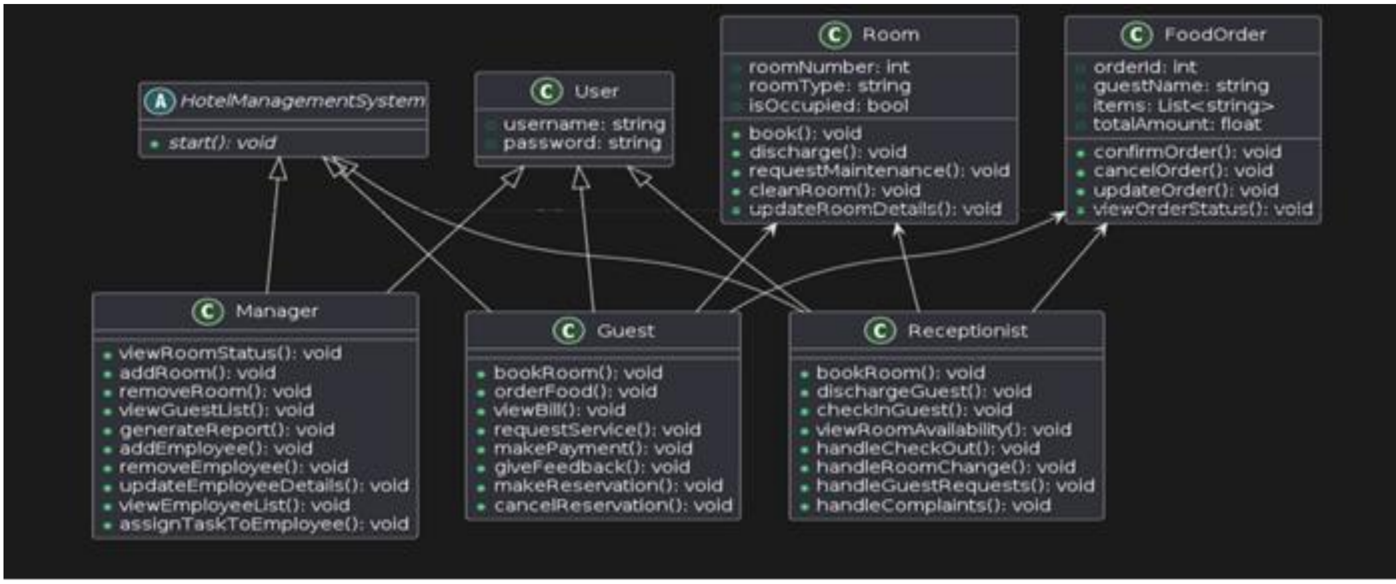
Interfaces to External Systems:

The system may interact with external APIs for functionalities such as payment processing or integration with external booking platforms.

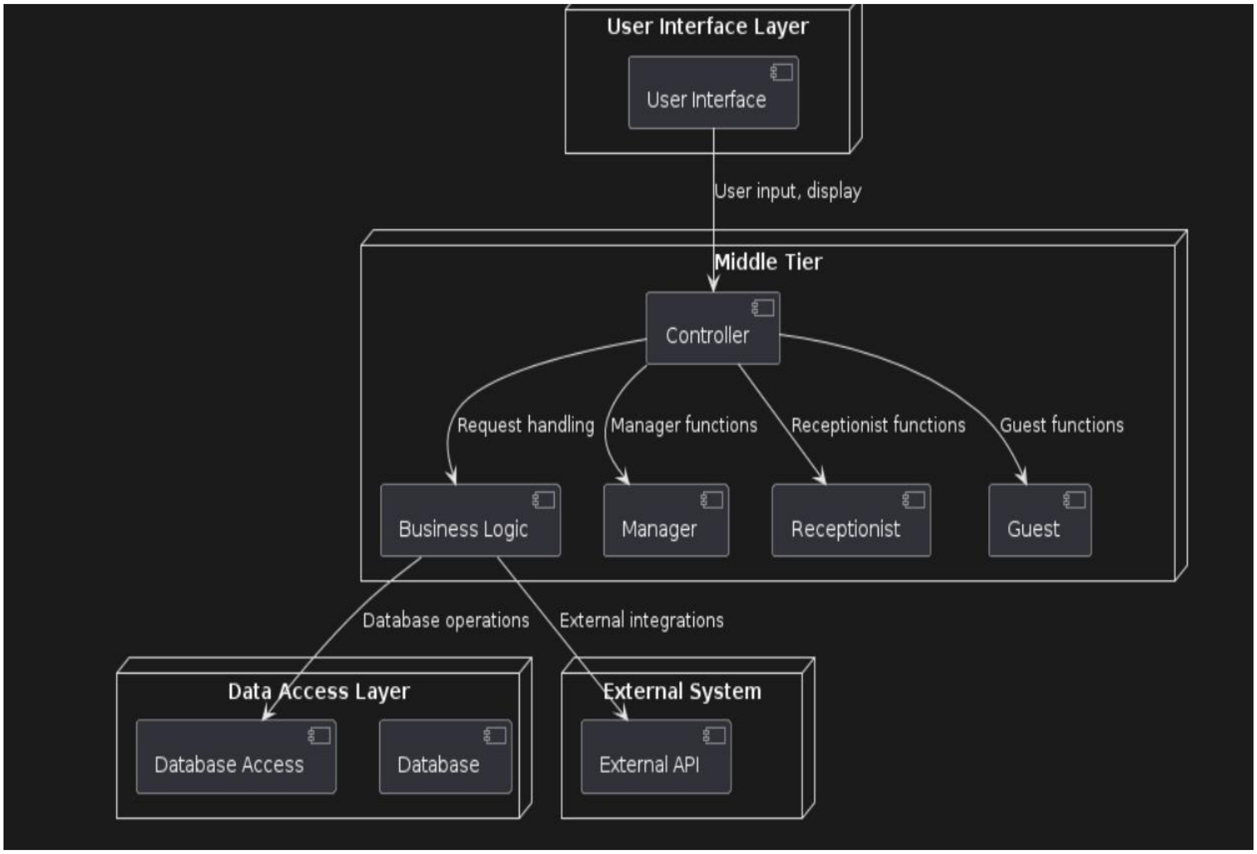
Major Physical Design Issues:
- The system comprises a User Interface Layer, Middle Tier, and Data Access Layer for managing interactions with users, core business logic, and data storage, respectively.

Global Design Strategies:
- Robust error handling mechanisms are implemented to ensure graceful degradation and user-friendly error messages.
- Security measures such as authentication and authorization are in place to protect sensitive data and functionalities.

This simplified overview provides a concise description of the system architecture, outlining its key components, relationships, interfaces, and design strategies.



1.2 Software Architecture



Design Strategy

Future System Extension or Enhancement:

- **Modularity**: The system architecture prioritizes modularity, allowing for easy addition or modification of functionalities without disrupting existing components.

- **Abstraction Layers**: Clear abstraction layers are employed to facilitate future enhancements by separating different concerns and promoting component reuse.

System Reuse:

*Component-Based Design**: Components are designed for reuse across the system, reducing redundancy and promoting code efficiency.

*Standardization**: Standardized interfaces and design patterns promote system reuse and interoperability with other systems.

User Interface Paradigms:

User-Centric Design**: The user interface prioritizes user experience, focusing on intuitive navigation and interactive elements.

Responsive Design: The interface is designed to be responsive, ensuring optimal display across various devices and screen sizes.**

Data Management:

** **Database Normalization**: Data organization follows normalized schemas to minimize redundancy and ensure data integrity.**

** **Scalability**: Provisions for scalability are incorporated into the data management strategy, allowing the system to handle growing data volumes effectively.**

Data Persistence: Persistent storage mechanisms guarantee data durability and availability, even in the event of system failures.**

Concurrency and Synchronization:

** **Thread-Safe Design**: Thread-safe design patterns and synchronization mechanisms manage concurrent access to shared resources, ensuring data consistency.**

** **Asynchronous Processing**: Asynchronous processing techniques enhance system responsiveness and scalability, particularly for resource-intensive tasks.**

Reasoning and Trade-Offs:

- **Flexibility vs. Complexity: Design choices balance system flexibility with complexity, aiming to accommodate**

future changes without overly complicating the architecture.

2 - **Performance vs. Maintainability**: Trade-offs are considered between system performance and maintainability, striving for an optimal balance between the two.

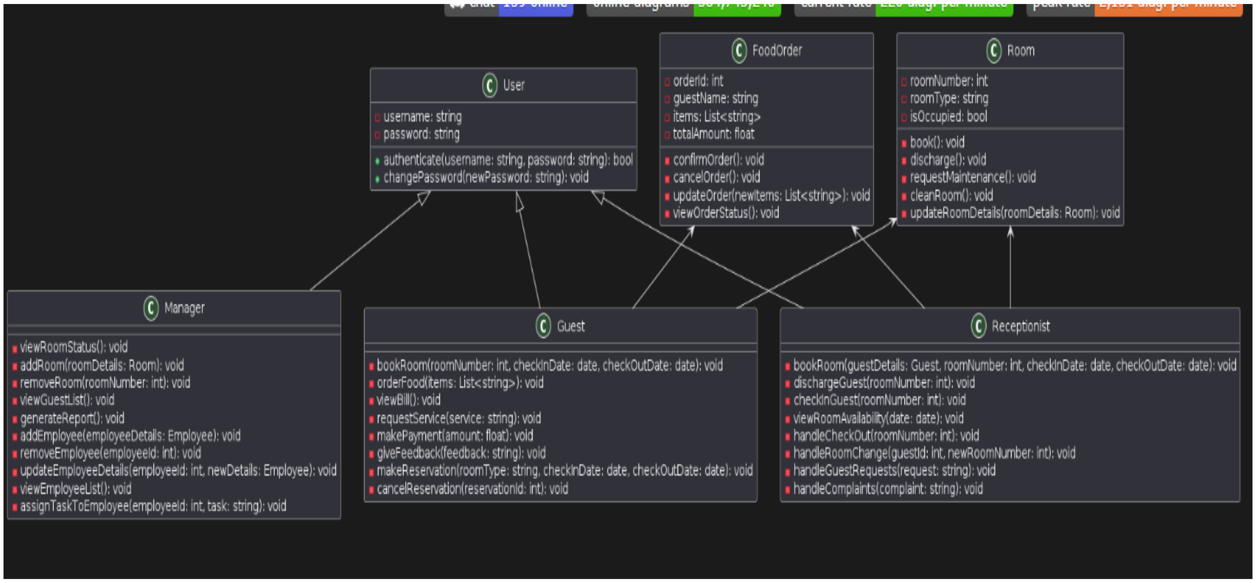
User Experience vs. System Efficiency: Design decisions prioritize user experience while also considering system efficiency and resource utilization.

Scalability vs. Resource Consumption: Scalability strategies are implemented while minimizing resource consumption and overhead.

These design strategies aim to create a robust and adaptable system architecture that can evolve to meet evolving requirements while maintaining usability, performance, and reliability.

Detailed System Design

2.1 *Database Design*


User
username: string
password: string
authenticate(username: string, password: string): bool
changePassword(newPassword: string): void
FoodOrder
orderId: int
guestName: string
items: List<string>
totalAmount: float
confirmOrder(): void
cancelOrder(): void
updateOrder(newItems: List<string>): void
viewOrderStatus(): void
Room
roomNumber: int
roomType: string
isOccupied: bool
book(): void
discharge(): void
requestMaintenance(): void
cleanRoom(): void
updateRoomDetails(roomDetails: Room): void
Manager
viewRoomStatus(): void
addRoom(roomDetails: Room): void
removeRoom(roomNumber: int): void
viewGuestList(): void
generateReport(): void
addEmployee(employeeDetails: Employee): void
removeEmployee(employeeId: int): void
updateEmployeeDetails(employeeId: int, newDetails: Employee): void
viewEmployeeList(): void
assignTaskToEmployee(employeeId: int, task: string): void
Guest
bookRoom(roomNumber: int, checkInDate: date, checkOutDate: date): void
orderFood(items: List<string>): void
viewBill(): void
requestService(service: string): void
makePayment(amount: float): void
giveFeedback(feedback: string): void
makeReservation(roomType: string, checkInDate: date, checkOutDate: date): void
cancelReservation(reservationId: int): void
Receptionist
bookRoom(guestDetails: Guest, roomNumber: int, checkInDate: date, checkOutDate: date): void
dischargeGuest(roomNumber: int): void
checkInGuest(roomNumber: int): void
viewRoomAvailability(date: date): void
handleCheckOut(roomNumber: int): void
handleRoomChange(guestId: int, newRoomNumber: int): void
handleGuestRequests(request: string): void
handleComplaints(complaint: string): void

2.1.1 ER Diagram

```
+-----------------------+
|       Guest           |
+-----------------------+
| GuestID (PK)
| Name
| Email
| Phone
| CheckInDa
| CheckOutDate
| RoomNumber (FK)
+-----------------------+
       |
       |
       |
       |
       |
+-----------------------+
|        Room        |
+-----------------------+
| RoomNumber (PK)
| Type
| Rate
| Status
+-----------------------+
```

Guest and Room are the main entities in the system.
Each Guest has a unique GuestID (primary key) and is associated with a specific RoomNumber (foreign key) during their stay.
Each Room has a unique RoomNumber (primary key) and attributes like Type, Rate, and Status.
The relationship between Guest and Room is depicted by the RoomNumber attribute in the Guest entity, indicating which room a guest is staying in.

2.1.2 Data Dictionary

Data 1	
Name	Phone
Alias	Contact Number
Where-used/how-used	Represents the phone number of a guest. Input to process: Guest registration - Output from process: Communication - Stored in the database as a column in the Guest table.
Content description	String (formatted as a phone number)

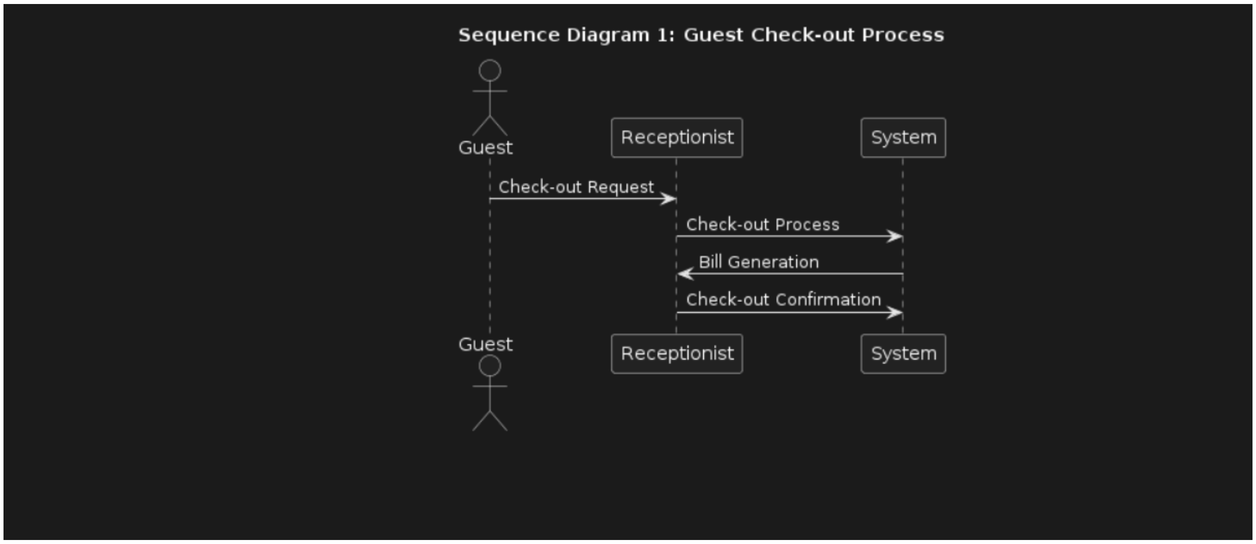
Data Element Name	Description	Data Type	Length	Constraints	Relationships	Usage
GuestID	Unique identifier for each guest	Integer		Unique, Not null		Primary key
GuestName	Name of the guest	String	50	Not null		Identification
RoomNumber	Number of the hotel room	Integer		Not null	Room(RoomNumber)	Identification
CheckInDate	Date of guest check-in	Date		Not null		Booking information
CheckOutDate	Date of guest check-out	Date		Not null		Booking information
ReservationID	Unique identifier for each reservation	Integer		Unique, Not null		Primary key
EmployeeID	Unique identifier for each employee	Integer		Unique, Not null		Primary key
EmployeeName	Name of the employee	String	50	Not null		Identification
TaskID	Unique identifier for	Integer		Unique, Not null		Primary key

	each task					
TaskDescription	Description of the task	String	100	Not null		Task information
FoodOrderID	Unique identifier for each food order	Integer		Unique, Not null		Primary key
OrderDate	Date when the food order was placed	Date		Not null		Order information
TotalAmount	Total amount of the food order	Decimal		Not null		Order information

2.2 Application Design

2.2.1 Sequence Diagram

2.2.1.1 <Sequence Diagram 1>

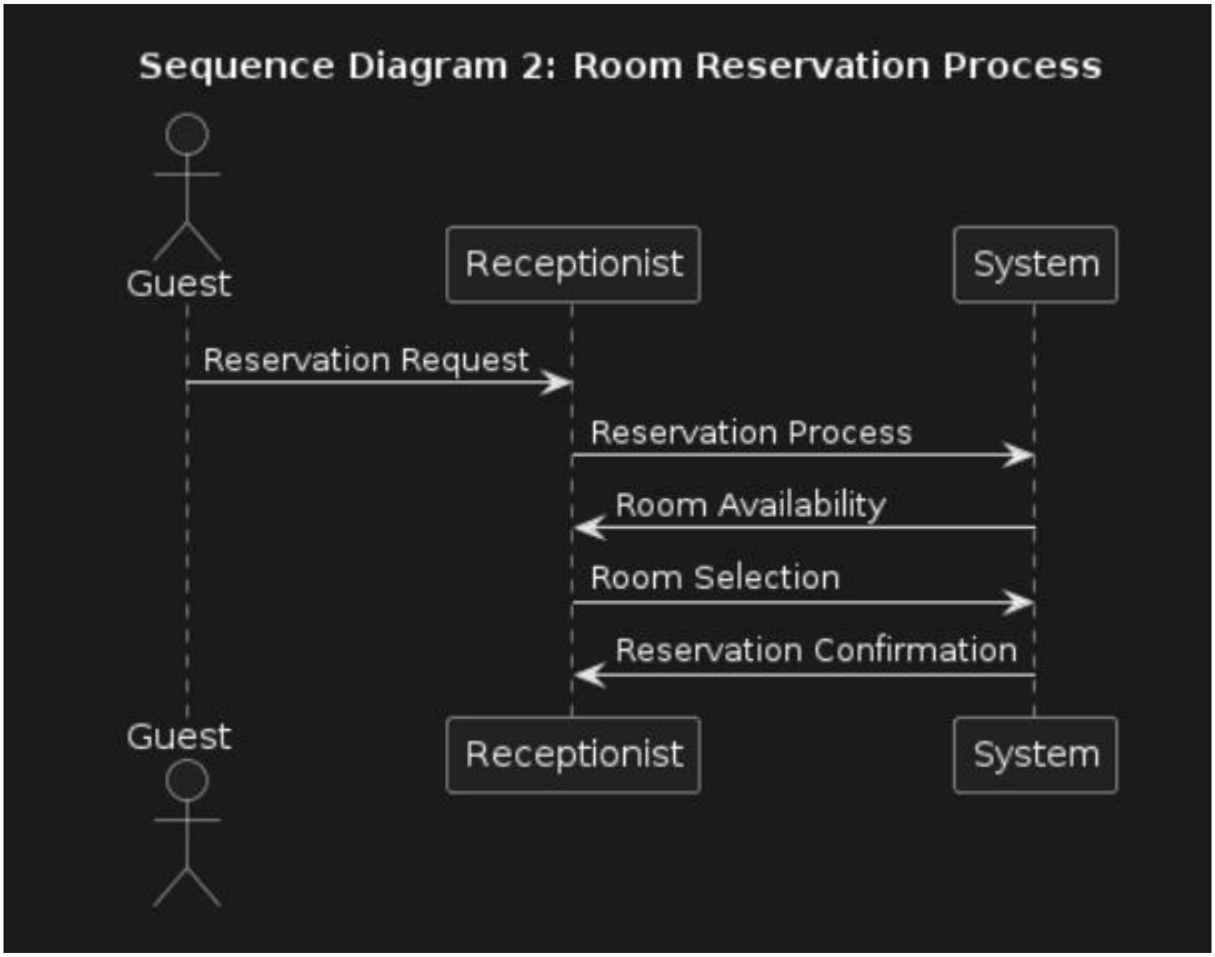


Sequence Diagram 1: Guest Check-in Process

This diagram illustrates the sequence of events involved in the Guest Check-in Process:

Actor: The Guest arrives at the hotel and interacts with the Receptionist.
Receptionist: The Receptionist initiates the check-in process by entering the guest's details into the system.
System: The Hotel Management System verifies the availability of rooms and prompts the Receptionist to select a suitable room for the guest.
Receptionist: The Receptionist selects a room and assigns it to the guest.
System: The Hotel Management System updates the room status as occupied and generates a room key for the guest.
Receptionist: The Receptionist hands over the room key to the guest, along with any additional information or services.
Guest: The Guest proceeds to their assigned room.
This sequence ensures a smooth and efficient check-in process for guests at the hotel.

2.2.1.2 <Sequence Diagram 2>



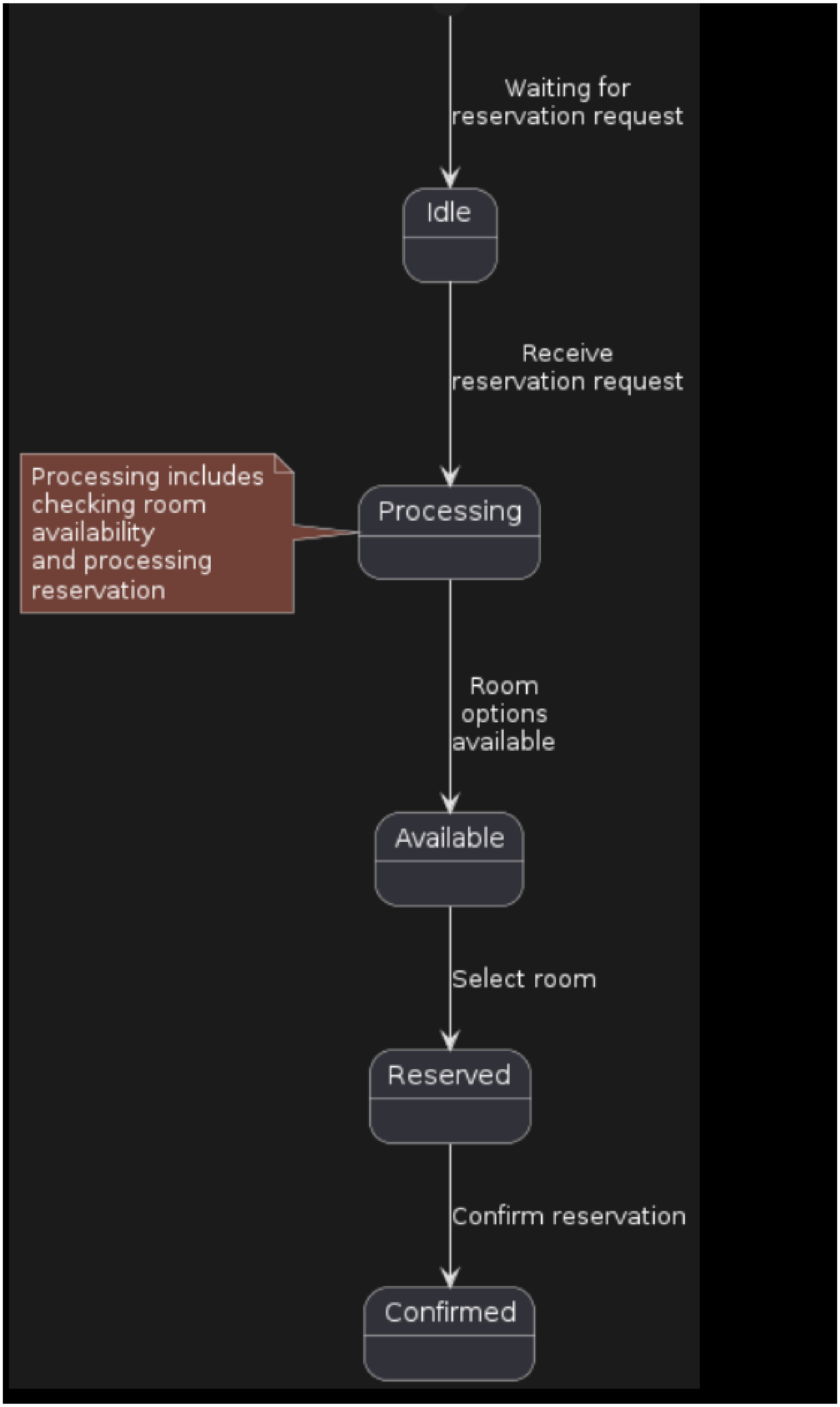
Sequence Diagram 2: Room Reservation Process

This diagram outlines the steps involved in the Room Reservation Process:

Actor: The Guest requests a room reservation from the Receptionist.
Receptionist: The Receptionist initiates the reservation process by interacting with the Hotel Management System.
System: The Hotel Management System checks the availability of rooms and presents the options to the Receptionist.
Receptionist: The Receptionist selects a room based on the guest's preferences and confirms the reservation.
System: The Hotel Management System sends a confirmation to the Receptionist and updates the reservation status.
Guest: The Guest receives confirmation of their room reservation

2.2.2 State Diagram

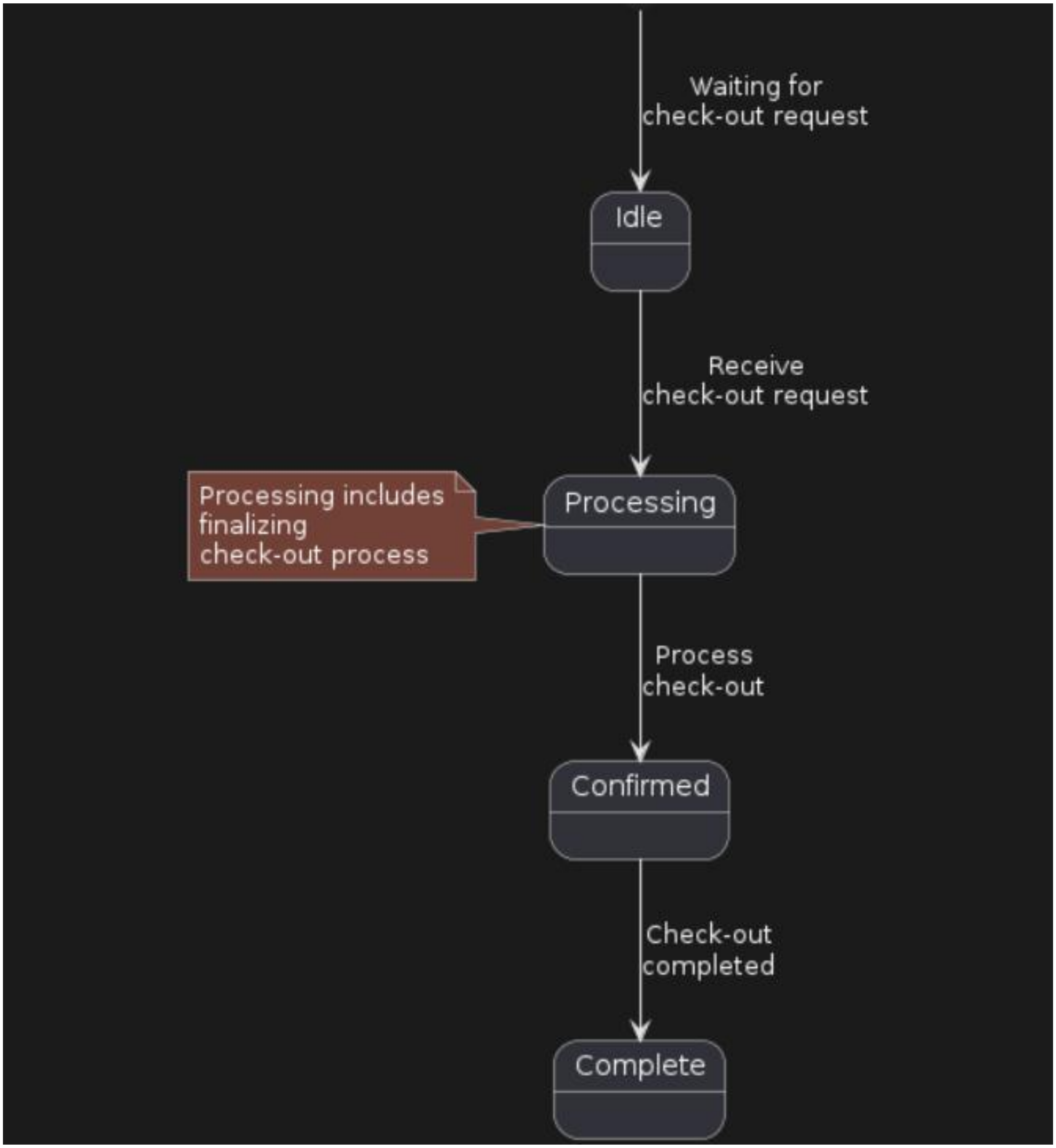
2.2.2.1 <State Diagram 1>


Waiting for reservation request
Idle
Receive reservation request
Processing includes checking room availability and processing reservation
Processing
Room options available
Available
Select room
Reserved
Confirm reservation
Confirmed

State Diagram 1: Room Reservation Process

Idle: This is the initial state where the system is waiting for a reservation request from a guest.
Processing: Upon receiving a reservation request, the system moves to the Processing state, where it checks the availability of rooms and processes the reservation.
Available: If rooms are available, the system transitions to the Available state, where it presents room options to the receptionist for selection.
Reserved: After the receptionist selects a room for the guest, the system transitions to the Reserved state, indicating that the room has been reserved.
Confirmed: Once the reservation is confirmed, the system moves to the Confirmed state, indicating that the reservation process is complete.

2.2.2.2 <State Diagram 2>.



State Diagram 2: Guest Check-out Process

Idle: This is the initial state where the system is waiting for a check-out request from a guest.
Processing: Upon receiving a check-out request, the system moves to the Processing state, where it finalizes the check-out process.

Confirmed: After the check-out process is completed, the system transitions to the Confirmed state, indicating that the check-out has been successfully processed.
Complete: Finally, the system moves to the Complete state, indicating that the check-out process is complete and the guest has officially checked out of the hotel.

3 References

No specific documents were referenced for the creation of this use case.

4 *As this is a hypothetical scenario, no specific documents or sources were utilized.*

Appendices

1. Sample Code Snippets:**

5 - Sample code snippets showcasing key functionalities, such as room reservation or billing calculations, to provide a deeper understanding of the system's implementation details.

6

7 **2. Simple System Diagram:**

8 - A simplified system diagram providing a basic visual representation of essential components and their interactions, aiding in understanding the system's architecture.

9

10 **3. Test Cases:**

11 - A list of test scenarios and their expected outcomes to assist in validating the system's functionality and ensuring robustness through comprehensive testing.

12

13 **4. Project Plan:**

14 - An outline of the project's scope, objectives, timeline, and tasks for each team member in a project plan providing clarity on project management and execution.

15

16 **5. Project Requirements Document:**

- A summary of stakeholder requirements and an outline of the system's features and functionalities in a requirements document serving as a reference point for system development.

6. Development Environment Setup Guide:

- Instructions for setting up the development environment necessary to run and modify the system's code, facilitating collaboration among team members.

Acknowledgments: Recognition of individuals or resources that contributed to the project's development, such as professors, peers, or online tutorials, expressing gratitude and acknowledging support received during the project's execution.detail that would be too distracting to include in the main body of the document.