MOYAI HOTEL
BOOKING SYSTEM

Group 2
Lecturer Name: Dr. Lizawati Binti Mi Yusuf

GROUP MEMBERS

AHMAD NABIL BIN AHMAD NAZRIL

(B23CS0020)

AQMAR ILHAN BIN MOHAMAD FADZILAH

(B23CS0027)

EIZKHAN BIN SAHARIZA

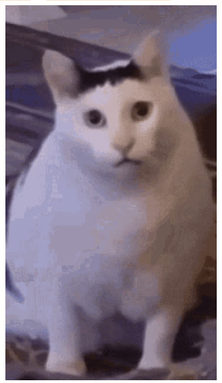
(B23CS0035)

MUHAMMAD HANIF AZRI BIN AZIZAN

(B23CS0057)

INTRODUCTION

Moyai Hotel Booking System is an upgraded system for staff to make booking for guest easier. The booking process is efficiently streamlined by this system and improve hotel management productivity and have an user-friendly interface. The main goal of the hotel booking system is to increase overall efficiency and, as a result, staff satisfaction by automating and streamlining operations related to hotel reservation.



PROBLEM STATEMENT

Moyai hotel current booking system is an outdated system that have many bugs and error with not user-friendly to use. In order to enhance system efficiency and hotel productivity, we decide to develop a new system with user-friendly interface for Moyai hotel staff to easy to use the system.

OBJECTIVE

USER-FRIENDLY INTERFACE:

Provide a user-friendly interface for staff to make room reservations easily and efficiently

ENHANCED SYSTEM STABILITY & PERFORMANCE:

Streamlining the booking process through automation validation to reduces human error, minimizes errors, and speeds up reservation procedures.

IMPROVE HOTEL PRODUCTIVITY:

This new system can increase the Moyai Hotel productivity in order to get good customer service from guest when making room reservation faster for guest.

OOP CONCEPTS

01

ENCAPSULATION & DATA HIDING

02

CLASS RELATIONSHIP (AGGREGATION, COMPOSITION & ASSOCIATION)

03

INHERITANCE

04

ABSTRACTION & POLYMORPHISM

05

TEXT FILE OPERATION

06

EXCEPTION HANDLING

OOP CONCEPTS

ENCAPSULATION & DATA HIDING


```java
// Inheritance: DeluxeRoom extends the Room class, inheriting its attributes and methods.
public class DeluxeRoom extends Room {

    // Enum for DeluxeRoomType with associated base price.
    public enum DeluxeRoomType {
        DELUXE(basePrice:200.0); // The start room number is now managed separately

        private final double basePrice;

        DeluxeRoomType(double basePrice) {
            this.basePrice = basePrice;
        }

        // Encapsulation: Provides access to the private basePrice field.
        public double getBasePrice() {
            return basePrice;
        }
    }

    // Constructor: Initializes DeluxeRoom with specific DeluxeRoomType, availability, and room price.
    public DeluxeRoom(DeluxeRoomType roomType, boolean availability, double roomPrice) {
        // Constructor chaining: Calls the superclass constructor to initialize common attributes.
        super(roomNumber:0, capacity:6, roomType:"Deluxe", roomPrice, availability);
    }

    // Polymorphism and Method Override: Overrides the description method in the base class Room.
    @Override
    public String description() {
        return "Deluxe Room with capacity of " + getCapacity() + " guest.";
    }
}
```

OOP CONCEPTS

AGGREGATION


```java
public class Report {
    UserInterface userInterface = new ConsoleScreenController();

    public void generateReport(List<BookingInfo> bookings) throws IOException {
        if (bookings.isEmpty()) {
            System.out.print("No bookings to generate. Please create a new booking!
            userInterface.loadingAnimation();
            userInterface.clearScreen();
            return;
        }

        FileWriter fw = new FileWriter("./report/Report.txt");
        BufferedWriter bw = new BufferedWriter(fw);
        PrintWriter pw = new PrintWriter(bw);

        LocalDateTime now = LocalDateTime.now();
        DateTimeFormatter dateFormatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");
        String currentDate = now.format(dateFormatter);
```

OOP CONCEPTS

COMPOSITION


```java
public class BookingInfo {

    // Composition: The class has member variables representing other classes, forming a composition.
    private Room bookedRoom;
    private Guest guest;
    private LocalDate checkInDate;
    private LocalDate checkOutDate;
    private Payment payment;

    // Constructor: Initializes the BookingInfo object with provided values during object creation.
    public BookingInfo(Room bookedRoom, Guest guest, LocalDate checkInDate, LocalDate checkOutDate, Payment payment) {
        // Encapsulation: Setting private fields using constructor parameters.
        this.bookedRoom = bookedRoom;
        this.guest = guest;
        this.checkInDate = checkInDate;
        this.checkOutDate = checkOutDate;
        this.payment = payment;
    }

    // Getter methods: Provide access to the private fields, following encapsulation principles.
    public Room getBookedRoom() {
        return bookedRoom;
    }

    public Guest getGuest() {
        return guest;
    }

    public LocalDate getCheckInDate() {
        return checkInDate;
    }

    public LocalDate getCheckOutDate() {
        return checkOutDate;
    }
```

OOP CONCEPTS

ASSOCIATION


```java
public void handleBooking(BookingInfo bookingInfo) {                    //Association - BookingInfo Class
    String fullName, contactNumber, email, checkInDateString, checkOutDateString;
    LocalDate checkInDate = null;
    LocalDate checkOutDate = null;
    double totalAmount;

    if (allRoomsOccupied()) {
        System.out.println(x:"Sorry, all rooms are currently occupied. No new bookings can be made at the moment.");
        return;
    }

    System.out.println(x:"----- Make New Booking -----");
    System.out.print(s:"\nChoose a room type \n(1 for Standard, 2 for Deluxe, 3 for Superior): ");

    int roomTypeChoice = getUserChoice();

    if (roomTypeChoice == 1 && allRoomsOccupiedByType(roomType:1)) {
        System.out.println(x:"Sorry, all Standard rooms are currently occupied. Please choose another room type.\n");
        return;
    } else if (roomTypeChoice == 2 && allRoomsOccupiedByType(roomType:2)) {
        System.out.println(x:"Sorry, all Deluxe rooms are currently occupied. Please choose another room type.\n");
        return;
    } else if (roomTypeChoice == 3 && allRoomsOccupiedByType(roomType:3)) {
        System.out.println(x:"Sorry, all Superior rooms are currently occupied. Please choose another room type.\n");
        return;
    }

    displayAvailableRoomsByType(roomTypeChoice);

    // Assign room number manually
    System.out.print(s:"Enter the desired room number: ");
    int roomNumber = scanner.nextInt();

    userInterface.clearScreen();
```

OOP CONCEPTS

INHERITANCE


```java
// Inheritance: DeluxeRoom extends the Room class, inheriting its attributes and methods.
public class DeluxeRoom extends Room {

    // Enum for DeluxeRoomType with associated base price.
    public enum DeluxeRoomType {
        DELUXE(basePrice:200.0); // The start room number is now managed separately

        private final double basePrice;

        DeluxeRoomType(double basePrice) {
            this.basePrice = basePrice;
        }

        // Encapsulation: Provides access to the private basePrice field.
        public double getBasePrice() {
            return basePrice;
        }
    }

    // Constructor: Initializes DeluxeRoom with specific DeluxeRoomType, availability, and room price.
    public DeluxeRoom(DeluxeRoomType roomType, boolean availability, double roomPrice) {
        // Constructor chaining: Calls the superclass constructor to initialize common attributes.
        super(roomNumber:0, capacity:6, roomType:"Deluxe", roomPrice, availability);
    }

    // Polymorphism and Method Override: Overrides the description method in the base class Room.
    @Override
    public String description() {
        return "Deluxe Room with capacity of " + getCapacity() + " guest.";
    }
}
```

OOP CONCEPTS

ABSTRACTION


```java
// Abstraction: Declares an abstract class named Room, serving as a base class for specific types of rooms.
abstract class Room {

    // Encapsulation: Protected fields to encapsulate the internal state, accessible to subclasses.
    protected int roomNumber;
    protected int capacity;
    protected String roomType;  // Added roomType
    protected double price;
    protected boolean isAvailable;

    // Encapsulation and Constructor: Initializes the common attributes of a room during object creation.
    public Room(int roomNumber, int capacity, String roomType, double price, boolean isAvailable) {
        this.roomNumber = roomNumber;
        this.capacity = capacity;
        this.roomType = roomType;  // Set roomType in the constructor
        this.price = price;
        this.isAvailable = isAvailable;
    }
```

OOP CONCEPTS

POLYMORPHISM


```java
// Inheritance: DeluxeRoom extends the Room class, inheriting its attributes and methods.
public class DeluxeRoom extends Room {

    // Enum for DeluxeRoomType with associated base price.
    public enum DeluxeRoomType {
        DELUXE(basePrice:200.0); // The start room number is now managed separately

        private final double basePrice;

        DeluxeRoomType(double basePrice) {
            this.basePrice = basePrice;
        }

        // Encapsulation: Provides access to the private basePrice field.
        public double getBasePrice() {
            return basePrice;
        }
    }

    // Constructor: Initializes DeluxeRoom with specific DeluxeRoomType, availability, and room price.
    public DeluxeRoom(DeluxeRoomType roomType, boolean availability, double roomPrice) {
        // Constructor chaining: Calls the superclass constructor to initialize common attributes.
        super(roomNumber:0, capacity:6, roomType:"Deluxe", roomPrice, availability);
    }

    // Polymorphism and Method Override: Overrides the description method in the base class Room.
    @Override
    public String description() {
        return "Deluxe Room with capacity of " + getCapacity() + " guest.";
    }
}
```

OOP CONCEPTS

TEXT FILE OPERATION : CREATE NEW FILE

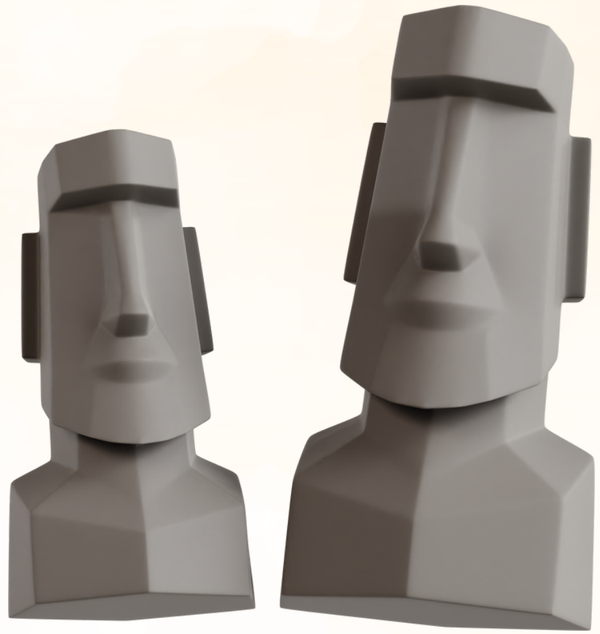

```java
public class Report {
    UserInterface userInterface = new ConsoleScreenController();

    public void generateReport(List<BookingInfo> bookings) throws IOException {
        if (bookings.isEmpty()) {
            System.out.print(s:"No bookings to generate. Please create a new booking! \nNavigating to the main menu");
            userInterface.loadingAnimation();
            userInterface.clearScreen();
            return;
        }

        FileWriter fw = new FileWriter(fileName:"./report/Report.txt"); //Create new folder with Report.txt in the folder
        BufferedWriter bw = new BufferedWriter(fw);
        PrintWriter pw = new PrintWriter(bw);

        LocalDateTime now = LocalDateTime.now();
        DateTimeFormatter dateFormatter = DateTimeFormatter.ofPattern(pattern:"dd-MM-yyyy");
        String currentDate = now.format(dateFormatter);

        // Calculate the center alignment for the hotel title
        int titleLength = "Moyai Hotel Booking Information Report".length();
        int padding = (130 - titleLength) / 2;
        String titlePadding = " ".repeat(Math.max(a:0, padding));

        // Print the report header
        pw.printf(format:"%s%s%s[ %3s ]%n", titlePadding, "Moyai Hotel Booking Information Report", titlePadding, currentDa
        pw.println(x:"-----------------------------------------------------------------------------------------------------------
        pw.printf(format:"%-2s | %-12s | %-25s | %-10s | %-16s | %-16s | %-8s | %-15s | %-17s%n",
                ...args:"#", "Room Number", "Guest Name", "Room Type", "Check-in Date", "Check-out Date","Night(s)", "Booki
        pw.println(x:"---+--------------+---------------------------+------------+------------------+------------------+---

        // Sort bookings by room number
        Collections.sort(bookings, new Comparator<BookingInfo>() {
            //@Override
            public int compare(BookingInfo b1, BookingInfo b2) {
                Integer roomNumber1 = b1.getBookedRoom().getRoomNumber();
                Integer roomNumber2 = b2.getBookedRoom().getRoomNumber();
                return roomNumber1.compareTo(roomNumber2);
```

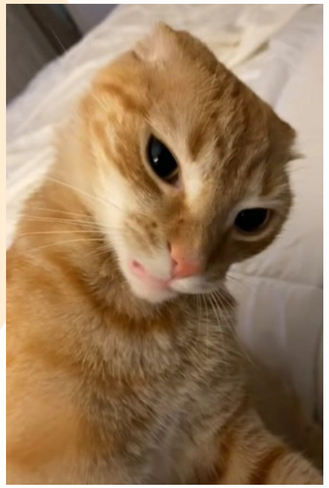
OOP CONCEPTS

TEXT FILE OPERATION : READ INPUT FILE


```java
private boolean checkCredentials(String username, String password) {
    try {
        File file = new File(pathname:"C:\\Users\\ASUS\\Desktop\\Degree UTM\\SEMESTER 1\\OOP\\Projek\\HotelBookingSystem
        Scanner fileScanner = new Scanner(file);

        while (fileScanner.hasNextLine()) {
            String line = fileScanner.nextLine();
            String[] credentials = line.split(regex:",");

            if (credentials.length == 2) {
                String storedUsername = credentials[0];
                String storedPassword = credentials[1];

                if (username.equals(storedUsername) && password.equals(storedPassword)) {
                    fileScanner.close();
                    return true;
                }
            } else {
                System.out.println(x:"Error: Invalid credentials format in the file.");
            }
        }

        fileScanner.close();
    } catch (FileNotFoundException e) {
        System.out.println(x:"Error: Credentials file not found. Please create new credential file");
        System.exit(status:0);
    }

    return false;
}
```

SYSTEM FUNTIONALITIES

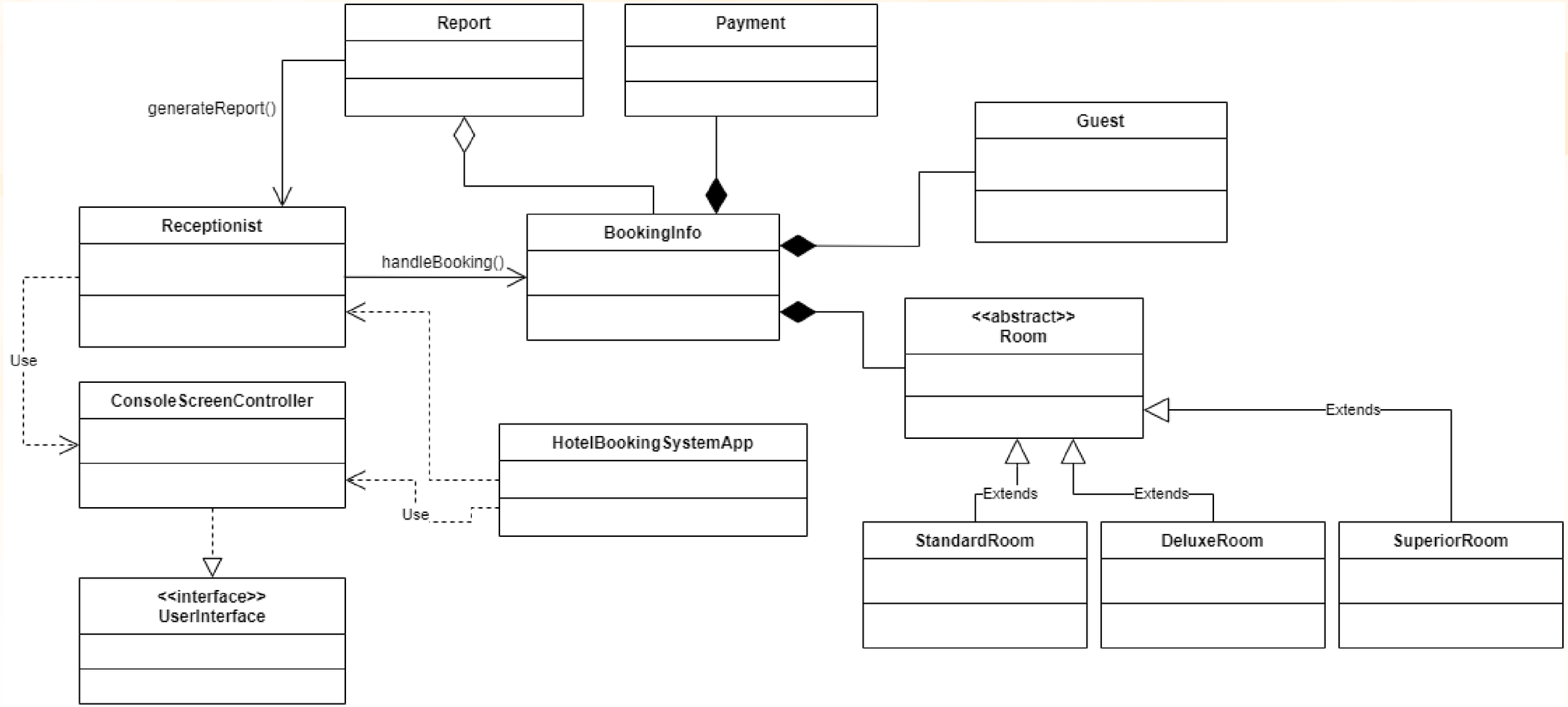




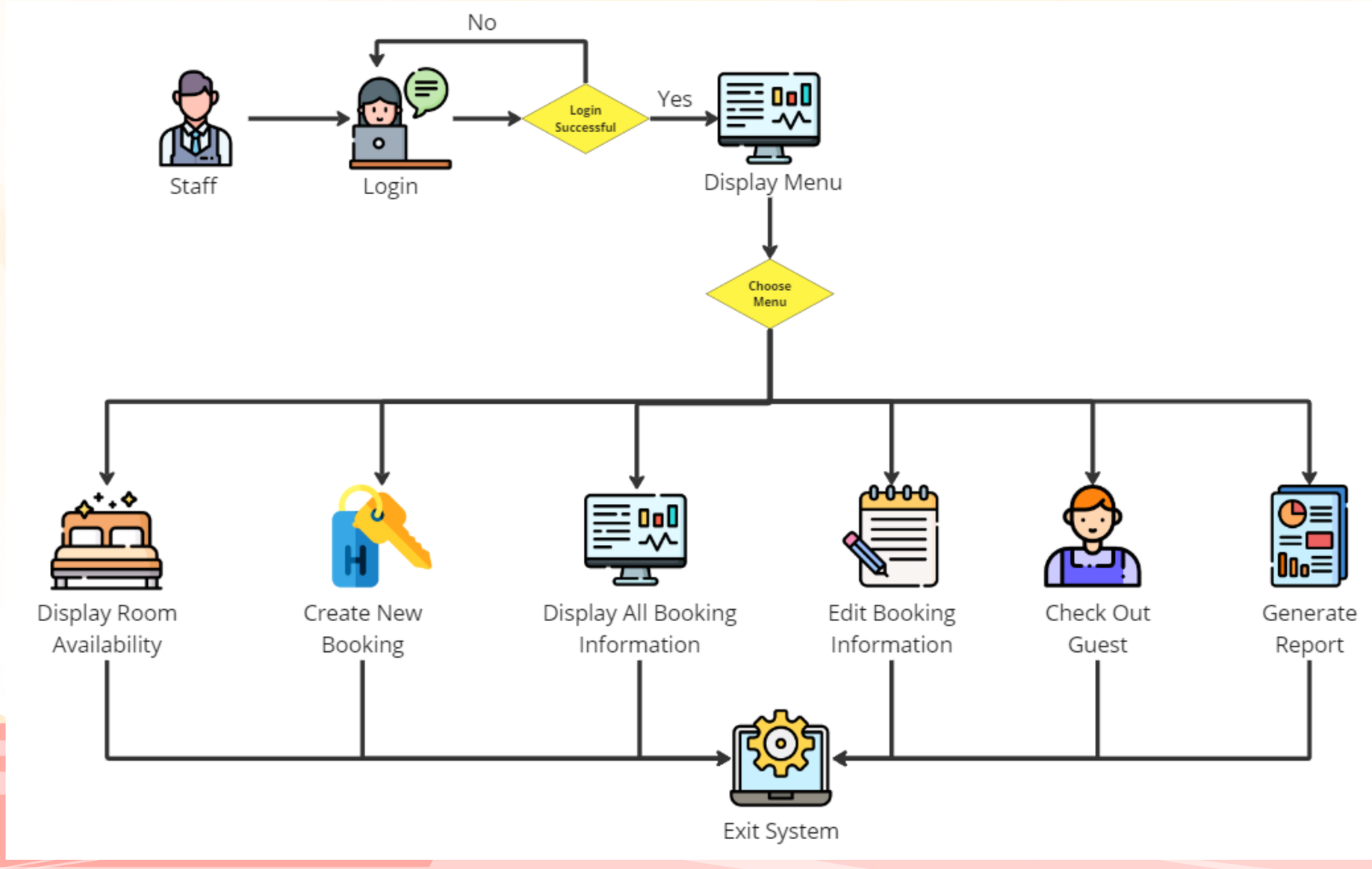
STAFF

- Login via validation and verification from credentials file.
- View available room in the hotel
- Create new booking for hotel guest
- Display all booking information
- Edit booking information
- Check-out guest
- Generate report

CLASS DIAGRAM



SYSTEM PRESENTATION



CONCLUSION

Moyai Hotel Booking System provide an efficient and user-friendly system that improve the hotel management productivity.

SYSTEM DEMONSTRATION

THANK YOU