State Model for
Hotel Management System

Name	Roll Number	Gr. Number	Div/Batch
Atharva Parikh	332044	21910716	B/B2
Sidhant Khamankar	332029	21910598	B/B2
Namrata Kadav	332040	21911024	B/B2

Objective

Identify States and events for your system.
Study state transitions and identify Guard conditions.
Draw State chart diagram with advanced UML 2.0 notations.
Implement the state model with a suitable object-oriented language.

Theory

A **state model** describes the timely behaviour of the class objects over a period of time. A state model has multiple **state diagrams** where each state diagram describes a class in the model.

State model shows these changes in the object with the help of **states, events, transitions** and **conditions**. Events are the incidents that occur to the object at a particular time whereas the state shows the value of the object at a particular time.

A state diagram describes the relation between events and states which are the significant elements of the state model.
Events are the incidents that take place at a particular time.
1. Signal Event

The signal event describes, sending and receiving of information from one object to another at a particular time. This event specifies one-way transmission.

2. Change Event
A change event is an event that occurs whenever a boolean expression is satisfied. This boolean expression is checked continuously and whenever the expression result changes from true to false or from false to true the change event occurs.

The UML representation of the change event is as follow:

when(room temperature < heating point)

In UML the change event is expressed using '**when**' keyword followed by the boolean expression.

3. Time Event
The time event is the event that occurs at a specified time or after the specified time elapsed. The absolute time event is represented in the UML using the '**when**' keyword followed by the parenthesis with a time expression. The Time interval event is represented by the '**after**' keyword followed parenthesis with expression evaluating the time duration.

when(time = 18:30)

after(10 seconds)

States
The state represents the values of the attributes of an object at a particular time. Thus, the state defines the behaviour of an object at a point in time. In UML the state of an object is represented with the *round box* containing the state name.

Transitions and Condition
When an object changes its current state to another state it is termed as the transition. Transition triggers the change in the original state of an object on

the occurrence of an event. Thereby the target state or the next state of the object depends on the object's original state and the event to which the object's original state has responded.

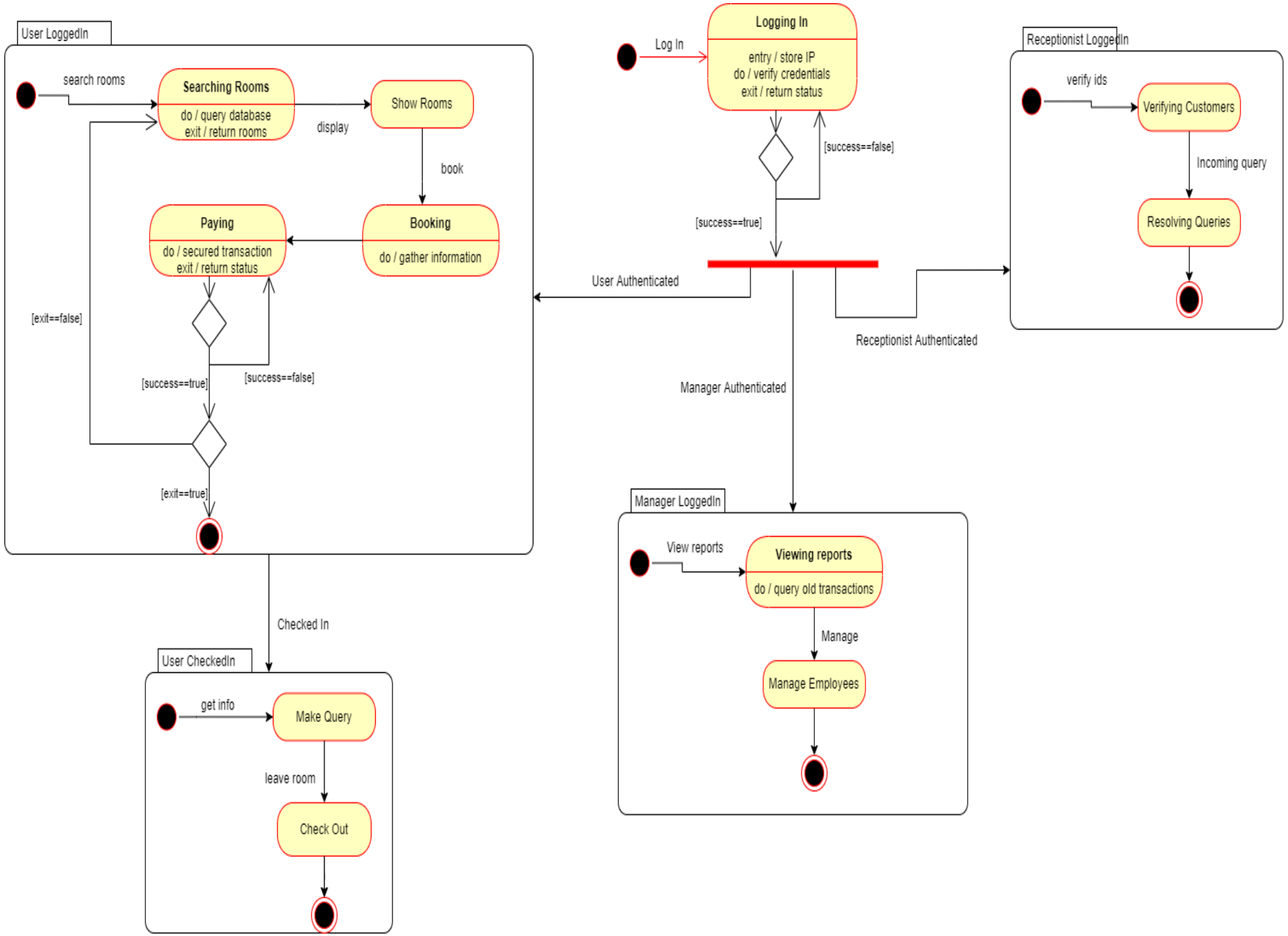
The occurrence of the transition also depends on the guard condition which is a boolean expression. The guard condition is checked only once when an event occurs and if it turns out true the transition takes place.

State Diagram

The state diagram in a state model represents the order of the events which causes the change in states sequentially. A state diagram is basically a graph, where every node in a graph denotes the state and every line connecting the nodes are the transitions that respond to an event and cause changes in the state.

Each state diagram in the state model is presented in the rectangular frame and the name of the corresponding state diagram is written in the pentagonal tag at the left corner of the rectangular frame. Guard conditions are optional are if required are written in square brackets just beside the events.

Diagram:


User LoggedIn
search rooms
Searching Rooms
do / query database
exit / return rooms
display
Show Rooms
book
Booking
do / gather information
Paying
do / secured transaction
exit / return status
[exit==false]
[success==true]
[success==false]
[exit==true]
Log In
Logging In
entry / store IP
do / verify credentials
exit / return status
[success==false]
[success==true]
User Authenticated
Receptionist Authenticated
Manager Authenticated
Receptionist LoggedIn
verify ids
Verifying Customers
Incoming query
Resolving Queries
Manager LoggedIn
View reports
Viewing reports
do / query old transactions
Manage
Manage Employees
Checked In
User CheckedIn
get info
Make Query
leave room
Check Out

Code:

```
import java.util.*;

public class States {
    public class Person {
        protected int id;
        protected int ph_no;
        protected String name;
        protected String username;
        protected String Password;
        protected String account_type;

        protected boolean login(String username, String password) {
            // .....
            if (username == this.username && password == this.Password)
                return true;

            return false;
        }
    };

    public class Customer extends Person {
        // following variables make up the userdata
        String Address;
        int UID_NO;
        int RoomNo;

        // boolean CheckIn(userdata){
        // //logic to save data to session storage or as cookies
        // if (data saved successfully){
        // return true;
        // }
        // return false; //some problem occured
        // }

        // void CheckOut(userdata){
        // //logic to remove user session data from device and browser
        // // call to display home screen;
        // }
```

```
Bill payBill() {
    // call to payment gateway for bill payment
    // bill = call to respective function of bill object
    Bill b = new Bill();
    if (b.Status()) {
        return b;
    }
    return null;
}

// int Book(userdata){
// //display all available rooms
// //attempt to book a room in the hotel
// //call to payBill function
// boolean roomBooked = book provided room;
// if(roomBooked){
// return 1;
// }
// else{
// return -1; //server error
// }
// }

// int CancelBooking(userdata){
// //search for user and remove the booking
// //apply the cancellation charges according to rules
// boolean userFound = find user;
// boolean dataRemoved = removeData;
// if(userFound && dataRemoved){
// return amount; //amount to be returned to user
// }
// else{
// return -1; //error
// }
// }

// boolean UpdateProfile(userdata){
// //search for a user
// if(user found){
```

```
        // //update data according to what user wants
        // return true;
        // }
        // else{
        // return false; //user doesn't exist
        // }
        // }

        // void ViewAvailability(roomData){
        // //search for rooms according to roomData provided
        // // boolean roomsfound = search for rooms;
        // if(roomsfound){
        // //display the rooms
        // if(interested){
        // //call for booking
        // }
        // }
        // else{
        // //show next available booking period
        // //call for book function
        // }
        // }
    };

    public class Manager extends Person {
        public void Record_complaints(Customer c) {
            // Store complaints from a customer
            Receptionist rep = new Receptionist();
            System.out.println(rep.Address_Queries(c));
        }

        public void View_Reports() {
            // View financial reports using Billing information
            Rooms R = new Rooms();
            for (Customer c : R.past_customers()) {
                System.out.println(c.payBill());
            }
        }

        public void Manage_Employees() {
```

```
            // Manage Employee leaves, payment, work assignment
        }
    }

    public class Receptionist extends Person {
        public void Accept_Customer_Feedback(Customer c) {
            // Accept the feedback and store it in database
            // along with customer id

        };

        public Boolean Verify_Customer(Customer c) {
            Customer cust = c;
            class Local {
                public Boolean verifiedCustomer(Customer cust) {
                    // Check for the Documents and booking details from
customer
                    // Check CheckIn date and time
                    // Check for advance payment

                    Boolean found = false;
                    if (found == true) {
                        return true;
                    }
                    return false;
                }
            }
            if (new Local().verifiedCustomer(cust) == true) {
                return true;
            }
            return false;
        };

        public String Address_Queries(Customer c) {
            String query_status = "Not_resolved!";
            Boolean resolved_Query = false;
            // try to resolve the query ..acordingly update the customer.
            if (resolved_Query) {
                query_status = "Your query has been resolved !";
            } else {
```

```
                query_status = "Sorry for inconveience, we will resolved
your query shortly !";
            }
            // return the status of query resolved or not or in process.
            return query_status;
        };

    }

    enum Status {
        Booked, Pending, Cancelled, CheckIn, CheckOut
    }

    public class RoomBooking {
        private String reservation_no;
        private Date StartDate;
        private int Duration_in_days;

        private Status st;

        private Date CheckIN;
        private Date CheckOut;

        public RoomBooking fetchDetails(String reservation_no) {
            RoomBooking r1 = new RoomBooking();

            return r1;
        }
    }

    public class Rooms {

        private int room_no;
        private String room_type;
        private String location;
        private int Room_price;
        private Date Last_Maintenance;

        public ArrayList<Customer> past_customers() {
            // Past customers of this Room
```

```
        ArrayList<Customer> arr = new ArrayList<Customer>(10);
        return arr;
    }
}

public class Bill {
    protected int BillID;
    protected boolean PaymentStatus;
    protected String TransactionID;
    protected Date date;
    protected int TotalAmount;

    boolean Status() {
        // implement the payment gateway
        // PaymentStatus = call to gateway;
        if (PaymentStatus == true) {
            // create the bill
            // show bill and allow printing and saving
            return true;
        } else {
            // ask user to try again
            return false;
        }
    }
};

public class CashTransaction extends Bill {
    String Name_on_card;
    String Zipcode;
}

public class CreditCardTransaction extends Bill {
    int cash_tendered;
}

public static void main(String[] args) {
    Rooms R[] = new Rooms[10];
    Person cust = new Customer();

    for (int i = 0; i < 10; i++) {
```

```
            R[i] = new Rooms();
        }

        R[0].room_no = 101;
        R[0].room_type = "Single";
        R[0].location = "A";
        R[0].Room_price = 1000;
        R[0].Last_Maintenance = new Date();

        cust.updateProfile("id: 101,"+
        "ph_no: 9876543210,"+
        "name: 'John',"+
        "username: 'john',"+
        "Password: 'john123',"+
        "account_type : Customer,");

        cust.ViewAvailability(R[0]);
        cust.bookRoom(R[0]);
        cust.cancelBooking(R[0]);

        cust.bookRoom(R[1]);
        cust.CheckIn(R[1], cust);

        Receptionist r = new Receptionist();
        r.Verify_Customer(cust);
        r.Address_Queries(cust);
        r.Accept_Customer_Feedback(cust);

        Manager m = new Manager();
        m.View_Reports();
        m.Manage_Employees(r);
        m.Record_complaints(cust);
    }
};
```