**NEIL GOGTE INSTITUTE OF TECHNOLOGY**

(​**A Unit of Keshav Memorial Technical Education (KMTES)** **(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad).**

**A**

**MINI PROJECT REPORT**

***on***

**HOTEL MANAGEMENT SYSTEM**

**BACHELOR OF ENGINEERING**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

**Submitted by**

|  |  |
| --- | --- |
| **S Lakshmidhar** | **: 2453-18-733-049** |
| **G V S Krishna Hrithik** | **: 2453-18-733-019** |
| **Yalla Satya Krishna Vamsee** | **: 2453-18-733-059** |

**Under the Guidance of**

**Dr. Md. Imran**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**Neil Gogte Institute of Technology**

Kachawanisingaram Village, Hyderabad, Telangana 500058.

**March 2021**

**NEIL GOGTE INSTITUTE OF TECHNOLOGY**

(​**A Unit of Keshav Memorial Technical Education (KMTES)** **(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad).**

**CERTIFICATE**



This is to certify that the project report titled “​**Hotel Management System**​” ​is being submitted by ​**S Lakshmidhar**​ (​**2453-18-733-049**​), **G V S Krishna Hrithik (2453-18-733-019)** ​, **Yalla Satya Krishna Vamsee (2453-18-733-059)** ​of III-year B.E.V Semester​**Computer Science and Engineering** ​is a record of bonafide work carried out by them.The results embodied in this report have not been submitted to any other University for the award of any degree.

**Internal Guide** **HOD**

**External Examiner**

**NEIL GOGTE INSTITUTE OF TECHNOLOGY**

(​**A Unit of Keshav Memorial Technical Education (KMTES)** **(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad).**

**DECLARATION**



We hereby declare that the Mini Project Report entitled, “​**Hotel Management System**​”​submitted for the B.E. degree is entirely ourwork and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree.

**Date:**

**S Lakshmidhar**

**2453-18-733-049**

**G V S Krishna Hrithik**

**2453-18-733-019**

**Yalla Satya Krishna Vamsee**

**2453-18-733-059**

**NEIL GOGTE INSTITUTE OF TECHNOLOGY**

(​**A Unit of Keshav Memorial Technical Education (KMTES)** **(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad).**

**ACKNOWLEDGEMENT**



We are happy to express our deep sense of gratitude to the principal of the college **Dr D Jaya Prakash,** ​Professor, Neil Gogte Institute of Technology, for havingprovided us with adequate facilities to pursue our project.

We would like to thank ​**Dr K V Ranga Rao,** ​Professor and Head, Department of Computer Science and Engineering, ​Neil Gogte Institute of Technology, for having provided the freedom to use all the facilities available in the department, especially the laboratories and the library.

We are very grateful to our project guide ​**Dr. Md. Imran,** ​Associate Professor in Computer Science and Engineering, Neil Gogte Institute of Technology, for his extensive patience and guidance throughout our project work.

We sincerely thank our seniors and all the teaching and non-teaching staff of the Department of Computer Science & Engineering and Information Technology for their timely suggestions, healthy criticism and motivation during this work.

We would also like to thank classmates for always being there whenever we needed help or moral support. With great respect and obedience, We thank our parents, sisters and brother who were the backbone behind our deeds.

Finally, We express our immense gratitude with pleasure to the other individuals who have either directly or indirectly contributed to our need at the right time for the development and success of this work.

**S LAKSHMIDHAR**

**G V S KRISHNA HRITHIK**

**YALLA SATYA KRISHNA VAMSEE**

**Hotel Management System- Abstract**

Hotel Management System is a fully functional **Desktop Application** designed keeping hotel receptionist in the mind when he wants to reserve for room to customer for temporal purposes.

Even today, with many of features that technology offers, many of the hotel still maintain books of records which won’t sustain very long as time goes on. Here efficiency of storing the information is one of key problems. The second key problem to be mentioned is that continuous corrections or updating records won’t be easy in case of books (i.e.) continuously updating on the books won’t be good idea though.

A solution to above mentioned problems is an Online Hotel Management system, which will help the managers to continuously monitor, update, maintain their records without much hassle (i.e.) sitting at one place. This system makes it convenient for the admin to add customer details and time of his stay in the hotel. It helps receptionist as well to easily identify the clean and empty rooms to allot for the customer. The software will help in easy updating of the information in the database, making admin and receptionist life easier to manage the vast facilities present in the hotel.

**NEIL GOGTE INSTITUTE OF TECHNOLOGY**

(​**A Unit of Keshav Memorial Technical Education (KMTES)** **(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad).**

**CONTENTS**



|  |  |  |  |
| --- | --- | --- | --- |
| **S NO.** |  | **TITLE** | **PAGE NO.** |
|  | **CERTIFICATE** | | **ii** |
|  | **DECLARATION** | | **iii** |
|  | **ACKNOWLEDGEMENTS** | | **iv** |
|  | **ABSTRACT** | | **v** |
|  | **TABLE OF CONTENTS** | | **vi** |
|  | **LIST OF FIGURES** | | **viii** |
| **1** | **INTRODUCTION** | |  |
|  | 1.1 | Motivation | 1 |
|  | 1.2 | Problem Statement | 1 |
|  | 1.3 | Solution | 1 |
|  | 1.4 | Objective | 2 |
|  | 1.5 | Limitations | 2 |
| **2** | **LITERATURE SURVEY** | |  |
|  | 2.1 | Surveys | 3 |
|  | 2.2 | Existing System | 3 |
|  | 2.3 | Disadvantages of Existing System | 3 |
|  | 2.4 | Proposed System | 4 |
|  | 2.5 | Advantages of Proposed System | 4 |
|  | 2.6 | Conclusion | 4 |

vi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **3** | **ANALYSIS** | |  |  |
|  | 3.1 | Software and Hardware requirements | 5 |  |
|  | 3.2 | Context Diagram |  |  |
|  |  | a) Flow Diagram | 6 |  |
|  |  |  |  |
|  | 3.3 Data Collection and preparation | | 8 |  |

|  |  |  |
| --- | --- | --- |
| **4** | **DESIGN** |  |
|  | 4.1 Introduction | 9 |
|  | 4.2 Class Diagram | 10 |
|  | 4.3 Use Case Diagram | 11 |
|  | 4.4 Activity Diagram | 12 |
|  | 4.5 Sequence Diagram | 13 |
|  | 4.6 ER Diagram | 14 |

|  |  |  |  |
| --- | --- | --- | --- |
| **5** | **IMPLEMENTATION** | |  |
|  | 5.1 | Implementation Steps | 15 |
|  | 5.2 | Source code | 15 |
| **6** | **TESTING AND VALIDATION** | |  |
|  | 6.1 | Project Screenshots | 33 |
|  | 6.2 | Result | 40 |
| **7** | **LIST OF TABLES AND FIGURES** | | 41 |
| **8** | **CONCLUSION AND FUTURE ENHANCEMENTS** | | 43 |

|  |  |  |
| --- | --- | --- |
| **9** | **REFERENCES** | 44 |

**INTRODUCTION**

**1.1 Motivation:**

Even in 2021, Most of the Hotels in our country use a manual system for the management and maintenance of critical information. The current system requires numerous paper forms which tend to age very badly. More often than not, information stored on them is either incomplete or does not follow management standards.

Papers or ledgers are also often lost in transit between various departments requiring a comprehensive auditing process to ensure that no vital information is lost.

Multiple copies of the same information exist in the manual ledgers and this may lead to inconsistencies in data in various data stores. This also hampers the efficiency of other people working in these hotels.

Using our **Hotel Management System**, A structured system can be enabled that will ensure perfection in daily work routine and also restore the consistent performance of the staff due to the systematic, organized approach of our system.

**1.2**​ ​**Problem Introduction:**

Most Hotels make use of various accounting books and diaries to keep a track of all the management. This includes every peon, cleaner, driver and even the high-level management Staff. It creates a system of confusion and disturbance as the Hotel grows its customers. Having a Digital record is much better solution to all the problems posed by this condition.

**1.3 Solution:**

The solution is to bring various operations performed on paper to a single platform that is online and can be scaled better than that on paper. This saves time and is a much better option than manual entry.

**1.4 Objective:**

To develop a modern application that will aid hotel managers.

To offer flexibility and scalability in hotel management. To enable those without any technical knowledge to operates the application. To provide for real-time overall effective, efficient and accurate hotel management in the local areas that are in line with global hotel management standards.

A

**1.5 Limitations:**

In the course of this study, a major constraint experienced was the incorrect system requirements. Others include the inevitability of human error and bias as some information were obtained via interpersonal interactions, interviews and research, making some inconsistent.

cities​. This is where NGIT Study

**LITERATURE SURVEY**

**2.1 Surveys:**

The first step in our survey is the identification of need of change to improve or enhance an existing system of manual entry. An initial investigation on existing system was carried out. The present system of hotel management is completely manual. Many problems were identified during the initial study of the existing system.

**2.2 Existing System:**

The existing system of hotel management makes extensive use of manual data entry. This has been observed even in most high-end hotels according to our survey.

The hotel manager makes note of all the staff’s attendance, their allocated workloads, customer entries, their room numbers, room occupancy status and the type of beds that are present in the hotel.

**2.3 Disadvantages of Existing System:**

*Lack of immediate updating:* The information that is manually updated is very difficult to update instantly as too much paper work is involved.

*Lack of immediate feedback:* The information generated by various customers takes time and efforts to be stored manually at one place.

*Error prone manual calculation:* Manual calculations are error prone and take a lot of time this may result in incorrect information.

*Lack of accountability:* Manual entries make both customer requests and employee’s concerns hard to implement.

*Error caused due to aging:* Paper scripts can be washed away, torn that causes loopholes to be generated in the hotel management system.

**2.4 Proposed System:**

Hotel Management System is designed for hotel of any size to replace their existing manual paper-based system. The new system is to control the information of hotel staff, customers and drivers. These services are to be provided in an efficient, cost effective manner, with the goal of reducing the time and resources currently required for such tasks.

This system provides online storage/updating facility. This system promises very less or no paper work and also provides help to the operational staff.

In this system everything is stored electronically so very less amount of paper work is required and information can be retrieved very easily without searching here and there into registers.

**2.5 Advantages of Proposed System:**

The advantages of the proposed project are that the online system simply automates hotel management system. It is faster and reliable, provides better services as compare to manual system. This system may increase the profit for diagnosis labs. The user friendly and interactive interface makes using this application easy for everyone. There is almost zero percent chance of report swapping or missing which has quite fair chances in manual system. The web version of the proposed application saves users to visit website again and again, they can check all details on finger tips.

**2.6 Conclusion:**

As our proposed system is designed for both small-scale and large-scale organization, it doesn’t require much resource at the users end. Countries like Sri Lanka is stepping towards the computerized management day by day but still most of our country is deprived of basic computer facility and high-speed internet connection as well as lack of skilled man system.

It is faster and reliable, provides better services as compare to manual system. This system power to operate this kind of sophisticated system.

As our goal is to establish this efficient system not only in urban hotels but also in rural hotels, we design a simple but elegant system that will require minimum resource and gives effective and insured results.

Below we describe our requirement for the system to run.

**ANALYSIS**

|  |  |
| --- | --- |
| **3.1 Software and Hardware requirements** |  |
| **Software requirements:** |  |
| OS: Windows 10 Home and Above |  |
| Java Editor: VS Code / Eclipse / NetBeans  Software: JDK 8 or Above & MySQL Database |  |
| **Hardware requirements:** |  |
| Processor: Intel® Core ™ i3-1003U CPU @ 2.00GHz. |  |
| RAM: 2.00 GB or Above  ROM: 200 GB or Above  System Type: 64-bit/32-bit OS & x64/x86-based Processor |  |

**Front End Technologies:**

JAVA AWT along with JAVA SWING API

**Back End Technologies:**

MYSQL Database

* The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages. The next section, What Can Java Technology Do? highlights what functionality some of the packages in the Java API provide.
* The following figure depicts a program that’s running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.



**Java Database Connectivity (JDBCTM)**: Provides uniform access to a wide range of relational databases. It has been used in our project in order to connect MySQL Database with Java Frontend.

**Java Swing API** is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

**Java AWT** (Abstract Window Toolkit) is an API to develop GUI or window-based applications in java.

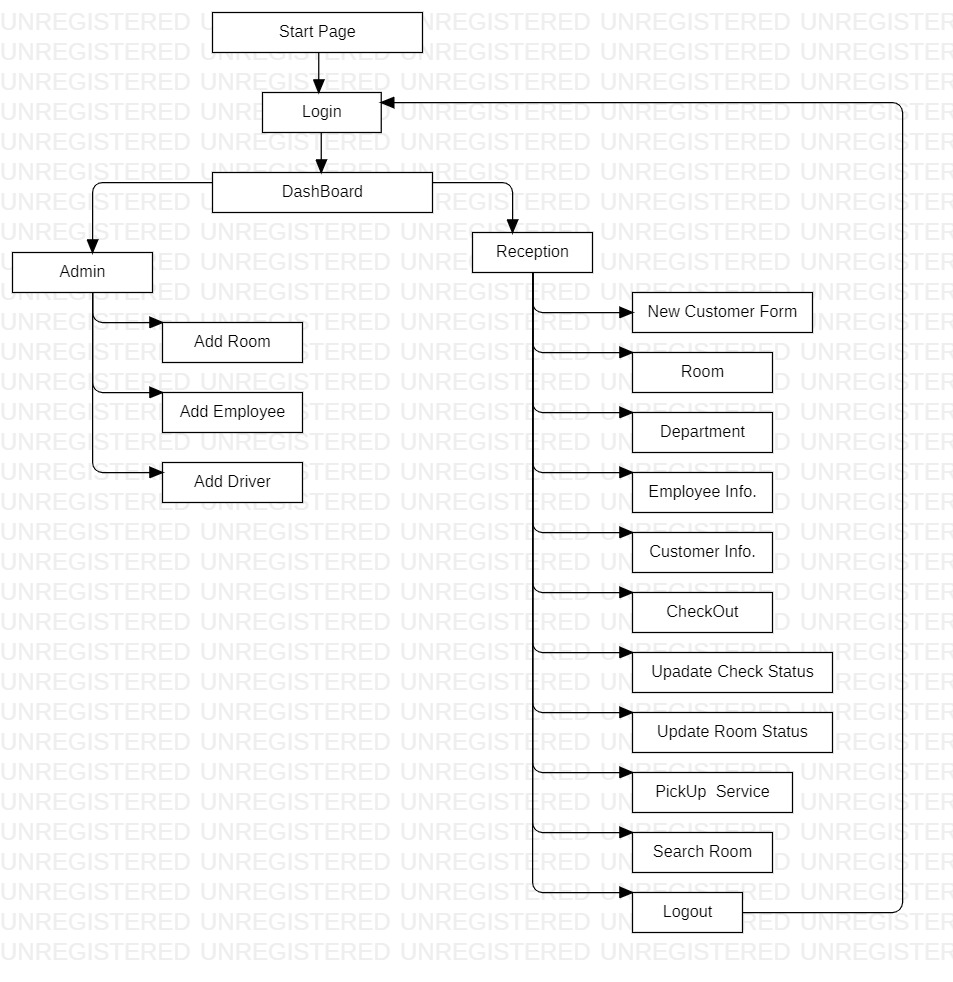
Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components are using the resources of OS.

**MySQL** is the most popular Open Source Relational SQL Database Management System. MySQL is one of the best RDBMS being used for developing various web-based software applications. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company.

* Open Sourced
* Very Customizable
* Works with ease in case of large databases
* Works on many OS and programming languages too
* Standard form of SQL database programming language

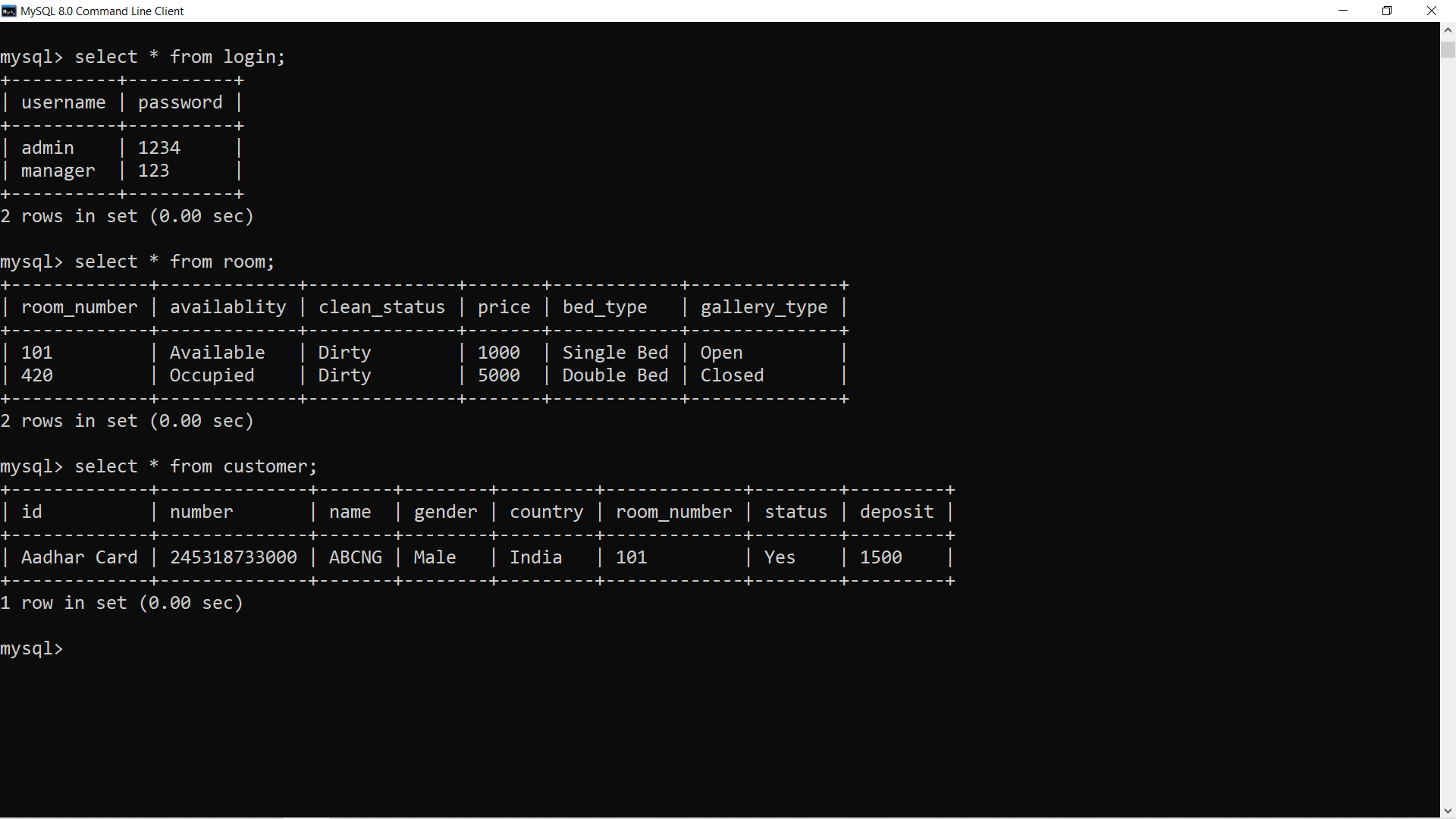
**3.2 Content Diagram**

Flow diagram



**3.3 Database Collection and Preparation**





**DESIGN**

**4.1 Introduction**

**UML Design**

The Unified Modelling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the software system and its components. It is a graphical language, which provides a vocabulary and set of semantics and rules. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. It is used to understand, design, configure, maintain, and control information about the systems.

The UML is a language for:

å​ Visualizing​

å​ Specifying​

å​ Constructing​

å​ Documenting​

**Visualizing**

Through UML we see or visualize an existing system and ultimately we visualize how the system is going to be after implementation. Unless we think, we cannot implement. UML helps to visualize how the components of the system communicate and interact with each other.

**Specifying**

Specifying means building, models that are precise, unambiguous and complete UML addresses the specification of all the important analysis design, implementation decisions that must be made in developing and deploying a software system.

**Constructing**

UML models can be directly connected to a variety of programming language through mapping a model from UML to a programming language like JAVA or C++ or VB. Forward Engineering and Reverse Engineering is possible through UML.

**Documenting**

The Deliverables of a project apart from coding are some Artifacts, which are critical in controlling, measuring and communicating about a system during its developing requirements, architecture, desire, source code, project plans, tests, prototypes releasers, etc...

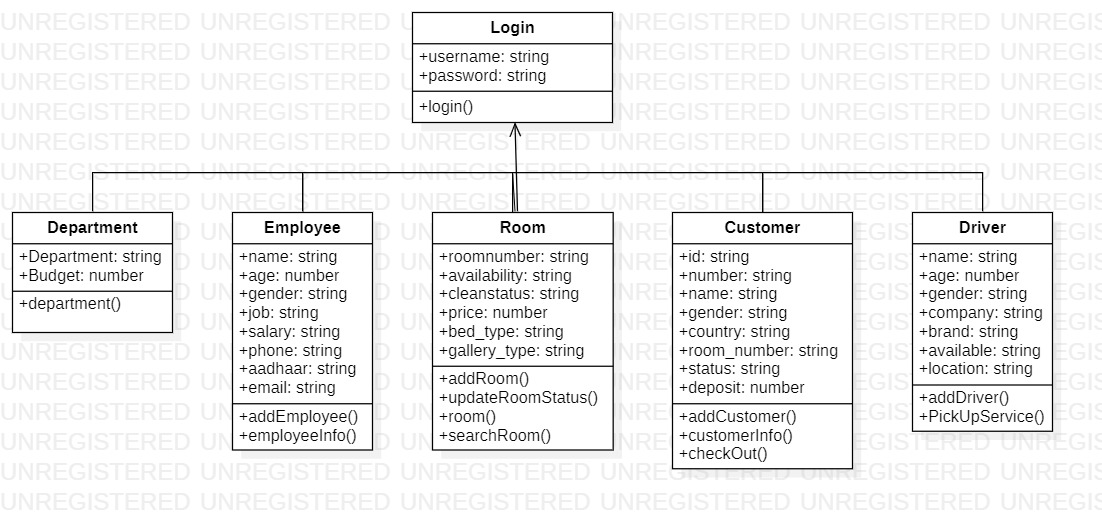
**4.2 Class Diagram**

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

Class diagram of our project:

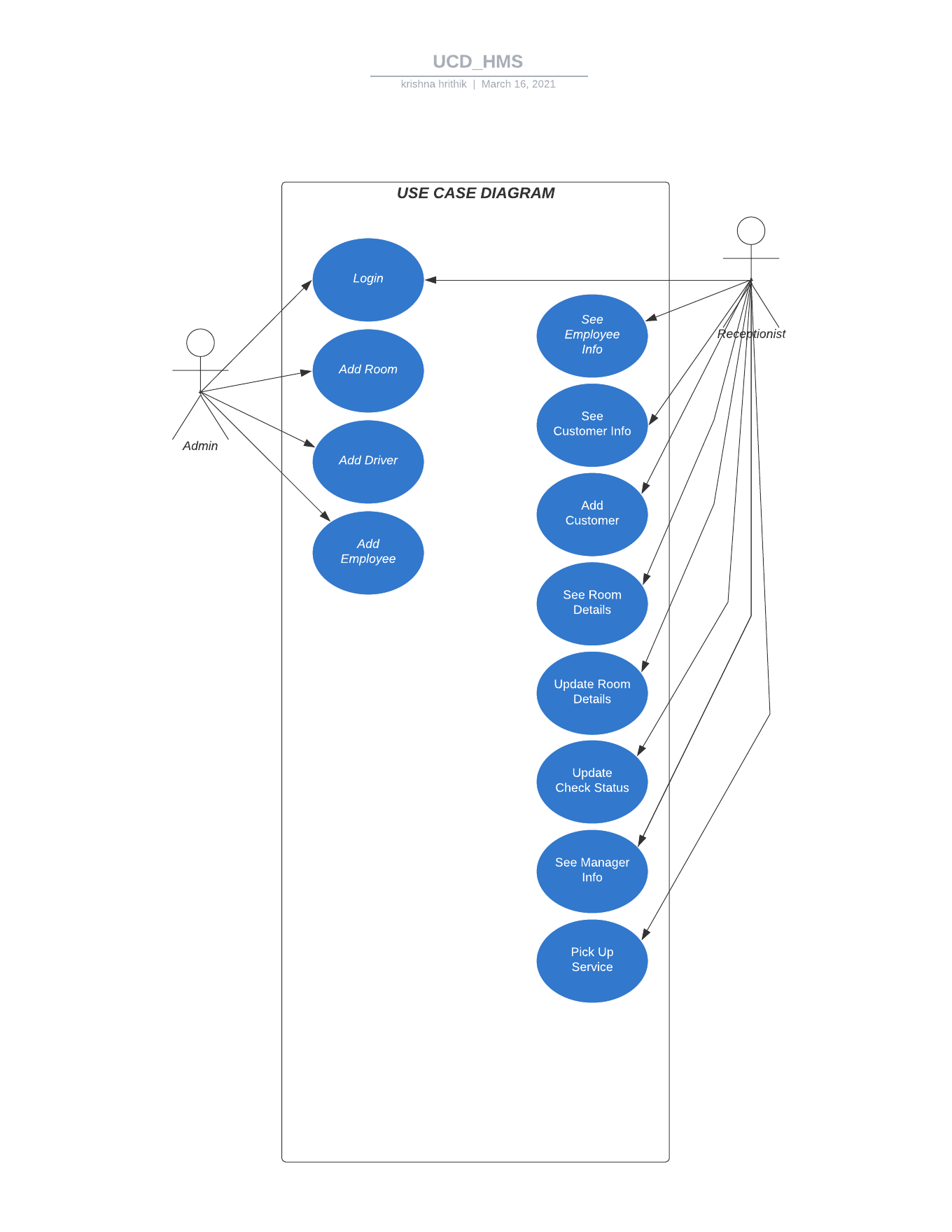


**4.3 Use Case diagram**

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

Use case diagrams are formally included in two modelling languages defined by the “The Unified Modelling language” (UML) and “The Systems Modelling language (sysML).

Use case diagram of our project:

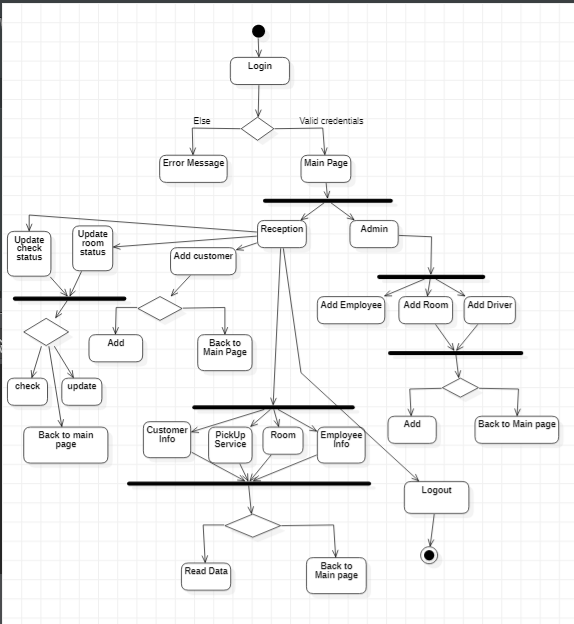
****

**4.4 Activity diagram**

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

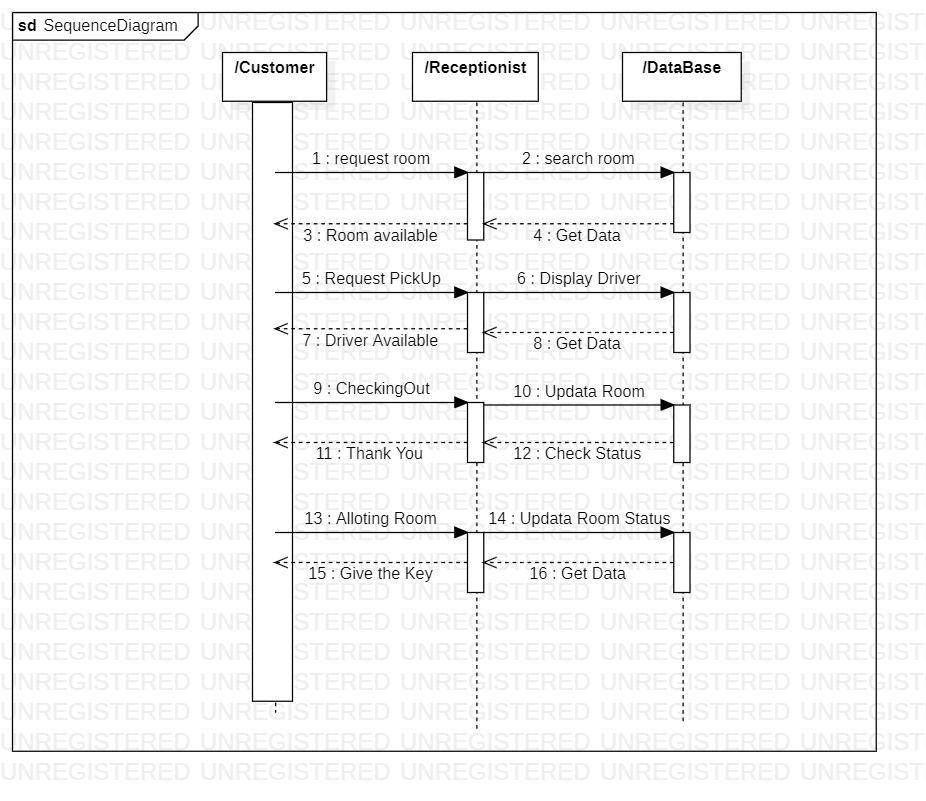
The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc



**4.5 Sequence diagram**

A Sequence Diagram is an interaction diagram that emphasizes the time ordering of messages; a collaboration diagram is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages. Sequence diagrams and collaboration diagrams are isomorphic, meaning that you can take one and transform it into the other.

Sequence diagram of our project:



**4.6 ER diagram**

Entity-Relationship Diagram is a graphical representation of entities and their relationship to each other’s. It describes how data is related to each other. An entity is a piece of data - an object or a concept about which data is stored. A relationship is how the data is shared between entities. In E-R Diagram, there are 3 main Components:



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Symbol** | | **Name** |  | **Description** |  |  |
|  |  |  |  |  | | |  |
|  |  |  | Entity | An entity can be any object, | | |  |
|  |  |  |  |
|  |  |  | place, person or anything. | | |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  | An | Attribute Describes a | |  |
|  |  |  | Attribute | property or characteristics of an | | |  |
|  |  |  |  | entity. | |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  | A | Relationship | Describes |  |
|  |  |  | Relationship | relation between entities. | | |  |
|  |  |  |  |  |  |  |  |

**SYSTEM IMPLEMENTATION**

**Introduction:**

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

**5.1 Implementation steps:**

* Install JDK 8 or Above and MySQL
* Create and execute the maven project
* Home page (App.java) is appeared.
* Write source code (Conn.java) for connecting the java pages to MySQL backend.
* Enter credentials in order to login.
* To view the data stored in the database
  + Use <database\_name>;
  + Show tables;
  + Select \* from <table\_name>;

**5.2 Source Code:**

1. **App.java**

package hotelSystem.version1;

import javax.swing.\*;

import javax.swing.plaf.FontUIResource;

import java.awt.\*;

import java.awt.event.\*;

public class App extends JFrame implements ActionListener

{

    App(){

        setBounds(50, 50, 1200, 800);

        ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("hotelSystem\\version1\\icons\\first.jpg"));

        JLabel j1 = new JLabel(i1);

        j1.setBounds(0, 0, 1200, 800);

        add(j1);

        JLabel j2 = new JLabel("HOTEL MANAGEMENT SOFTWARE");

        j2.setBounds(620, 550, 1000, 70);

        j2.setForeground(Color.MAGENTA);

        j2.setFont(new FontUIResource("Arial", Font.PLAIN, 30));

        j1.add(j2);

        JButton b1 = new JButton("Next");

        b1.setBackground(Color.WHITE);

        b1.setForeground(Color.BLACK);

        b1.setBounds(810, 500, 300, 30);

        b1.addActionListener(this);

        j1.add(b1);

        setLayout(null);

        setVisible(true);

        while(true){

            j2.setVisible(false);

            try{

                Thread.sleep(500);

            }catch(Exception e){

            }

            j2.setVisible(true);

            try{

                Thread.sleep(500);

            }catch(Exception e){

            }

        }

    }

    public void actionPerformed(ActionEvent ae){

        new Login().setVisible(true);

        this.setVisible(false);

    }

    public static void main(String[] args)

    {

        new App();

    }

}

1. **Conn.java**

package hotelSystem.version1;

import java.sql.\*;

public class Conn {

    Connection c;

    Statement s;

    Conn(){

        try {

            Class.forName("com.mysql.cj.jdbc.Driver");

            c = DriverManager.getConnection("jdbc:mysql:///projecthms", "root", "1234");

            s = c.createStatement();

        }catch (Exception e) {

            e.printStackTrace();

        }

    }

}

1. **Login.java**

package hotelSystem.version1;

import java.awt.Color;

import java.awt.Image;

import java.awt.event.\*;

import java.sql.ResultSet;

import javax.swing.\*;

public class Login extends JFrame implements ActionListener{

    JLabel l1, l2;

    JButton b1, b2;

    JTextField t1;

    JPasswordField t2;

    Login(){

        l1 = new JLabel("Username");

        l1.setBounds(40, 20, 70, 30);

        add(l1);

        l2 = new JLabel("Password");

        l2.setBounds(40, 80, 70, 30);

        add(l2);

        t1 = new JTextField();

        t1.setBounds(120, 20, 100, 30);

        add(t1);

        t2 = new JPasswordField();

        t2.setBounds(120, 80, 100, 30);

        add(t2);

        b1 = new JButton("Login");

        b1.setBackground(Color.WHITE);

        b1.setForeground(Color.BLACK);

        b1.setBounds(40, 140, 70, 45);

        b1.addActionListener(this);

        add(b1);

        b2 = new JButton("Cancel");

        b2.setBackground(Color.WHITE);

        b2.setForeground(Color.BLACK);

        b2.setBounds(130, 140, 90, 45);

        b2.addActionListener(this);

        add(b2);

        ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("hotelSystem\\version1\\icons\\second.jpg"));

        Image i2 = i1.getImage().getScaledInstance(150, 150, Image.SCALE\_DEFAULT);

        i1 = new ImageIcon(i2);

        JLabel l3 = new JLabel(i1);

        l3.setBounds(280, 20, 150, 150);

        add(l3);

        getContentPane().setBackground(Color.WHITE);

        setLayout(null);

        setBounds(500, 300, 500, 250);

        setVisible(true);

    }

    public void actionPerformed(ActionEvent ae){

        if(ae.getSource() == b1){

            try{

                Conn c = new Conn();

                String un = t1.getText(); // username

                StringBuilder pwd= new StringBuilder();

                char[] p = t2.getPassword(); // password

                for(char ch: p){

                    pwd.append(ch);

                }

                String qry = "select \* from login where username='"+un+"' and password='"+pwd.toString()+"'";

                ResultSet rs = c.s.executeQuery(qry);

                if(rs.next()){

                    this.setVisible(false);

                    new Dashboard().setVisible(true);

                }else{

                    JOptionPane.showMessageDialog(null, "Invalid login");

                    this.setVisible(false);

                    // new App().setVisible(true);

                }

            }catch(Exception e){

            }

        }else if(ae.getSource() == b2){

            System.exit(0);

        }

    }

    public static void main(String[] args) {

        new Login();

    }

}

1. **Reception.java**

package hotelSystem.version1;

import javax.swing.\*;

import java.sql.\*;

import java.awt.event.\*;

import java.awt.\*;

public class Reception extends JFrame {

    private JPanel contentPane;

    public static void main(String[] args) {

        new Reception();

    }

    public Reception() {

        setBounds(530, 200, 850, 570);

        contentPane = new JPanel();

        setContentPane(contentPane);

        contentPane.setLayout(null);

        ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("hotelSystem\\version1\\icons\\fourth.jpg"));

        Image i3 = i1.getImage().getScaledInstance(500, 500, Image.SCALE\_DEFAULT);

        ImageIcon i2 = new ImageIcon(i3);

        JLabel l1 = new JLabel(i2);

        l1.setBounds(250, 30, 500, 470);

        add(l1);

        JButton btnNewCustomerForm = new JButton("New Customer Form");

        btnNewCustomerForm.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {

                    NewCustomer custom = new NewCustomer();

                    custom.setVisible(true);

                    setVisible(false);

                } catch (Exception e1) {

                    e1.printStackTrace();

                }

            }

        });

        btnNewCustomerForm.setBounds(10, 30, 200, 30);

        btnNewCustomerForm.setBackground(Color.BLACK);

        btnNewCustomerForm.setForeground(Color.WHITE);

        contentPane.add(btnNewCustomerForm);

        JButton btnNewButton = new JButton("Room");

        btnNewButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent arg0) {

                try {

                    Room room = new Room();

                    room.setVisible(true);

                    setVisible(false);

                } catch (Exception e) {

                    e.printStackTrace();

                }

            }

        });

        btnNewButton.setBounds(10, 70, 200, 30);

        btnNewButton.setBackground(Color.BLACK);

        btnNewButton.setForeground(Color.WHITE);

        contentPane.add(btnNewButton);

        JButton btnNewButton\_1 = new JButton("Department");

        btnNewButton\_1.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {

                    Department dept = new Department();

                    dept.setVisible(true);

                    setVisible(false);

                } catch (Exception e1) {

                    e1.printStackTrace();

                }

            }

        });

        btnNewButton\_1.setBounds(10, 110, 200, 30);

        btnNewButton\_1.setBackground(Color.BLACK);

        btnNewButton\_1.setForeground(Color.WHITE);

        contentPane.add(btnNewButton\_1);

        JButton btnNewButton\_2 = new JButton("All Employee Info");

        btnNewButton\_2.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {

                    Employee em = new Employee();

                    em.setVisible(true);

                    setVisible(false);

                } catch (Exception e1) {

                    e1.printStackTrace();

                }

            }

        });

        btnNewButton\_2.setBounds(10, 150, 200, 30);

        btnNewButton\_2.setBackground(Color.BLACK);

        btnNewButton\_2.setForeground(Color.WHITE);

        contentPane.add(btnNewButton\_2);

        JButton btnNewButton\_3 = new JButton("Customer Info");

        btnNewButton\_3.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {

                    CustomerInfo customer = new CustomerInfo();

                    customer.setVisible(true);

                    setVisible(false);

                } catch (Exception e1) {

                    e1.printStackTrace();

                }

            }

        });

        btnNewButton\_3.setBounds(10, 190, 200, 30);

        btnNewButton\_3.setBackground(Color.BLACK);

        btnNewButton\_3.setForeground(Color.WHITE);

        contentPane.add(btnNewButton\_3);

        JButton btnManagerInfo = new JButton("Manager Info");

        btnManagerInfo.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {

                    ManagerInfo mana = new ManagerInfo();

                    mana.setVisible(true);

                    setVisible(false);

                } catch (Exception e1) {

                    e1.printStackTrace();

                }

            }

        });

        btnManagerInfo.setBounds(10, 230, 200, 30);

        btnManagerInfo.setBackground(Color.BLACK);

        btnManagerInfo.setForeground(Color.WHITE);

        contentPane.add(btnManagerInfo);

        JButton btnNewButton\_4 = new JButton("Check Out");

        btnNewButton\_4.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                CheckOut check;

                try {

                    check = new CheckOut();

                    check.setVisible(true);

                    setVisible(false);

                } catch (SQLException e1) {

                    e1.printStackTrace();

                }

            }

        });

        btnNewButton\_4.setBounds(10, 270, 200, 30);

        btnNewButton\_4.setBackground(Color.BLACK);

        btnNewButton\_4.setForeground(Color.WHITE);

        contentPane.add(btnNewButton\_4);

        JButton btnNewButton\_5 = new JButton("Update Check Status");

        btnNewButton\_5.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {

                    UpdateCheck update = new UpdateCheck();

                    update.setVisible(true);

                    setVisible(false);

                } catch (Exception e1) {

                    e1.printStackTrace();

                }

            }

        });

        btnNewButton\_5.setBounds(10, 310, 200, 30);

        btnNewButton\_5.setBackground(Color.BLACK);

        btnNewButton\_5.setForeground(Color.WHITE);

        contentPane.add(btnNewButton\_5);

        JButton btnNewButton\_6 = new JButton("Update Room Status");

        btnNewButton\_6.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {

                    UpdateRoom room = new UpdateRoom();

                    room.setVisible(true);

                    setVisible(false);

                } catch (Exception s) {

                    s.printStackTrace();

                }

            }

        });

        btnNewButton\_6.setBounds(10, 350, 200, 30);

        btnNewButton\_6.setBackground(Color.BLACK);

        btnNewButton\_6.setForeground(Color.WHITE);

        contentPane.add(btnNewButton\_6);

        JButton btnPickUpSerice = new JButton("Pick up Service");

        btnPickUpSerice.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent arg0) {

                try {

                    PickUp pick = new PickUp();

                    pick.setVisible(true);

                    setVisible(false);

                } catch (Exception e) {

                    e.printStackTrace();

                }

            }

        });

        btnPickUpSerice.setBounds(10, 390, 200, 30);

        btnPickUpSerice.setBackground(Color.BLACK);

        btnPickUpSerice.setForeground(Color.WHITE);

        contentPane.add(btnPickUpSerice);

        JButton btnSearchRoom = new JButton("Search Room");

        btnSearchRoom.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {

                    SearchRoom search = new SearchRoom();

                    search.setVisible(true);

                    setVisible(false);

                } catch (Exception ss) {

                    ss.printStackTrace();

                }

            }

        });

        btnSearchRoom.setBounds(10, 430, 200, 30);

        btnSearchRoom.setBackground(Color.BLACK);

        btnSearchRoom.setForeground(Color.WHITE);

        contentPane.add(btnSearchRoom);

        JButton btnNewButton\_7 = new JButton("Log Out");

        btnNewButton\_7.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent ae) {

                try {

                    new Login().setVisible(true);

                    setVisible(false);

                } catch (Exception e) {

                    e.printStackTrace();

                }

            }

        });

        btnNewButton\_7.setBounds(10, 470, 200, 30);

        btnNewButton\_7.setBackground(Color.BLACK);

        btnNewButton\_7.setForeground(Color.WHITE);

        contentPane.add(btnNewButton\_7);

        getContentPane().setBackground(Color.WHITE);

        setVisible(true);

    }

}

1. **Dashboard.java**

package hotelSystem.version1;

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

public class Dashboard extends JFrame {

    public static void main(String[] args) {

        new Dashboard().setVisible(true);

    }

    public Dashboard() {

        super("HOTEL MANAGEMENT SYSTEM");

        setForeground(Color.CYAN);

        setLayout(null);

        ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("hotelSystem\\version1\\icons\\third.jpg"));

        Image i2 = i1.getImage().getScaledInstance(1950, 1000, Image.SCALE\_DEFAULT);

        ImageIcon i3 = new ImageIcon(i2);

        JLabel NewLabel = new JLabel(i3);

        NewLabel.setBounds(0, 0, 1950, 1000);

        add(NewLabel);

        JLabel HotelManagementSystem = new JLabel("THE TAJ GROUP WELCOMES YOU");

        HotelManagementSystem.setForeground(Color.WHITE);

        HotelManagementSystem.setFont(new Font("Tahoma", Font.PLAIN, 46));

        HotelManagementSystem.setBounds(600, 60, 1000, 85);

        NewLabel.add(HotelManagementSystem);

        JMenuBar menuBar = new JMenuBar();

        setJMenuBar(menuBar);

        JMenu HotelSystem = new JMenu("HOTEL MANAGEMENT");

        HotelSystem.setForeground(Color.BLUE);

        menuBar.add(HotelSystem);

        JMenuItem EmpDetails = new JMenuItem("RECEPTION");

        HotelSystem.add(EmpDetails);

        JMenu HotelSystemHello = new JMenu("ADMIN");

        HotelSystemHello.setForeground(Color.RED);

        menuBar.add(HotelSystemHello);

        JMenuItem EmpDetailshello1 = new JMenuItem("ADD EMPLOYEE");

        HotelSystemHello.add(EmpDetailshello1);

        EmpDetailshello1.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent ae) {

                try {

                    new AddEmployee().setVisible(true);

                } catch (Exception e) {

                }

            }

        });

        JMenuItem EmpDetailshello2 = new JMenuItem("ADD ROOMS");

        HotelSystemHello.add(EmpDetailshello2);

        EmpDetailshello2.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent ae) {

                try {

                    new AddRoom().setVisible(true);

                } catch (Exception e) {

                }

            }

        });

        EmpDetails.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent ae) {

                new Reception();

            }

        });

        JMenuItem EmpDetailshello3 = new JMenuItem("ADD DRIVERS");

        HotelSystemHello.add(EmpDetailshello3);

        EmpDetailshello3.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent ae) {

                try {

                    new AddDrivers().setVisible(true);

                } catch (Exception e) {

                }

            }

        });

        setSize(1950, 1090);

        setVisible(true);

        getContentPane().setBackground(Color.WHITE);

    }

}

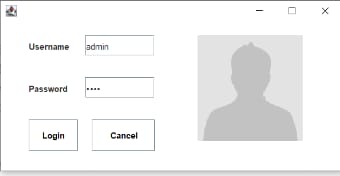
**TESTING AND VALIDATION**

**6.1 Project Screenshots**

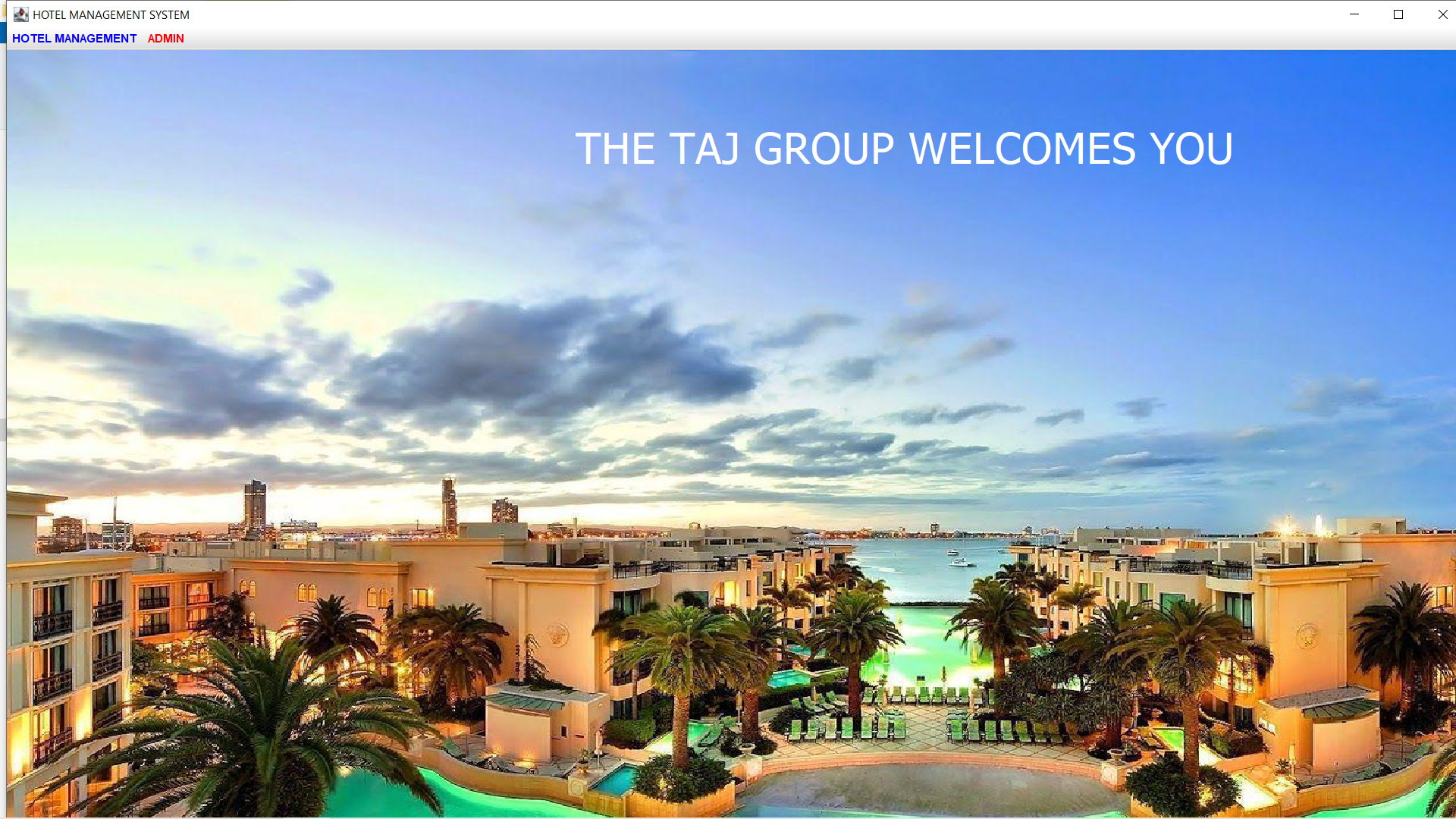
1. **App.java**



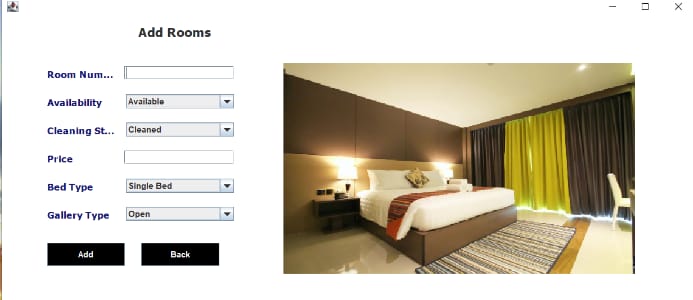
1. **Login.java**



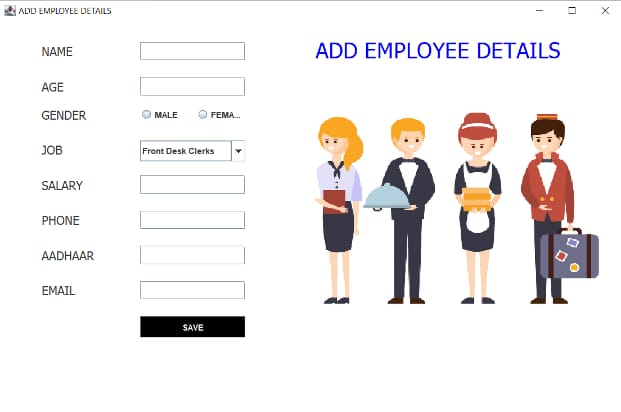
1. **Dashboard.java**



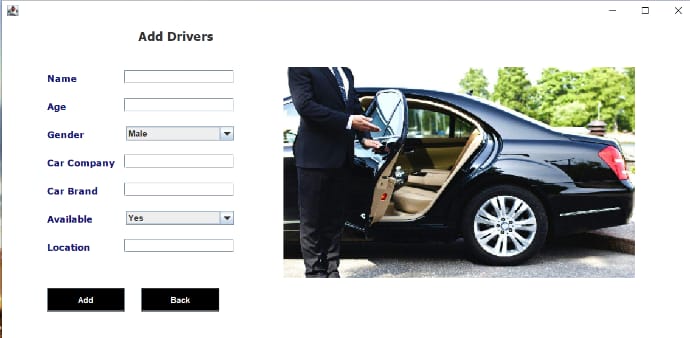
1. **addRoom.java**



1. **addEmployee.java**



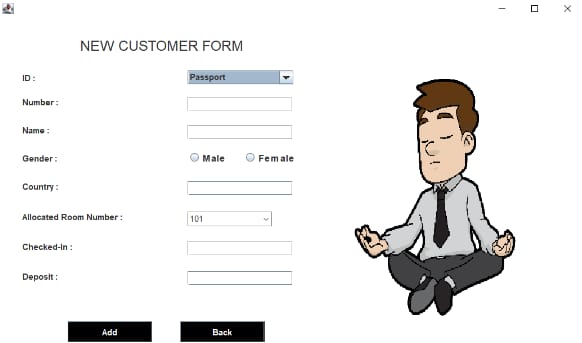
1. **addDriver.java**



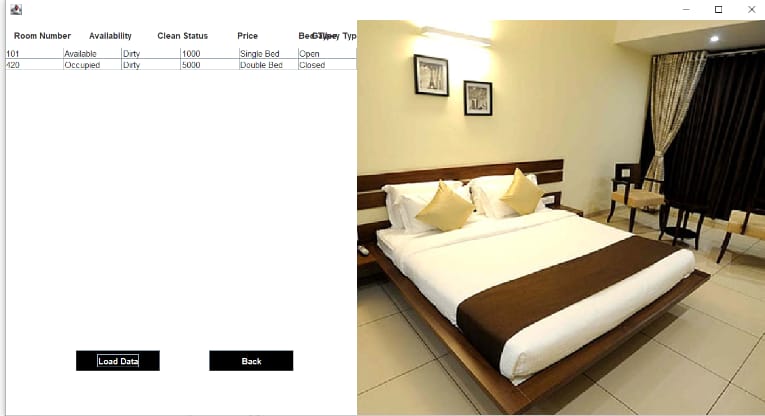
1. **Reception.java**



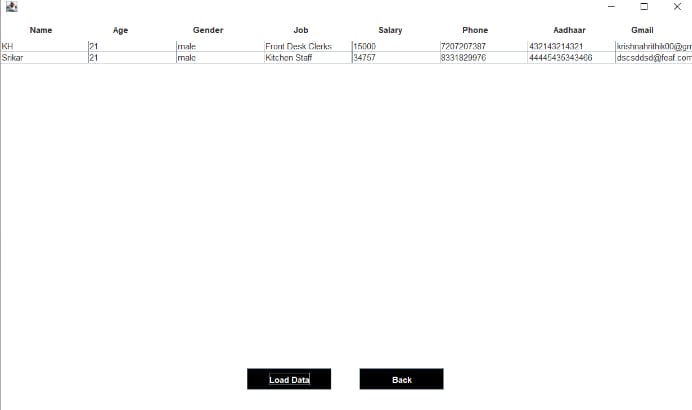
1. **NewCustomer.java**



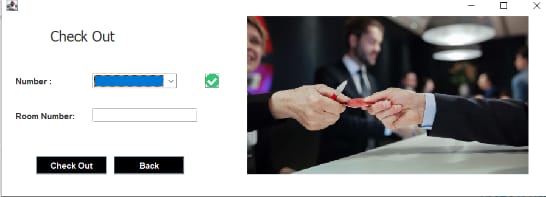
1. **Room.java**



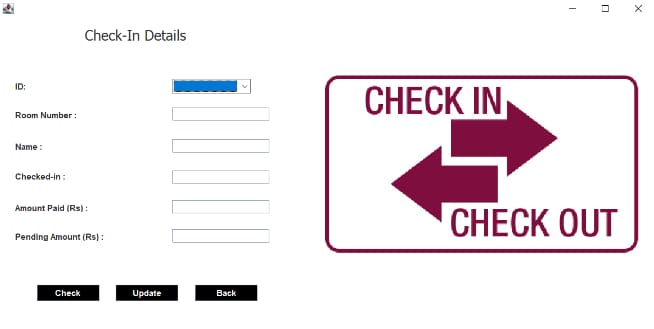
1. **EmployeeInfo.java**



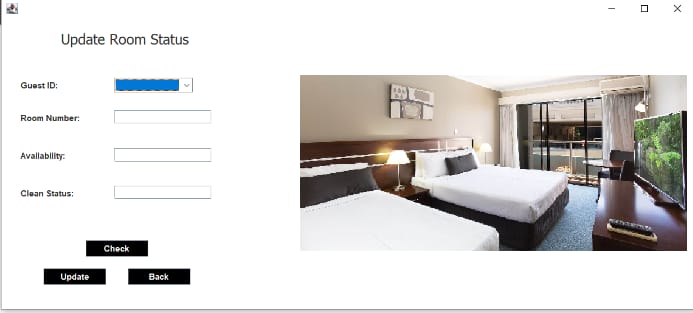
1. **CheckOut.java**



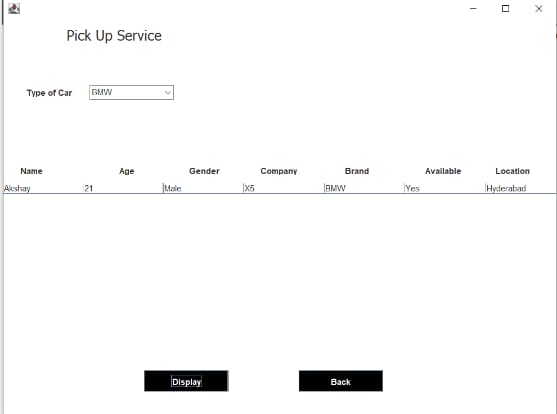
1. **updateCheckStatus.java**



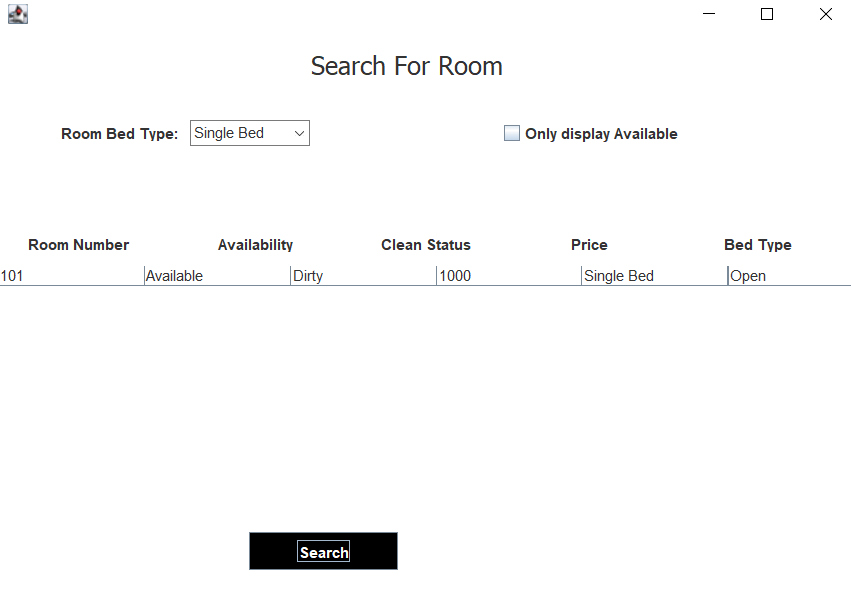
1. **updateRoomStatus.java**



1. **PickUpService.java**

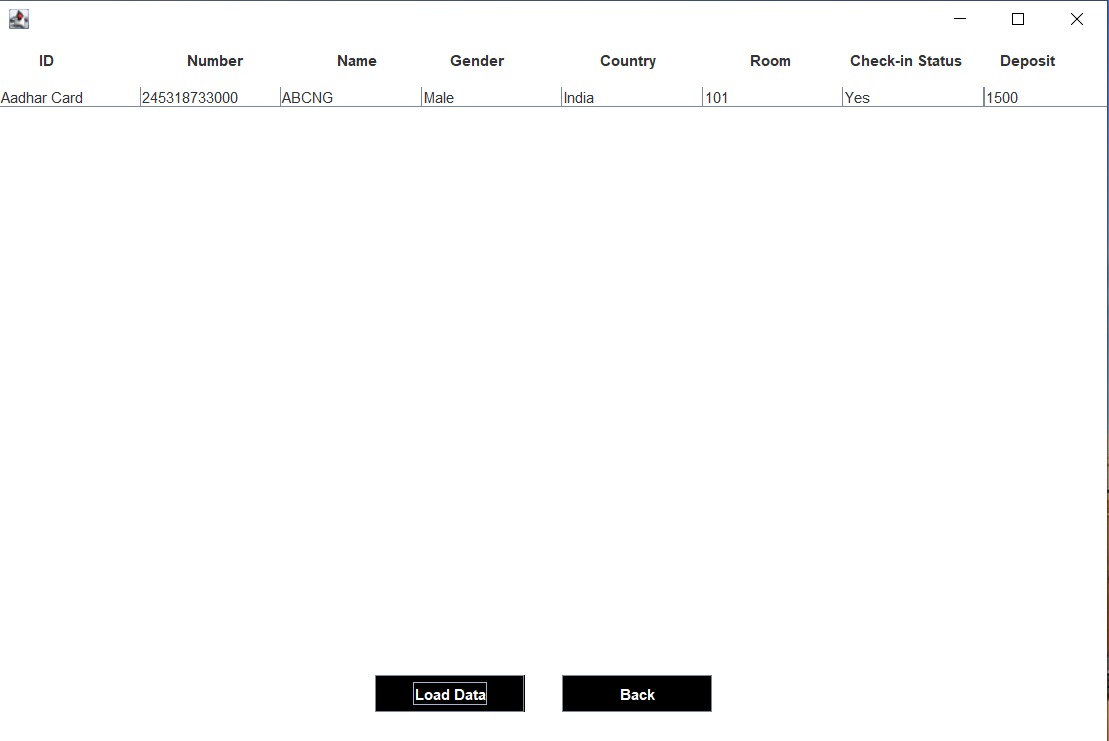


1. **SearchRoom.java**

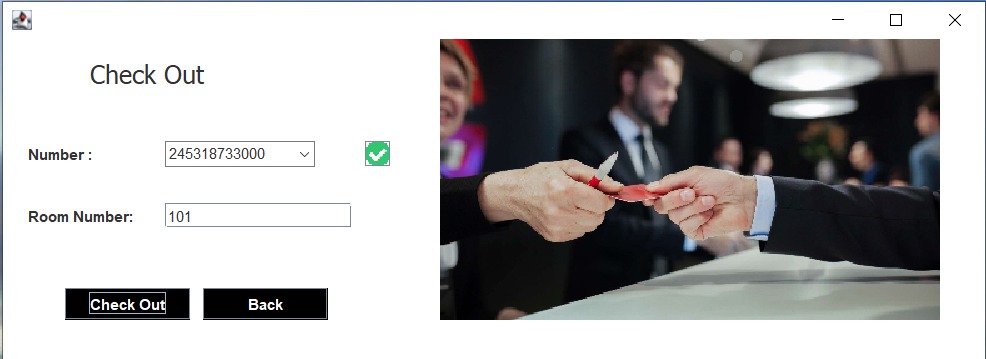


**6.2 Results**

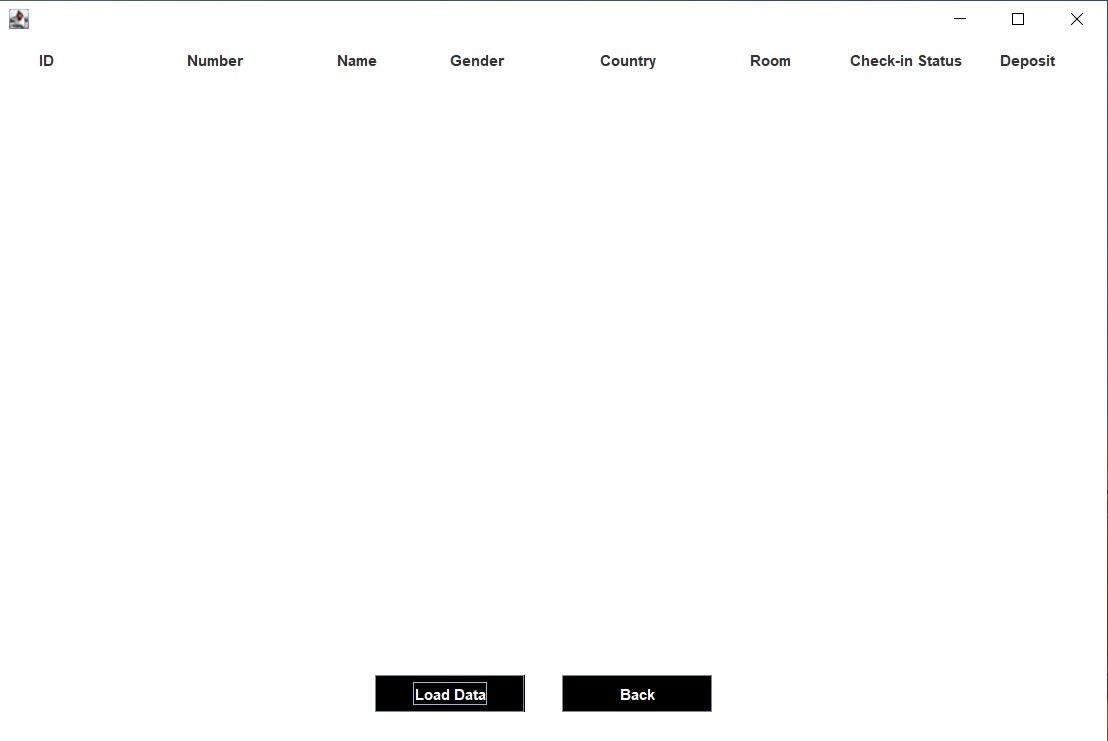
1. Customer Info in the Hotel



1. Customer Checking Out



1. Updated Customer Info



**LIST OF TABLES AND FIGURES**

**7.1 List of Tables**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S No** | **Fig No** | **Table Content** | **Page** |  |
|  |  |  | **no** |  |
|  |  |  |  |  |
| 1 | 3.1 | HARDWARE REQUIREMENTS FOR | 5 |  |
|  |  |  |  |
|  |  | PRESENT PROJECT |  |  |
|  |  |  |  |  |
| 2 | 3.1 | ​SOFTWARE REQUIREMENTS FOR | 5 |  |
|  |  |  |  |
|  |  | PRESENT PROJECT |  |  |
|  |  |  |  |  |

**7.2 List of Figures**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **Fig No** | **Name of the figure** | **Page** |
|  |  |  | **No** |
|  |  |  |  |
| **0** | **3.2** | **Flow Diagram** | 6 |
|  |  |  |  |
| **1** | **4.2** | **Class Diagram** | 9 |
|  |  |  |  |
| **2** | **4.3** | **Use Case Diagram** | 10 |
|  |  |  |  |
| **3** | **4.4** | **Activity Diagram** | 11 |
|  |  |  |  |
| **4** | **4.5** | **Sequence Diagram** | 12 |
|  |  |  |  |
| **5** | **4.6** | **ER Diagram** | 13 |
|  |  |  |  |
| **6** | **6.1a** | **Home page** | 33 |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **7** | **6.1b** | **Login page** | 33 |
|  |  |  |  |
| **8** | **6.1c** | **Dashboard Page** | 34 |
|  |  |  |  |
| **9** | **6.1d** | **Add Room** | 34 |
|  |  |  |  |
| **10** | **6.1e** | **Add Employee** | 35 |
|  |  |  |  |
| **11** | **6.1f** | **Add Driver** | 35 |
|  |  |  |  |
| **12** | **6.1g** | **Reception** | 36 |
|  |  |  |  |
| **13** | **6.1h** | **New Customer** | 36 |
|  |  |  |  |
| **14** | **6.1 i** | **Room** | 37 |
|  |  |  |  |
| **15** | **6.1 j** | **Employee Info** | 37 |
|  |  |  |  |
| **16** | **6.1 k** | **Check Out** | 38 |
|  |  |  |  |
| **17** | **6.1 l** | **Update Check Status** | 38 |
|  |  |  |  |
| **18** | **6.1 m** | **Update Room Status** | 38 |
|  |  |  |  |
| **19** | **6.1 n** | **Pick Up Service** | 39 |
|  |  |  |  |
| **20** | **6.1 o** | **Search Room** | 39 |
| **21** | **6.2** | **Results** | 40 |
|  |  |  |  |

**CONCLUSION AND FUTURE**

**ENHANCEMENTS**

**Conclusion:**

Hotel Management System in short, is a minimalistic desktop application which was developed in order to make hotel management easy in many folds. However some desirable mentions are:

* The first one is that customer, who should be treated as a king by businessmen, is handled in most of the aspects effectively from allotting a room to checking out of the room
* The second important to be discussed is that people working for hotel are also being managed by adding them into database
* Here, each department present in the hotel are being allocated with particular budget, are also stored in the database. Therefore, It helps hotel management officials to handle the key financial matters with matter of several clicks

**Future Enhancements:**

Though there are notable features present in the current version of Hotel Management

System, we can further **elaborate the powers of admin** by giving an option to “Remove

Employee”, “Remove Driver” etc. **“Food” feature** would also be beneficial for both the

customer and receptionist.

**REFERENCES**

* JAVA SWING API: <https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html>
* JAVA AWT: <https://www.javatpoint.com/java-awt>
* MySQL: <https://www.geeksforgeeks.org/mysql-common-mysql-queries/>
* VS Code: <https://code.visualstudio.com/docs/java/java-project/>