**Dr. AMBEDKAR INSTITUTE OF TECHNOLOGY**

Near Jana Bharathi Campus, Mallathahally, Bengaluru 560056



MINI PROJECT REPORT

ON

# “HOTEL MANAGEMENT SYSTEM”

BACHELOR OF ENGINEERING

IN

# COMPUTER SCIENCE AND ENGINEERING

SUBMITTED

BY

**Chethana S Murthy 1DA21CS180**

**Keerthi B S 1DA21CS182**

UNDER THE GUIDANCE OF

**Mrs.Veena Potdar Mrs.Megha L**

**Assistant.Prof, Assistant.Prof,**

**Dept. of CSE Dr.AIT Dept. of CSE Dr.AIT**

**Department of Computer Science & Engineering**

**2021-22**

# Dr. AMBEDKAR INSTITUTE OF TECHNOLOGY

Near Jana Bharathi Campus, Mallathahally, Bengaluru 560056



# CERTIFICATE

This is to certify that the project entitled “**HOTEL MANAGEMENT SYSTEM**” submitted in the partial fulfillment of the requirement of the 5th semester DBMS laboratory curriculum during the year 2023-24 is a result of bonafide work carried out by-

## Chethana S Murthy 1DA21CS180 Keerthi B S 1DA21CS182

Signature of the guide

**Mrs. Veena Potdar**  **Mrs. Megha L**

**Assistant. Prof., Assistant. Prof.,**

**Dept. of CSE Dr. AIT Dept. of CSE Dr. AIT**

1. Internal Examiner

1. External Examiner

### Dr. Siddaraju,

Head of Department

Department of CSE, Dr.AIT

**ACKNOWLEDGEMENT**

The sense of contention and elation that accompanies the success of this seminar and the report could be incomplete without mentioning the names of people who have helped me in accomplishing them, people whose constant guidance, support and encouragement resulted in the realization.

We consider ourselves privileged to express our gratitude and respect towards all those who guided us through the project, “**HOTEL MANAGEMENT SYSTEM**”.

We take this opportunity to thank **Dr. M Meenkashi, PRINCIPAL Dr. Ambedkar Institute of Technology, Bengaluru** for his support and encouragement.

We are grateful to **Dr. Siddaraju, Head of Department, CSE, Dr. Ambedkar Institute of Technology, Bengaluru** for providing encouragement and support.

We consider ourselves privileged to express our gratitude and respect towards our guide

**Mrs. Veena Potdar, Assistant Proffessor, Department of CSE, Dr. Ambedkar Institute of Technology** and **Mrs. Megha L, Assistant Proffessor, Department of CSE Dr. Ambedkar Institute of Technology** for constant guidance and support for the completion of the project.

Lastly, we thank all the members of the staff both teaching and non-teaching, friends and last but not the least our parents and family, for helping me directly or indirectly in the completion of the project.

**CHETHANA S MURTHY**

**KEERTHI B S**

# ABSTRACT

The paper developed an automated system that is used to manage car information and its administration. This was with a view to eliminate the problem of inappropriate data keeping, inaccurate reports, time wastage in storing, processing and reserving information encountered by the traditional car rental system in order to improve the overall efficiency of the organization. The design provides excellent car rental management service and improved information structure.

|  |  |  |
| --- | --- | --- |
|  | **CONTENTS** |  |
| **Chapter No.** | **Title** | **Page No.** |
| **Chapter 1** | **Introduction** | 1 |
| **Chapter 2** | **Literature Review** | 2 |
| 2.1 | Database Management System | 2 |
| 2.2 | Structured Query Language | 3 |
| 2.3 | MySQL | 3 |
| 2.3.1 | MySQL Workbench | 4 |
| 2.4 | Entity Relationship Diagram | 4 |
| 2.5 | Relational Schema | 6 |
| 2.6 | Normalization | 7 |
| 2.7 | Uses of DBMS | 8 |
| 2.8 | Application of Database | 8 |
| **Chapter 3** | **Requirement Specification** | 9 |
| 3.1 | Hardware Requirements | 9 |
| 3.2 | Software Requirements | 9 |
| **Chapter 4** | **Description** | 10 |
| 4.1 | E-R Diagram | 11 |
| 4.2 | Relational Schema | 12 |
| **Chapter 5** | **Coding** | 15 |
| 5.1 | Table Creation and Insertion | 15 |
| 5.2 | Queries | 20 |
| 5.3 | Front-end Code and Snapshots | 24 |
|  | **Conclusion** | 70 |
|  | **Bibliography** | 71 |

**CHAPTER 1**

## INTRODUCTION

A Hotel Management System is looking to develop a state-of-the-art hotel portfolio management system which is able to track their hotel room booking history and billing the details. This system facilitates the owner of the Hotel to retrieve, update and track and delete the bookings and hotel efficiently. At the same time, can utilize this system to monitor their financial management. Currently, different locations of the hotels have their own separated systems leading to lack of communication and inefficient data sharing. For example, the branch of hotel located in Mumbai uses simple Microsoft Excel to keep the record of their customers, details of room and date they checked in and out which is inconvenient to retrieve and update the hotel and customer information. In the hotel located at Dharwad, maintain a book-based ledger to keep a record of their customers and employee and the insurance for the same. The main branch of the company located in Bengaluru has to keep the customer and employee details of all their branch offices on their own computer system. While each statement serves a distinctive process, there is no coordination, assimilation and representation of data. The systems may have duplicate data which leads to waste of space. The different systems also may have different application programs which cause incompatible files.

Due to these disadvantages of the current system, a hotel management system is proposed. Hotel Management System is a database management system (DBMS) which is based on computer networks, using the advance database technology to construct, maintain and update various kinds of data in data base system. The DBMS can track and update all the information of the branches of hotel and customers during a particular time span. The major advantages of the DBMS are easy to retrieve and update information, efficient data sharing and communication and reliable backup and security .Information about customer is done by just writing the their name, customer id and room booked details. Whenever a new customer is added to the company its information is stored freshly. Bills are generated by recording the customer. Details of the customer are updated on a written sheet and at last, they all summed up. Room usage details are recorded on the document, which contains the customer information. It is destroyed after sometime time period to decrease the paper load in the office. The admin themselves have to track the customer details which is a tedious job.

**CHAPTER 2**

## LITERATURE REVIEW

**2.1 DATABASE MANAGEMENT SYSTEM**

Data can be defined as a representation of facts, concepts, or instructions in a formalized manner, which should be suitable for communication, interpretation, or processing by human or electronic machine.

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system Let us discuss few examples. An online telephone directory uses a database to store data of people, phone numbers, and other contact details. Your electricity service provider uses a database to manage billing, client-related issues, handle fault data, etc. Let us also consider Facebook. It needs to store, manipulate, and present data related to members, their friends, member activities, messages, advertisements, and a lot more. We can provide a countless number of examples for the usage of databases.

Database Management System (DBMS) is a software for storing and retrieving users' data while considering appropriate security measures. It consists of a group of programs which manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large systems, a DBMS helps users and other third-party software to store and retrieve data. DBMS allows users to create their own databases as per their requirement. The term “DBMS” includes the user of the database and other application programs. It provides an interface between the data and the software application. There are 4 major types of DBMS. Let’s look into them in detail.

* **HIERARCHICAL** -In a Hierarchical database, model data is organized in a tree-like structure. Data is Stored Hierarchically (top down or bottom up) format. Data is represented using a parent- child relationship. In Hierarchical DBMS parent may have many children, but children have only one parent.
* **NETWORK DBMS -**The network database model allows each child to have multiple parents. It helps you to address the need to model more complex relationships like as the orders/parts many- to-many relationship. In this model, entities are organized in a graph which can be accessed through several paths.
* **RELATIONAL MODEL**-Relational DBMS is the most widely used DBMS model because it is one of the easiest. This model is based on normalizing data in the rows and columns of the tables. Relational model stored in fixed structures and manipulated using SQL.
* **OBJECT-ORIENTED MODEL**-In Object-oriented Model data stored in the form of objects. The structure which is called classes which display data within it. It defines a database as a collection of objects, which stores both data members values and operations.

* 1. **SQL – STRUCTURED QUERY LANGUAGE**

SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS). It is particularly useful in handling structured data, i.e., data incorporating relations among entities and variables. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc. Although most database systems use SQL, most of them also have their own additional proprietary extensions that are usually only used on their system.

* 1. **MySQL**

MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open- source and free software under the GNU license. It is supported by **Oracle Company**.

MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database. It is commonly used in conjunction with [PHP s](https://www.javatpoint.com/php-tutorial)cripts for creating powerful and dynamic server-side or web-based enterprise applications. It is developed, marketed, and supported by **MySQL AB, a Swedish company**, and written in C programming language and [C++ programming language*.*](https://www.javatpoint.com/cpp-tutorial) Many small and big companies use MySQL. MySQL supports many Operating Systems like [Windows,](https://www.javatpoint.com/windows) [Linux, M](https://www.javatpoint.com/linux-tutorial)acOS, etc. with C, C++,and [Java languages.](https://www.javatpoint.com/java-tutorial)

MySQL is [a Relational Database Management System (](https://www.javatpoint.com/what-is-rdbms)RDBMS) software that provides many things, which are as follows:

* It allows us to implement database operations on tables, rows, columns, and indexes.
* It defines the database relationship in the form of tables (collection of rows and columns), also known as relations.
* It provides the Referential Integrity between rows or columns of various tables.
* It allows us to updates the table indexes automatically.
* It uses many SQL queries and combines useful information from multiple tables for the end-users.

**2.3 MySQL Workbench**

MySQL Workbench is a unified visual database designing or graphical user interface tool used for working with database architects, developers, and Database Administrators. It is developed and maintained by Oracle. It provides SQL development, data modeling, data migration, and comprehensive administration tools for server configuration, user administration, backup, and many more. We can use this Server Administration for creating new physical data models, E-R diagrams, and for SQL development (run queries, etc.). It is available for all major operating systems like Mac OS, Windows, and Linux.

With MySQL Workbench, you use an intuitive, browser-based interface, to:

* Administer the database
* Create tables, views, and other database objects
* Import, export, and view table data• Run queries and SQL scripts
* Generate reports.

**2.4 ENTITY RELATIONSHIP DIAGRAM**

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

* The properties that are used to describe an entity are known as Attributes; for example, an Employee entity may have a Name, Gender, Date of Birth of his/her attributes.
* If book is regarded as an entity then Author’s name, Price, published by, etc. are its various attributes.

A specific entity will have a value for each of its attributes. Thus, an entity has a value for each attribute. A diagram representing entities and relationships among them is known as entity relationship diagram. The major elements used in ER diagram are entities, attributes, identifiers and relationships that express a reality for which database is designed.

|  |  |
| --- | --- |
|  |  |
|  |  |

**Fig. 2.4.1 - ER Diagram**

**ENTITY TYPE:**

It symbolizes anything in the real world that has multiple existence.

* **WEAK ENTITY TYPE:** The weak entity in DBMS do not have a primary key and are dependent on the parent entity. It mainly depends on other entities.
* **RELATIONSHIP TYPE:** A diamond box is used to represent the relationship between two entities. Relationships can be one-to-one, one-to-many or many-to-many.
* **IDENTIFYING RELATIONSHIP TYPE:** The relationship type that is used to relate a weak entity type to its owner is shown by double lined diamond shaped box.

**ATTRIBUTE:**

Entities are represented by means of their properties, called **attributes**. All attributes have values. For example, a student entity may have name, class, and age as attributes. There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

* **KEY ATTRIBUTE**: A key attribute is one for which each entity has a unique value. It is represented by an oval shape with the attribute name underlined.
* **MULTI VALUED ATTRIBUTE**: An entity that has multiple values for that attribute is called multivalued attribute.
* **DERIEVED ATTRIBUTE:** As discussed earlier, an attribute whose value depends upon the value of the stored attribute. It is represented using a dashed oval shape.

In a database system, we deal with various types of keys as follows:

* **CANDIDATE KEY:** Minimal set of attributes that uniquely identifies each occurrence of an entity type.
* **PRIMARY KEY:** Candidate key selected to uniquely identify each occurrence of an entity type.
* **UNIQUE KEY:** Can accept unique of null values.
* **COMPOSITE KEY:** A key that consists of two or more attributes and removal of even one of them would result in loss of intended information.

* 1. **RELATIONAL SCHEMA**

The relational schema is the primary element of the relational database. These databases are managed using language and structure that is consistent with first-order logic. This allows for database management based on entity relationships, making them easy to organize according to volume. Relational schema refers to the meta-data that describes the structure of data within a certain domain. It is the blueprint of a database that outlines the way its structure organizes data into tables. There are two steps to creating a relational database schema: creating the logical schema and creating the physical schema. The logical schema depicts the structure of the database, showing the tables, columns and relationships with other tables in the database and can be created with modeling tools or spreadsheet and drawing software. The physical schema is created by actually generating the tables, columns and relationships in the relational database management software (RDBMS).

* 1. **NORMALIZATION**

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables. Here are the most commonly used normal forms.

* First Normal Form
* Second Normal Form
* Third Normal Form
* Boyce & Codd Normal Form.

**First Normal Form (1NF)**

For a table to be in the First Normal Form, it should follow the following 4 rules:

* It should only have single(atomic) valued attributes/columns.
* Values stored in a column should be of the same domain
* All the columns in a table should have unique names.
* The order in which data is stored, does not matter.

**Second Normal Form (2NF)**

For a table to be in the Second Normal Form, it should follow the following 4 rules:

* It should be in the First Normal form.
* It should not have Partial Dependency.

**Third Normal Form (3NF)**

A table is said to be in the Third Normal Form when,

* It is in the Second Normal form.
* It doesn't have Transitive Dependency.

**Boyce and Codd Normal Form (BCNF)**

**Boyce and Codd Normal Form** is a higher version of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF. For a table to be in BCNF, following conditions must be satisfied:

* R must be in 3rd Normal Form
* For each functional dependency (X → Y), X should be a super key.

**2.7 USES OF DBMS**

Data that is well organized and integrated is very useful in decision making. We can infer some of the following uses of DBMS.

* Effective and efficient management of data.
* Query processing and management.
* Easy to understand and user friendly.
* Security and integrity of data.
* Better decision making.
* Data sharing and storage.
* Better access to accurate data.
* Ensures error free information.

**2.8 APPLICATIONS OF DATABASE**

Databases are used to support internal operations of organizations and to underpin online interactions with customers and suppliers’ databases are used to hold administrative information and more specialized data, such as engineering data or economic models.

Databases touch all aspects of our lives. Some of the major areas of application are as follows:

* Banking
* Airlines
* Universities
* E- Commerce
* Human Resources

**CHAPTER 3**

## REQUIREMENT SPECIFICATION

The hardware and software components of a computer system that are required to install and use software efficiently are specified in the SRS. The minimum system requirements need to be met for the programs to run at all times on the system.

**3.1 HARDWARE REQUIREMENTS**

The hardware requirements specify the necessary hardware which provides us the platform to implement our programs.

* 2.2 GHz processor (Pentium).
* GB RAM (System Memory).
* 20 GB of hard-drive space.
* VGA capable of 1024 x 768 screen resolution. • Necessary computer peripherals such as keyboard etc.

**3.2 SOFTWARE REQUIREMENTS**

The software requirement specifies the pre-installed software needed to run the code being implemented in this project.

* Windows Operating System
* MySQL Workbench
* MySQL Shell
* MySQL Server
* Connector/P
* Python

**CHAPTER 4**

## DESCRIPTION

A Hotel System is looking to develop a state-of-the-art hotel portfolio management system which is able to track their hotel room booking history and billing details. This system facilitates the owner of the Hotel to retrieve, update and track and delete the bookings and customer efficiently. At the same time, room utilize this system to monitor their financial management.

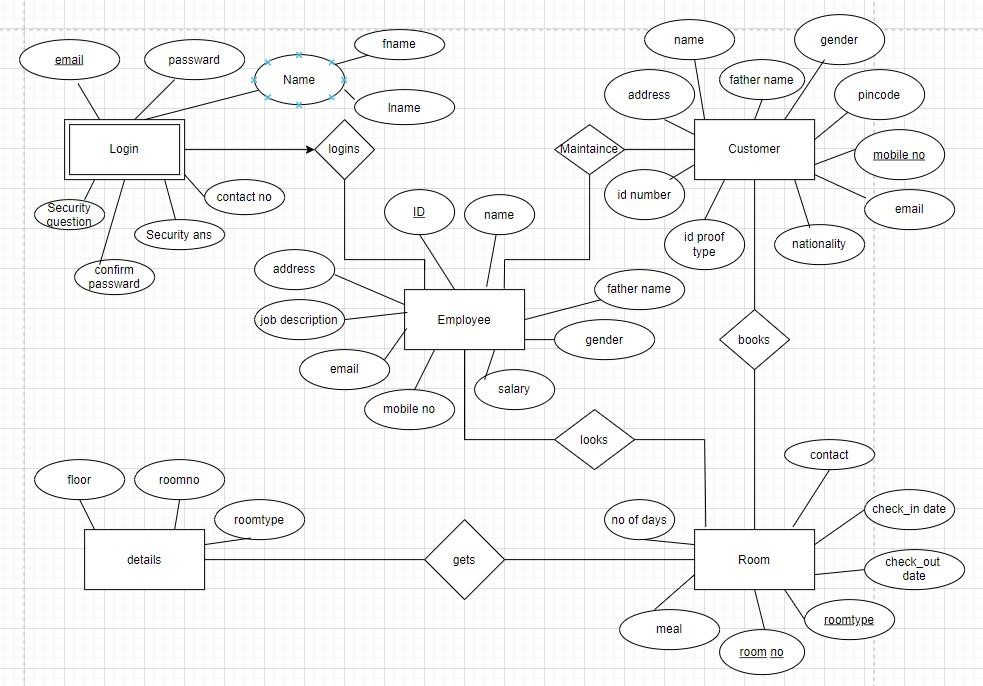
Currently, different locations of the car rental company have their own separated systems leading to lack of communication and inefficient data sharing. For example, the branch of hotel located in Basveshwarnagar uses simple Microsoft Excel to keep the record of their customers, room and employee details which is inconvenient to retrieve and update the room and customer's information. In hotel located at Whitefield, maintain a book-based ledger to keep a re cord of their customers and room,employee and the insurance for the same. The main branch of the company located in Rajajinagar has to keep the customer and room details of all their branch offices on their own computer system. While each statement serves a distinctive process, there is no coordination, assimilation and representation of data. The systems may have duplicate data which leads to waste of space. The different systems also may have different application programs which cause incompatible files.

Due to these disadvantages of the current system, hotel management system is proposed. Hotel Management System is a database management system (DBMS) which is based on computer networks, using the advance database technology to construct, maintain and update various kinds of data in data base system. The DBMS can track and update all the information of the cars available and customers during a particular time span. The major advantages of the DBMS are easy to retrieve and update information, efficient data sharing and communication and reliable backup and security.

**4.1 E-R DIAGRAM**

|  |  |
| --- | --- |
| ***Entity*** | ***Attributes*** |
| Registration | Name, email, contact no, security key, security answer, password Confirm password. |
| Customer | Id, name, father name, gender, pincode, Mobile no, Email, Nationality, Id proof type, Id number, Address. |
| Room Booking | contact, check\_in date, check\_out date, room type, room number, meal. |
| Room Details | room type, room number, floor |
| Employee | Id, name, father name, gender, salary, Mobile no, Email,  Job description, Id proof type, Id number, Address |

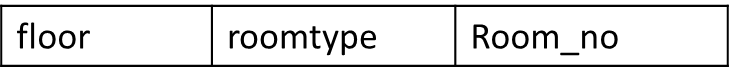
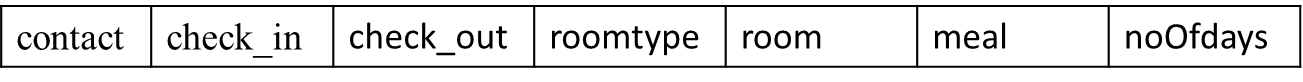
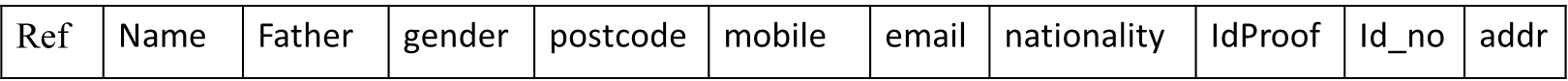
**Fig. 4.1.1 - E-R Diagram**



**Fig. 4.1.1 - E-R diagram for “HOTEL MANAGEMENT SYSTEM”**

### 4.2 RELATIONAL SCHEMA

**S em m**



onfirm password

Password

contact

Security

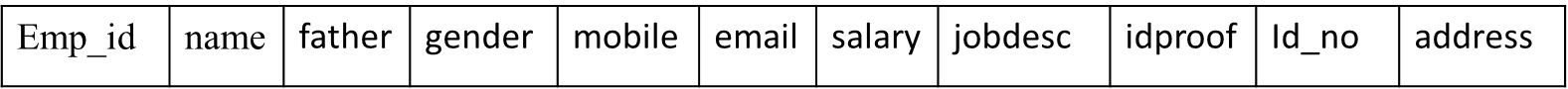
ans

Security qn

email

lname

fname



**Fig. 4.2.1 - Relation Schema for “Hotel Management System”**

**LOGIN TABLE:**

**Registration**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Email | Contact no | Security key | Security answer | Passward | Confirm passward |

**Attributes:**

**o**

* **Name:** Entering the username. It is of the type varchar.
* **Email:** Email address of the username is of type varchar.
* **Contact:** Contact number of the user is of type varchar.
* **Security key:** Security key containing questions. It is of the type varchar.
* **Security answer:** Security answer is present. It is of the type varchar. • **Password:**  Password should be set for security purpose. It is of the type varchar.

**Customer registration table:**

**CUSTOMER**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Id | Name | father name | gender | pincode | Mobile no | Email | Nationality | Id proof type | Id  number | Addresss |

**Attributes:**

* **ID:** This is used to uniquely identify a row in the table. It is of the type int.
* **Name:** Every customer has name to be identified differently. It is of the type varchar.
* **Father name:** Customer father name should be added. It is of the type varchar.
* **Gender :** To indicate the gender of the customer. It is of the type varchar.
* **Pincode:** Pincode of the residential address. It is of the type int.
* **Mobile:** Mobile no will be unique for each customer. It is of type int.
* **Email:** Customer email is required. It is of type varchar.
* **Nationality:** To identify from which nation the customer has came. It is of type varchar.
* **Idproof Number:** This is used to uniquely identify a row in the table. It is of the type int.
* **Address:** Customer address is required. It is of type varchar.

**Room booking:**

**Room booking**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Contact | Check\_in | Check\_out | Roomtype | Meal |

**Attributes:**

* **Contact:** it is used fetch the data of the customer. It is type of number.
* **Check\_in**: it is a type of date.
* **Check\_out**: it is a type of date.
* **Roomtype**: it is a checkbox used know the type of room.
* **Meal**: it is a type of varchar

**Details:**

**Details of the room:**

|  |  |  |
| --- | --- | --- |
| **+**Room no | Room type | Floor |

**Attributes:**

* **Floor:** it contains floor number of room. It is a type of number**.**
* **Roomtype:** it is type of varchar.
* **Room\_no**: it is used to know the room no.

**Employee:**

**Employee details**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Id** | Name | Father name | Gender | Salary | Mobile | Email | Job  Description | Id proof number | Address |

**Attributes:**

* **ID:** This is used to uniquely identify a row in the table. It is of the type int.
* **Name:** Every customer has name to be identified differently. It is of the type varchar.
* **Father name:** Customer father name should be added. It is of the type varchar.
* **Gender:** To indicate the gender of the customer. It is of the type varchar.
* **Salary:** Salary of Employee. It is of the type int.
* **Mobile:** Mobile no will be unique for each customer. It is of type int.
* **Email:** Customer email is required. It is of type varchar.
* **Job Description:** To identify the job of the employee. It is of type varchar.
* **Idproof Number:** This is used to uniquely identify a row in the table. It is of the type int.
* **Address:** Customer address is required. It is of type varchar.

.

**CHAPTER 5**

# CODING

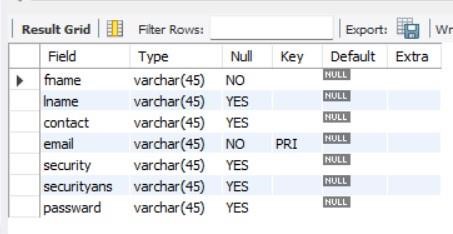
## 5.1 Table Creation and Insertion

**Registration Table:**

Create Table register(

Name varchar(45), email varchar(45),contact no varchar(45),

Securitykey varchar(45),securityanswer varchar(45),password varchar(45), Confirm password(45),primary key(email));



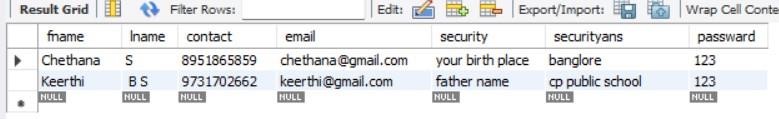
**Inserting:**

Insert into register values (‘chethana’,’s’,’8951865859’,’chethana@gmail.com’,

’enter birth place’,’banglore’,123,123)

Insert into register values (‘Keerthi’,’B S’,’9731702662’,’keerthi@gmail.com’,

’enter primary school,’cp public school’,123,123)



**Customer Table:**

Create Table Customer(

Name varchar(10),Fath\_Nam varchar(10),

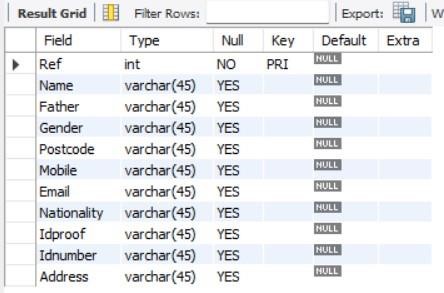
Gender Varchar(2),Pincode varchar(10),

Mobile\_no varchar(10),Email varchar(20),

Nationality varchar(10),Id proof Type varchar(10),

Id number varchar(10),Addresss(10),Cust\_Id number(10),

Primary key(Cust\_id,Mobile\_no));



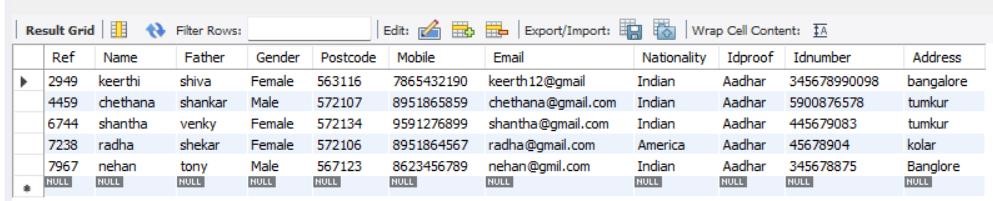
**Insertion:**

Insert into customer values(`keeerthi’,`shiva`,`female`,7865432190`,keerthi@gamail.com `Indian`,`aadhar`,`345678990098`, **`**banglore`)

Insert into customer values(`chethana’,`shankar`,`female`,8951865859`, chethana@gamail.com,``Indian`,`aadhar`,`5900876543`, **`**tumkur`)

Insert into customer values(`shantha’,`venky`,`female`,9591276899`, shantha@gamail.com,``Indian`,`aadhar`,`445678083`, **`**tumkur`)

Insert into customer values(`radha’,`shekar,`male`,8951864567`, radha@gamail.com,`American`,`aadhar`,`45678904`, **`**kolar`)



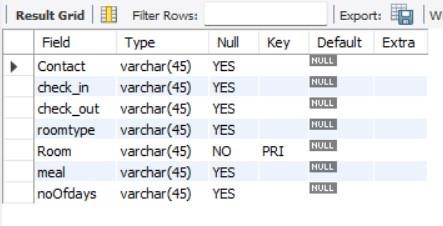
**Room Booking Table:**

Create Table Room(

Contact varchar(10),check\_in varchar(45),

Check\_out varchar(45),roomtype varchar(45), Roomnumber varchar(10),meal varchar(10),

noOfdays varchar(10), Primary key(Roomnumber));



**Insertion:**

Insert into roombooking table values('9591276899', '12/01/2023', '18/01/2023', 'Laxary',

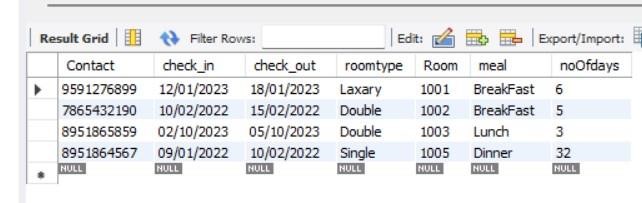
'1001', 'BreakFast', '6')

Insert into roombooking table values('7865432190', '10/02/2022', '15/02/2022', 'Double', `1002', 'BreakFast', '5')

Insert into roombooking table values('8951865859', '02/10/2023', '05/10/2023', 'Double',

'1003', 'Lunch', '3')

Insert into roombooking table values('8951864567', '09/01/2022', '10/02/2022', 'Single', '1005', 'Dinner', '32')



**Inserting:**

**Details Table:**

Create Table details(

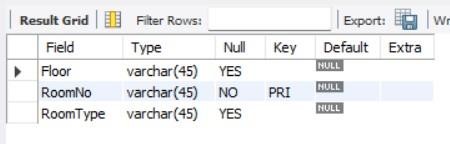
Roomno varchar(45),

Room type varchar(45),

Floor varchar(45),

Primary key(Roomno),

Foreign key(Roomno)References Room(Roomnumber));



**Insertion:**

Insert into details values('2', '1001', 'single')

Insert into details values('1', '1002', 'double')

Insert into details values('2', '1003', 'single')

Insert into details values('3', '2004', 'double')



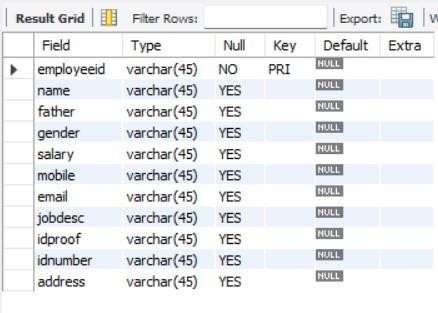
**Employee Details :**

Create Table Employee(

Name varchar(10),father varchar(45), gender Varchar(45),Salary varchar(45), mobile\_no varchar(45),email varchar(45), jobdesc varchar(45),idproof varchar(45),

idnumber varchar(45),addresss(405),employeeid varchar(45), Primary key(employeeid));

**Inserting:**



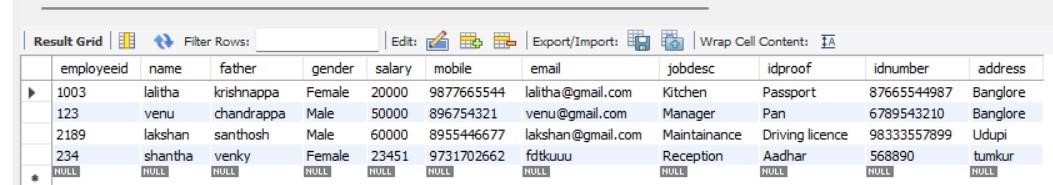
**Insertion:**

Insert into values Employee('1003', 'lalitha', 'krishnappa', 'Female', '20000', '9877665544', 'lalitha@gmail.com', 'Kitchen', 'Passport', '87665544987', 'Banglore')

Insert into values Employee('123', 'venu', 'chandrappa', 'Male', '50000', '896754321', 'venu@gmail.com', 'Manager', 'Pan', '6789543210', 'Banglore')

Insert into values Employee('2189', 'lakshan', 'santhosh', 'Male', '60000', '8955446677', 'lakshan@gmail.com', 'Maintainance', 'Driving licence', '98333557899', 'Udupi')

Insert into values Employee '234', 'shantha', 'venky', 'Female', '23451', '9731702662', 'fdtkuuu', 'Reception', 'Aadhar', '568890', 'tumkur')



**5.2 QUERIES**

The most common operation in SQL, the query, makes use of the declarative. SELECT statement. SELECT retrieves data from one or more tables, or expressions. Standard SELECT statements have no persistent effects on the database. Some non-standard implementations of SELECT can have persistent effects, such as the SELECT INTO Syntax provided in some databases.

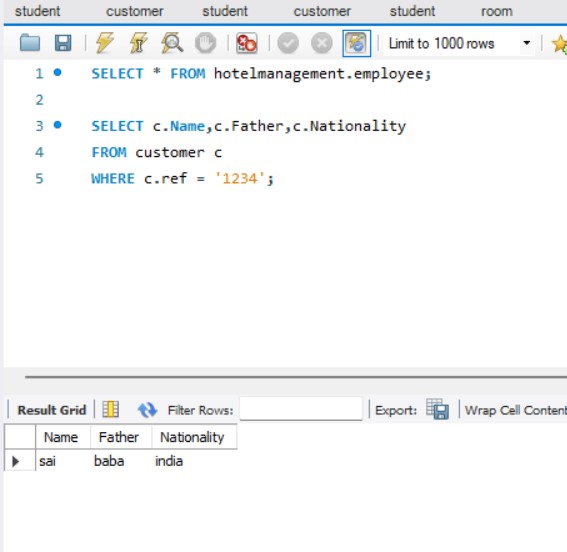
Queries allow the user to describe desired data, leaving the database management system

(DBMS) to carry out planning, optimizing, and performing the physical operations necessary to produce that result, normally immediately following the SELE T keyword. An asterisk (“**\***”) can be used to specify that the query should return all columns of the queried tables. Select is the most complex statement in SQL, with optional keywords and clauses that include:

* The **FROM** clause, which indicates the table(s) to retrieve data from. The **FROM** clause can include optional JOIN subclauses to specify the rules for joining tables.
* The **WHERE** clause includes a comparison predicate, which restricts the rows returned by the query. The **WHERE** clause eliminates all rows from the result set where the comparison predicate does not evaluate to True.
* The **GROUP BY** clause projects rows having common values into a smaller set of rows. **GROUP BY** is often used in conjunction with SQL aggregation functions or to eliminate duplicate rows from a result set. The **WHERE** clause is applied before the GROUP BY clause.
* The **HAVING** clause includes a predicate used to filter rows resulting from the **GROUP BY** clause. Because it acts on the results of the **GROUP BY** clause, aggregation functions can be used in the **HAVING** clause predicate.
* The **ORDER BY** clause identifies which column[s] to use to sort the resulting data, and in which direction to sort them (ascending or descending). Without an **ORDER BY** clause, the order of rows returned by an SQL query is undefined.
* The **DISTINCT** keyword eliminates duplicate data.

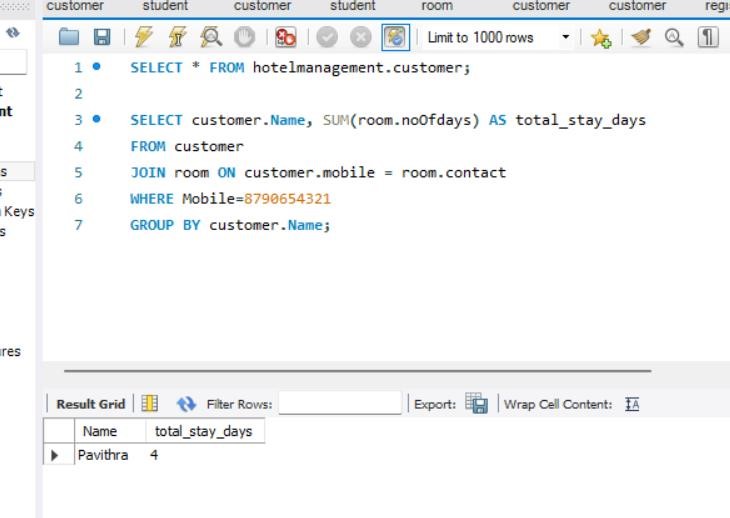
**QUERY 1**

* 1. **Retrieve customer name, father and nationality of ref=1234**



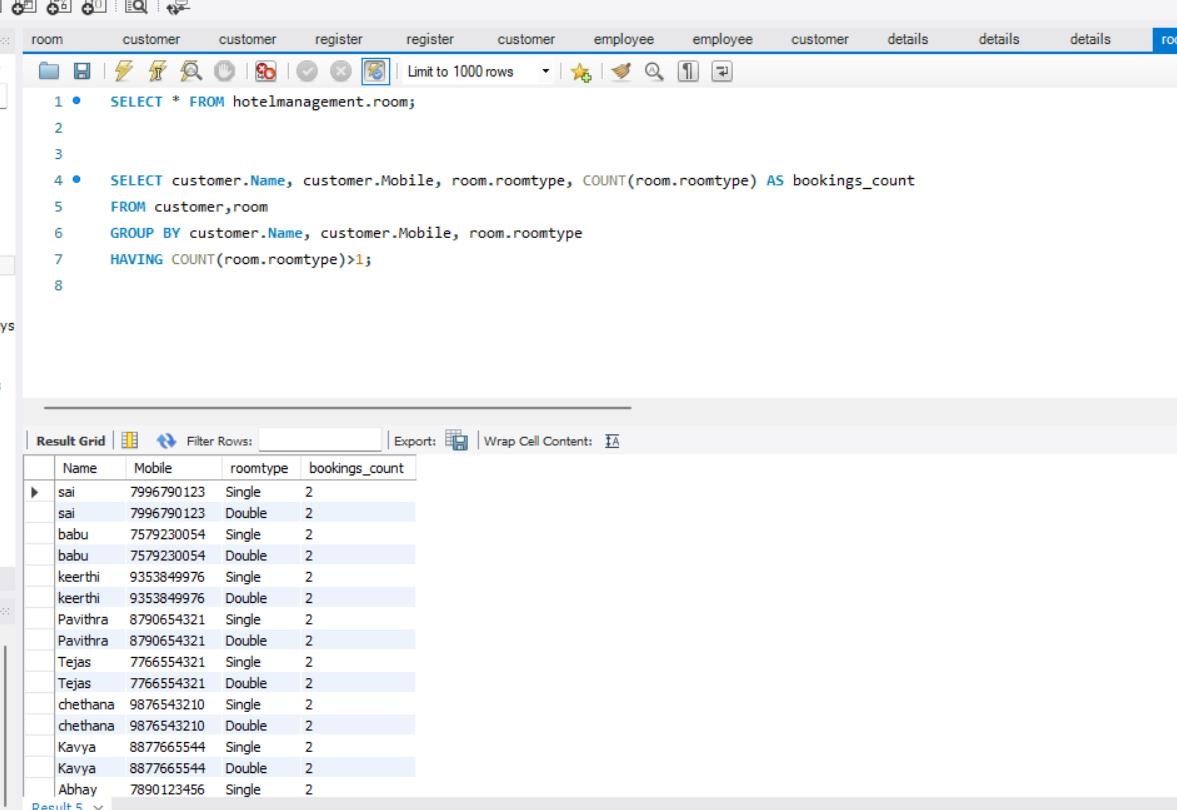
**QUERY 2**

* 1. **Find the total number of days each customer has stayed in the hotel**.



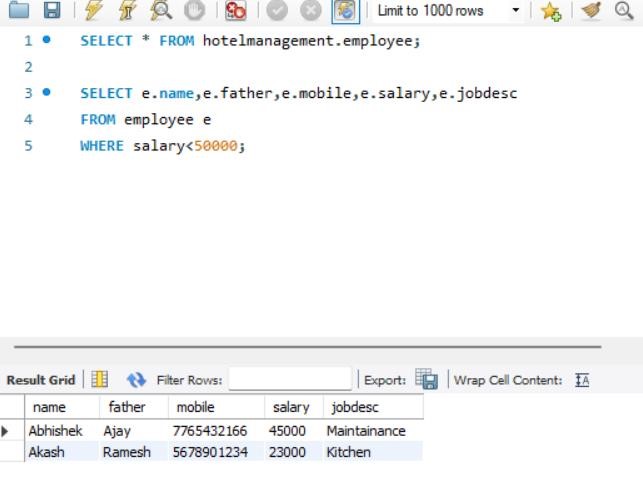
**QUERY 3**

**3.List the customers and their contact details who have booked the same room type more than once.**



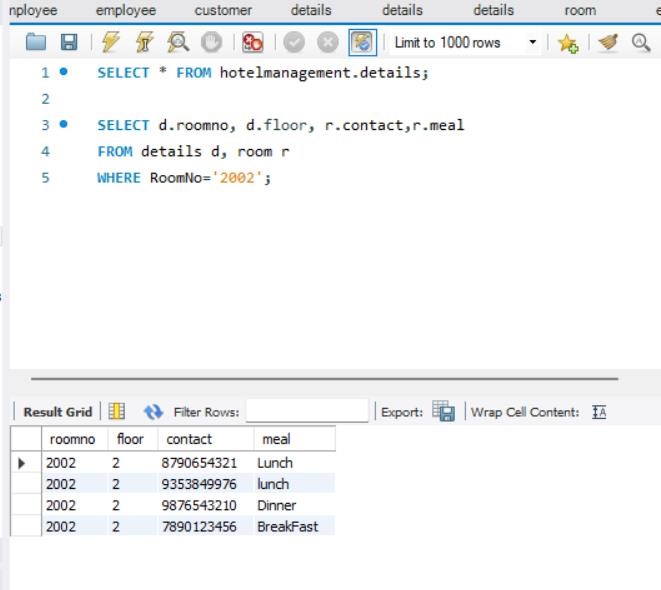
**QUERY 4**

1. **Retrieve the employee name, father, jobdesc, salary whose salary is less than 50000.**



**QUERY 5**

1. **Retrieve the details of roomno, floor, where roomtype=double.**



## 5.3 Front end Code and Snapshots

We have 5 python classes with their source code and Design that are under the package hotel that is being imported in starting of every code.

Login.py(Which uses user name and password to login into the hotel main page source and design file)

Hotel.py (Which has links to other four python classes) customer.py (Where we can add customer to be registered) roombooking.java(Where we have info about the roombooking type) Details.py (Where we have info about the room) employee.py (Where we have info about the employee)

So that we can access every code from anywhere. In Hotel.py we have created a function which on clicking opens the individual design and runs the code of that particular python class.

### Register.py

from tkinter import\* from tkinter import ttk

from PIL import Image,ImageTk #pip install poillow from tkinter import messagebox import mysql.connector from customer import Cust\_Win from room import Roombooking from details import DetailsRoom from employee import Employee

#from hotel import HotelManagementSystem

def main(): win=Tk()

app=login\_window(win)

win.mainloop()

class login\_window:

def \_\_init\_\_(self,root): self.root=root self.root.title("Login")

self.root.geometry("1550x800+0+0")

self.bg=ImageTk.PhotoImage(file=r"D:\mini project\hotel images/SDT\_Zoom-

Backgrounds\_April-8\_Windansea-1-logo-1.jpg") lbl\_bg=Label(self.root,image=self.bg) lbl\_bg.place(x=0,y=0,relwidth=1,relheight=1)

frame=Frame(self.root,bg="black")

frame.place(x=610,y=170,width=340,height=450)

img1=Image.open(r"D:\mini project\hotel images/LoginIconAppl.png") img1=img1.resize((100,100),Image.LANCZOS) self.photoimage1=ImageTk.PhotoImage(img1)

lbling1=Label(image=self.photoimage1,bg="black",borderwidth=0) lbling1.place(x=730,y=175,width=100,height=100)

get\_str=Label(frame,text="Get Started",font=("times new roman",20,"bold"),fg="white",bg="black")

get\_str.place(x=95,y=100)

#label

username=lbl=Label(frame,text="Username",font=("times new roman",15,"bold"),fg="white",bg="black")

username.place(x=70,y=155)

self.txtuser=ttk.Entry(frame,font=("times new roman",15,"bold")) self.txtuser.place(x=40,y=180,width=270)

password=lbl=Label(frame,text="Password",font=("times new

roman",15,"bold"),fg="white",bg="black")

password.place(x=70,y=225)

self.txtpass=ttk.Entry(frame,font=("times new roman",15,"bold")) self.txtpass.place(x=40,y=250,width=270)

#----------------icon img--------------

img2=Image.open(r"D:\mini project\hotel images/LoginIconAppl.png") img2=img2.resize((25,25),Image.LANCZOS) self.photoimage2=ImageTk.PhotoImage(img2)

lbling1=Label(image=self.photoimage2,bg="black",borderwidth=0) lbling1.place(x=650,y=323,width=25,height=25)

img3=Image.open(r"D:\mini project\hotel images/lock-512.png") img3=img3.resize((25,25),Image.LANCZOS) self.photoimage3=ImageTk.PhotoImage(img3)

lbling1=Label(image=self.photoimage3,bg="black",borderwidth=0) lbling1.place(x=650,y=395,width=25,height=25)

#loginbtn

loginbtn=Button(frame,text="Login",command=self.login,font=("timesnew roman",15,"bold"),bd=3,relief=RIDGE,fg="white",bg="red",activeforeground="white", activebackground="red")

loginbtn.place(x=110,y=300,width=120,height=35)

#registerbtn

registerbtn=Button(frame,text="New User Register",command=self.register\_window,font=( "timesneroman",10,"bold"),border=0,fg="white",bg="black",activeforeground="white",activeback ground="black")

registerbtn.place(x=15,y=350,width=160)

#forgotbtn

forgotbtn=Button(frame,text="Forgot

Password",command=self.forget\_passward\_window,font=("times new

roman",10,"bold"),border=0,fg="white",bg="black",activeforeground="white",activebackground=" black")

forgotbtn.place(x=10,y=370,width=160)

def register\_window(self): self.new\_window=Toplevel(self.root) self.app=Register(self.new\_window)

def login(self): if self.txtuser.get()=="" or self.txtpass.get()=="" :

messagebox.showerror("Error","all field required") elif self.txtuser.get()=="keerthi" or self.txtpass.get()=="keer": messagebox.showinfo("success","welcome to hotel") else:

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database=

"hotelmanegement") my\_cursor=conn.cursor()

my\_cursor.execute("select \* from register where email=%s and passward=%s",( self.txtuser.get(), self.txtpass.get()

))

row=my\_cursor.fetchone() if row==None: messagebox.showerror("error","invalid username and passward") else:

open\_main=messagebox.askyesno("YesNo","Access only admin") if open\_main>0: self.new\_window=Toplevel(self.root) self.app=HotelManagementSystem(self.new\_window) else: if not open\_main:

return

conn.commit() conn.close()

#=============reset passward============ def reset\_passward(self): if self.combo\_security.get()=="Select": messagebox.showerror("Error","select the security question") elif self.txt\_securityans.get=="":

messagebox.showerror("error","please enter the answer") elif self.txt\_new\_passward.get=="": messagebox.showerror("error","please enter the new passward") else:

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",databas e="hotelmanegement")

my\_cursor=conn.cursor()

query=("select \* from register where email=%s and security=%s and securityans=%s") value=(self.txtuser.get(),self.combo\_security.get(),self.txt\_securityans) my\_cursor.execute(query,value) row=my\_cursor.fetchone() if row==None: messagebox.showerror("error","please enter the correct answer") else:

query=("update register set passward=%s where email=%s") value=(self.txt\_new\_passward.get(),self.txtuser.get()) my\_cursor.execute(query,value)

conn.commit() conn.close()

messagebox.showerror("error","new passward has set successfully")

#=============forgot passward=============

def forget\_passward\_window(self): if self.txtuser.get()=="": messagebox.showerror("Error","please enter the email address to reset passward") else:

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database=

"hotelmanegement") my\_cursor=conn.cursor()

query=("select \* from register where email=%s") value=(self.txtuser.get(),) m y\_cursor.execute(query,value) row=my\_cursor.fetchone()

# print(row)

if row==None: messagebox.showerror=("error","please enter the valid user name") else:

conn.close() self.root2=Toplevel()

self.root2.title("forgot passward")

self.root2.geometry("340x450+610+170")

l=Label(self.root2,text="Forgot passward",font=("times new roman",20,"bold"),fg="red",bg="white")

l.place(x=0,y=10,relwidth=1) security=Label(self.root2,text="Security Question",font=("times new roman",15,"bold"),fg="black",bg="white")

security.place(x=50,y=80)

self.combo\_security=ttk.Combobox(self.root2,textvariable=self.combo\_security,font=("tim es new roman",15,"bold"),state="readonly") self.combo\_security["values"]=("Select","your birth place","father name",

"primary school name")

self.combo\_security.place(x=50,y=110,width=250) self.combo\_security.current(0)

securityans=Label(self.root2,text="Security Question Answer",font=("times new roman",15,"bold"),fg="black",bg="white")

securityans.place(x=50,y=150) self.txt\_securityans=ttk.Entry(self.root2,textvariable= self.var\_securityans,font=("times new roman",15))

self.txt\_securityans.place(x=50,y=180,width=250)

new\_passward=Label(self.root2,text="New passward",font=("times new roman",15,"bold"),fg="black",bg="white")

new\_passward.place(x=50,y=220)

self.txt\_new\_passward=ttk.Entry(self.root2,font=("times new roman",15)) self.txt\_new\_passward.place(x=50,y=250,width=250)

btn=Button(self.root2,text="Reset",command=self.reset\_passward,font=("times new roman",15,"bold"),fg="black",bg="white")

btn.place(x=100,y=300)

class Register: def \_\_init\_\_(self,root): self.root=root

self.root.title("Register")

self.root.geometry("1600x900+0+0")

#==============variables=========

self.var\_fname=StringVar() self.var\_lname=StringVar() self.var\_contact=StringVar() self.var\_email=StringVar() self.var\_passward=StringVar() self.var\_cpassward=StringVar() self.var\_security=StringVar() self.var\_securityans=StringVar()

#====back ground image============ self.bg=ImageTk.PhotoImage(file=r"D:\mini project\hotel images\0-3450\_3d-nature-wallpaper- hd-1080p-free-download-new.jpg")

bg\_lbl=Label(self.root,image=self.bg) bg\_lbl.place(x=0,y=0,relwidth=1,relheight=1)

#=============left image======= self.bg1=ImageTk.PhotoImage(file=r"D:\mini project\hotel images\thought-good-morningmessages-LoveSove.jpg")

left\_lbl=Label(self.root,image=self.bg1)

left\_lbl.place(x=50,y=100,width=470,height=550)

frame=Frame(self.root,bg="white")

frame.place(x=520,y=100,width=800,height=550)

register\_lbl=Label(frame,text="Register Here",font=("times new roman",20,"bold"),fg="dark

green",bg="white")

register\_lbl.place(x=20,y=20)

#=============label and entry========== fname=Label(frame,text="First Name",font=("times new roman",15,"bold"),fg="black",bg="white")

fname.place(x=50,y=100)

fname\_entry=ttk.Entry(frame,textvariable=self.var\_fname,font=("times new roman",15,"bold")) fname\_entry.place(x=50,y=130,width=250)

lname=Label(frame,text="Last Name",font=("times new

roman",15,"bold"),fg="black",bg="white")

lname.place(x=370,y=100)

self.txt\_lname=ttk.Entry(frame,textvariable=self.var\_lname,font=("times new roman",15)) self.txt\_lname.place(x=370,y=130,width=250)

contact=Label(frame,text="Contact no",font=("times new

roman",15,"bold"),fg="black",bg="white")

contact.place(x=50,y=170)

self.txt\_contact=ttk.Entry(frame,textvariable=self.var\_contact,font=("times new roman",15)) self.txt\_contact.place(x=50,y=200,width=250)

email=Label(frame,text="Email",font=("times new roman",15,"bold"),fg="black",bg="white") email.place(x=370,y=170)

self.txt\_email=ttk.Entry(frame,textvariable=self.var\_email,font=("times new roman",15)) self.txt\_email.place(x=370,y=200,width=250)

security=Label(frame,text="Security Question",font=("times new roman",15,"bold"),fg="black",bg="white")

security.place(x=50,y=240) self.combo\_security=ttk.Combobox(frame,textvariable=self.var\_security,font=("times new roman",15,"bold"),state="readonly")

self.combo\_security["values"]=("Select","your birth place","father name","primary school name") self.combo\_security.place(x=50,y=270,width=250) self.combo\_security.current(0)

securityans=Label(frame,text="Security Question Answer",font=("times new

roman",15,"bold"),fg="black",bg="white")

securityans.place(x=370,y=240) self.txt\_securityans=ttk.Entry(frame,textvariable=self.var\_securityans,font=("times new roman",15))

self.txt\_securityans.place(x=370,y=270,width=250)

passward=Label(frame,text="Passward",font=("times new

roman",15,"bold"),fg="black",bg="white")

passward.place(x=50,y=310)

self.txt\_passward=ttk.Entry(frame,textvariable=self.var\_passward,font=("times new roman",15)) self.txt\_passward.place(x=50,y=340,width=250)

cpassward=Label(frame,text="Conform Passward",font=("times new

roman",15,"bold"),fg="black",bg="white")

cpassward.place(x=370,y=310) self.txt\_cpassward=ttk.Entry(frame,textvariable=self.var\_cpassward,font=("times new roman",15))

self.txt\_cpassward.place(x=370,y=340,width=250)

self.var\_check=IntVar() self.checkbtn=Checkbutton(frame,variable=self.var\_check,text="I agree the terms and conditions",font=("times new roman",15,"bold"),onvalue=1,offvalue=0,bg="white")

self.checkbtn.place(x=50,y=380)

img=Image.open(r"D:\mini project\hotel images\register-now-button1.jpg") img=img.resize((100,50),Image.LANCZOS) self.photoimage=ImageTk.PhotoImage(img)

b1=Button(frame,image=self.photoimage,command=self.register\_data,borderwidth=0,cursor="hand2",bg

="white") b1.place(x=10,y=470,width=300)

img1=Image.open(r"D:\mini project\hotel images\loginpng.png") img1=img1.resize((100,50),Image.LANCZOS) self.photoimage1=ImageTk.PhotoImage(img1)

b1=Button(frame,image=self.photoimage1,borderwidth=0,cursor="hand2",bg="white") b1.place(x=330,y=470,width=300)

#===================function declaration============== def register\_data(self): if self.var\_fname.get()=="" or self.var\_email.get()=="" or self.var\_security.get()=="Select": messagebox.showerror("error","all fields are required") elif self.var\_passward.get()!=self.var\_cpassward.get(): messagebox.showerror("Error","Passward and confirm passward must be same") elif self.var\_check.get()==0:

messagebox.showerror("error","please agree our terms and conditions") else:

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database=

"hotelmanegement") my\_cursor=conn.cursor()

query=("select \* from register where email=%s")

value=(self.var\_email.get(),) my\_cursor.execute(query,value) row=my\_cursor.fetchone() if row!=None: messagebox.showerror("Error","user already exits,please try another email") else:

my\_cursor.execute("insert into register values(%s,%s,%s,%s,%s,%s,%s)",(

self.var\_fname.get(), self.var\_lname.get(), self.var\_contact.get(), self.var\_email.get(), self.var\_security.get(), self.var\_securityans.get(), self.var\_passward.get()

)) conn.commit() conn.close()

messagebox.showinfo("success","Register Successfully")

**Hotel.py** class HotelManagementSystem: def \_\_init\_\_(self,root): self.root=root

self.root.title("Hotel Management System") self.root.geometry("1550x800+0+0")

# ===================1st img=================

img1=Image.open(r"D:\mini project\hotel images\hotel1.png")

img1=img1.resize((1550,140), Image.LANCZOS) self.photoimg1=ImageTk.PhotoImage(img1)

lbling=Label(self.root,image=self.photoimg1,bd=4,relief=RIDGE) lbling.place(x=0,y=0,width=1550,height=140)

# ===================logo=================

img2=Image.open(r"D:\mini project\hotel images\logohotel.png") img2=img2.resize((230,140), Image.LANCZOS) self.photoimg2=ImageTk.PhotoImage(img2)

lbling=Label(self.root,image=self.photoimg2,bd=4,relief=RIDGE) lbling.place(x=0,y=0,width=230,height=140)

# =======================title============== lbl\_title=Label(self.root,text="HOTEL MANAGEMENT SYSTEM",font=("times new

roman",40,"bold"),bg="black",fg="gold",bd=4,relief=RIDGE)

lbl\_title.place(x=0,y=140,width=1550,height=50)

# ==============main frame================= main\_frame=Frame(self.root,bd=4,relief=RIDGE) main\_frame.place(x=0,y=190,width=1550,height=620)

# ====================menu=============== lbl\_menu=Label(main\_frame,text="MENU",font=("times new roman",20,"bold"),bg="black",fg="gold",bd=4,relief=RIDGE)

lbl\_menu.place(x=0,y=0,width=230)

# ================btn frame============ btn\_frame=Frame(main\_frame,bd=4,relief=RIDGE) btn\_frame.place(x=0,y=35,width=228,height=190)

cust\_btn=Button(btn\_frame,text="CUSTOMER",command=self.cust\_details,width=22,font=("tim es new roman",14,"bold"),bg="black",fg="gold",bd=0,cursor="hand1")

cust\_btn.grid(row=0,column=0,pady=1)

room\_btn=Button(btn\_frame,text="ROOM",command=self.roombooking,width=22,font=("times new roman",14,"bold"),bg="black",fg="gold",bd=0,cursor="hand1")

room\_btn.grid(row=1,column=0,pady=1)

details\_btn=Button(btn\_frame,text="DETAILS",command=self.details\_room,width=22,font=("tim es new roman",14,"bold"),bg="black",fg="gold",bd=0,cursor="hand1")

details\_btn.grid(row=2,column=0,pady=1)

report\_btn=Button(btn\_frame,text="EMPLOYEE",command=self.emp\_details,width=22,font=("ti mes new roman",14,"bold"),bg="black",fg="gold",bd=0,cursor="hand1")

report\_btn.grid(row=3,column=0,pady=1)

logout\_btn=Button(btn\_frame,text="------",width=22,font=("times new

roman",14,"bold"),bg="black",fg="gold",bd=0,cursor="hand1")

logout\_btn.grid(row=4,column=0,pady=1)

# ===============right side img============ img3=Image.open(r"D:\mini project\hotel images\slide3.jpg")

img3=img3.resize((1310,590), Image.LANCZOS) self.photoimg3=ImageTk.PhotoImage(img3)

lbling1=Label(main\_frame,image=self.photoimg3,bd=4,relief=RIDGE) lbling1.place(x=225,y=0,width=1310,height=590)

# =============down img====================

img4=Image.open(r"D:\mini project\hotel images\myh.jpg") img4=img4.resize((230,210), Image.LANCZOS) self.photoimg4=ImageTk.PhotoImage(img4)

lbling1=Label(main\_frame,image=self.photoimg4,bd=4,relief=RIDGE) lbling1.place(x=0,y=225,width=230,height=210)

img5=Image.open(r"D:\mini project\hotel images\khana.jpg") img5=img5.resize((230,190), Image.LANCZOS) self.photoimg5=ImageTk.PhotoImage(img5)

lbling1=Label(main\_frame,image=self.photoimg5,bd=4,relief=RIDGE)

lbling1.place(x=0,y=420,width=230,height=190)

def cust\_details(self):

self.new\_window=Toplevel(self.root) self.app=Cust\_Win(self.new\_window)

def roombooking(self): self.new\_window=Toplevel(self.root)

self.app=Roombooking(self.new\_window)

def details\_room(self): self.new\_window=Toplevel(self.root)

self.app=DetailsRoom(self.new\_window)

def emp\_details(self):

self.new\_window=Toplevel(self.root) self.app=Employee(self.new\_window)

def login(self):

self.new\_window=Toplevel(self.root)

self.app=login\_window(self.new\_window)

if \_\_name\_\_ == "\_\_main\_\_": main()

### Customer.py

from tkinter import\*

from PIL import Image,ImageTk #pip install pillow

from tkinter import ttk import random import mysql.connector

from tkinter import messagebox

class Cust\_Win:

def \_\_init\_\_(self,root):

self.root=root

self.root.title("Hotel Management System") self.root.geometry("1295x550+0+0")

#=============var===============

self.var\_ref=StringVar() x=random.randint(1000,9999) self.var\_ref.set(str(x)) self.var\_cust\_name=StringVar()

self.var\_father=StringVar() self.var\_gender=StringVar() self.var\_post=StringVar() self.var\_mobile=StringVar() self.var\_email=StringVar() self.var\_nationality=StringVar() self.var\_address=StringVar() self.var\_id\_proof=StringVar() self.var\_id\_number=StringVar()

# =========title===============

lbl\_title=Label(self.root,text="ADD CUSTOMER DETAILS",font=("times new

roman",18,"bold"),bg="black",fg="gold",bd=4,relief=RIDGE) lbl\_title.place(x=0,y=0,width=1295,height=50)

# ===============logo=================

img2=Image.open(r"C:\Users\SAIKEERTHIBS\OneDrive\Desktop\dbms project\hotel

images\logohotel.png") img2=img2.resize((100,40), Image.LANCZOS) self.photoimg2=ImageTk.PhotoImage(img2)

lbling=Label(self.root,image=self.photoimg2,bd=4,relief=RIDGE) lbling.place(x=5,y=2,width=100,height=40)

# ====================labelFrame==============

labelframeleft=LabelFrame(self.root,bd=2,relief=RIDGE,text="Customer Details",font=("times

new roman",12,"bold"),bg="white",fg="black",padx=2,) labelframeleft.place(x=5,y=50,width=425,height=490)

# ==========labels and entry==================

# custref

lbl\_cust\_ref=Label(labelframeleft,font=("arial",12,"bold"),text="Customer Ref:",padx=2,pady=6) lbl\_cust\_ref.grid(row=0,column=0,sticky=W)

enty\_ref=ttk.Entry(labelframeleft,textvariable=self.var\_ref,font=("arial",13,"bold"),width=29,state="read only") enty\_ref.grid(row=0,column=1)

# cust name

cname=Label(labelframeleft,text="Customer Name:",font=("arial",12,"bold"),padx=2,pady=6) cname.grid(row=1,column=0,sticky=W)

txtcname=ttk.Entry(labelframeleft,textvariable=self.var\_cust\_name,font=("arial",13,"bold"),width=29) txtcname.grid(row=1,column=1)

# father name lblFname=Label(labelframeleft,text="Father Name:",font=("arial",12,"bold"),padx=2,pady=6) lblFname.grid(row=2,column=0,sticky=W)

txtFname=ttk.Entry(labelframeleft,textvariable=self.var\_father,font=("arial",13,"bold"),width=29) txtFname.grid(row=2,column=1)

# gender combobox

lbl\_gender=Label(labelframeleft,text="Gender:",font=("arial",12,"bold"),padx=2,pady=6) lbl\_gender.grid(row=3,column=0,sticky=W)

combo\_gender=ttk.Combobox(labelframeleft,textvariable=self.var\_gender,font=("arial",13,"bold"),width

=27,state="readonly") combo\_gender["value"]=("Male","Female","Other") combo\_gender.current(0)

combo\_gender.grid(row=3,column=1)

# postcode

lblPostCode=Label(labelframeleft,text="Pincode:",font=("arial",12,"bold"),padx=2,pady=6) lblPostCode.grid(row=4,column=0,sticky=W)

txtPostCode=ttk.Entry(labelframeleft,textvariable=self.var\_post,font=("arial",13,"bold"),width=29) txtPostCode.grid(row=4,column=1)

# mobile number

lblMobile=Label(labelframeleft,text="Mobile No:",font=("arial",12,"bold"),padx=2,pady=6) lblMobile.grid(row=5,column=0,sticky=W)

txtMobile=ttk.Entry(labelframeleft,textvariable=self.var\_mobile,font=("arial",13,"bold"),width=29) txtMobile.grid(row=5,column=1)

# email

lblEmail=Label(labelframeleft,text="Email:",font=("arial",12,"bold"),padx=2,pady=6) lblEmail.grid(row=6,column=0,sticky=W)

txtEmail=ttk.Entry(labelframeleft,textvariable=self.var\_email,font=("arial",13,"bold"),width=29) txtEmail.grid(row=6,column=1)

# nationality

lblNationality=Label(labelframeleft,text="Nationality:",font=("arial",12,"bold"),padx=2,pady=6) lblNationality.grid(row=7,column=0,sticky=W)

combo\_Nationality=ttk.Combobox(labelframeleft,textvariable=self.var\_nationality,font=("arial",13,"bold

"),width=27,state="readonly") combo\_Nationality["value"]=("Indian","USA","Dubai") combo\_Nationality.current(0)

combo\_Nationality.grid(row=7,column=1)

# idproof combobox lblIdProof=Label(labelframeleft,text="Id Proof Type",font=("arial",12,"bold"),padx=2,pady=6) lblIdProof.grid(row=8,column=0,sticky=W)

combo\_id=ttk.Combobox(labelframeleft,textvariable=self.var\_id\_proof,font=("arial",13,"bold"),width=27

,state="readonly") combo\_id["value"]=("Aadhar","Pan","Driving licence","Passport") combo\_id.current(0)

combo\_id.grid(row=8,column=1)

# idproof no

lblIdNumber=Label(labelframeleft,text="Id Number",font=("arial",12,"bold"),padx=2,pady=6) lblIdNumber.grid(row=9,column=0,sticky=W)

txtIdNumber=ttk.Entry(labelframeleft,textvariable=self.var\_id\_number,font=("arial",13,"bold"),width=29

)

txtIdNumber.grid(row=9,column=1)

#address

lblAddress=Label(labelframeleft,text="Address",font=("arial",12,"bold"),padx=2,pady=6) lblAddress.grid(row=10,column=0,sticky=W)

txtAddress=ttk.Entry(labelframeleft,textvariable=self.var\_address,font=("arial",13,"bold"),width=29) txtAddress.grid(row=10,column=1)

# ---------------------btn---------------------------

btn\_frame=Frame(labelframeleft,bd=2,relief=RIDGE) btn\_frame.place(x=0,y=400,width=412,height=40)

btnAdd=Button(btn\_frame,text="Add",command=self.add\_data,font=("arial",12,"bold"),bg="black",fg=" gold",width=9) btnAdd.grid(row=0,column=0,padx=1)

btnUpdate=Button(btn\_frame,text="Update",command=self.Update,font=("arial",12,"bold"),bg="black",f g="gold",width=9) btnUpdate.grid(row=0,column=1,padx=1)

btnReset=Button(btn\_frame,text="Reset",command=self.reset,font=("arial",12,"bold"),bg="black",fg="go ld",width=9) btnReset.grid(row=0,column=2,padx=1)

btnDelete=Button(btn\_frame,text="Delete",command=self.mDelete,font=("arial",12,"bold"),bg="black",f g="gold",width=9) btnDelete.grid(row=0,column=3,padx=1)

# ===========table================== Table\_Frame=LabelFrame(self.root,bd=2,relief=RIDGE,text="View Details And Search

System",font=("times new roman",12,"bold"),bg="white",fg="black",padx=2,) Table\_Frame.place(x=435,y=50,width=860,height=490)

lblSearch=Label(Table\_Frame,font=("arial",12,"bold"),text="Search By:",bg="red",fg="white") lblSearch.grid(row=0,column=0,sticky=W,padx=2) self.Search\_var=StringVar()

combo\_Search=ttk.Combobox(Table\_Frame,textvariable=self.Search\_var,font=("arial",13,"bold"),width=

24,state="readonly") combo\_Search["value"]=("Mobile","Ref") combo\_Search.current(0)

combo\_Search.grid(row=0,column=1,padx=2)

self.txt\_search=StringVar()

txtSearch=ttk.Entry(Table\_Frame,textvariable=self.txt\_search,font=("arial",13,"bold"),width=24) txtSearch.grid(row=0,column=2,padx=2)

btnSearch=Button(Table\_Frame,text="Search",command=self.Search,font=("arial",11,"bold"),bg="green

",fg="white",width=9) btnSearch.grid(row=0,column=3,padx=1)

btnShowAll=Button(Table\_Frame,text="Show

All",command=self.fetch\_data,font=("arial",11,"bold"),bg="green",fg="white",width=9) btnShowAll.grid(row=0,column=4,padx=1)

# ======shw data table========

details\_table=Frame(Table\_Frame,bd=2,relief=RIDGE) details\_table.place(x=0,y=50,width=860,height=350)

scroll\_x=ttk.Scrollbar(details\_table,orient=HORIZONTAL) scroll\_y=ttk.Scrollbar(details\_table,orient=VERTICAL)

self.Cust\_Details\_Table=ttk.Treeview(details\_table,column=("ref","name","father","gender","post","mob ile","email","nationality","idproof","idnumber","address"),xscrollcommand=scroll\_x.set,yscrollcommand

=scroll\_y) scroll\_x.pack(side=BOTTOM,fill=X) scroll\_y.pack(side=RIGHT,fill=Y)

scroll\_x.config(command=self.Cust\_Details\_Table.xview) scroll\_y.config(command=self.Cust\_Details\_Table.yview)

self.Cust\_Details\_Table.heading("ref",text="Ref No") self.Cust\_Details\_Table.heading("name",text="Name") self.Cust\_Details\_Table.heading("father",text="Father") self.Cust\_Details\_Table.heading("gender",text="Gender") self.Cust\_Details\_Table.heading("post",text="Post") self.Cust\_Details\_Table.heading("mobile",text="Mobile") self.Cust\_Details\_Table.heading("email",text="Email")

self.Cust\_Details\_Table.heading("nationality",text="Nationality") self.Cust\_Details\_Table.heading("idproof",text="Idproof") self.Cust\_Details\_Table.heading("idnumber",text="IdNumber") self.Cust\_Details\_Table.heading("address",text="Address")

self.Cust\_Details\_Table["show"]="headings" self.Cust\_Details\_Table.column("ref",width=100) self.Cust\_Details\_Table.column("name",width=100) self.Cust\_Details\_Table.column("father",width=100) self.Cust\_Details\_Table.column("gender",width=100) self.Cust\_Details\_Table.column("post",width=100) self.Cust\_Details\_Table.column("mobile",width=100) self.Cust\_Details\_Table.column("email",width=100) self.Cust\_Details\_Table.column("nationality",width=100) self.Cust\_Details\_Table.column("idproof",width=100) self.Cust\_Details\_Table.column("idnumber",width=100) self.Cust\_Details\_Table.column("address",width=100)

self.Cust\_Details\_Table.pack(fill=BOTH,expand=1)

self.Cust\_Details\_Table.bind("<ButtonRelease-1>",self.get\_cursor) self.fetch\_data()

def add\_data(self): if self.var\_mobile.get()=="" or self.var\_father.get()=="": messagebox.showerror("Error","All fields are required",parent=self.root) else: try:

conn=mysql.connector.connect(host="localhost",username="root",password="Keerthi@182!",database=" hotelmanagement")

my\_cursor=conn.cursor()

my\_cursor.execute("insert into customer values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",( self.var\_ref.get(), self.var\_cust\_name.get(), self.var\_father.get(), self.var\_gender.get(), self.var\_post.get(), self.var\_mobile.get(), self.var\_email.get(), self.var\_nationality.get(), self.var\_id\_proof.get(), self.var\_id\_number.get(), self.var\_address.get()

)) conn.commit() self.fetch\_data() conn.close() messagebox.showinfo("Success","Customer has been added",parent=self.root) except Exception as es:

messagebox.showwarning("warning","some thing went wrong:{str(es)}",parent=self.root)

def fetch\_data(self):

conn=mysql.connector.connect(host="localhost",username="root",password="Keerthi@182!",database=" hotelmanagement") my\_cursor=conn.cursor()

my\_cursor.execute("select \* from customer") rows=my\_cursor.fetchall() if len(rows)!=0:

self.Cust\_Details\_Table.delete(\*self.Cust\_Details\_Table.get\_children()) for i in rows:

self.Cust\_Details\_Table.insert("",END,values=i) conn.commit() conn.close() def get\_cursor(self,event=""): cursor\_row=self.Cust\_Details\_Table.focus() content=self.Cust\_Details\_Table.item(cursor\_row) row=content["values"]

self.var\_ref.set(row[0]), self.var\_cust\_name.set(row[1]), self.var\_father.set(row[2]), self.var\_gender.set(row[3]), self.var\_post.set(row[4]), self.var\_mobile.set(row[5]), self.var\_email.set(row[6]), self.var\_nationality.set(row[7]), self.var\_id\_proof.set(row[8]), self.var\_id\_number.set(row[9]),

self.var\_address.set(row[10])

def Update(self):

if self.var\_mobile.get()=="": messagebox.showerror("Error","Please enter mobile number",parent=self.root) else:

conn=mysql.connector.connect(host="localhost",username="root",password="Keerthi@182!",database=" hotelmanagement") my\_cursor=conn.cursor()

my\_cursor.execute("update customer set

Name=%s,Father=%s,Gender=%s,PostCode=%s,Mobile=%s,Email=%s,Nationality=%s,Idproof=%s,Idn umber=%s,Address=%s where Ref=%s",(

self.var\_cust\_name.get(),

self.var\_father.get(),

self.var\_gender.get(),

self.var\_post.get(),

self.var\_mobile.get(),

self.var\_email.get(),

self.var\_nationality.get(),

self.var\_id\_proof.get(),

self.var\_id\_number.get(),

self.var\_address.get(),

self.var\_ref.get()

)) conn.commit() self.fetch\_data() conn.close()

messagebox.showinfo("update","Customer details has been updated

successfully",parent=self.root)

def mDelete(self):

mDelete=messagebox.askyesno("Hotel Management System","do you want delete

customer",parent=self.root) if mDelete>0:

conn=mysql.connector.connect(host="localhost",username="root",password="Keerthi@182!",database=" hotelmanagement") my\_cursor=conn.cursor()

query="delete from customer where Ref=%s"

value=(self.var\_ref.get(),) my\_cursor.execute(query,value) else: if not mDelete:

return

conn.commit() self.fetch\_data()

conn.cl.set("")

def reset(self):

# self.var\_ref.set(""), self.var\_cust\_name.set(""), self.var\_father.set(""), # self.var\_gender.set(""), self.var\_post.set(""), self.var\_mobile.set(""), self.var\_email.set(""),

# self.var\_nationality.set(""), # self.var\_id\_proof.set(""), self.var\_id\_number.set(""),

self.var\_address.set("")

x=random.randint(1000,9999) self.var\_ref.set(str(x))

def Search(self):

conn=mysql.connector.connect(host="localhost",username="root",password="Keerthi@182!",database=" hotelmanagement") my\_cursor=conn.cursor()

my\_cursor.execute("Select \* from customer where " + str(self.Search\_var.get()) + " LIKE '%" +

str(self.txt\_search.get()) + "%'")

rows=my\_cursor.fetchall() if len(rows)!=0:

self.Cust\_Details\_Table.delete(\*self.Cust\_Details\_Table.get\_children()) for i in rows:

self.Cust\_Details\_Table.insert("",END,values=i) conn.commit() conn.close()

if \_\_name\_\_ == "\_\_main\_\_":

root=Tk() obj=Cust\_Win(root)

root.mainloop()

Room Booking.py

from tkinter import\*

from PIL import Image,ImageTk #pip install pillow from tkinter import ttk import random from time import strftime from datetime import datetime import mysql.connector

from tkinter import messagebox

class Roombooking: def \_\_init\_\_(self,root): self.root=root

self.root.title("Hotel Management System") self.root.geometry("1295x550+0+0")

#======variables======= self.var\_contact=StringVar() self.var\_checkin=StringVar() self.var\_checkout=StringVar() self.var\_roomtype=StringVar() self.var\_roomavailable=StringVar() self.var\_meal=StringVar() self.var\_noofdays=StringVar() self.var\_paidtax=StringVar() self.var\_actualtotal=StringVar()

self.var\_total=StringVar()

# =========title===============

lbl\_title=Label(self.root,text="ROOM BOOKING DETAILS",font=("times new

roman",18,"bold"),bg="black",fg="gold",bd=4,relief=RIDGE) lbl\_title.place(x=0,y=0,width=1295,height=50)

# ===============logo=================

img2=Image.open(r"D:\mini project\hotel images\logohotel.png") img2=img2.resize((100,40), Image.LANCZOS) self.photoimg2=ImageTk.PhotoImage(img2)

lbling=Label(self.root,image=self.photoimg2,bd=4,relief=RIDGE) lbling.place(x=5,y=2,width=100,height=40)

# ====================labelFrame==============

labelframeleft=LabelFrame(self.root,bd=2,relief=RIDGE,text="Room booking

Details",font=("times new roman",12,"bold"),bg="white",fg="black",padx=2,) labelframeleft.place(x=5,y=50,width=425,height=490)

# ==========labels and entry==================

# custref

lbl\_cust\_contact=Label(labelframeleft,font=("arial",12,"bold"),text="Customer

Contac:",padx=2,pady=6) lbl\_cust\_contact.grid(row=0,column=0,sticky=W)

enty\_contact=ttk.Entry(labelframeleft,textvariable=self.var\_contact,font=("arial",13,"bold"),width=20) enty\_contact.grid(row=0,column=1,sticky=W)

#featch data button

btnFetchData=Button(labelframeleft,command=self.Fetch\_contact,text="Fetch

Data",font=("arial",8,"bold"),bg="black",fg="gold",width=8) btnFetchData.place(x=347,y=4)

#check in date

check\_in\_date=Label(labelframeleft,font=("arial",12,"bold"),text="Check\_in

Date:",padx=2,pady=6) check\_in\_date.grid(row=1,column=0,sticky=W)

txtcheck\_in\_date=ttk.Entry(labelframeleft,textvariable=self.var\_checkin,font=("arial",13,"bold"),width=2

9) txtcheck\_in\_date.grid(row=1,column=1)

#check out date

lbl\_Check\_out=Label(labelframeleft,font=("arial",12,"bold"),text="Check\_Out Date:",padx=2,pady=6)

lbl\_Check\_out.grid(row=2,column=0,sticky=W)

txt\_Check\_out=ttk.Entry(labelframeleft,textvariable=self.var\_checkout,font=("arial",13,"bold"),width=29

)

txt\_Check\_out.grid(row=2,column=1)

#room type

label\_RoomType=Label(labelframeleft,font=("arial",12,"bold"),text="Room

Type:",padx=2,pady=6) label\_RoomType.grid(row=3,column=0,sticky=W)

combo\_RoomType=ttk.Combobox(labelframeleft,textvariable=self.var\_roomtype,font=("arial",12,"bold"

),width=27,state="readonly") combo\_RoomType["value"]=("Single","Double","Laxary") combo\_RoomType.current(0)

combo\_RoomType.grid(row=3,column=1)

#available room

lblRoomAvailable=Label(labelframeleft,font=("arial",12,"bold"),text="Room

number:",padx=2,pady=6) lblRoomAvailable.grid(row=4,column=0,sticky=W)

txtRoomAvailable=ttk.Entry(labelframeleft,textvariable=self.var\_roomavailable,font=("arial",13,"bold"), width=29) txtRoomAvailable.grid(row=4,column=1)

#conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database ="hotelmanegement")

#my\_cursor=conn.cursor()

#my\_cursor.execute("select RoomNo from details")

#rows=my\_cursor.fetchall()

#combo\_RoomNo=ttk.Combobox(labelframeleft,textvariable=self.var\_roomavailable,font=("arial",12,"b old"),width=27,state="readonly")

#combo\_RoomNo["value"]=rows

#combo\_RoomNo.current(0)

#combo\_RoomNo.grid(row=4,column=1)

#meal

lblMeal=Label(labelframeleft,font=("arial",12,"bold"),text="Meal",padx=2,pady=6) lblMeal.grid(row=5,column=0,sticky=W)

txtMeal=ttk.Entry(labelframeleft,textvariable=self.var\_meal,font=("arial",13,"bold"),width=29) txtMeal.grid(row=5,column=1)

#no of days

lblNoOfDays=Label(labelframeleft,font=("arial",12,"bold"),text="No Of Days",padx=2,pady=6) lblNoOfDays.grid(row=6,column=0,sticky=W) txtNoOfDays=ttk.Entry(labelframeleft,textvariable=self.var\_noofdays,font=("arial",13,"bold"),width=29) txtNoOfDays.grid(row=6,column=1)

#paid tax

lblIdNumber=Label(labelframeleft,font=("arial",12,"bold"),text="Paid Tax",padx=2,pady=6) lblIdNumber.grid(row=7,column=0,sticky=W)

txtIdNumber=ttk.Entry(labelframeleft,textvariable=self.var\_paidtax,font=("arial",13,"bold"),width=29) txtIdNumber.grid(row=7,column=1)

#sub total1

lblNoOfDays=Label(labelframeleft,font=("arial",12,"bold"),text="Sub Total",padx=2,pady=6) lblNoOfDays.grid(row=8,column=0,sticky=W)

txtNoOfDays=ttk.Entry(labelframeleft,textvariable=self.var\_actualtotal,font=("arial",13,"bold"),width=29

)

txtNoOfDays.grid(row=8,column=1)

#total cost

lblIdNumber=Label(labelframeleft,font=("arial",12,"bold"),text="Total Cost:",padx=2,pady=6) lblIdNumber.grid(row=9,column=0,sticky=W)

txtIdNumber=ttk.Entry(labelframeleft,textvariable=self.var\_total,font=("arial",13,"bold"),width=29) txtIdNumber.grid(row=9,column=1)

#bill btn==========

btnBill=Button(labelframeleft,text="Bill",command=self.total,font=("arial",12,"bold"),bg="black",fg="go ld",width=9) btnBill.grid(row=10,column=0,padx=1,sticky=W)

#btns

btn\_frame=Frame(labelframeleft,bd=2,relief=RIDGE) btn\_frame.place(x=0,y=400,width=412,height=40)

btnAdd=Button(btn\_frame,text="Add",command=self.add\_data,font=("arial",12,"bold"),bg="black",fg=" gold",width=9) btnAdd.grid(row=0,column=0,padx=1)

btnUpdate=Button(btn\_frame,text="Update",command=self.Update,font=("arial",12,"bold"),bg="black",f g="gold",width=9) btnUpdate.grid(row=0,column=1,padx=1)

btnReset=Button(btn\_frame,text="Reset",command=self.reset,font=("arial",12,"bold"),bg="black",fg="go ld",width=9) btnReset.grid(row=0,column=2,padx=1) btnDelete=Button(btn\_frame,text="Delete",command=self.mDelete,font=("arial",12,"bold"),bg="black",f g="gold",width=9) btnDelete.grid(row=0,column=3,padx=1)

#==========================RIGHT SIDE IMAGE====================== img3=Image.open(r"D:\mini project\hotel images\bed.jpg")

img3=img3.resize((520,300), Image.LANCZOS) self.photoimg3=ImageTk.PhotoImage(img3)

lblimg=Label(self.root,image=self.photoimg3,bd=4,relief=RIDGE) lblimg.place(x=750,y=55,width=520,height=200)

# ===========table from search system==================

Table\_Frame=LabelFrame(self.root,bd=2,relief=RIDGE,text="View Details And Search

System",font=("times new roman",12,"bold"),bg="white",fg="black",padx=2,) Table\_Frame.place(x=435,y=280,width=860,height=260)

lblSearch=Label(Table\_Frame,font=("arial",12,"bold"),text="Search By:",bg="red",fg="white") lblSearch.grid(row=0,column=0,sticky=W,padx=2) self.Search\_var=StringVar()

combo\_Search=ttk.Combobox(Table\_Frame,textvariable=self.Search\_var,font=("arial",13,"bold"),width=

24,state="readonly") combo\_Search["value"]=("Contact","Room") combo\_Search.current(0)

combo\_Search.grid(row=0,column=1,padx=2)

self.txt\_search=StringVar()

txtSearch=ttk.Entry(Table\_Frame,textvariable=self.txt\_search,font=("arial",13,"bold"),width=24) txtSearch.grid(row=0,column=2,padx=2)

btnSearch=Button(Table\_Frame,text="Search",command=self.Search,font=("arial",11,"bold"),bg="green

",fg="white",width=9) btnSearch.grid(row=0,column=3,padx=1)

btnShowAll=Button(Table\_Frame,text="Show

All",command=self.fetch\_data,font=("arial",11,"bold"),bg="green",fg="white",width=9) btnShowAll.grid(row=0,column=4,padx=1)

# ======shw data table========

details\_table=Frame(Table\_Frame,bd=2,relief=RIDGE) details\_table.place(x=0,y=50,width=860,height=180)

scroll\_x=ttk.Scrollbar(details\_table,orient=HORIZONTAL) scroll\_y=ttk.Scrollbar(details\_table,orient=VERTICAL)

self.room\_table=ttk.Treeview(details\_table,column=("contact","checkin","checkout","roomtype"

,"roomavailable","meal","noOfdays" ),xscrollcommand=scroll\_x.set, yscrollcommand=scroll\_y)

scroll\_x.pack(side=BOTTOM,fill=X) scroll\_y.pack(side=RIGHT,fill=Y)

scroll\_x.config(command=self.room\_table.xview) scroll\_y.config(command=self.room\_table.yview)

self.room\_table.heading("contact",text="Contact") self.room\_table.heading("checkin",text="Check In") self.room\_table.heading("checkout",text="Check Out") self.room\_table.heading("roomtype",text="RoomType") self.room\_table.heading("roomavailable",text="Room No") self.room\_table.heading("meal",text="Meal")

self.room\_table.heading("noOfdays",text="NoOfDays")

self.room\_table["show"]="headings" self.room\_table.column("contact",width=100) self.room\_table.column("checkin",width=100) self.room\_table.column("checkout",width=100) self.room\_table.column("roomtype",width=100) self.room\_table.column("roomavailable",width=100) self.room\_table.column("meal",width=100) self.room\_table.column("noOfdays",width=100) self.room\_table.pack(fill=BOTH,expand=1)

self.room\_table.bind("<ButtonRelease-1>",self.get\_cursor)

self.fetch\_data()

#add data

def add\_data(self): if self.var\_contact.get()=="" or self.var\_checkin.get()=="": messagebox.showerror("Error","All fields are required",parent=self.root) else: try:

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database=

"hotelmanegement")

my\_cursor=conn.cursor()

my\_cursor.execute("insert into room values(%s,%s,%s,%s,%s,%s,%s)",(

self.var\_contact.get(), self.var\_checkin.get(), self.var\_checkout.get(), self.var\_roomtype.get(), self.var\_roomavailable.get(), self.var\_meal.get(), self.var\_noofdays.get()

)) conn.commit()

self.fetch\_data()

conn.close() messagebox.showinfo("Success","Room booked",parent=self.root) except Exception as es:

messagebox.showwarning("warning","some thing went wrong:{str(es)}",parent=self.root)

#=====fetch data============== def fetch\_data(self):

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database=

"hotelmanegement") my\_cursor=conn.cursor()

my\_cursor.execute("select \* from room") rows=my\_cursor.fetchall() if len(rows)!=0:

self.room\_table.delete(\*self.room\_table.get\_children()) for i in rows:

self.room\_table.insert("",END,values=i) conn.commit()

conn.close()

#get\_cursor def get\_cursor(self,event=""): cursor\_row=self.room\_table.focus() content=self.room\_table.item(cursor\_row) row=content["values"]

self.var\_contact.set(row[0]), self.var\_checkin.set(row[1]), self.var\_checkout.set(row[2]), self.var\_roomtype.set(row[3]), self.var\_roomavailable.set(row[4]), self.var\_meal.set(row[5]),

self.var\_noofdays.set(row[6])

#====update function============== def Update(self): if self.var\_contact.get()=="": messagebox.showerror("Error","Please enter mobile number",parent=self.root) else:

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database=

"hotelmanegement") my\_cursor=conn.cursor()

my\_cursor.execute("update room set

check\_in=%s,check\_out=%s,roomtype=%s,roomavailable=%s,meal=%s,noOfdays=%s where Contact=%s",(

self.var\_checkin.get(),

self.var\_checkout.get(),

self.var\_roomtype.get(),

self.var\_roomavailable.get(),

self.var\_meal.get(),

self.var\_noofdays.get(),

self.var\_contact.get()

)) conn.commit() self.fetch\_data() conn.close()

messagebox.showinfo("Update","Room details has been updated successfully",parent=self.root)

#=============delete======== def mDelete(self):

mDelete=messagebox.askyesno("Hotel Management System","do you want delete

customer",parent=self.root) if mDelete>0:

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database=

"hotelmanegement") my\_cursor=conn.cursor()

query="delete from room where Contact=%s"

value=(self.var\_contact.get(),) my\_cursor.execute(query,value) else: if not mDelete:

return

conn.commit() self.fetch\_data()

conn.cl.set("")

def reset(self): self.var\_contact.set(""), self.var\_checkin.set(""), self.var\_checkout.set(""), self.var\_roomtype.set(""), self.var\_roomavailable.set(""), self.var\_meal.set(""), self.var\_noofdays.set("") self.var\_paidtax.set("") self.var\_actualtotal.set("")

self.var\_total.set("")

#================all data fetch========

def Fetch\_contact(self):

if self.var\_contact.get()=="": messagebox.showerror("Error","please enter contact number",parent=self.root) else: conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database=

"hotelmanegement") my\_cursor=conn.cursor()

query=("select Name from customer where Mobile=%s") value=(self.var\_contact.get(),) my\_cursor.execute(query,value) row=my\_cursor.fetchone()

if row==None:

messagebox.showerror("Error","This number not found",parent=self.root) else:

conn.commit()

conn.close()

showDataframe=Frame(self.root,bd=4,relief=RIDGE,padx=2) showDataframe.place(x=450,y=55,width=290,height=180)

lblName=Label(showDataframe,text="Name:",font=("arial",12,"bold")) lblName.place(x=0,y=0)

lbl=Label(showDataframe,text=row,font=("arial",12,"bold")) lbl.place(x=90,y=0)

#========================gender===========================

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database=

"hotelmanegement") my\_cursor=conn.cursor()

query=("select Gender from customer where Mobile=%s") value=(self.var\_contact.get(),) my\_cursor.execute(query,value) row=my\_cursor.fetchone()

lblGender=Label(showDataframe,text="Gender:",font=("arial",12,"bold")) lblGender.place(x=0,y=30)

lbl2=Label(showDataframe,text=row,font=("arial",12,"bold")) lbl2.place(x=90,y=30)

#===============email================

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database=

"hotelmanegement") my\_cursor=conn.cursor()

query=("select Email from customer where Mobile=%s") value=(self.var\_contact.get(),) my\_cursor.execute(query,value) row=my\_cursor.fetchone()

lblmail=Label(showDataframe,text="Email:",font=("arial",12,"bold")) lblmail.place(x=0,y=60) lbl3=Label(showDataframe,text=row,font=("arial",12,"bold")) lbl3.place(x=90,y=60)

#============================nationality================

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database=

"hotelmanegement") my\_cursor=conn.cursor()

query=("select Nationality from customer where Mobile=%s") value=(self.var\_contact.get(),) my\_cursor.execute(query,value) row=my\_cursor.fetchone()

lblNationality=Label(showDataframe,text="Nationality:",font=("arial",12,"bold")) lblNationality.place(x=0,y=90)

lbl4=Label(showDataframe,text=row,font=("arial",12,"bold")) lbl4.place(x=90,y=90)

#====================address===============

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database=

"hotelmanegement") my\_cursor=conn.cursor()

query=("select Address from customer where Mobile=%s") value=(self.var\_contact.get(),) my\_cursor.execute(query,value) row=my\_cursor.fetchone()

lblAddress=Label(showDataframe,text="Adress:",font=("arial",12,"bold")) lblAddress.place(x=0,y=120)

lbl5=Label(showDataframe,text=row,font=("arial",12,"bold")) lbl5.place(x=90,y=120)

#Search system==================== def Search(self):

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database=

"hotelmanegement") my\_cursor=conn.cursor()

my\_cursor.execute("Select \* from room where " + str(self.Search\_var.get()) + " LIKE '%" +

str(self.txt\_search.get()) + "%'")

rows=my\_cursor.fetchall() if len(rows)!=0:

self.room\_table.delete(\*self.room\_table.get\_children()) for i in rows:

self.room\_table.insert("",END,values=i) conn.commit()

conn.close()

def total(self):

inDate=self.var\_checkin.get() outDate=self.var\_checkout.get() inDate=datetime.strptime(inDate,"%d/%m/%Y") outDate=datetime.strptime(outDate,"%d/%m/%Y") self.var\_noofdays.set(abs(outDate-inDate).days)

if (self.var\_meal.get()=="BreakFast" and self.var\_roomtype.get()=="Laxary"):

q1=float(300) q2=float(700)

q3=float(self.var\_noofdays.get()) q4=float(q1+q2) q5=float(q3+q4)

Tax="Rs."+str("%.2f"%((q5)\*0.1))

ST="Rs."+str("%.2f"%((q5)))

TT="Rs."+str("%.2f"%(q5+((q5)\*0.9)))

self.var\_paidtax.set(Tax) self.var\_actualtotal.set(ST) self.var\_total.set(TT)

elif (self.var\_meal.get()=="Lunch" and self.var\_roomtype.get()=="Single"):

q1=float(300) q2=float(700)

q3=float(self.var\_noofdays.get()) q4=float(q1+q2) q5=float(q3+q4)

Tax="Rs."+str("%.2f"%((q5)\*0.1))

ST="Rs."+str("%.2f"%((q5)))

TT="Rs."+str("%.2f"%(q5+((q5)\*0.9)))

self.var\_paidtax.set(Tax) self.var\_actualtotal.set(ST)

self.var\_total.set(TT)

elif (self.var\_meal.get()=="Lunch" and self.var\_roomtype.get()=="Laxary"):

q1=float(300) q2=float(700)

q3=float(self.var\_noofdays.get()) q4=float(q1+q2) q5=float(q3+q4)

Tax="Rs."+str("%.2f"%((q5)\*0.1))

ST="Rs."+str("%.2f"%((q5)))

TT="Rs."+str("%.2f"%(q5+((q5)\*0.9)))

self.var\_paidtax.set(Tax) self.var\_actualtotal.set(ST) self.var\_total.set(TT)

elif (self.var\_meal.get()=="Dinner" and self.var\_roomtype.get()=="Laxary"):

q1=float(300) q2=float(700)

q3=float(self.var\_noofdays.get()) q4=float(q1+q2) q5=float(q3+q4) Tax="Rs."+str("%.2f"%((q5)\*0.1))

ST="Rs."+str("%.2f"%((q5)))

TT="Rs."+str("%.2f"%(q5+((q5)\*0.9)))

self.var\_paidtax.set(Tax) self.var\_actualtotal.set(ST)

self.var\_total.set(TT)

elif (self.var\_meal.get()=="BreakFast" and self.var\_roomtype.get()=="Single"):

q1=float(300) q2=float(700)

q3=float(self.var\_noofdays.get()) q4=float(q1+q2) q5=float(q3+q4)

Tax="Rs."+str("%.2f"%((q5)\*0.1))

ST="Rs."+str("%.2f"%((q5)))

TT="Rs."+str("%.2f"%(q5+((q5)\*0.9)))

self.var\_paidtax.set(Tax) self.var\_actualtotal.set(ST) self.var\_total.set(TT)

elif (self.var\_meal.get()=="Dinner" and self.var\_roomtype.get()=="Single"):

q1=float(300) q2=float(700)

q3=float(self.var\_noofdays.get()) q4=float(q1+q2) q5=float(q3+q4)

Tax="Rs."+str("%.2f"%((q5)\*0.1))

ST="Rs."+str("%.2f"%((q5)))

TT="Rs."+str("%.2f"%(q5+((q5)\*0.9)))

self.var\_paidtax.set(Tax) self.var\_actualtotal.set(ST)

self.var\_total.set(TT)

elif (self.var\_meal.get()=="BreakFast" and self.var\_roomtype.get()=="Double"):

q1=float(300) q2=float(700)

q3=float(self.var\_noofdays.get()) q4=float(q1+q2) q5=float(q3+q4)

Tax="Rs."+str("%.2f"%((q5)\*0.1))

ST="Rs."+str("%.2f"%((q5)))

TT="Rs."+str("%.2f"%(q5+((q5)\*0.9)))

self.var\_paidtax.set(Tax) self.var\_actualtotal.set(ST) self.var\_total.set(TT)

elif (self.var\_meal.get()=="Lunch" and self.var\_roomtype.get()=="Double"):

q1=float(300) q2=float(700)

q3=float(self.var\_noofdays.get()) q4=float(q1+q2) q5=float(q3+q4) Tax="Rs."+str("%.2f"%((q5)\*0.1))

ST="Rs."+str("%.2f"%((q5)))

TT="Rs."+str("%.2f"%(q5+((q5)\*0.9)))

self.var\_paidtax.set(Tax) self.var\_actualtotal.set(ST)

self.var\_total.set(TT)

elif (self.var\_meal.get()=="Dinner" and self.var\_roomtype.get()=="Double"):

q1=float(300) q2=float(700)

q3=float(self.var\_noofdays.get()) q4=float(q1+q2) q5=float(q3+q4)

Tax="Rs."+str("%.2f"%((q5)\*0.1))

ST="Rs."+str("%.2f"%((q5)))

TT="Rs."+str("%.2f"%(q5+((q5)\*0.9)))

self.var\_paidtax.set(Tax)

self.var\_actualtotal.set(ST)

if \_\_name\_\_ == "\_\_main\_\_": root=Tk() obj=Roombooking(root) root.mainloop()

**Details.py** from tkinter import\*

from PIL import Image,ImageTk #pip install pillow from tkinter import ttk import random from time import strftime from datetime import datetime import mysql.connector from tkinter import messagebox

class DetailsRoom:

def \_\_init\_\_(self,root): self.root=root

self.root.title("Hotel Management System") self.root.geometry("1295x550+0+0")

# =========title===============

lbl\_title=Label(self.root,text="ROOM BOOKING DETAILS",font=("times new

roman",18,"bold"),bg="black",fg="gold",bd=4,relief=RIDGE) lbl\_title.place(x=0,y=0,width=1295,height=50)

# ===============logo=================

img2=Image.open(r"D:\mini project\hotel images\logohotel.png") img2=img2.resize((100,40), Image.LANCZOS) self.photoimg2=ImageTk.PhotoImage(img2) lbling=Label(self.root,image=self.photoimg2,bd=4,relief=RIDGE) lbling.place(x=5,y=2,width=100,height=40)

# ====================labelFrame==============

labelframeleft=LabelFrame(self.root,bd=2,relief=RIDGE,text="New Room Add",font=("times

new roman",12,"bold"),bg="white",fg="black",padx=2,) labelframeleft.place(x=5,y=50,width=540,height=350)

# ==========labels and entry==================

# floor

lbl\_floor=Label(labelframeleft,font=("arial",12,"bold"),text="Floor:",padx=2,pady=6) lbl\_floor.grid(row=0,column=0,sticky=W,padx=20)

self.var\_floor=StringVar()

enty\_floor=ttk.Entry(labelframeleft,textvariable=self.var\_floor,font=("arial",13,"bold"),width=20) enty\_floor.grid(row=0,column=1,sticky=W)

####room number=====================

lbl\_RoomNumber=Label(labelframeleft,font=("arial",12,"bold"),text="Room

Number:",padx=2,pady=6) lbl\_RoomNumber.grid(row=1,column=0,sticky=W,padx=20)

self.var\_roomno=StringVar()

enty\_RoomNumber=ttk.Entry(labelframeleft,textvariable=self.var\_roomno,font=("arial",13,"bold"),width

=20) enty\_RoomNumber.grid(row=1,column=1,sticky=W)

#============room Type====================

lbl\_RoomType=Label(labelframeleft,font=("arial",12,"bold"),text="Room Type:",padx=2,pady=6) lbl\_RoomType.grid(row=2,column=0,sticky=W,padx=20)

self.var\_RoomType=StringVar()

enty\_RoomType=ttk.Entry(labelframeleft,textvariable=self.var\_RoomType,font=("arial",13,"bold"),width

=20) enty\_RoomType.grid(row=2,column=1,sticky=W)

# ---------------------btn---------------------------

btn\_frame=Frame(labelframeleft,bd=2,relief=RIDGE) btn\_frame.place(x=0,y=250,width=412,height=40)

btnAdd=Button(btn\_frame,text="Add",command=self.add\_data,font=("arial",12,"bold"),bg="black",fg=" gold",width=9) btnAdd.grid(row=0,column=0,padx=1) btnUpdate=Button(btn\_frame,text="Update",command=self.Update,font=("arial",12,"bold"),bg="black",f g="gold",width=9) btnUpdate.grid(row=0,column=1,padx=1)

btnReset=Button(btn\_frame,text="Reset",command=self.reset,font=("arial",12,"bold"),bg="black",fg="go ld",width=9) btnReset.grid(row=0,column=2,padx=1)

btnDelete=Button(btn\_frame,text="Delete",command=self.mDelete,font=("arial",12,"bold"),bg="black",f g="gold",width=9) btnDelete.grid(row=0,column=3,padx=1)

# ===========table from search system==================

Table\_Frame=LabelFrame(self.root,bd=2,relief=RIDGE,text="Show room details",font=("times new roman",12,"bold"),bg="white",fg="black",padx=2,)

Table\_Frame.place(x=600,y=55,width=600,height=350)

scroll\_x=ttk.Scrollbar(Table\_Frame,orient=HORIZONTAL) scroll\_y=ttk.Scrollbar(Table\_Frame,orient=VERTICAL)

self.room\_table=ttk.Treeview(Table\_Frame,column=("floor","roomno","roomtype"

),xscrollcommand=scroll\_x.set, yscrollcommand=scroll\_y)

scroll\_x.pack(side=BOTTOM,fill=X) scroll\_y.pack(side=RIGHT,fill=Y)

self.room\_table.heading("floor",text="Floor") self.room\_table.heading("roomno",text="Room Number") self.room\_table.heading("roomtype",text="RoomType")

self.room\_table["show"]="headings"

self.room\_table.column("floor",width=100) self.room\_table.column("roomno",width=100) self.room\_table.column("roomtype",width=100)

self.room\_table.pack(fill=BOTH,expand=1)

self.room\_table.bind("<ButtonRelease-1>",self.get\_cursor) self.fetch\_data()

#add data

def add\_data(self): if self.var\_floor.get()=="" or self.var\_RoomType.get()=="":

messagebox.showerror("Error","All fields are required",parent=self.root) else: try:

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database= "hotelmanegement")

my\_cursor=conn.cursor() my\_cursor.execute("insert into details values(%s,%s,%s)",(

self.var\_floor.get(), self.var\_roomno.get(), self.var\_RoomType.get(),

)) conn.commit() self.fetch\_data()

conn.close()

messagebox.showinfo("Success","New Room Added Successfully",parent=self.root) except Exception as es: messagebox.showwarning("warning","some thing went wrong:{str(es)}",parent=self.root)

#=====fetch data============== def fetch\_data(self):

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database=

"hotelmanegement") my\_cursor=conn.cursor()

my\_cursor.execute("select \* from details") rows=my\_cursor.fetchall() if len(rows)!=0:

self.room\_table.delete(\*self.room\_table.get\_children()) for i in rows:

self.room\_table.insert("",END,values=i) conn.commit()

conn.close()

#get\_cursor def get\_cursor(self,event=""): cursor\_row=self.room\_table.focus() content=self.room\_table.item(cursor\_row) row=content["values"]

self.var\_floor.set(row[0]), self.var\_roomno.set(row[1]),

self.var\_RoomType.set(row[2])

#====update function============== def Update(self): if self.var\_floor.get()=="":

messagebox.showerror("Error","Please enter mobile number",parent=self.root) else:

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database=

"hotelmanegement") my\_cursor=conn.cursor()

my\_cursor.execute("update details set Floor=%s,RoomType=%s where RoomNo=%s",( self.var\_floor.get(), self.var\_RoomType.get(), self.var\_roomno.get(),

)) conn.commit() self.fetch\_data() conn.close()

messagebox.showinfo("Update"," New Room details has been updated

successfully",parent=self.root)

#=============delete======== def mDelete(self):

mDelete=messagebox.askyesno("Hotel Management System","do you want delete Room

details",parent=self.root) if mDelete>0:

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database=

"hotelmanegement") my\_cursor=conn.cursor()

query="delete from details where RoomNo=%s"

value=(self.var\_roomno.get(),) my\_cursor.execute(query,value) else: if not mDelete:

return

conn.commit() self.fetch\_data() conn.cl.set("")

def reset(self): self.var\_floor.set(""), self.var\_roomno.set(""), self.var\_RoomType.set("")

if \_\_name\_\_ == "\_\_main\_\_":

root=Tk()

obj=DetailsRoom(root) root.mainloop()

### Employee.py

from tkinter import\*

from PIL import Image,ImageTk #pip install pillow from tkinter import ttk

from time import strftime from datetime import datetime import mysql.connector

from tkinter import messagebox

class Employee:

def \_\_init\_\_(self,root): self.root=root

self.root.title("Hotel Management System") self.root.geometry("1295x550+0+0")

#=============VARIABLES===========

self.var\_emp\_id=StringVar() self.var\_emp\_name=StringVar() self.var\_efather=StringVar() self.var\_egender=StringVar() self.var\_salary=StringVar() self.var\_emobile=StringVar() self.var\_eemail=StringVar() self.var\_jobdesc=StringVar() self.var\_eaddress=StringVar() self.var\_eid\_type=StringVar() self.var\_eid\_number=StringVar()

# =========title===============

lbl\_title=Label(self.root,text="Employee details",font=("times new

roman",18,"bold"),bg="black",fg="gold",bd=4,relief=RIDGE) lbl\_title.place(x=0,y=0,width=1295,height=50)

# ===============logo=================

img2=Image.open(r"D:\mini project\hotel images\logohotel.png") img2=img2.resize((100,40), Image.LANCZOS) self.photoimg2=ImageTk.PhotoImage(img2)

lbling=Label(self.root,image=self.photoimg2,bd=4,relief=RIDGE) lbling.place(x=5,y=2,width=100,height=40)

# ====================labelFrame==============

labelframeleft1=LabelFrame(self.root,bd=2,relief=RIDGE,text="New Employee

Details",font=("times new roman",12,"bold"),bg="white",fg="black",padx=2,) labelframeleft1.place(x=5,y=50,width=425,height=490)

# ==========labels and entry==================

# employee id

lbl\_cust\_ref=Label(labelframeleft1,font=("arial",12,"bold"),text="Employee ID:",padx=2,pady=6) lbl\_cust\_ref.grid(row=0,column=0,sticky=W)

enty\_ref=ttk.Entry(labelframeleft1,textvariable=self.var\_emp\_id,font=("arial",13,"bold"),width=29) enty\_ref.grid(row=0,column=1)

# EMPLOYEE name

ename=Label(labelframeleft1,text="Employee Name:",font=("arial",12,"bold"),padx=2,pady=6) ename.grid(row=1,column=0,sticky=W) txtename=ttk.Entry(labelframeleft1,textvariable=self.var\_emp\_name,font=("arial",13,"bold"),width=29) txtename.grid(row=1,column=1)

# father name

lblFrname=Label(labelframeleft1,text="Father Name:",font=("arial",12,"bold"),padx=2,pady=6) lblFrname.grid(row=2,column=0,sticky=W)

txtFrname=ttk.Entry(labelframeleft1,textvariable=self.var\_efather,font=("arial",13,"bold"),width=29) txtFrname.grid(row=2,column=1)

# gender combobox

lbl\_gender=Label(labelframeleft1,text="Gender:",font=("arial",12,"bold"),padx=2,pady=6) lbl\_gender.grid(row=3,column=0,sticky=W)

combo\_gender=ttk.Combobox(labelframeleft1,textvariable=self.var\_egender,font=("arial",13,"bold"),wid th=27,state="readonly") combo\_gender["value"]=("Male","Female","Other") combo\_gender.current(0)

combo\_gender.grid(row=3,column=1)

# Salary

lblPostCode1=Label(labelframeleft1,text="Salary:",font=("arial",12,"bold"),padx=2,pady=6) lblPostCode1.grid(row=4,column=0,sticky=W)

txtPostCode1=ttk.Entry(labelframeleft1,textvariable=self.var\_salary,font=("arial",13,"bold"),width=29) txtPostCode1.grid(row=4,column=1)

# mobile number

lblMobile=Label(labelframeleft1,text="Mobile No:",font=("arial",12,"bold"),padx=2,pady=6) lblMobile.grid(row=5,column=0,sticky=W)

txtMobile=ttk.Entry(labelframeleft1,textvariable=self.var\_emobile,font=("arial",13,"bold"),width=29) txtMobile.grid(row=5,column=1)

# email

lblEmail1=Label(labelframeleft1,text="Email:",font=("arial",12,"bold"),padx=2,pady=6) lblEmail1.grid(row=6,column=0,sticky=W)

txtEmail=ttk.Entry(labelframeleft1,textvariable=self.var\_eemail,font=("arial",13,"bold"),width=29) txtEmail.grid(row=6,column=1)

# job desc

lbljobdesc=Label(labelframeleft1,text="Job Description",font=("arial",12,"bold"),padx=2,pady=6) lbljobdesc.grid(row=7,column=0,sticky=W)

combo\_jobdesc=ttk.Combobox(labelframeleft1,textvariable=self.var\_jobdesc,font=("arial",13,"bold"),wi dth=27,state="readonly") combo\_jobdesc["value"]=("Reception","Maintainance","Kitchen","Manager") combo\_jobdesc.current(0)

combo\_jobdesc.grid(row=7,column=1)

# idproof combobox

lblIdProof1=Label(labelframeleft1,text="Id Proof Type",font=("arial",12,"bold"),padx=2,pady=6) lblIdProof1.grid(row=8,column=0,sticky=W)

combo\_id1=ttk.Combobox(labelframeleft1,textvariable=self.var\_eid\_type,font=("arial",13,"bold"),width=

27,state="readonly") combo\_id1["value"]=("Aadhar","Pan","Driving licence","Passport") combo\_id1.current(0)

combo\_id1.grid(row=8,column=1)

# idproof no

lblIdNumber=Label(labelframeleft1,text="Id Number",font=("arial",12,"bold"),padx=2,pady=6) lblIdNumber.grid(row=9,column=0,sticky=W)

txtIdNumber=ttk.Entry(labelframeleft1,textvariable=self.var\_eid\_number,font=("arial",13,"bold"),width=

29) txtIdNumber.grid(row=9,column=1)

#address

lblAddress=Label(labelframeleft1,text="Address",font=("arial",12,"bold"),padx=2,pady=6) lblAddress.grid(row=10,column=0,sticky=W)

txtAddress=ttk.Entry(labelframeleft1,textvariable=self.var\_eaddress,font=("arial",13,"bold"),width=29) txtAddress.grid(row=10,column=1)

#btns============================= btn\_frame=Frame(labelframeleft1,bd=2,relief=RIDGE) btn\_frame.place(x=0,y=400,width=412,height=40)

btnAdd=Button(btn\_frame,text="Add",command=self.add\_data,font=("arial",12,"bold"),bg="black",fg=" gold",width=9) btnAdd.grid(row=0,column=0,padx=1)

btnUpdate=Button(btn\_frame,text="Update",command=self.Update,font=("arial",12,"bold"),bg="black",f g="gold",width=9) btnUpdate.grid(row=0,column=1,padx=1)

btnReset=Button(btn\_frame,text="Reset",command=self.reset,font=("arial",12,"bold"),bg="black",fg="go ld",width=9) btnReset.grid(row=0,column=2,padx=1) btnDelete=Button(btn\_frame,text="Delete",command=self.mDelete,font=("arial",12,"bold"),bg="black",f g="gold",width=9) btnDelete.grid(row=0,column=3,padx=1)

# ===========table==================

Table\_Frame=LabelFrame(self.root,bd=2,relief=RIDGE,text="View Details And Search

System",font=("times new roman",12,"bold"),bg="white",fg="black",padx=2,) Table\_Frame.place(x=435,y=50,width=860,height=490)

lblSearch=Label(Table\_Frame,font=("arial",12,"bold"),text="Search By:",bg="red",fg="white") lblSearch.grid(row=0,column=0,sticky=W,padx=2)

self.search\_var=StringVar()

combo\_Search=ttk.Combobox(Table\_Frame,textvariable=self.search\_var,font=("arial",13,"bold"),width=

24,state="readonly") combo\_Search["value"]=("Mobile") combo\_Search.current(0)

combo\_Search.grid(row=0,column=1,padx=2)

self.txt\_search=StringVar()

txtSearch=ttk.Entry(Table\_Frame,textvariable=self.txt\_search,font=("arial",13,"bold"),width=24) txtSearch.grid(row=0,column=2,padx=2)

btnSearch=Button(Table\_Frame,text="Search",command=self.Search,font=("arial",11,"bold"),bg="green

",fg="white",width=9) btnSearch.grid(row=0,column=3,padx=1)

btnShowAll=Button(Table\_Frame,text="Show

All",command=self.fetch\_data,font=("arial",11,"bold"),bg="green",fg="white",width=9) btnShowAll.grid(row=0,column=4,padx=1)

# ======shw data table========

details\_table=Frame(Table\_Frame,bd=2,relief=RIDGE) details\_table.place(x=0,y=50,width=840,height=350) scroll\_x=ttk.Scrollbar(details\_table,orient=HORIZONTAL) scroll\_y=ttk.Scrollbar(details\_table,orient=VERTICAL)

self.Emp\_Details\_Table=ttk.Treeview(details\_table,column=("employeeid","name","father","gender","sal ary","mobile","email","jobdesc","idproof","idnumber","address"),xscrollcommand=scroll\_x.set,yscrollco mmand=scroll\_y) scroll\_x.pack(side=BOTTOM,fill=X) scroll\_y.pack(side=RIGHT,fill=Y)

scroll\_x.config(command=self.Emp\_Details\_Table.xview) scroll\_y.config(command=self.Emp\_Details\_Table.yview) self.Emp\_Details\_Table.heading("employeeid",text="Emp ID") self.Emp\_Details\_Table.heading("name",text="Name") self.Emp\_Details\_Table.heading("father",text="Father") self.Emp\_Details\_Table.heading("gender",text="Gender") self.Emp\_Details\_Table.heading("salary",text="Salary") self.Emp\_Details\_Table.heading("mobile",text="Mobile") self.Emp\_Details\_Table.heading("email",text="Email") self.Emp\_Details\_Table.heading("jobdesc",text="Job description") self.Emp\_Details\_Table.heading("idproof",text="Idproof") self.Emp\_Details\_Table.heading("idnumber",text="IdNumber") self.Emp\_Details\_Table.heading("address",text="Address")

self.Emp\_Details\_Table["show"]="headings" self.Emp\_Details\_Table.column("employeeid",width=100) self.Emp\_Details\_Table.column("name",width=100) self.Emp\_Details\_Table.column("father",width=100) self.Emp\_Details\_Table.column("gender",width=100) self.Emp\_Details\_Table.column("salary",width=100) self.Emp\_Details\_Table.column("mobile",width=100) self.Emp\_Details\_Table.column("email",width=100) self.Emp\_Details\_Table.column("jobdesc",width=100) self.Emp\_Details\_Table.column("idproof",width=100) self.Emp\_Details\_Table.column("idnumber",width=100) self.Emp\_Details\_Table.column("address",width=100)

self.Emp\_Details\_Table.pack(fill=BOTH,expand=1)

self.Emp\_Details\_Table.bind("<ButtonRelease-1>",self.get\_cursor) self.fetch\_data()

def add\_data(self):

if self.var\_emobile.get()=="" or self.var\_efather.get()=="": messagebox.showerror("Error","All fields are required",parent=self.root) else: try:

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database=

"hotelmanegement")

my\_cursor=conn.cursor()

my\_cursor.execute("insert into employee values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",( self.var\_emp\_id.get(), self.var\_emp\_name.get(), self.var\_efather.get(), self.var\_egender.get(), self.var\_salary.get(), self.var\_emobile.get(), self.var\_eemail.get(), self.var\_jobdesc.get(), self.var\_eid\_type.get(), self.var\_eid\_number.get(), self.var\_eaddress.get())) conn.commit()

self.fetch\_data() conn.close() messagebox.showinfo("Success","Employee has been added",parent=self.root) except Exception as es: messagebox.showwarning("warning","some thing went wrong:{str(es)}",parent=self.root)

def fetch\_data(self):

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database=

"hotelmanegement") my\_cursor=conn.cursor()

my\_cursor.execute("select \* from employee") rows=my\_cursor.fetchall() if len(rows)!=0:

self.Emp\_Details\_Table.delete(\*self.Emp\_Details\_Table.get\_children()) for i in rows:

self.Emp\_Details\_Table.insert("",END,values=i) conn.commit() conn.close() def get\_cursor(self,event=""): cursor\_row=self.Emp\_Details\_Table.focus() content=self.Emp\_Details\_Table.item(cursor\_row) row=content["values"]

self.var\_emp\_id.set(row[0]), self.var\_emp\_name.set(row[1]), self.var\_efather.set(row[2]), self.var\_egender.set(row[3]), self.var\_salary.set(row[4]), self.var\_emobile.set(row[5]), self.var\_eemail.set(row[6]), self.var\_jobdesc.set(row[7]), self.var\_eid\_type.set(row[8]), self.var\_eid\_number.set(row[9]), self.var\_eaddress.set(row[10])

def Update(self): if self.var\_emobile.get()=="": messagebox.showerror("Error","Please enter mobile number",parent=self.root) else:

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database=

"hotelmanegement") my\_cursor=conn.cursor()

my\_cursor.execute("update employee set

name=%s,father=%s,gender=%s,salary=%s,mobile=%s,email=%s,jobdesc=%s,idproof=%s,idnumber=%s ,address=%s where employeeid=%s",(

self.var\_emp\_name.get(),

self.var\_efather.get(),

self.var\_egender.get(),

self.var\_salary.get(),

self.var\_emobile.get(),

self.var\_eemail.get(),

self.var\_jobdesc.get(),

self.var\_eid\_type.get(),

self.var\_eid\_number.get(),

self.var\_eaddress.get(),

self.var\_emp\_id.get(),

)) conn.commit() self.fetch\_data() conn.close()

messagebox.showinfo("update","Employee details has been updated

successfully",parent=self.root)

def mDelete(self):

mDelete=messagebox.askyesno("Hotel Management System","do you want delete

employee",parent=self.root) if mDelete>0:

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database=

"hotelmanegement") my\_cursor=conn.cursor()

query="delete from employee where employeeid=%s" value=(self.var\_emp\_id.get(),) my\_cursor.execute(query,value) else: if not mDelete:

return

conn.commit() self.fetch\_data() conn.cl.set("")

def reset(self):

self.var\_emp\_id.set(""), self.var\_emp\_name.set(""), self.var\_efather.set(""), #self.var\_egender.set(""), self.var\_salary.set(""), self.var\_emobile.set(""), self.var\_eemail.set(""), #self.var\_jobdesc.set(""), #self.var\_eid\_type.set(""), self.var\_eid\_number.set(""),

self.var\_eaddress.set("")

def Search(self):

conn=mysql.connector.connect(host="localhost",username="root",password="Chethana@123",database=

"hotelmanegement") my\_cursor=conn.cursor()

my\_cursor.execute("Select \* from employee where " + str(self.search\_var.get()) + " LIKE '%" +

str(self.txt\_search.get()) + "%'")

rows=my\_cursor.fetchall() if len(rows)!=0:

self.Emp\_Details\_Table.delete(\*self.Emp\_Details\_Table.get\_children()) for i in rows:

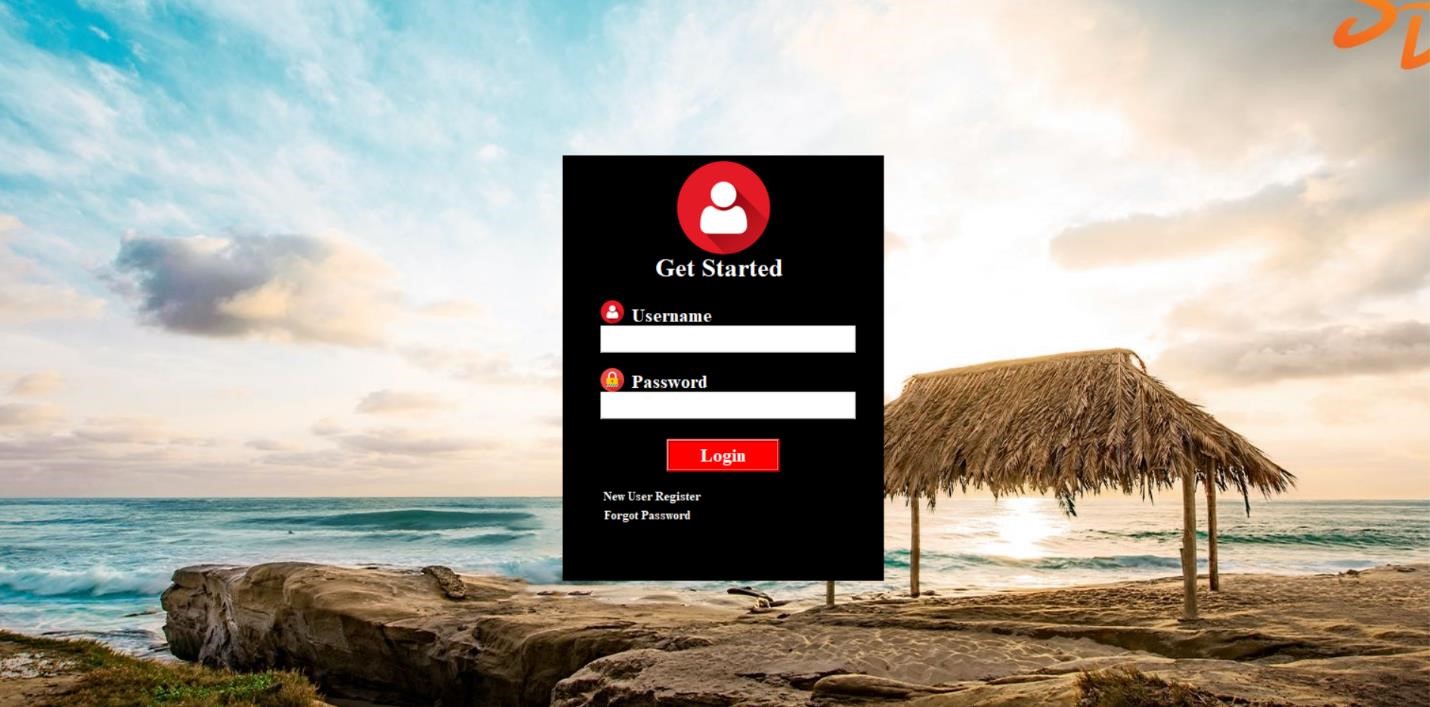
self.Emp\_Details\_Table.insert("",END,values=i) conn.commit()

conn.close()

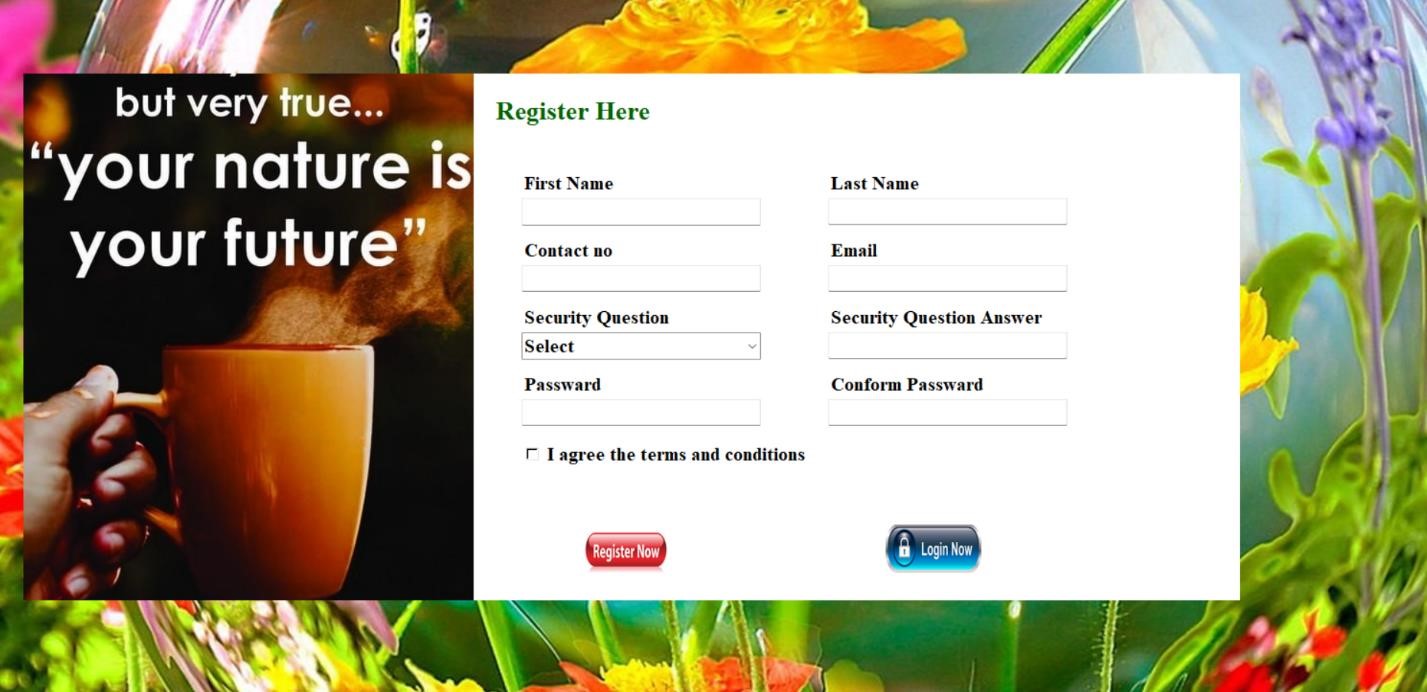
if \_\_name\_\_ == "\_\_main\_\_":

root=Tk() obj=Employee(root) root.mainloop()

**Snapshots:**



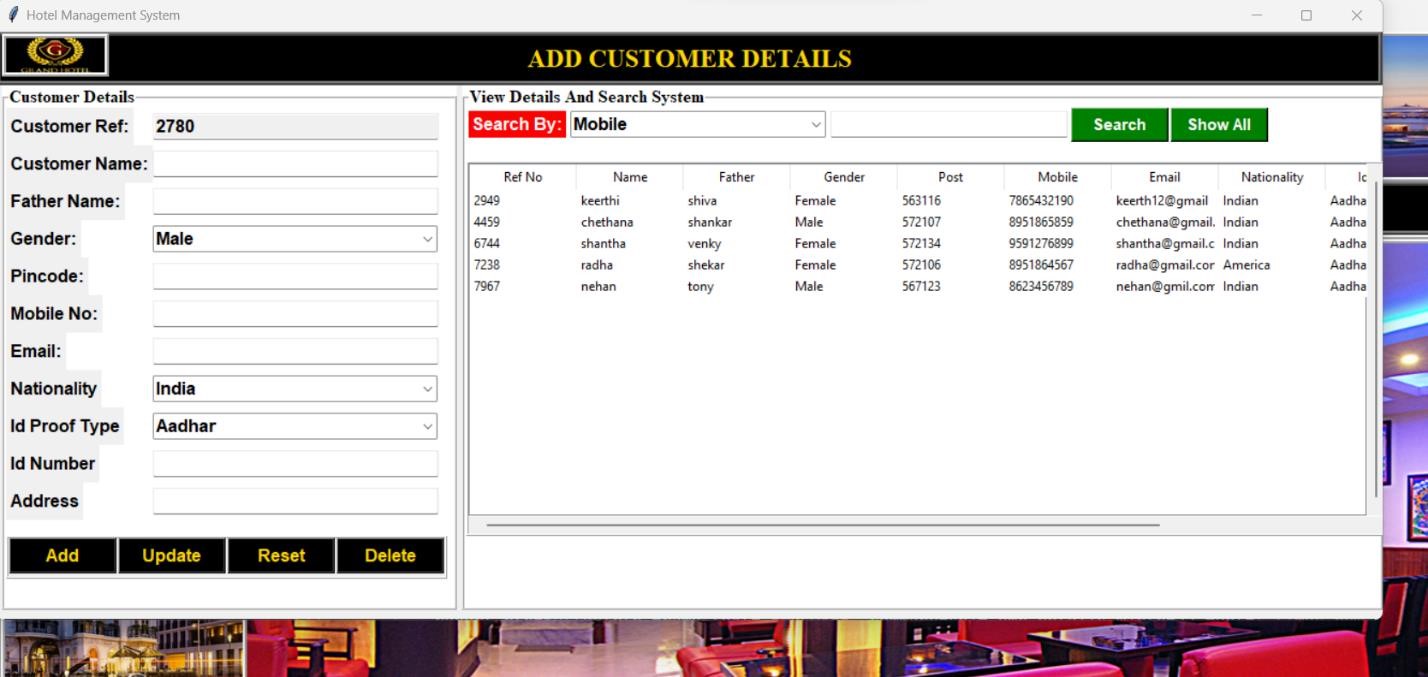
## Login page



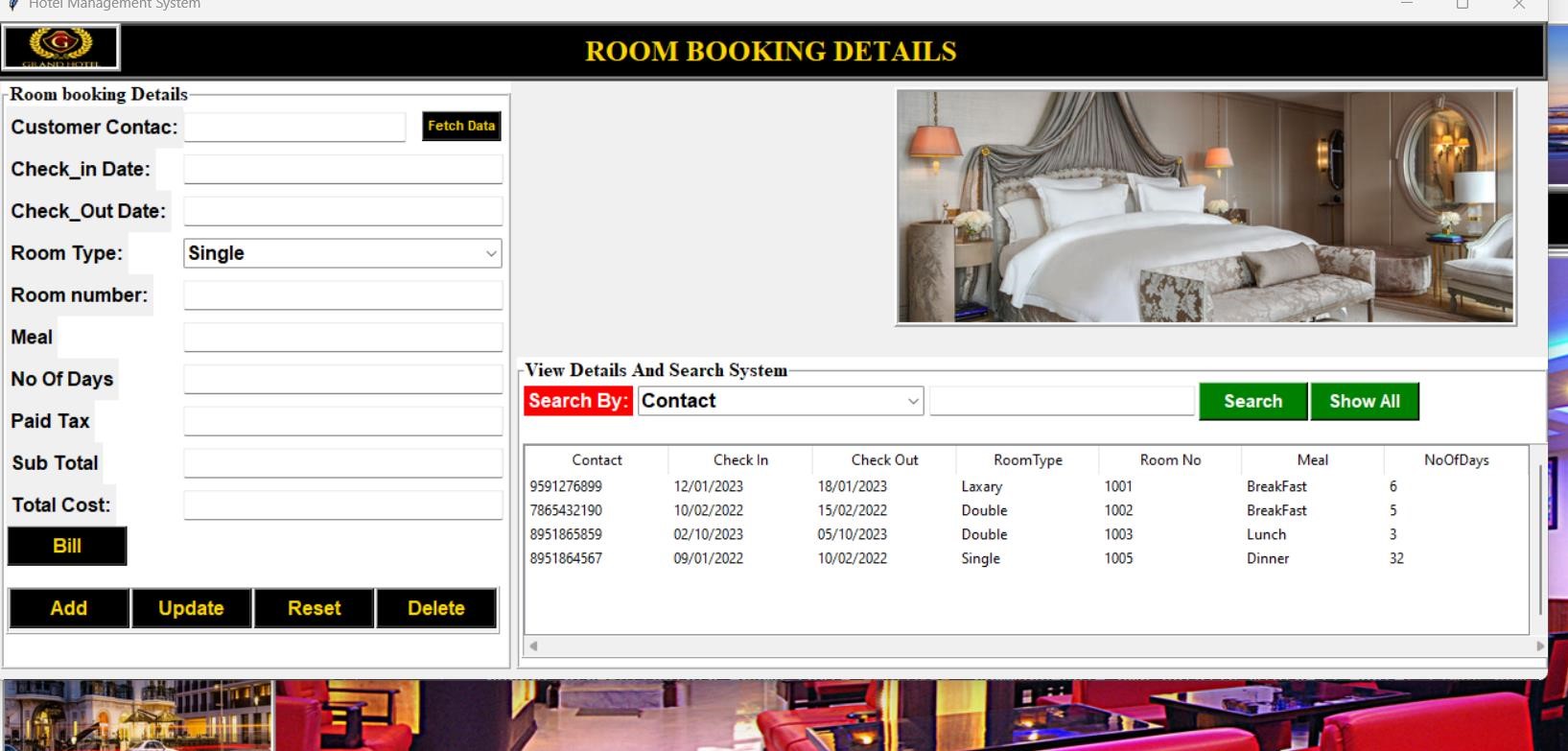
**Registration page**



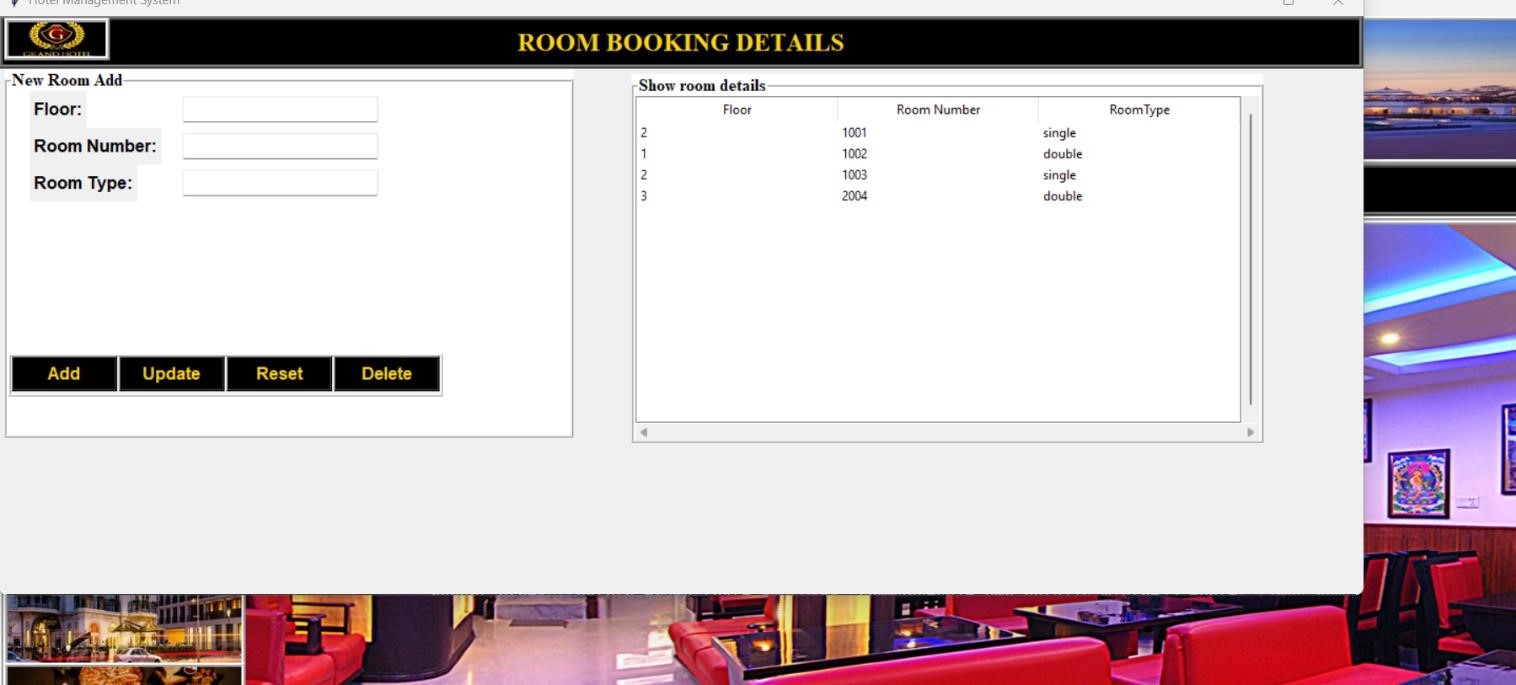
**Hotel home page**



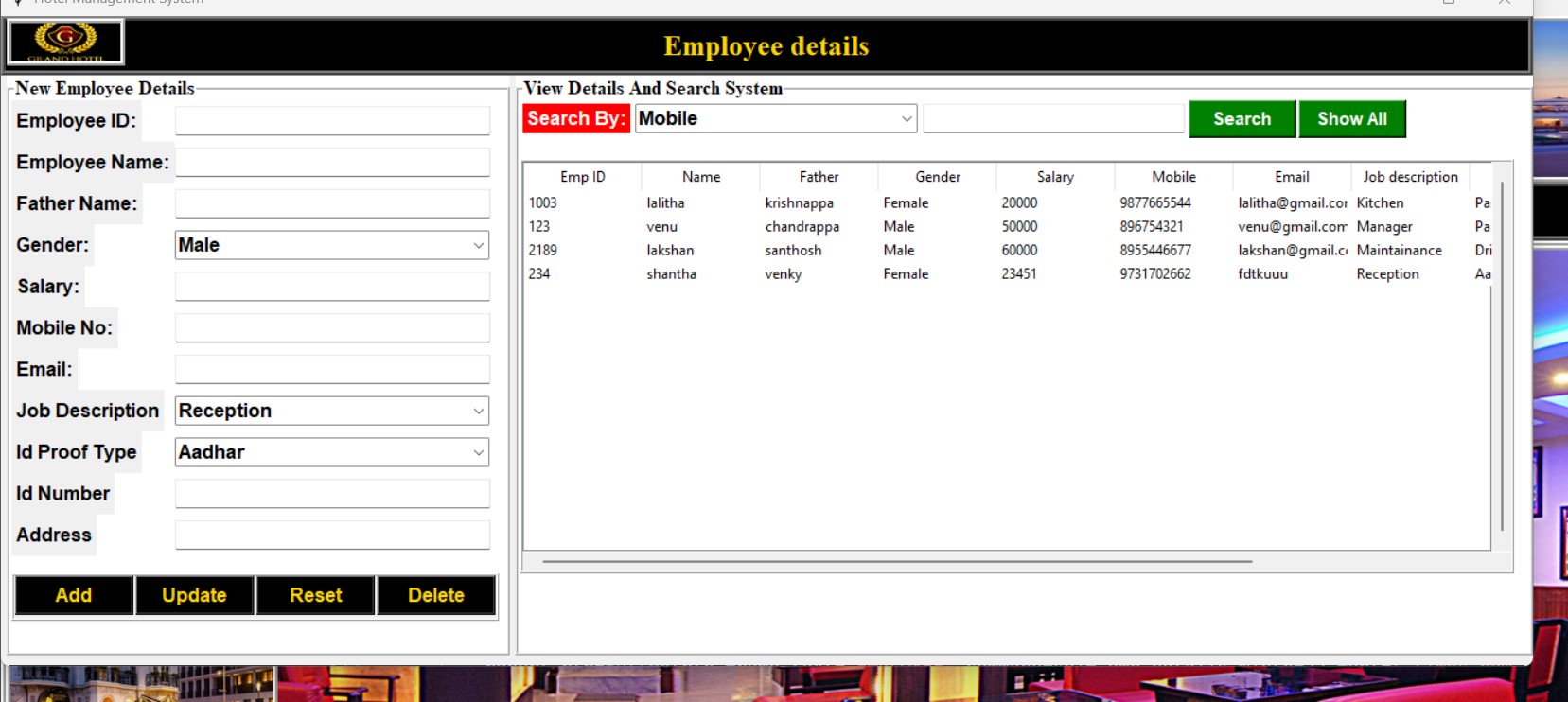
**Customer Registration table**



**Room Booking Details table**



### Room Details table



**Employee Details table**

### CONCLUSION

The project Hotel Management System (HMS) is for computerizing the working in a hotel. The software takes care of all the requirements of an average hotel and is capable to provide easy and effective storage of information related to customer, employee and rooms that have been booked

It generates bills, provides room details including room type. It also provides customer details such as check\_in date, check\_out date, meal etc. The system also provides the facility of backup as per requirement.

### BIBLIOGRAPHY

Few of the book(s) and websites that were instrumental in helping us to complete this project are as mentioned below.

### BOOK

1. Fundamental of Database System by Elmasri and Navathe ,5th Edition, Addison-Wesley,2007.
2. Database System Concepts by Avi Silberschatz, Henry F Korth, and S. Sudharshan,1996.
3. Concepts of Database Management by Philip J. Pratt,2008
4. Modern Database Management by Jeffery A Hoffer,2010

### URL

1. https:/[/www.w3schools.com](http://www.w3schools.com/)
2. https:/[/www.youtube.com](http://www.youtube.com/)
3. https:/[/www.google.co.in](http://www.google.co.in/)
4. https:/[/www.wikipedia.org](http://www.wikipedia.org/)

### BIBLIOGRAPHY