Fourth Semester Project Report on

"HOTEL MANAGEMENT SYSTEM"

Submitted in Partial Fulfilment of the Requirements
for the Award of the Degree of

BACHELOR OF VOCATION IN SOFTWARE DEVELOPMENT (B.Voc.)

(University of Calicut)

Submitted by

NITHIN A.
FKAVBVW039

under the guidance of
Mrs. MUBEENA V.

2022-2023

Department of Software Development

Farook College (Autonomous)

Farook College P.O., Kozhikode, Kerala - 673 632

March 2023

DEPARTMENT OF SOFTWARE DEVELOPMENT
FAROOK COLLEGE (AUTONOMOUS)

Certificate

This is to certify that the project work entitled ***Hotel Management System*** is a bonafide record of the work done by **Nithin A.** Reg No. FKAVBVW039 in partial fulfilment of the awards of degree of Bachelor of Vocation in Software Development, University of Calicut, under my guidance during the year 2022-2023.

Mrs. MUBEENA V. **Mrs. MUBEENA V.**

Internal Guide **Head of Department**

Certified that the candidate were examined by us in the Project Viva Voice, examination held on _____ and his Register Number is:_____

Date:

External Examiners:

1)

2)

DECLARATION

I, Nithin A. hereby declare that this Project Report entitled “Hotel Management System” submitted to Farook College (Autonomous) in partial fulfilment of the requirements for the award of degree of Bachelor of Vocation in Software Development under the faculty of B.Voc is a bonafide record of work done by me under the guidance of Mrs. Mubeena V. Assistant Professor, B.Voc. Software Development, Farook College (Autonomous), Kozhikode – 673 632.

I also declare that this report has not been submitted anywhere else for the award of any degree, diploma, associateship, fellowship or other academic recognition.

Farook College

28/03/2023 Nithin A.

ACKNOWLEDGEMENT

I take this opportunity to express my sincere thanks and deep gratitude to all those people who extended their wholehearted co-operation and helped us in completing this project successfully.

First and foremost I praise and thank God,the foundation of all wisdom from the depth of my heart for being the unfailing source of strength. I sincerely express my special thanks to Farook College

I am pleased to thank the principal of Farook College **Dr. K. M. Naseer**, Our HOD **Mrs. Mubeena V.**, deep sense of gratitude to **Mrs. Rini Fernandez** , **Mr. Umarsabik U.K.** and **Mr. Rahul Das H.** for showing their keen and personal interest in our work, and other faculty member for giving us wonderful opportunity to work on the project. This project helps to know more about existing features of Python and MySQL. Project has provided a platform to implement things we have learnt and also learn more things in detail, which shall help me in future.

I would also like to thank all my colleagues, friends and classmates for helping me directly or indirectly throughout the project.

Nithin A.

ABSTRACT

This paper presents a comprehensive hotel management system designed to improve the efficiency and productivity of hotel operations, while enhancing the guest experience. The system includes a range of automated front-desk tasks, such as check-in and check-out, room assignments, and billing. It also provides staff with real-time information about guest preferences, history, and contact information to help personalize the guest experience. The system can manage room inventory, schedule and manage staff, provide inventory control, reporting, marketing and customer relation management. Additionally, it offers a mobile integration and robust security features to protect sensitive customer information. The system was implemented and tested in a hotel setting, and results demonstrate significant improvements in staff productivity and guest satisfaction

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

A hotel management system (HMS) is a software application designed to streamline and automate the operations of a hotel or hospitality business. The system helps hotel managers to manage their day-to-day operations, such as room reservations, guest checkin and check-out, inventory management, and accounting.

HMSs are used in various types of hotels, from small bed and breakfasts to large resorts. The system can be integrated with other hotel technology solutions, such as point-of-sale systems, online booking engines, and customer relationship management software.

1.1 Hotel Management System

This software is very light weight and easy to use. We provide efficient and secured system with lowest price to the customer. The modules used are easy to understand and maintenance cost comparatively less. The aim is to automate its existing manual system by the help of computerized equipments and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same.

1.2 Features of Existing Systems

1. Faster Payement Gateway System
2. Can maintain data of customers for longer time period
3. Can maintain good customer relation-ship management

1.3 Limitations of Existing Systems

1. Must have proper training to use the software product
2. Lack of Flexibility: Some hotel management systems may lack flexibility and customization options, which can be a problem for hotels with unique needs and requirements.
3. High Cost: Many hotel management systems can be expensive, especially for smaller hotels with limited budgets. This can be a barrier to entry for smaller businesses looking to improve their operations

1.4 Area and Category of the Project Work

The project is developed to manage various aspects of a hotels. This software are used in the reception area of the hotels. The area of the project is hotel management, and the category is software development using the Python programming language. The project involves developing a software system to manage various aspects of a hotel, including room reservations, check-ins, check-outs, room inventory, and billing. The system will provide a user-friendly interface for the hotel staff to perform these tasks efficiently, and it will also generate detailed reports and analytics to help the hotel management make informed decisions. The use of Python will allow the development of a robust, scalable, and maintainable software system.

CHAPTER 2

PROBLEM DEFINITION AND METHODOLOGY

2.1 Problem Definition

Lack of proper interface : Many hotel management systems have a poor user interface that can be difficult to navigate and use. This can lead to frustration and errors, as well as a decrease in productivity

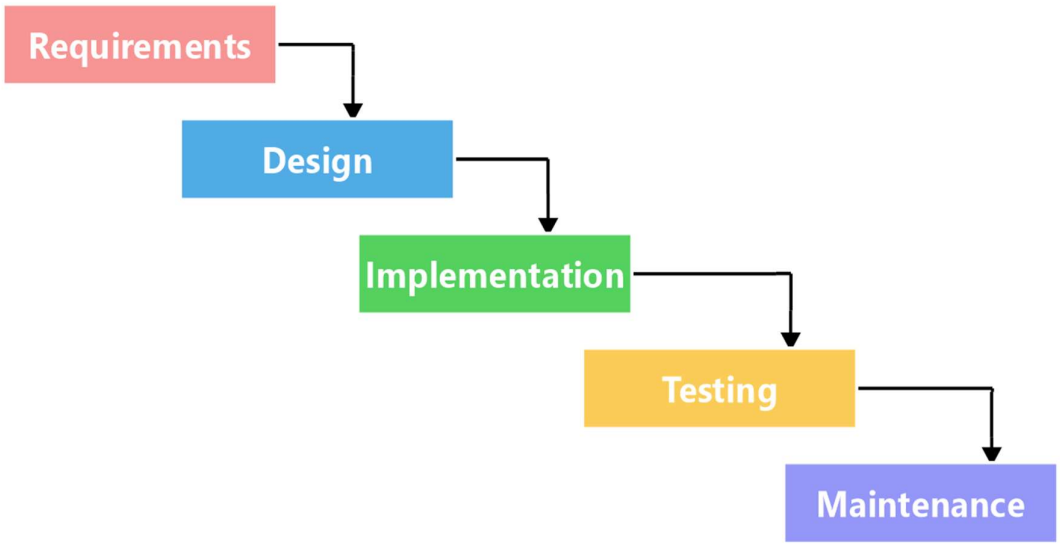
2.2 Objectives

The main objective behind creating a hotel management system with a better user interface is to improve the overall experience of both the hotel staff and the guests. A user-friendly interface can streamline the operations of a hotel, allowing staff members to manage bookings, reservations, and customer interactions more efficiently. This can result in increased productivity, reduced errors, and improved customer satisfaction.

2.3 Motivation

Primary motivation is to improve the overall guest experience. With a user-friendly interface, guests can easily make reservations, access information about their stay, and interact with hotel staff. This can lead to increased guest satisfaction, positive reviews, and repeat business.

2.4 Methodology



1. Requirements Gathering: Gather the requirements of the hotel management system by consulting with the hotel management team, staff, and guests. This will help to identify the key features and functionalities needed in the system.

2. Analysis and Design: Analyze the requirements and design the hotel management system architecture. This involves defining the system components, database schema, and user interface.

3. Development: Develop the hotel management system software using appropriate programming languages and technologies. Ensure that the system is scalable and can handle a large number of users.

4. Testing: Conduct rigorous testing to ensure that the hotel management system meets the requirements and is free of bugs and errors. This includes functional testing, performance testing, and security testing.

5. Deployment: Deploy the hotel management system to the hotel's servers or cloud infrastructure. Train the hotel staff on how to use the system.

6. Maintenance and Support: Provide ongoing maintenance and support for the hotel management system. This includes bug fixes, upgrades, and customer support.

Water fall model is used because the needs of the stakeholders are already well known.In this model we can follow a sequential approach through out the phase. Overall, the key to a successful hotel management system is to understand the specific requirements of the hotel and its stakeholders, and to design and develop a system that meets those requirements efficiently and effectively.

2.5 Scope

By providing software with better user-interface can reduce training cost to the institution

CHAPTER 3

ANALYSIS

3.1 Requirement Analysis

The requirement analysis is the process of identifying system requirements through the observation of current system, discussin with the potential users etc. A proper analysis reduced the ambiguity and effort in developing the system.

3.2 Existing System

The current hotel management system have some kind of complicated user interface and the software cost may be high. So the every owners can't afford the software.

3.3 Proposed System

This software is a light weight desktop based application made using python and its framework tkinter with intractive and easily understandable by the users. We provide roboust security to the customer details that we collect and can be stored for a long term. As it is a light weight software we can run the software in the devices with basic specification

3.4 Requirement Specification

3.4.1 Functional Requirements

1. User Interface: The system should have a user-friendly interface that is easy to navigate, with clear and intuitive options for all users.

2. Room Management: The system should allow for the management of room inventory, including room types, rates, availability, and occupancy.

3. Reservation Management: The system should enable the management of reservations, including booking, cancellation, modification, and payment.

4. Guest Management: The system should allow for the management of guest information, including guest profiles, preferences, and history.

5. Billing and Payment: The system should enable the management of billing and payment, including invoicing, payment processing, and reporting.

3.4.2 Non-functional Requirements

1. Integration: The system should be able to integrate with other systems, such as accounting, inventory, and sales, to provide a complete and accurate view of hotel operations.

2. Data Security: The system should include measures to ensure data security, such as encryption, access control, and backup and recovery procedures.

3. Scalability: The system should be scalable to accommodate the changing needs of the hotel, with the ability to add new features and functionality as required.

3.4.3 Environment Details

Tool	: PyCharm
Language	: Python,MySQL
Operating System	: Windows/ Linux

3.4.4 Hardware Requirements

1. Processor : Pentium IV
2. Memory : 1GB
3. Hard disk : 40GB

3.4.5 Software Requirements

1. Back End : MySQL
2. Front End : : Python
3. Platform : Windows 8 or higher / macOS / linux

3.5 Feasibility Study

3.5.1 Technical Feasibility

The first step is to assess whether the project is technically feasible. This involves analyzing the hardware, software, and infrastructure requirements for the system, as well as evaluating the development resources and expertise required to implement the system.

3.5.2 Economical Feasibility

The next step is to evaluate the economic feasibility of the project. This involves analyzing the cost of development, implementation, and maintenance of the system, as well as assessing the potential benefits and return on investment (ROI) of the system.

3.5.3 Operational Feasibility

The third step is to assess the operational feasibility of the system. This involves analyzing how the system will be integrated into the hotel's existing operations, the impact on staff and guests, and the potential risks and challenges that may arise during implementation.

CHAPTER 4

DESIGN

4.1 Architecture Diagrams/DFD

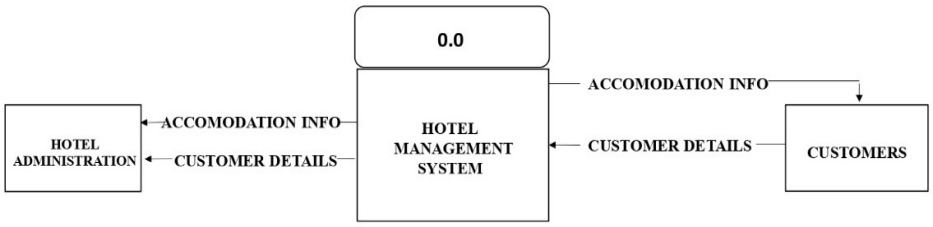


Fig.1 : Data Flow Daigram

This DFD shows the flow of data through systems in which the details collected from the users are used to assign rooms to their preferences and datas are stored for a long term for better customization in the rooms.

4.2 User Interface Layout

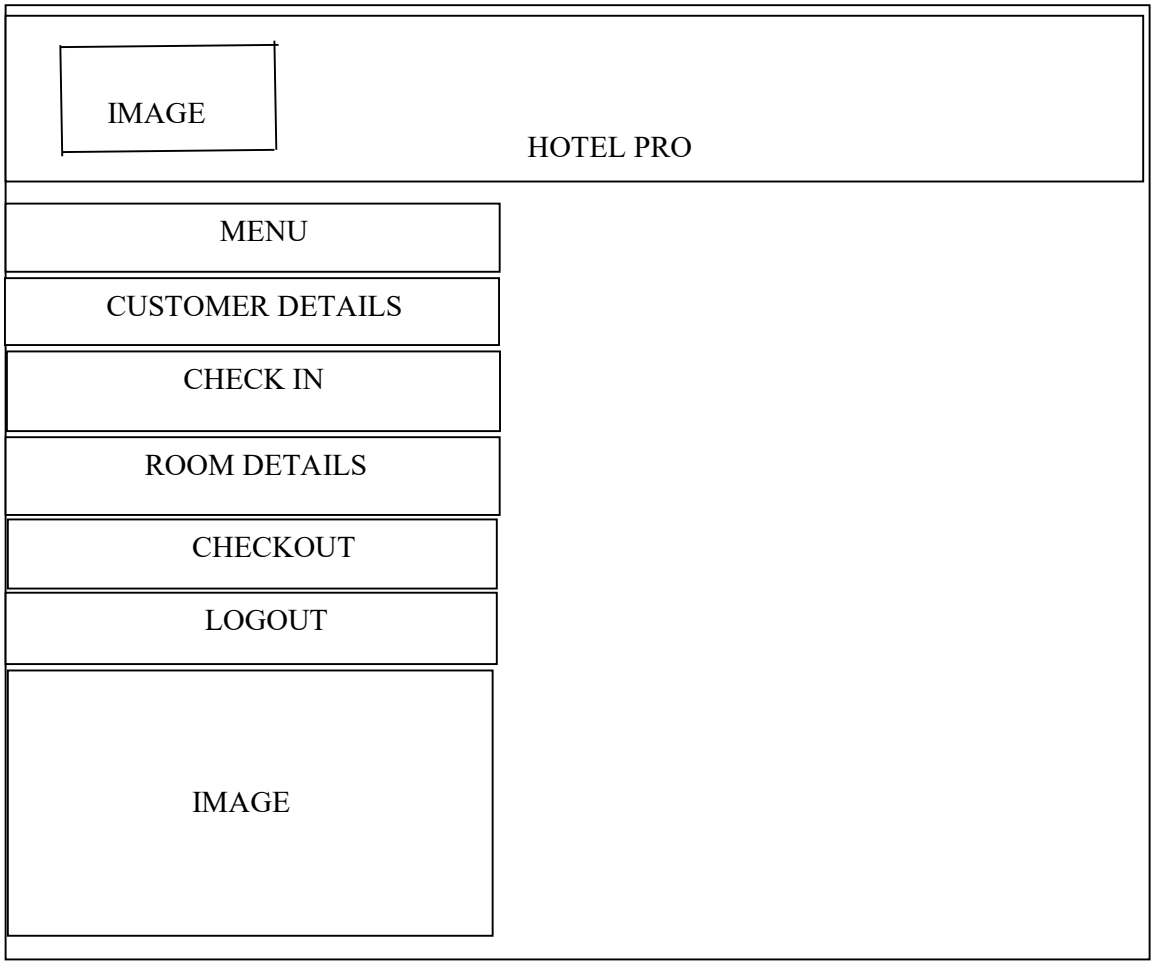


Fig.2 : User Interface Layout

4.3 Snapshots

1. Home Page :

From this page we can access to Customer details, Room details , Check in and Check out. By clicking on logout the user get logout from the interface and get returned to the login page .

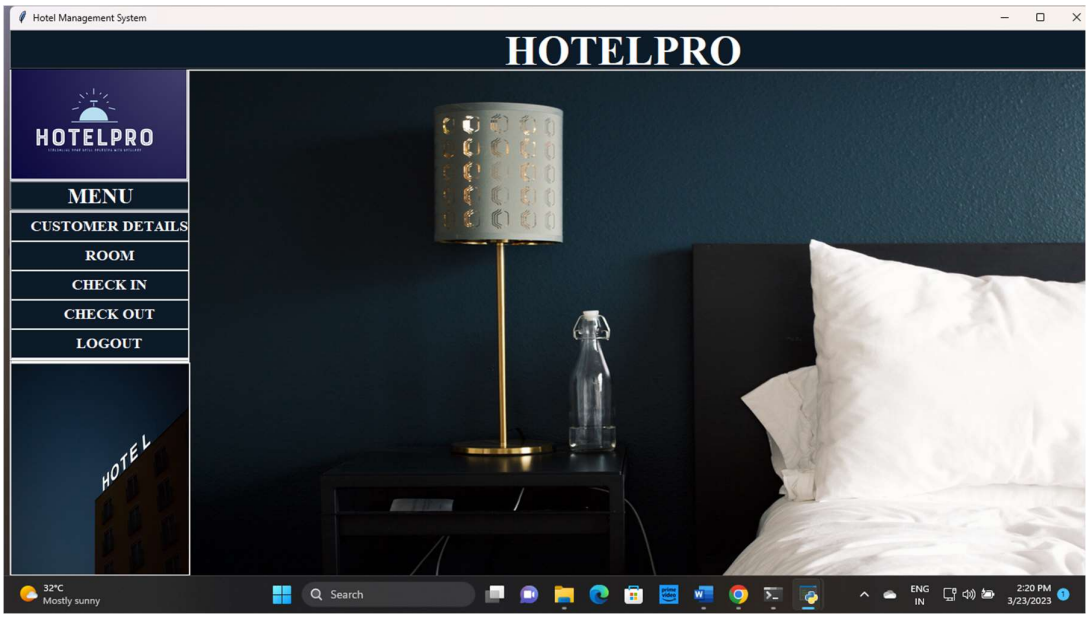


Fig 3: Home Page

2. Customer Details :

In the page Customer Details collected from the customers are entered and displayed in the right.The “Add” button is used to add data in the table, “Update” button is used alter the content in the table, “Delete” button is used to delete content in the table and “Reset” button removes the contents entered in the form. The Search button is used to search customer. Search operation can be performed on the basis of mobile number and reference number.

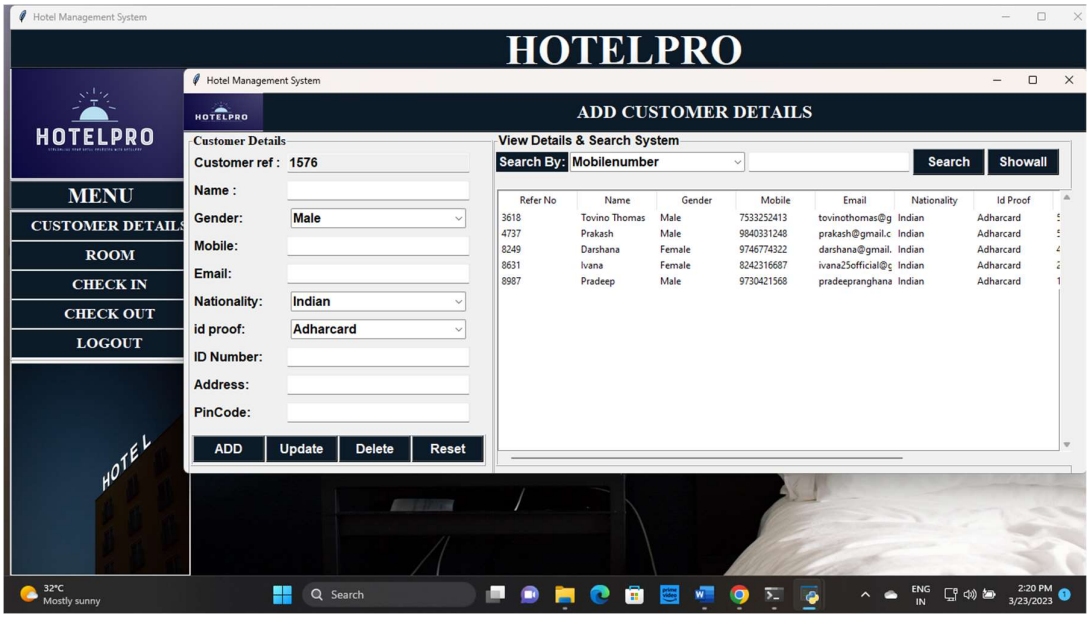


Fig.4: Customer Details

3. Check In :

In this page we enter the checkin informations such as room number floor number, number of days type of room, price etc. and the entered data are displayed in the right. The Fetch button displays the information the details such as Name,email, address etc in the small box which is situated in the top right side of the window. Search button are used to search customer it can be searched based on contact and room number. Bill button is used to calculate amount based on number of days and type of the room. Add button is used to add data in the table, Update button is used alter the content in the table, Delete button is used to delete content in the table and Reset button removes the contents entered in the form.

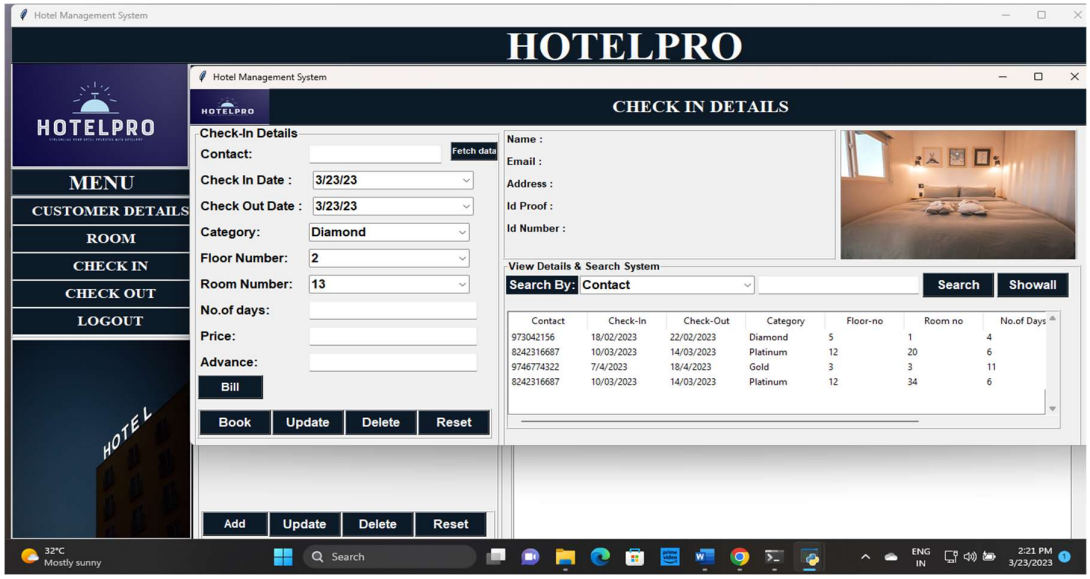


Fig.5: Check in

Structure of Tables

Login			
Field Name	**Data Type**	**Width**	**Constraints**
Staff id	Int	9	Not Null
Name	VarChar	45	Not Null
Email	VarChar	45	Primary key
Security	VarChar	45	Not Null
Password	VarChar	45	Not Null

Table.1 : Login Details

Customer			
Field Name	**Data Type**	**Width**	**Constraints**
Ref	Int	9	Primary key
Name	VarChar	45	Not Null
Gender	VarChar	45	Not Null
Mobile	VarChar	12	NotNull,Unique
Email	VarChar	45	Not Null
Nationality	VarChar	45	Not Null
Id Proof	VarChar	45	Not Null
Id Number	VarChar	45	Not Null
Address	VarChar	45	Not Null
PinCode	Int	9	Not Null

Table.2 : Customer Details

Check In			
Field Name	**Data Type**	**Width**	**Constraints**
Contact	VarChar	12	Not Null
Check-in	VarChar	45	Not Null
Check-out	VarChar	45	Not Null
Floor number	VarChar	45	NotNull
Room Number	Int	9	Not Null,PK
No.of days	VarChar	45	Not Null
Advance	VarChar	45	Not Null
Price	VarChar	45	Not Null

Table.3 : Check-in Details

Room Details			
Field Name	**Data Type**	**Width**	**Constraints**
Floor	VarChar	45	Not Null
Room no.	Int	9	Not Null , PK
Room Type	VarChar	45	Not Null

Table 4 : Room Details

CHAPTER 5

CODING

Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics developed by Guido van Rossum. It was originally released in 1991. Designed to be easy as well as fun, the name "Python" is a nod to the British comedy group Monty Python.Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems

PyCharm

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development.

Vs Code

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development.

Some important codes are given below

Customer Details.py

```
 def add_data(self):
       if self.var_mobile.get()==""or self.var_email.get()=="":
              messagebox.showerror("Error","All fields are required",parent=self.root)
       else:
             try:

conn=mysql.connector.connect(host="localhost",username="root",password="nithi
n1234#",database="project")
                my_cursor=conn.cursor()
                my_cursor.execute("insert into customer
values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",(
                                                        self.var_ref.get(),
                                                        self.var_cust_name.get(),
                                                        self.var_gender.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_pincode.get()

                                                        ))
                conn.commit()
                self.fetch_data()
                conn.close()
                messagebox.showinfo("Success"," Customer details have been
added",parent=self.root)
            except Exception as es:
                messagebox.showwarning("Warning",f"Something went wrong please
try again:{str(es)}",parent=self.root)
```

Check-in.py

```
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="nithin
1234#",database="project")
                    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","No data
found",parent=self.root)

                    else:
                            conn.commit()
                            conn.close()

            showDataframe=Frame(self.root,bd=2,relief=RIDGE,padx=2)
            showDataframe.place(x=395,y=55,width=420,height=175)

            lblName=Label(showDataframe,text="Name :",font=("ariel",10,"bold"))
            lblName.place(x=0,y=0)

            lbl=Label(showDataframe,text=row,font=("ariel",10,"bold"))
            lbl.place(x=60,y=0)
```

```
        #email

conn=mysql.connector.connect(host="localhost",username="root",password="nithin
1234#",database="project")
        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()



        lblGender=Label(showDataframe,text="Email :",font=("ariel",10,"bold"))
        lblGender.place(x=0,y=30)

        lbl2=Label(showDataframe,text=row,font=("ariel",10,"bold"))
        lbl2.place(x=60,y=30)

        #address
conn=mysql.connector.connect(host="localhost",username="root",password="nithin1234#",
database="project")
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="Address :",font=("ariel",10,"bold"))
lbladdress.place(x=0,y=60)

lbl3=Label(showDataframe,text=row,font=("ariel",10,"bold"))
lbl3.place(x=60,y=60)
# id proof

conn=mysql.connector.connect(host="localhost",username="root",password="nithin1234#",
database="project")
my_cursor=conn.cursor()
query=('select idproof from customer where mobile=%s')
value=(self.var_contact.get(),)
my_cursor.execute(query,value)
row=my_cursor.fetchone()



lblIdproof=Label(showDataframe,text="Id Proof :",font=("ariel",10,"bold"))
lblIdproof.place(x=0,y=90)

lbl4=Label(showDataframe,text=row,font=("ariel",10,"bold"))
lbl4.place(x=60,y=90)
```

```
#id Number

conn=mysql.connector.connect(host="localhost",username="root",password="nithin1234#",
database="project")
my_cursor=conn.cursor()
query=('select idnumber from customer where mobile=%s')
value=(self.var_contact.get(),)
my_cursor.execute(query,value)
row=my_cursor.fetchone()
conn.commit()
conn.close()


lblidnumber=Label(showDataframe,text="Id Number :",font=("ariel",10,"bold"))
lblidnumber.place(x=0,y=120)

            lbl5=Label(showDataframe,text=row,font=("ariel",10,"bold"))
            lbl5.place(x=80,y=120)

    def add_data(self):
        if self.var_contact.get()==""or self.var_noofdays.get()=="":
                messagebox.showerror("Error","All fields are required",parent=self.root)
        else:
            try:

conn=mysql.connector.connect(host="localhost",username="root",password="nithin1234#",
database="project")
                my_cursor=conn.cursor()
                my_cursor.execute("insert into checkin
values(%s,%s,%s,%s,%s,%s,%s,%s,%s)",(
                                                        self.var_contact.get(),
                                                        self.var_checkin.get(),
                                                        self.var_checkout.get(),
                                                        self.var_category.get(),
                                                        self.var_floorno.get(),
                                                        self.var_roomavailable.get(),
                                                        self.var_noofdays.get(),
                                                        self.var_price.get(),
                                                        self.var_advance.get()



                                                                ))
                conn.commit()
                self.fetch_data()
                conn.close()
                messagebox.showinfo("Success"," Customer details have been
added",parent=self.root)
            except Exception as es:
```

```
            messagebox.showwarning("Warning",f"Something went wrong please try
again:{str(es)}",parent=self.root)
```

Check Out.py

```
    def welcome(self):
        self.textarea.delete(1.0,END)
        self.textarea.insert(END,"\t\t\Hotel Pro")
        self.textarea.insert(END,f"\n Bill Number:{self.bill_no.get()}")
        self.textarea.insert(END,f"\n Customer Name:{self.c_name.get()}")
        self.textarea.insert(END,f"\n Phone Number:{self.var_contact.get()}")


        self.textarea.insert(END,"\n-----------------------------------------------------------------------")
        self.textarea.insert(END,f"\n Dishes\t\t\tQuantity\t\tPrice")
        self.textarea.insert(END,"\n-----------------------------------------------------------------------")
    def AddItem(self):
        Tax=1
        self.n=self.prices.get()

        self.s=self.var_payable.get()
        self.m=self.qty.get()*self.n
        self.l.append(self.m)


        self.textarea.insert(END,f"\n  {self.product.get()}\t\t{self.qty.get()}\t\t{self.m}")



        self.sub_total.set(str('Rs.%.2f'%(sum(self.l))))
        self.tax_input.set(str('Rs.%.2f'%((((sum(self.l)) - (self.prices.get()))*Tax)/100)))
        self.total.set(str("Rs.%.2f"%(((sum(self.l))  + ((((sum(self.l)) -
(self.prices.get()))*Tax)/100)))))


    def adroom(self):

        q1=float(self.var_price.get())
        q2=float(self.var_advance.get())
        q3=float(q1-q2)
        AP="Rs."+str("%.2f"%(q3))
        self.var_payable.set(AP)
        self.textarea.insert(END,"\n-----------------------------------------------------------------------
----------------------")
        self.textarea.insert(END,f"\n Price of Room :  {self.var_payable.get()}")
        self.textarea.insert(END,"\n-----------------------------------------------------------------------
----------------------")
```

```
    def gen_bill(self):
        if self.product.get()=="":
            messagebox.showerror("Error","Please add room bill")
        else:
            text=self.textarea.get(10.0,(10.0+float(len(self.l))))
            self.welcome()
            self.textarea.insert(END,text)
            self.textarea.insert(END,"\n-----------------------------------------------------------------------------
------------------------")
            self.textarea.insert(END,f"\n Sub Amount:\t\t\t{self.sub_total.get()}")
            self.textarea.insert(END,f"\n Tax Amount:\t\t\t{self.tax_input.get()}")
            self.textarea.insert(END,f"\n Total Amount:\t\t\t{self.total.get()}")
            self.textarea.insert(END,"\n\-----------------------------------------------------------------------------
-------------------------")

    def savebill(self):
        op=messagebox.askyesno("Saave bill","Do you want to save the bill")
        if op>0:
            self.bill_data=self.textarea.get(1.0,END)
            f1=open('savebills/'+str(self.bill_no.get())+'.txt','w')

            f1.write(self.bill_data)
            op=messagebox.showinfo("Saved",f"Bill No:{self.bill_no.get()} saved successfully
",)
            f1.close()

    def iprint(self):
        q=self.textarea.get(1.0,"end-1c")
        filename=tempfile.mktemp('.txt')
        open(filename,'w').write(q)
        os.startfile(filename,"print")


    def find_bill(self):
        found="yes"
        for i in os.listdir('savebills/'):
            if i.split('.')[0]==self.search_bill.get():
                    f1=open(f'savebills/{i}','r')
                    self.textarea.delete(1.0,END)
                    for d in f1:
                        self.textarea.insert(END,d)
                    f1.close()
                    found="yes"
            if found=='no':
                messagebox.showerror('Error',"Invalid bill number")


    def clear(self):
        self.textarea.delete(1.0,END)
        self.c_name.set("")
```

```
        self.c_email.set("")
        self.c_phone.set("")
        self.bill_no.set("")
        z=random.randint(1000,9999)
        self.bill_no.set(z)
        self.search_bill.set('')
        self.product.set('')
        self.prices.set('')
        self.qty.set(0)
        self.l=[0]
        self.sub_total.set('')
        self.tax_input.set('')
        self.total.set('')
        self.var_price.set('')
        self.var_advance.set('')
        self.var_payable.set('')

    def vacate(self):
        vactate=messagebox.askyesno("Hotel Management System","Do you want to delete this
customer",parent=self.root)
        if vactate>0:

conn=mysql.connector.connect(host="localhost",username="root",password="nithin1234#",
database="project")
            my_cursor=conn.cursor()
            query=("delete from checkin where Contact=%s")
            value=(self.var_contact.get(),)
            my_cursor.execute(query,value)

        else:
            if not vactate:
              return
        conn.commit()
        conn.close()

    def exit(self):
        self.root.destroy()
        os.system("main.py")
```

CHAPTER 6

SYSTEM TESTING

6.1 Introduction

System testing is carried out in a planned manner according to the system test plan document. The system test plan identifies all testing-related activities that must be performed, specifies the schedule of testing, and allocates resources. It also lists all the test cases and the expected outputs for each test case. It ensures that the system works accurately and efficiently, before live operation commences. It helps in the noting and correction of errors.

6.2 Testing Methods

6.2.1 Unit Testing

Unit testing focusses on verification efforts made on the smallest unit of software design, a module. The modules are tested separately and this testing is carried out during the programming stage itself. The project can be divided into modules: Customer Details, Check in, Check out and Room details. Each of these modules contain multiple forms, which have been individually tested. The module interface is tested to ensure that information properly flows in and out of the program unit under test. It is ensured that each module works satisfactorily with regards to the output expected from it.

6.2.2 IntegrationTesting

Integration testing is used to take the unit tested modules and build a program structure. All the modules are combined and then tested as a whole. Data can be lost across interfaces and one module may have an adverse effect on others. Moreover, sub functions when combined may not produce the desired major functions. All these make integration testing necessary. Here, error correction is difficult because their isolation from the entire program becomes complicated.

6.2.3 Alpha Testing

Alpha testing takes place at the developer's side by the internal teams, before release to the external customers. Developers observe the working of the application and note problems. Generally, this testing is performed without the involvement of the development

teams. It is done when development is about to complete. Minor design changes can still be made as a result of alpha testing. The focus of this testing is to simulate real users by using blackbox and whitebox techniques and the aim is to carry out the tasks that a typical user might perform, diagnose any errors that arise and rectify them.

6.2.4 Beta Testing

Beta testing of a product is performed by the real users of the software application in a real environment and can be considered as a form of external user acceptance testing. Beta version of the software is released to a limited number of end-users of the product to obtain feedback on the product quality. It reduces the risk of product failure and provides increased quality of the end product through customer validation. It is the final test before shipping a product to the customers. Direct feedback from customers is a major advantage of this testing in addition to helping to test the product in a real-time environment. It creates a goodwill with the customers and increases customer satisfaction.

Since this project has been developed using an iterative model, it has been monitored by the client, Eagle Environmental service and pest control establishment, periodically to notify the desired changes to be made. This served as a customer validation and there by helped in creating a better solution to the clients.

6.3 Test Cases

A test case in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not. The test cases in our project context are:

I) Login Form

Input	Expected Result	Actual result	Remarks
Invalid username and password	Unsuccessful login	Unsuccessful login	Pass
Valid username, invalid password	Unsuccessful login	Unsuccessful login	Pass
Invalid username, valid password	Unsuccessful login	Unsuccessful login	Pass
Valid username and password	Successful login	Successful login	Pass

II) Data Management (Creation of masters)

a) Mandatory Fields

Input	Expected Result	Actual result	Remarks
Blank entry	Unsuccessful insertion	Unsuccessful insertion	Pass
Text/Number entry	Successful insertion	Successful insertion	Pass

b) Optional Fields

Input	Expected Result	Actual result	Remarks
Blank entry	Successful insertion	Successful insertion	Pass
Text/Number Entry	Successful insertion	Successful insertion	Pass

CHAPTER 7

CONCLUSION

In conclusion, we have successfully developed a hotel management system using Python that meets the requirements of our client. Our system allows the hotel staff to manage various tasks such as room reservations, check-ins, check-outs, room inventory, and billing, all through a user-friendly interface. The use of Python programming language has allowed us to develop a robust, scalable, and maintainable system that can be easily modified and extended in the future. Overall, our project has been a success, and we believe that our hotel management system will greatly benefit our client and their guests..

In conclusion, a well-designed and implemented hotel management system can provide numerous benefits to hotels, including increased efficiency, improved guest experience, and better decision-making capabilities.

FUTURE ENHANCEMENTS

- Integration with payment systems: To improve the payment process for guests, a hotel management system could be integrated with payment gateways like PayPal or Stripe to enable easy and secure online payments.
- Automated room allocation: Using machine learning algorithms, a hotel management system could be developed to automatically allocate rooms based on guests' preferences, availability, and room prices.
- Online check-in and check-out: By integrating with the hotel's booking system and mobile apps, guests can check in and out of their rooms without visiting the reception. This can save time and enhance guests' experience.
- Real-time guest feedback: By integrating with social media platforms, a hotel management system can collect real-time feedback from guests and respond to their queries and complaints promptly. This can help enhance guests' experience and improve the hotel's reputation.

BIBLIOGRAPHY

1. References

- https://stackoverflow.com/
- www.geeksforgeeks.org
- https://www.youtube.com/@codewithkiran
- https://www.youtube.com/@webcode4805