

DATABASE SYSTEM ASSIGNMENT

(CS F212)

Group Members:

Khirthana Sundaram	2021A7PS0064U
Keane Anthony Coutinho	2021A7PS0080U
Jannath Shaik	2021A7PS0132U
Mithil Dudam	2021A7PS0142U
Neha John	2021A7PS0163U

Acknowledgment

We would like to express our sincere gratitude to all the professors of the Computer Science Department for allowing us to apply our expertise in this assignment and for giving us the required knowledge to program using MySQL.

Our humble and noteworthy appreciation is due to Dr Tamizharasan Periyasamy for being there to answer our queries. We are thankful for his guidance and assistance he gave for the completion of this assignment.

Table Of Contents

1.Introduction

- 1.1 Authorization
- 1.2 Historical Background
- 1.3 Objective
- 1.4 Methodology
- 1.5 Report Preview

2. Discussion and Analysis

- 2.1 Problem Statement
- 2.2 MySQL Queries
 - 2.2.1 Triggers
 - 2.2.2 Functions
- 2.3 ER Diagram
- 2.4 Relational Model
 - 2.4.1 Staff View
 - 2.4.2 Login View
 - 2.4.3 Guest View
 - 2.4.4 Room View
 - 2.4.5 Booking View
 - 2.4.6 Members View
 - 2.4.7 Payment View
- 2.5 Normalization
- 2.6 Output

3. Conclusion

Introduction

Authorization

This project has been authorized by Dr. Tamizharasan Periyasamy, Professor in the Department of Computer Science, at BITS Pilani Dubai Campus, on 2 March 2023

Historical Background

Database Management Systems (DBMS) have been around since the early days of computing. The first DBMS, called the Integrated Data Store (IDS), was developed in the 1960s by General Electric. Other early DBMSs included the Information Management System (IMS) developed by IBM, and the Data Manipulation System (DMS) developed by the Massachusetts Institute of Technology (MIT).

In the 1970s, the concept of a relational database was introduced, and the first commercially available relational database management system (RDBMS) was developed by IBM. The Structured Query Language (SQL) was also developed in the 1970s and became the standard language for interacting with RDBMSs.

In the 1980s and 1990s, the use of RDBMSs became more widespread and popular as businesses began to recognize the value of data management. This led to the development of various RDBMSs such as Oracle, Microsoft SQL Server, and MySQL, which are still widely used today. In recent years, the rise of big data and the advent of cloud computing has led to the development of new types of DBMSs such as NoSQL databases and cloud-based database management systems.

Overall, the evolution of DBMSs has been driven by changing business needs, technological advancements, and the growing importance of data management in today's digital age.

Objective

- To design and implement a database management system for a hotel that will provide efficient management of guest information, room reservations and billing.
- The assignment aims to apply the concepts of database design, normalization, SQL and database management to create a robust and user - friendly hotel management system that meets the needs of both guests and hotel staff.

Methodology

- Class notes and PPT's
- Images and videos from the net
- Websites

Report Preview

Including the introduction this report is divided into three sections. Chapter-2, Discussion and Analysis, discuss the logic the program is based on , as well as displays code and output. Chapter 3, Conclusion, concludes the report and provides further suggestions for improvement.

Discussion and Analysis

Problem Statement

To create an efficient Hotel Database Management System that is normalized to 3rd Normal Form to ensure preserved dependency along with an ER diagram and Relational model.

MySQL Queries

Triggers

1. setprice: A Trigger to set the total price of the room in the payments table from the room price and the duration.

```
delimiter !
create trigger setprice
after INSERT ON BOOKING
for each row
begin
    DECLARE RP int;
    DECLARE DUR INT;
    DECLARE pid varchar(5);
    set RP=(select room.price from room NATURAL JOIN BOOKING where ROOM.R_NO=NEW.R_NO);
    SET DUR=(SELECT DURATION FROM BOOKING WHERE B_ID=NEW.B_ID);
    set pid=(SELECT generate_id());
    insert into payment(P_ID,B_ID,R_PRICE,TIP,EXTRA_CHARGES)
    VALUES (pid,NEW.B_ID,(DUR*RP),0,0);
end !
delimiter ;
```

2. UPDATE_STATUS: Changes the room status to occupied when a room is booked.

```
DELIMITER !
CREATE TRIGGER UPDATE_STATUS
AFTER INSERT ON BOOKING
FOR EACH ROW
BEGIN
    update ROOM SET STATUS="OCCUPIED" WHERE R_NO=NEW.R_NO;
END!
DELIMITER ;
```

3. UPDATE_STATUS_2: Changes the room status to vacant when a value is inserted into bookings and the checkout date has been past.

```
delimiter !
create trigger UPDATE_STATUS_2
AFTER INSERT ON BOOKING
FOR EACH ROW
BEGIN
    declare d date;
    SET d=(select CURRENT_DATE());
    update ROOM SET STATUS="VACANT" WHERE R_NO IN (SELECT R_NO FROM BOOKING WHERE CHECK_OUT< d);
END!
DELIMITER ;
```

Functions

1. generate_id(): Function which randomly generates a payment id for set price trigger to input into the payment table.

```
delimiter !
CREATE FUNCTION generate_id() RETURNS VARCHAR(4)
BEGIN
    DECLARE chars VARCHAR(10) DEFAULT '0123456789';
    DECLARE id VARCHAR(4);
    SET id = CONCAT('P', FLOOR(RAND() * 1000));
    RETURN id;
END!
delimiter ;
```

2. calc_tot(PID): Function returns the total amount the customer has to pay at the end of the trip including room prices and the extra charges.

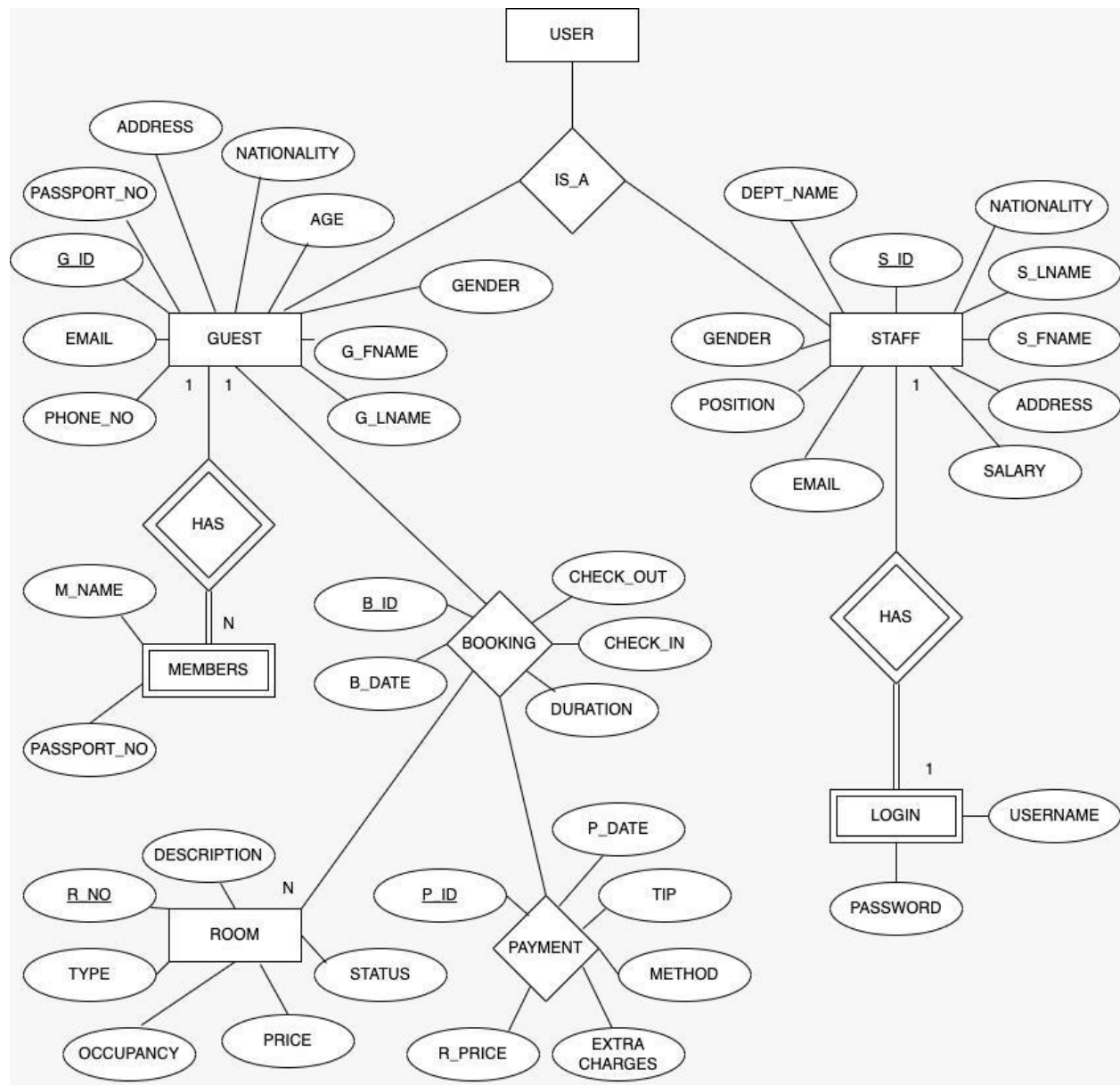
```
delimiter !
• create function calc_tot(PID VARCHAR(5)) returns int
begin
    declare total int;
    set total=((select R_PRICE FROM PAYMENT WHERE P_ID=PID)+(select TIP FROM PAYMENT WHERE P_ID=PID)+(select EXTRA_CHARGES FROM PAYMENT WHERE P_ID=PID));
    RETURN total;
END !
DELIMITER ;
```

Normalization:

TABLE	REASONING	NORMALISATION FORM
GUEST	1NF : value of attributes in tuple is atomic (not multivalued or composite) 2NF : G_ID is the only primary key and there are no partial dependencies 3NF : Each attribute seems to depend solely on the "G_ID" (No transitive dependencies)	3NF
MEMBERS	UNF (Unnormalized form): No primary key to uniquely identify values, and relation cannot be established between other tables using foreign key.	UNF
STAFF	1NF : value of attributes in tuple is atomic (not multivalued or composite) 2NF : S_ID is the only primary key and there are no partial dependencies 3NF : Each attribute seems to depend solely on the "S_ID" (No transitive dependencies)	3NF
LOGIN	UNF (Unnormalized form): No primary key to uniquely identify values, and relation cannot be established between other tables using foreign key.	UNF
ROOM	1NF : value of attributes in tuple is atomic (not multivalued or composite) 2NF : R_NO is the only primary key and there are no partial dependencies 3NF : Each attribute seems to depend solely on the "R_NO" (No transitive dependencies)	3NF

BOOKING	<p>1NF: value of attributes in tuple is atomic (not multivalued or composite)</p> <p>2NF: B_ID is the only primary key and there are no partial dependencies</p> <p>3NF: Each attribute seems to depend solely on the “B_ID” (No transitive dependencies)</p>	3NF
PAYMENT	<p>1NF: value of attributes in tuple is atomic (not multivalued or composite)</p> <p>2NF: P_ID is the only primary key and there are no partial dependencies</p> <p>3NF: Each attribute seems to depend solely on the “P_ID” (No transitive dependencies)</p>	3NF

ER Diagram



RELATIONAL DATABASE MODEL

STAFF VIEW:

	Field	Type	Null	Key	Default	Extra
►	S_ID	varchar(5)	NO	PRI	NULL	
	S_FNAME	varchar(20)	YES		NULL	
	S_LNAME	varchar(20)	YES		NULL	
	SALARY	int	YES		NULL	
	GENDER	varchar(6)	YES		NULL	
	NATIONALITY	varchar(25)	YES		NULL	
	ADDRESS	varchar(40)	YES		NULL	
	EMAIL	varchar(100)	YES		NULL	
	DEPT_NAME	varchar(15)	YES		NULL	
	POSITION	varchar(20)	YES		NULL	

LOGIN VIEW:

	Field	Type	Null	Key	Default	Extra
►	S_ID	varchar(5)	YES	MUL	NULL	
	USERNAME	varchar(20)	YES		NULL	
	PASSWORD	varchar(20)	YES		NULL	

GUEST VIEW:

Field	Type	Null	Key	Default	Extra
G_ID	varchar(5)	NO	PRI	NULL	
G_FNAME	varchar(20)	YES		NULL	
G_LNAME	varchar(20)	YES		NULL	
AGE	int	YES		NULL	
GENDER	varchar(6)	YES		NULL	
NATIONALITY	varchar(25)	YES		NULL	
PHONE_NUMBER	varchar(15)	YES		NULL	
EMAIL	varchar(100)	YES		NULL	
ADDRESS	varchar(30)	YES		NULL	
PASSPORT_NUMBER	varchar(10)	YES		NULL	

ROOM VIEW:

	Field	Type	Null	Key	Default	Extra
►	R_NO	varchar(5)	NO	PRI	NULL	
	TYPE	varchar(10)	YES		NULL	
	PRICE	int	YES		NULL	
	OCCUPANCY	int	YES		NULL	
	STATUS	varchar(15)	YES		NULL	
	DESCRIPTION	varchar(100)	YES		NULL	

BOOKING VIEW:

	Field	Type	Null	Key	Default	Extra
►	B_ID	varchar(5)	NO	PRI	NULL	
	G_ID	varchar(5)	YES	MUL	NULL	
	R_NO	varchar(5)	YES	MUL	NULL	
	B_DATE	timestamp	YES		NULL	
	CHECK_IN	date	YES		NULL	
	CHECK_OUT	date	YES		NULL	
	DURATION	int	YES		NULL	VIRTUAL GENERATED

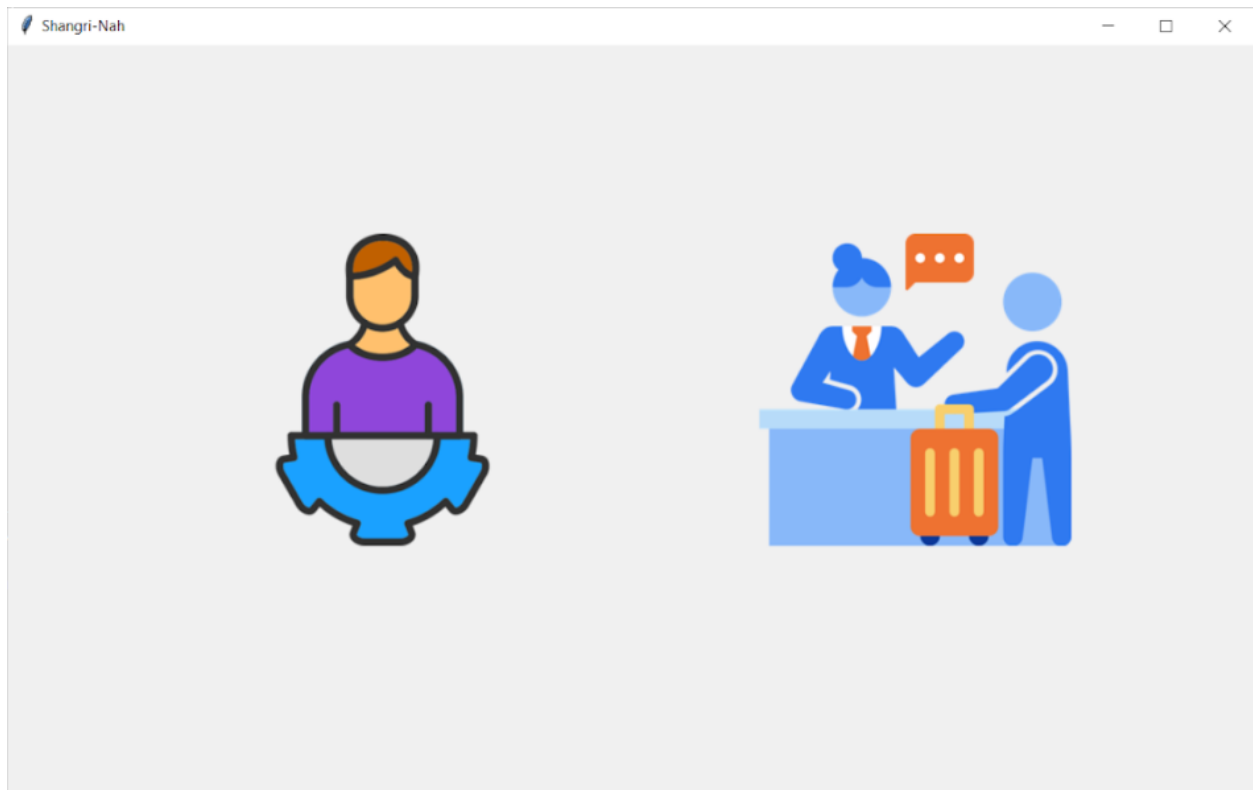
MEMBERS VIEW:

	Field	Type	Null	Key	Default	Extra
►	G_ID	varchar(5)	YES	MUL	NULL	
	M_NAME	varchar(30)	YES		NULL	
	PASSPORT_NUMBER	varchar(10)	YES		NULL	

PAYMENT VIEW:

	Field	Type	Null	Key	Default	Extra
►	P_ID	varchar(5)	NO	PRI	NULL	
	B_ID	varchar(5)	YES	MUL	NULL	
	P_DATE	date	YES		NULL	
	METHOD	varchar(10)	YES		NULL	
	R_PRICE	int	YES		NULL	
	TIP	int	YES		NULL	
	EXTRA_CHARGES	int	YES		NULL	

Output




Customer Page:



New Guest:

Customer



Shangri-Nah Hotel

First Name: Phone:


Last Name: Address:

Age: Passport No:

Gender: Nationality:

Email:

Customer

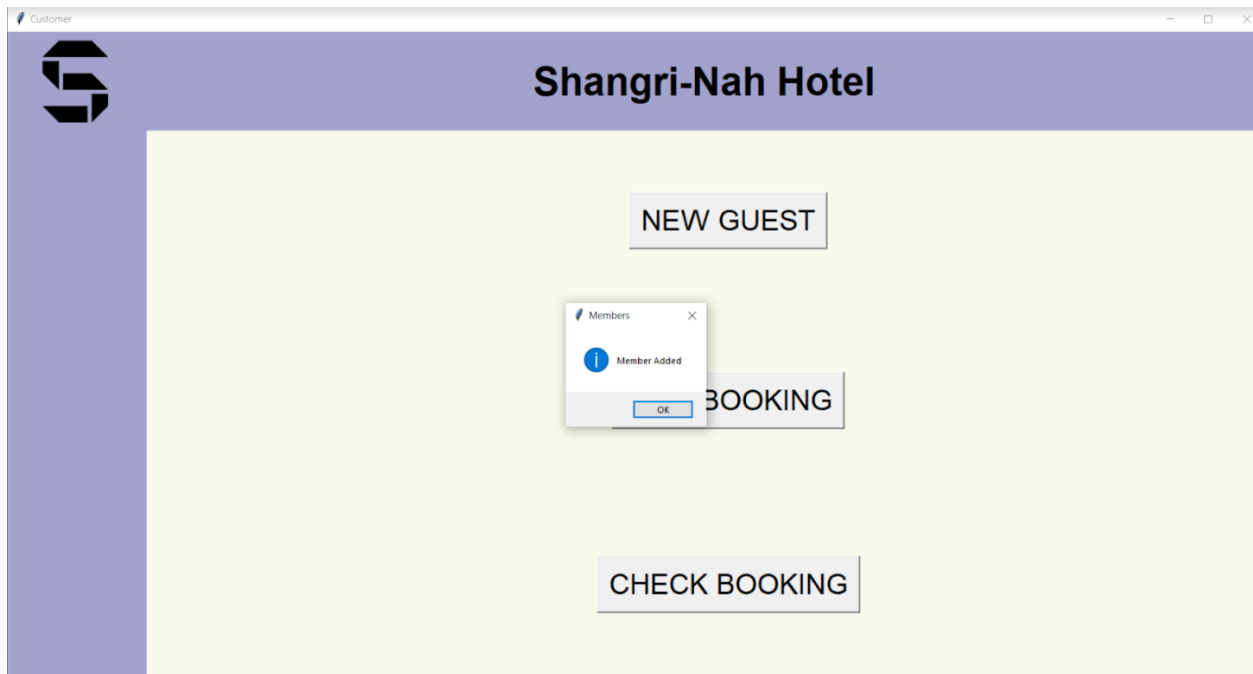


Shangri-Nah Hotel

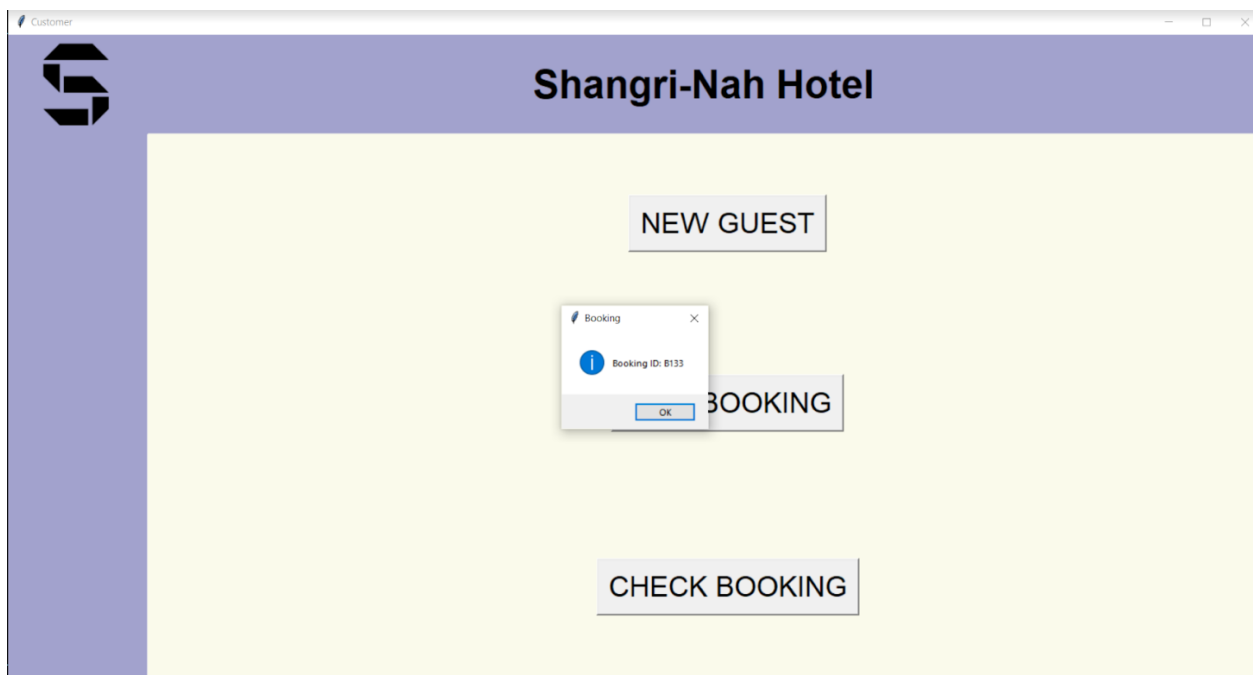
GID:

Name:

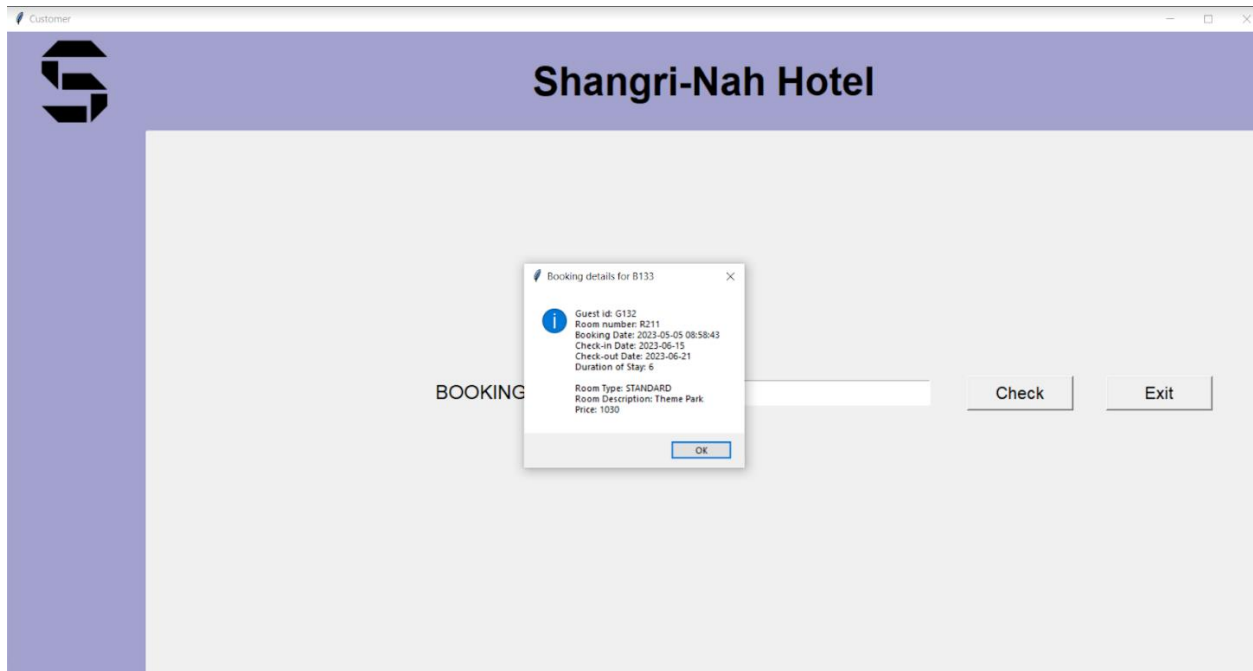
Passport No:



New Booking:



Check Booking:




Staff Page:




Guest Info:

Staff Login

 **Shangri-Nah Hotel**

GUEST ID:

Guest details

 Guest name: Subalakshmi Sundaram
Age: 25
Gender: Female
Nationality: Algeria
Phone number: 7483249832
Email: khitsucks@gmail.com
Address: international city
Passport number: s747574

Update Guest:

Staff Login

 **Shangri-Nah Hotel**

Guest ID: Phone:

First Name: Address:

Last Name: Passport No:

Age: Nationality:

Email:

Staff Login

Shangri-Nah Hotel

Guest ID: Phone:

First Name: Address:

Last Name: Passport No:

Age: Nationality:

Email:

Updates

Updated

OK

Payment:

Staff Login

Shangri-Nah Hotel

Booking ID: Tip:

Method: Extra Charges

Staff Login

Shangri-Nah Hotel

Booking ID:

Tip:

Method:

Extra Charges

Payment details for B133

Tip: 20
Method: Cash
Extra: 50
Total payment: 6250

OK

Booking Info:

Staff Login

Shangri-Nah Hotel

BOOKING NUMBER:

Booking details for B133

Guest id: G132
Room number: R211
Booking Date: 2023-05-05 08:58:43
Check-in Date: 2023-06-15
Check-out Date: 2023-06-21
Duration of Stay: 6

OK

Conclusion

In conclusion, a hotel management database system can be developed using Python programming language and a suitable database system (DBMS) such as MySQL can contain tables such as "GUEST", "STAFF", "ROOM", "BOOKING", "PAYMENT"; normalized to at least the third normal (3NF). Data integrity and accuracy are ensured.

To develop a GUI interface for the system, Python uses libraries such as Tkinter that allow user-friendly graphical interfaces. This library provides various buttons, text boxes, labels, menus and other widgets that can be used to create the interface.

The GUI allows hoteliers to manage room availability, make bookings, process payments, manage employee profiles, create booking information, etc. The system can also provide security features such as user authentication and access control has been added to ensure that only authorized personnel can access sensitive data.

Overall, a hotel management database system with a GUI interface using Python can be an efficient and practical solution for managing hotel operations and increasing customer satisfaction

In conclusion, a hotel management database system can be developed using Python programming language and a suitable database system (DBMS) such as MySQL can contain tables such as "GUEST", "STAFF", "ROOM", "BOOKING", "PAYMENT"; normalized to at least the third normal (3NF). Data integrity and accuracy are ensured.

To develop a GUI interface for the system, Python uses libraries such as Tkinter that allow user-friendly graphical interfaces. This library provides various buttons, text boxes, labels, menus and other widgets that can be used to create the interface.

The GUI allows hoteliers to manage room availability, make bookings, process payments, manage employee profiles, create booking information, etc. The system can also provide security features such as user authentication and access control has been added to ensure that only authorized personnel can access sensitive data.

Overall, a hotel management database system with a GUI interface using Python can be an efficient and practical solution for managing hotel operations and increasing customer satisfaction.