A

Project Report

On

Hungry House

Submitted in partial fulfillment of requirements

For the degree of

Bachelor of Technology in Computer Science & Engineering

Submitted by

Ankit Saxena (1674810005)

Nitya Singh (1674810010)

Shivam Singh (1674810024)

Under the supervision of

Ms. NIDHI SHUKLA

(HOD)

Department of Computer Science & Engineering

Dr. A.P.J. Abdul Kalam Technical University, Lucknow

Year-2020

ACKNOWLEDGEMENT

I wish to express my heartfelt gratitude to the all the people who have played a crucial role in the research for this project, without their active cooperation the preparation of this project could not have been completed within the specified time limit.

I am thankful to our respected Director, Dr. Prashant Kumar Pandey for motivating me to complete this project with complete focus and attention.

I am also thankful to my project guide Lecturer who supported me throughout this project with utmost cooperation and patience and for helping me in doing this Project.

Abstract

The restaurant management software is a capstone project that aims towards developing an all-in-one application that addresses the various problems and challenges faced by high-end restaurant owners today . In order to achieve this goal, this project addresses various aspects of the modern business . It allows its users to access a variety of functionalities that are essential to the culinary business . Given that this project encapsulates more than one interface, it is by nature rich when it comes to the number of functionalities that are offered and these vary depending on the user's role. For example, it allows the waiter to take user orders through a straightforward, user-friendly, and smooth interface . All in all, this projects main aim is to reduce the time overhead in highend management restaurants by providing an alternative to the traditional management system based on physical record keeping and paper work.

CONTENT

CHAPTER 1

INTRODUCTION

The organizations in eateries are currently developing continually. Simultaneously, the requirement for dealing with its activities and undertakings emerges. The most ideal approach to enhance these exercises is developing the business online too. The present age energizes cutting edge benefits particularly over the Internet.

The undertaking "Hungry House" is actualized to lessen the manual work and improves the exactness of work in an eatery.

Hungry House is Desktop Application to eatery the board. Thus the undertaking is grown capably to support café. This framework wake to offer support office to café and proprietors mechanize their BILLING OPERATIONS.

This framework altogether decreases the superfluous time squander inside the inn just as it diminishes pointless computation.

According to the new standard Goods and Service Tax (GST) demand on both AC and non-AC eateries to 5%. Each café charges 5% GST separation as 2.5% State GST (SGST) and 2.5 % Central GST (CGST) with no other assistance charge and some other VAT charges.. We have built up a RESTAURANT MANAGEMENT SYSTEM with inbuilt GST count in the bill, just as it shows the sum in rupees the amount GST is applied.

RESTAURENT MANAGEMENT SYSTEM venture completely created in python language which is as of now requested in showcase utilizing python GUI Tkinter.

CHAPTER-2

System Requirement

Software Requirement : Python

Python is an extensively used generally valuable, raised level programming language. It was from the start arranged by Guido van Rossum in 1991 and made by Python Software Foundation. It was basically made for emphasis on code understandability, and its accentuation licenses programming architects to impart thoughts in less lines of code.
Python is a programming language that lets you work quickly and organize structures even more capably.

Python is a raised level, decoded, insightful and object-arranged scripting language. Python is proposed to be significantly understandable. It uses English watchwords sometimes where as various tongues use complement, and it has less linguistic improvements than various vernaculars.

- **Python is Interpreted** − Python is set up at runtime by the middle person. You don't need to mastermind your program before executing it. This resembles PERL and PHP.
- **Python is Interactive** − You can truly sit at a Python impel and speak with the interpreter direct to form your ventures.
- **Python is Object-Oriented** − Python supports Object-Oriented style or method of programming that typifies code inside articles.
- **Python is a Beginner's Language** − Python is an exceptional language for the fledgling level programming designers and supports the improvement of a wide extent of employments from clear substance taking care of to WWW projects to games.

History of Python

Python was made by Guido van Rossum in the last piece of the eighties and mid nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is gotten from various tongues, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting vernaculars.

Python is copyrighted. Like Perl, Python source code is directly open under the GNU General Public License (GPL).

Python is directly kept up by an inside progression bunch at the association, notwithstanding the way that Guido van Rossum still holds a basic activity in organizing its empowering.

Python Features

Python's features include −

- **Easy-to-learn** − Python has scarcely any watchwords, clear structure, and a clearly described sentence structure. This allows the understudy to get the language quickly.
- **Easy-to-read** − Python code is even more clearly described and recognizable to the eyes.
- **Easy-to-maintain** − Python's source code is truly easy to-keep up.
- **A broad standard library** − Python's primary piece of the library is really adaptable and cross-stage immaculate on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an instinctive mode which licenses keen testing and examining of bits of code.
- **Portable** Python can run on a wide arrangement of hardware organizes and has a comparative interface on all stages.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programming designers to add to or adjust their gadgets to be more successful.
- **Databases** − Python offers interfaces to all noteworthy business databases.

- **GUI Programming -** Python supports GUI applications that can be made and ported to various system calls, libraries and windows structures, for instance, Windows MFC, Macintosh, and the X Window game plan of Unix.
- **Scalable** − Python gives an unrivaled structure and support for tremendous activities than shell scripting.

Beside the recently referenced features, Python has a significant once-over of good features, few are recorded underneath –

- It supports utilitarian and composed programming systems similarly as OOP.
- It can be used as a scripting language or can be orchestrated to byte-code for building colossal applications.
- It gives uncommonly raised level remarkable data types and supports dynamic sort checking.
- It supports automatic garbage collection.
- It can be integrated with C, C++, COM, ActiveX, CORBA, and Java easily.

Python graphical user interfaces (GUIs)

- **Tkinter** − Tkinter is the Python interface to the Tk GUI tool kit sent with Python. We would look this decision in this area.
- **wxPython** − This is an open-source Python interface for wxWindows.
- **JPython** − JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine.

There are many other interfaces available, which you can find them on the net.

Python Tkinter GUI

Fig. 2.1.1 Tkinter Graphical User Interface

Tkinter is the standard GUI library for Python. Python when gotten together with Tkinter gives a snappy and straightforward way to deal with make GUI applications. Tkinter gives an astounding thing organized interface to the Tk GUI tool compartment.

Making a GUI application using Tkinter is a straightforward task. You ought to just play out the going with propels –

- Import the Tkinter module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Example :

```
import tkinter
top = tkinter.Tk()
# Code to add widgets will go here...
top.mainloop()
```

This would create a following window –

Fig. 2.1.2 Tkinter Application Window

Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table :

Table 2.1 Tkinter operator description [8]

Sr No	Operator & Description
1	**Button** The Button widget is used to display buttons in your application.
2	**Canvas** The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.
3	**Checkbutton**

	The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.
4	**Entry** The Entry widget is used to display a single-line text field for accepting values from a user.
5	**Frame** The Frame widget is used as a container widget to organize other widgets.
6	**Label** The Label widget is used to provide a single-line caption for other widgets. It can also contain images.
7	**Listbox** The Listbox widget is used to provide a list of options to a user.
8	**Menubutton** The Menubutton widget is used to display menus in your application.
9	**Menu** The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.
10	**Message** The Message widget is used to display multiline text fields for accepting values from a user.
11	**Radiobutton** The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.

12	**Scale** The Scale widget is used to provide a slider widget.
13	**Scrollbar** The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.
14	**Text** The Text widget is used to display text in multiple lines.
15	**Toplevel** The Toplevel widget is used to provide a separate window container.
16	**Spinbox** The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.
17	**PanedWindow** A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.
18	**LabelFrame** A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.
19	**tkMessageBox** This module is used to display message boxes in your applications.

Geometry Management

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

- **pack()** : This geometry manager organizes widgets in blocks before placing them in the parent widget.
- **grid()** : This geometry manager organizes widgets in a table-like structure in the parent widget.
- **place()** : This geometry manager organizes widgets by placing them in a specific position in the parent widget.

MySQL

A database management, or DBMS, gives the user access to their data and helps them transform the data into information. Such database management systems include dBase, paradox, IMS, MySQL Server and SQL Server. These systems allow users to create, update and extract information from their database. A database is a structured collection of data. Data refers to the characteristics of people, things and events.[5]

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 scripts 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 and C++ programming languages. The official pronunciation of MySQL is not the My Sequel; it is My Ess Que Ell. However, you can pronounce it in your way. Many small and big companies use MySQL. MySQL supports many Operating Systems like Windows, Linux, MacOS, etc. with C, C++, and Java languages.

MySQL is a Relational Database Management System (RDBMS) software that provides many things, which are as follows:

- It allows us to implement the 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.[6]

MySQL Server Relations

MySQL Server stores records relating to each other in a table. Different tables are created for the various groups of information. Related tables are grouped together to form a database.

Relational Database - Sometimes all the information of interest to a business operation can be stored in one table. MySQL Server makes it very easy to link the data in multiple tables. Matching an employee to the department in which they work is one example. This is what makes MySQL Server a relational database management system, or RDBMS. It stores data in two or more tables and enables you to define relationships between the table and enables you to define relationships between the tables.

Primary Key - Every table in MySQL Server has a field or a combination of fields that uniquely identifies each record in the table. The Unique identifier is called the Primary Key, or simply the Key. The primary key provides the means to distinguish one record from all other in a table. It allows the user and the database system to identify, locate and refer to one particular record in the database.

Foreign Key- When a field is one table matches the primary key of another field is referred to as a foreign key. A foreign key is a field or a group of fields in one table whose values match those of the primary key of another table.

Referential Integrity- Not only does MySQL Server allow you to link multiple tables, it also maintains consistency between them. Ensuring that the data among related tables is correctly matched is referred to as maintaining referential integrity.

Data Abstraction- A major purpose of a database system is to provide users with an abstract view of the data. This system hides certain details of how the data is stored and maintained. Data abstraction is divided into three levels.

- **Physical level:** This is the lowest level of abstraction at which one describes how the data are actually stored.
- **Conceptual Level:** At this level of database abstraction all the attributed and what data are actually stored is described and entries and relationship among them.
- **View level:** This is the highest level of abstraction at which one describes only part of the database.[5]

Hardware Requirement

- **Processor :** Intel Pentium Family and above
- **RAM :** above 712 MB
- **Hard Disk :** 1.5 GB and above

CHAPTER-3

Feasibility Study

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

Technical Feasibility

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipments have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?
- ·Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web based user interface. Thus it provides an easy access to the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hard requirements for the development of this project are not many and are available as free as open source. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

Operational Feasibility

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are

to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following:

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

Economic Feasibility

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies. There is nominal expenditure and economical feasibility for certain.

CHAPTER-4

Methodology

What to use ?

Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).

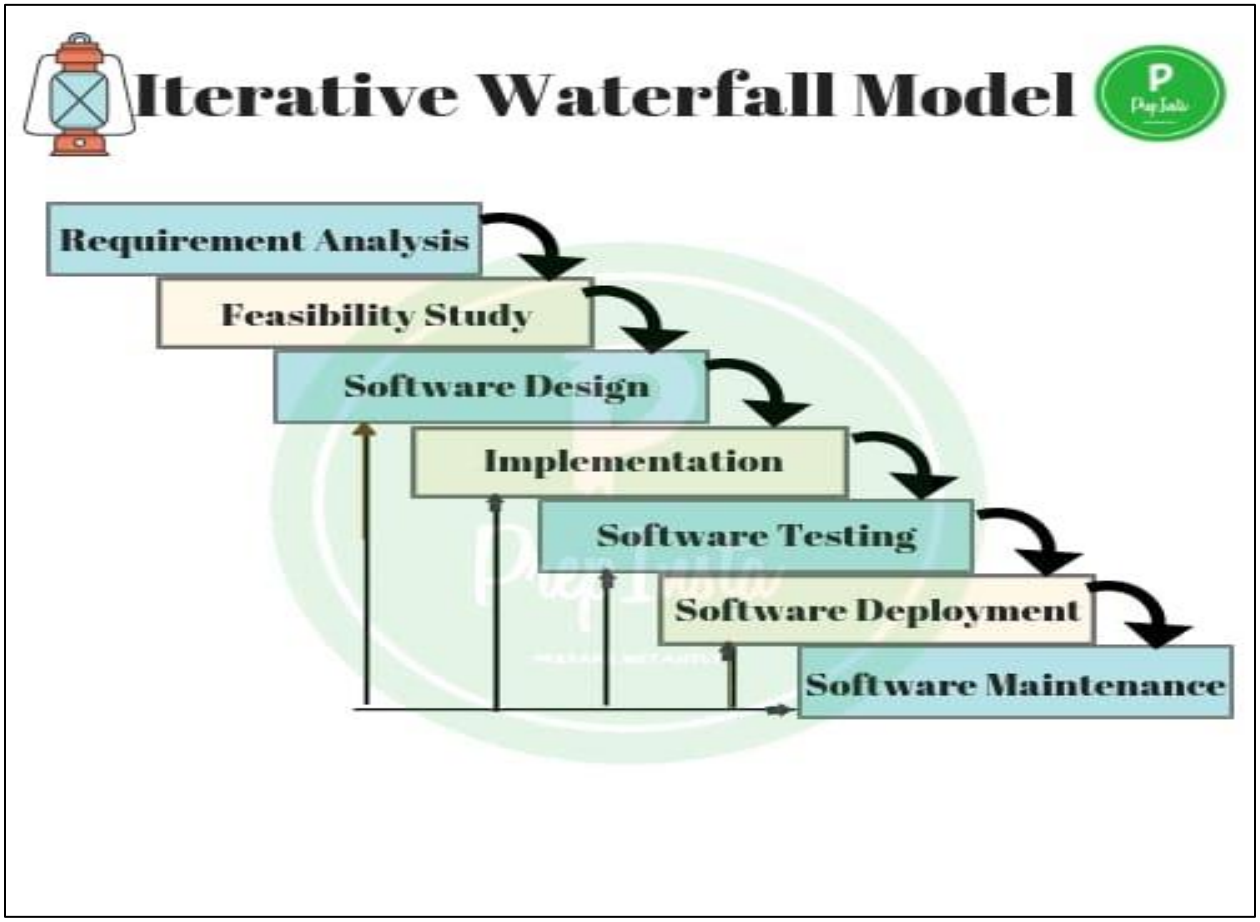


Fig. 5.1.1 Iterative Model [11]

Iterative and Incremental development is a combination of both iterative design or iterative method and incremental build model for development. "During software development, more than one iteration of the software development cycle may be in progress at the same time." This process may be described as an "evolutionary acquisition" or "incremental build" approach."

In this incremental model, the whole requirement is divided into various builds. During each iteration, the development module goes through the requirements, design, implementation and testing phases. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is ready as per the requirement. [12]

Why to use ?

The key to a successful use of an iterative software development lifecycle is rigorous validation of requirements, and verification & testing of each version of the software against those requirements within each cycle of the model. As the software evolves through successive cycles, tests must be repeated and extended to verify each version of the software. [12]

Section of methodology

Project Management Methodology consists of five stages:

- Initiating
- Planning
- Executing
- Controlling
- Closing

CHAPTER- 5

Implementation And Process Description

Use Case Diagram

Use case diagram consists of use cases and actors and shows the interaction between them.

The key points are:

- The main purpose is to show the interaction between the use cases and the actor.
- To represent the system requirement from user's perspective.
- The use cases are the functions that are to be performed in the module.
- An actor could be the end-user of the system or an external system.

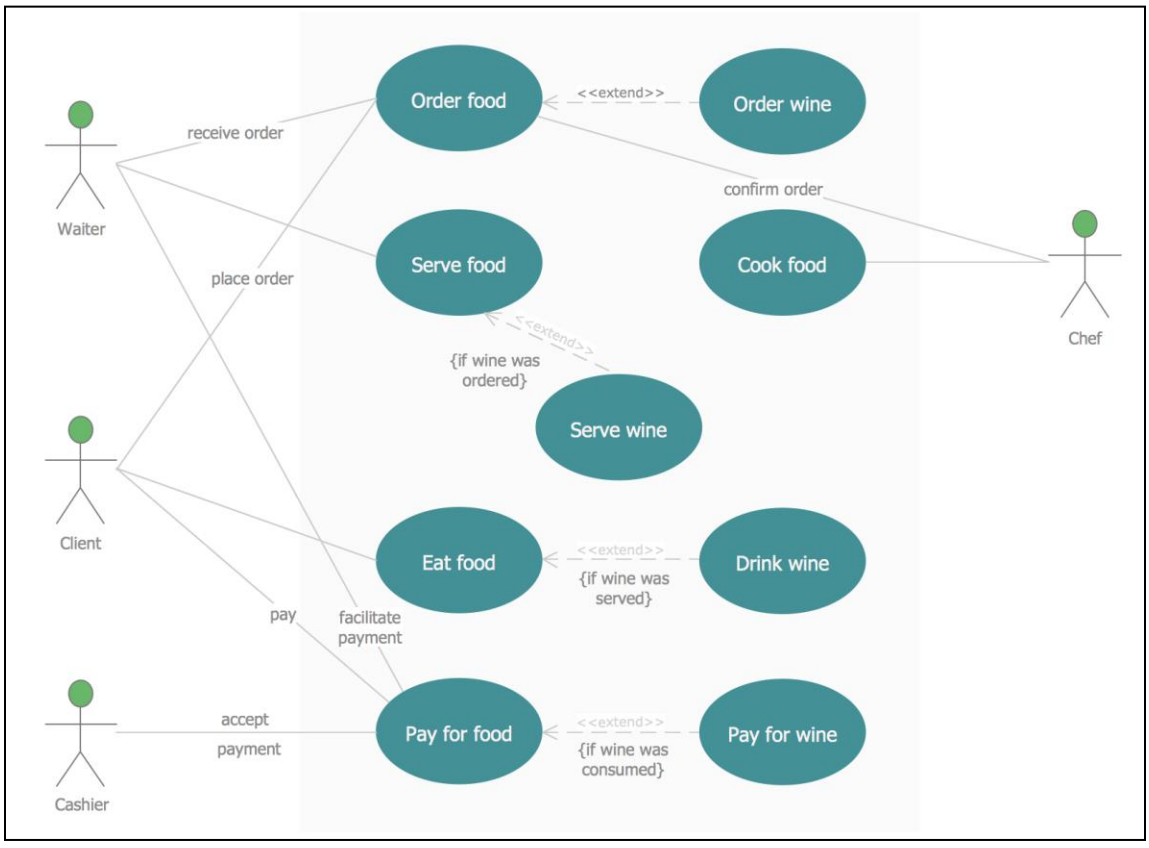


Fig. 6.1.1 Use Case Diagram [7]

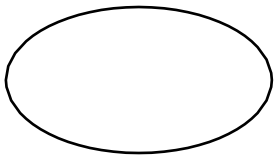
Data Flow Diagram (DFD)

A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. Using two familiar notations Yourdon, Gane and Sarson notation develops the data flow diagrams. Each component in a DFD is labeled with a descriptive name. Process is further identified with a number that will be used for identification purpose. The development of DFD'S is done in several levels.

A DFD is also known as a "bubble Chart" has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

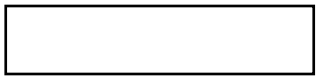
DFD SYMBOLS:

In the DFD, few important symbols are



Process that transforms data flow

Data Flow



Source or Destination of data

0 Level DFD

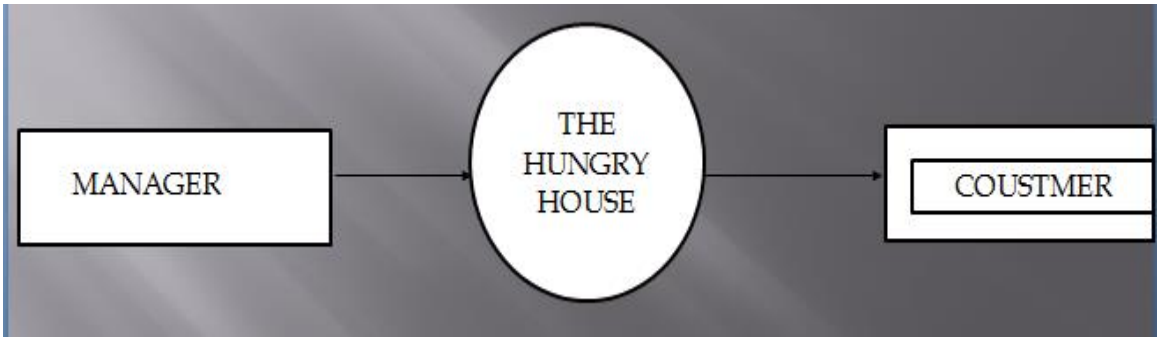


Fig. 6.2.1 0 Level DFD

1 Level DFD

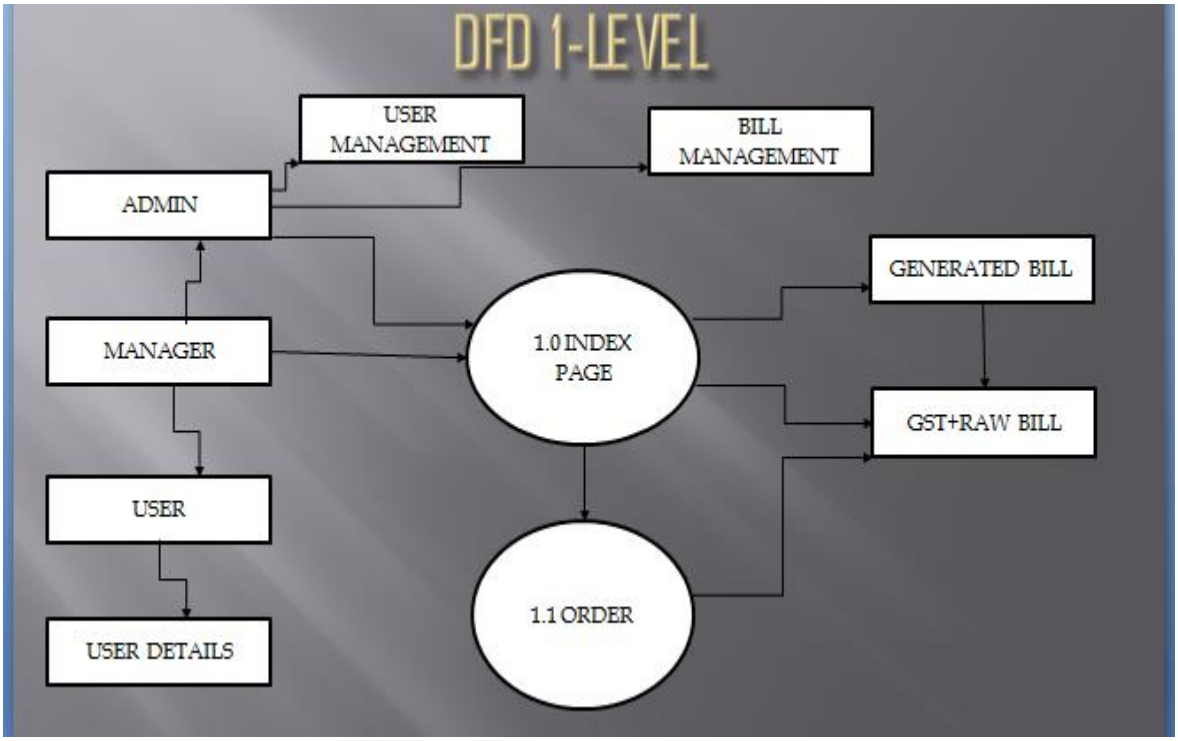


Fig. 6.2.2 1 Level DFD[7]

2 Level DFD

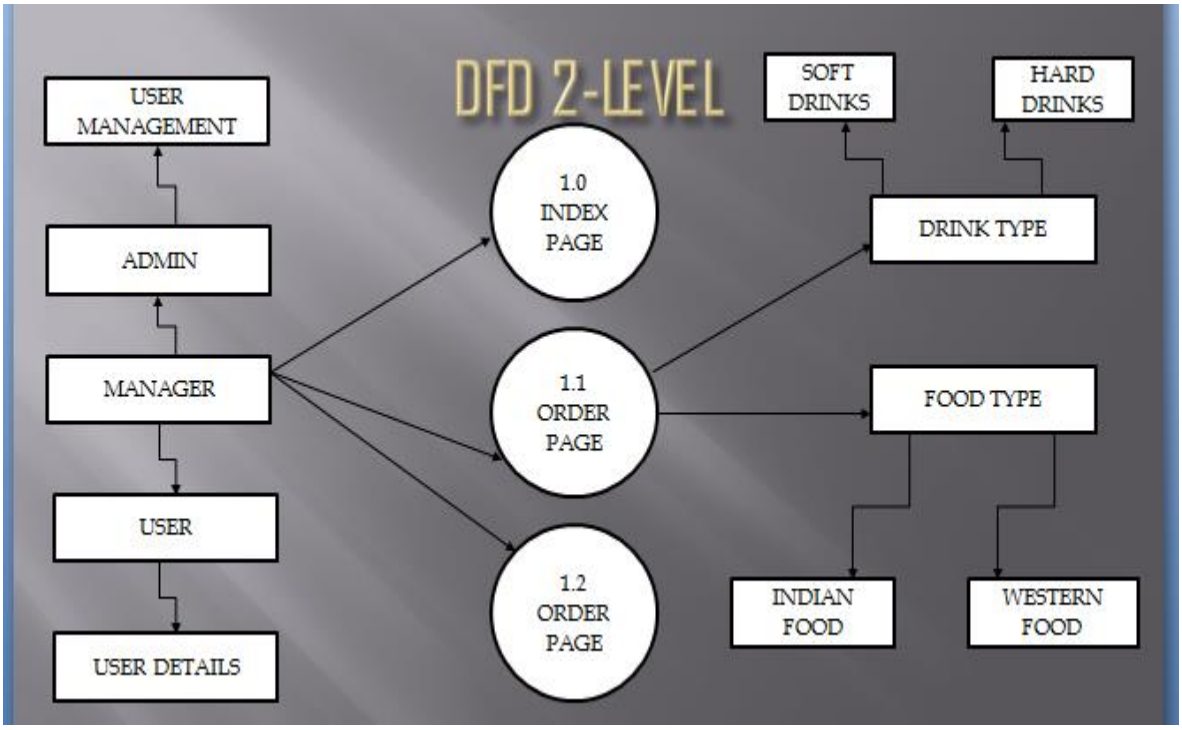


Fig. 6.2.3 2 Level DFD[7]

ER Diagram

The relation upon the system is structure through a conceptual ER-Diagram, which not only specifics the existential entities but also the standard relations through which the system exists and the cardinalities that are necessary for the system state to continue. The entity Relationship Diagram (ERD) depicts the relationship between the data objects. The ERD is the notation that is used to conduct the date modeling activity the attributes of each data object noted is the ERD can be described resign a data object descriptions.

The set of primary components that are identified by the ERD are

- Data object
- Relationships
- Attributes
- Various types of indicators

The primary purpose of the ERD is to represent data objects and their relationships.

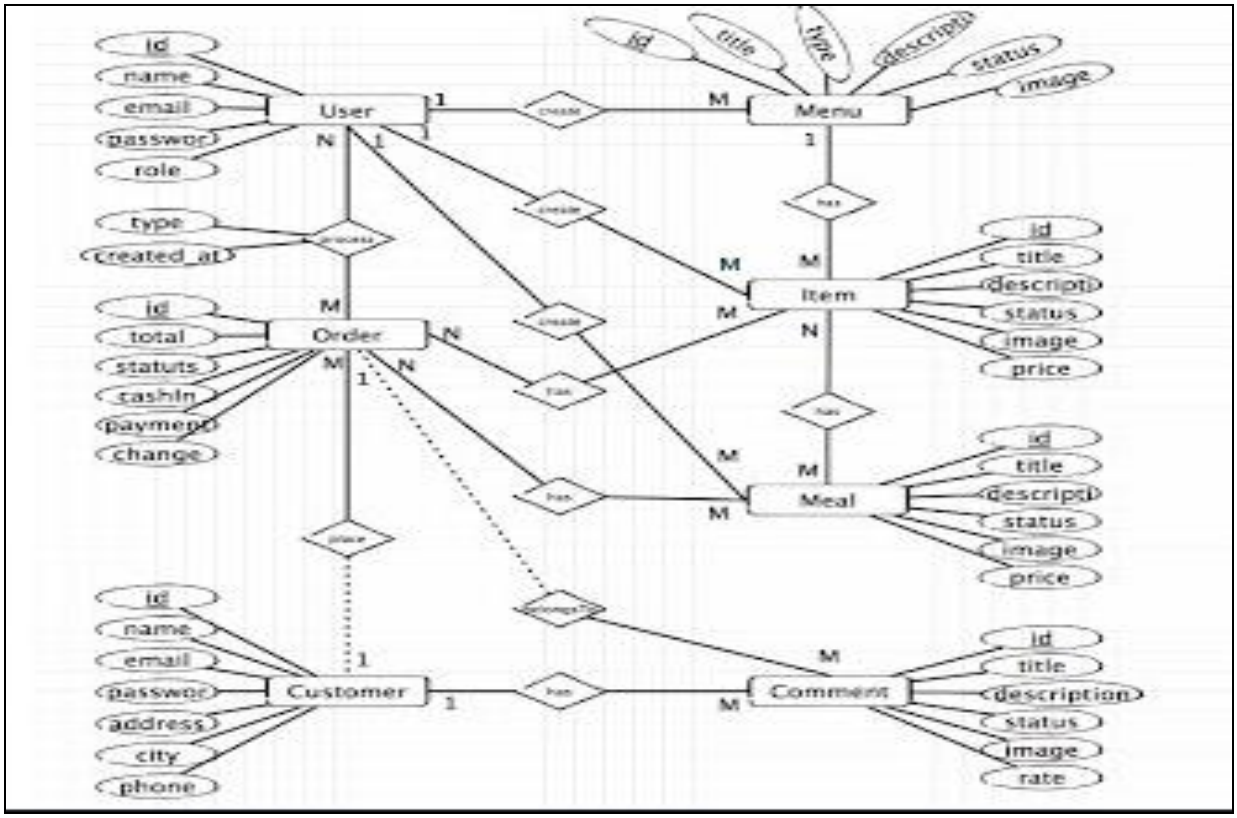


Fig. 6.3.1 ER Diagram

Code Implementation

```
from tkinter import *
import random
import time
import datetime

root=Tk()
root.geometry("1600x8000")
root.title("Restaurant Management System")

text_Input = StringVar()
operator = ""
```

```
Tops=Frame(root, width=1600,relief=SUNKEN)
Tops.pack(side=TOP)

f1=Frame(root,width=800,height=700,relief=SUNKEN)
f1.pack(side=LEFT)

f2 = Frame(root,width=300, height=700,bg="powder blue", relief=SUNKEN)
f2.pack(side=RIGHT)

#================================================================
#                              CALCULATOR
#================================================================

defbtnclick(numbers):
#global operator
        operator =operator + str(numbers)
        text_Input.set(operator)

defbtnClearDisplay():
#global operator
        operator = ""
        text_Input.set("")

defbtnEqualsInput():
#global operator
        sumup= str(eval(operator))
        text_Input.set(sumup)
        operator = ""
```

```
txtDisplay = Entry( f2,  font=('arail', 20, 'bold'), textvariable=text_Input, bd=30, insertwidth=4,  bg="powder blue",  justify='right')
txtDisplay.grid(columnspan=4)

btn7=Button(f2,padx=16,pady=16,  fg="black",  font=('arail',20,'bold'),text="7", bg="powder blue", command=lambda: btnclick(7))
btn7.grid(row=2,column=0)

btn8=Button(f2,padx=16,pady=16,  fg="black",  font=('arail',20,'bold'),text="8", bg="powder blue", command=lambda: btnclick(8))
btn8.grid(row=2,column=1)

btn9=Button(f2,padx=16,pady=16,  fg="black",  font=('arail',20,'bold'),text="9", bg="powder blue", command=lambda: btnclick(9))
btn9.grid(row=2,column=2)

Addition=Button(f2,padx=16,pady=16,  fg="black",  font=('arail',20,'bold'),text="+", bg="powder blue", command=lambda: btnclick("+"))
Addition.grid(row=2,column=3)

btn4=Button(f2,padx=16,pady=16,  fg="black",  font=('arail',20,'bold'),text="4", bg="powder blue", command=lambda: btnclick(4))
btn4.grid(row=3,column=0)

btn5=Button(f2,padx=16,pady=16,  fg="black",  font=('arail',20,'bold'),text="5", bg="powder blue", command=lambda: btnclick(5))
btn5.grid(row=3,column=1)

btn6=Button(f2,padx=16,pady=16,  fg="black",  font=('arail',20,'bold'),text="6", bg="powder blue", command=lambda: btnclick(6))
```

```
btn6.grid(row=3,column=2)

Subtraction=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="-",
bg="powder blue", command=lambda: btnclick("-"))
Subtraction.grid(row=3,column=3)

btn1=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="1",
bg="powder blue", command=lambda: btnclick(1))
btn1.grid(row=4,column=0)

btn2=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="2",
bg="powder blue", command=lambda: btnclick(2))
btn2.grid(row=4,column=1)
btn3=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="3",
bg="powder blue", command=lambda: btnclick(3))
btn3.grid(row=4,column=2)

Multiply=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="*",
bg="powder blue", command=lambda: btnclick("*"))
Multiply.grid(row=4,column=3)

btn0=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="0",
bg="powder blue", command=lambda: btnclick(0))
btn0.grid(row=5,column=0)

btnClear=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="C",
bg="powder blue", command=btnClearDisplay)
btnClear.grid(row=5,column=1)
```

```
btnEquals=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="=",
bg="powder blue", command=btnEqualsInput)
btnEquals.grid(row=5,column=2)

Division=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="/",
bg="powder blue", command=lambda: btnclick("/"))
Division.grid(row=5,column=3)

#=============================================================
#                          TIME AND HEADING NAME
#=============================================================
localtime=time.asctime(time.localtime(time.time()))

lblInfo=Label( Tops, font=('arial',50,'bold'),text="HUNGRY HOUSE ", fg="Steel Blue",
bd=10, anchor='w')
lblInfo.grid(row=0,column=0)

lblInfo=Label( Tops, font=('arial',20,'bold'), text=localtime, fg="SteelBlue", bd=10,
anchor='w')
lblInfo.grid(row=1,column=0)

#=============================================================
#                          BILL CALCULATIONS
#=============================================================
def Ref():
    x=random.randint(10908,500876)
    randomRef=str(x)
    rand.set(randomRef)
```

```
if (Idly.get()==""):
        CoIdly=0
else:
        CoIdly=float(Idly.get())


if (Dosa.get()==""):
        CoDosa=0
else:
        CoDosa=float(Dosa.get())


if (IceCream.get()==""):
        CoIceCream=0
else:
        CoIceCream=float(IceCream.get())


if (Pulav.get()==""):
        CoPulav=0
else:
        CoPulav=float(Pulav.get())


if (Tea.get()==""):
        CoTea=0
else:
        CoTea=float(Tea.get())
```

```
if (Drinks.get()==""):
        CoD=0
else:
        CoD=float(Drinks.get())


CostofIdly = CoIdly * 25
CostofDrinks= CoD * 20
CostofDosa = CoDosa* 25
CostofIceCream = CoIceCream * 30
CostPulav = CoPulav* 50
CostTea = CoTea * 5


Central_GST=
(((CostofIdly+CostofDrinks+CostofDosa+CostofIceCream+CostPulav+CostTea)*
2.5)/100)

State_GST=(((CostofIdly+CostofDrinks+CostofDosa+CostofIceCream+CostPulav+Cost
Tea)* 2.5)/100)

Total_cost                                                                    =
(CostofIdly+CostofDrinks+CostofDosa+CostofIceCream+CostPulav+CostTea)

CostofMeal= "Rs", str('%.2f' % (CostofIdly + CostofDrinks + CostofDosa +
CostofIceCream + CostPulav+CostTea))
C_gst = "Rs", str ('%.2f' % Central_GST)
S_gst = "Rs", str ('%.2f' % State_GST)
OverAllCost ="Rs", str ('%.2f' % (Total_cost+Central_GST+State_GST))
```

```
Sgst.set(S_gst)
Cost.set(CostofMeal)
Cgst.set(C_gst)
Total.set(OverAllCost)

defqExit():
        root.destroy()

def Reset():
        Tea.set("")
        Idly.set("")
        Dosa.set("")
        IceCream.set("")
        Pulav.set("")
        Drinks.set("")

rand.set("")

Total.set("")
Sgst.set("")
Cgst.set("")
Cost.set("")

#==============================================================
#               RESTAURANT MENU
#==============================================================
Tea=StringVar()
Idly=StringVar()
Dosa=StringVar()
```

```
IceCream=StringVar()
Pulav=StringVar()
Drinks=StringVar()
rand = StringVar()
Cost=StringVar()
Sgst=StringVar()
Cgst=StringVar()
Total=StringVar()


lblTea= Label(f1, font=('arial', 16, 'bold'),text="Tea",bd=16,anchor="w")
lblTea.grid(row=0, column=0)

lblTea=Entry(f1,
font=('arial',16,'bold'),textvariable=Tea,bd=10,insertwidth=4,bg="white",justify='right')
lblTea.grid(row=0,column=1)

lblDrinks= Label(f1, font=('arial', 16, 'bold'),text="Drinks",bd=16,anchor="w")
lblDrinks.grid(row=1, column=0)

txtDrinks=Entry(f1,
font=('arial',16,'bold'),textvariable=Drinks,bd=10,insertwidth=4,bg="white",justify='right'
)
txtDrinks.grid(row=1,column=1)

lblIceCream= Label(f1, font=('arial', 16, 'bold'),text="Ice-Cream",bd=16,anchor="w")
lblIceCream.grid(row=2, column=0)
```

```
lblIceCream=Entry(f1,
font=('arial',16,'bold'),textvariable=IceCream,bd=10,insertwidth=4,bg="white",justify='ri
ght')
lblIceCream.grid(row=2,column=1)

lblIdly= Label(f1, font=('arial', 16, 'bold'),text="Idly",bd=16,anchor="w")
lblIdly.grid(row=3, column=0)

txtIdly=Entry(f1,
font=('arial',16,'bold'),textvariable=Idly,bd=10,insertwidth=4,bg="white",justify='right')
txtIdly.grid(row=3,column=1)

lblDosa= Label(f1, font=('arial', 16, 'bold'),text="Dosa",bd=16,anchor="w")
lblDosa.grid(row=4, column=0)

txtDosa=Entry(f1,
font=('arial',16,'bold'),textvariable=Dosa,bd=10,insertwidth=4,bg="white",justify='right')
txtDosa.grid(row=4,column=1)

lblPulav= Label(f1, font=('arial', 16, 'bold'),text="Rice-Plate",bd=16,anchor="w")
lblPulav.grid(row=5, column=0)

txtPulav=Entry(f1,
font=('arial',16,'bold'),textvariable=Pulav,bd=10,insertwidth=4,bg="white",justify='right')
txtPulav.grid(row=5,column=1)
```

```
#============================================================
#               RESTAURANT BILL INFO
#============================================================
lblReference= Label(f1, font=('arial', 16, 'bold'),text="Reference",bd=16,anchor="w")
lblReference.grid(row=0, column=2)

txtReference=Entry(f1,
font=('arial',16,'bold'),textvariable=rand,bd=10,insertwidth=4,bg="powder
blue",justify='right')
txtReference.grid(row=0,column=3)

lblCost= Label(f1, font=('arial', 16, 'bold'),text="Cost of Meal",bd=16,anchor="w")
lblCost.grid(row=1, column=2)

txtCost=Entry(f1,
font=('arial',16,'bold'),textvariable=Cost,bd=10,insertwidth=4,bg="powder
blue",justify='right')
txtCost.grid(row=1,column=3)

lblSgst= Label(f1, font=('arial', 16, 'bold'),text="SGST",bd=16,anchor="w")
lblSgst.grid(row=2, column=2)

txtSgst=Entry(f1,
font=('arial',16,'bold'),textvariable=Sgst,bd=10,insertwidth=4,bg="powder
blue",justify='right')
txtSgst.grid(row=2,column=3)

lblCgst= Label(f1, font=('arial', 16, 'bold'),text="CGST",bd=16,anchor="w")
lblCgst.grid(row=3, column=2)
```

```
txtCgst=Entry(f1,
font=('arial',16,'bold'),textvariable=Cgst,bd=10,insertwidth=4,bg="powder
blue",justify='right')
txtCgst.grid(row=3,column=3)

lblTotalCost= Label(f1, font=('arial', 16, 'bold'),text="Total Cost",bd=16,anchor="w")
lblTotalCost.grid(row=4, column=2)

txtTotalCost=Entry(f1,
font=('arial',16,'bold'),textvariable=Total,bd=10,insertwidth=4,bg="powder
blue",justify='right')
txtTotalCost.grid(row=4,column=3)


#=============================================================
#               BUTTONS
#=============================================================
btnTotal=Button(f1,padx=16,pady=8,bd=16,fg="black",font=('arial',16,'bold'),width=10,t
ext="Total",bg="powder blue",command=Ref).grid(row=7,column=1)

btnReset=Button(f1,padx=16,pady=8,bd=16,fg="black",font=('arial',16,'bold'),width=10,t
ext="Reset",bg="powder blue",command=Reset).grid(row=7,column=2)

btnExit=Button(f1,padx=16,pady=8,bd=16,fg="black",font=('arial',16,'bold'),width=10,te
xt="Exit",bg="powder blue",command=qExit).grid(row=7,column=3)

root.mainloop()
```

CHAPTER-6

Software Testing

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirement analysis where the information domain, functions, behavior, performance, constraints and validation criteria for software are established. Moving inward along the spiral, we come to design and finally to coding. To develop computer software we spiral in along streamlines that decrease the level of abstraction on each turn.

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing progress by moving outward along the spiral to integration testing, where the focus is on the design and the construction of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed. Finally we arrive at system testing, where the software and other system elements are tested as a whole.

Unit Testing

Testing an application to its smallest unit is called Unit Testing. Again, testing each module of an application which numerous test cases and checking validations against unforeseen scenarios is what unit testing is all about. Once a bug is detected, that is recorded in the bug tracker, a ticket is raised, this bug is fixed, and again new unit test cases are written to perform unit testing over the debugged piece of code. Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing we have is white box oriented and some modules the steps are conducted in parallel.

White Box Testing

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

- Guarantee that all independent paths have been executed.
- Execute all logical decisions on their true and false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Execute internal data structures to ensure their validity.

Black Box Testing

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structure or external database access
- Performance errors
- Initialization and termination errors.

In this testing only the output is checked for correctness. The logical flow of the data is not checked.

Integrating Testing

Integration testing ensures that software and subsystems work together as a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together. Once each individual part of the system is tested, every smallest unit

is tested, different modules of the system are now integrated together and tested. Whether the integration works or whether a part of the system that is functional individually starts failing when integrated with another part is what integration testing is all about.

System Testing

Involves in-house testing of the entire system before delivery to the user. Its aim is to satisfy the user the system meets all requirements of the client's specifications. That an integrated system meets all its specifications and requirements is decided by system testing.

Acceptance Testing

It is a pre-delivery testing in which entire system is tested at client's site on real world data to find errors.

Validation

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled.

In case of erroneous input corresponding error messages are displayed.

Compilation Test

It was a good idea to do our stress testing early on, because it gave us time to fix some of the unexpected deadlocks and stability problems that only occurred when components were exposed to very high transaction volumes.

Execution Test

This program was successfully loaded and executed. Because of good programming there was no execution error. [3]

Regression Testing

Once the system is debugged, it is tested again to see if it is compatible with the changes made and compatible with any changes made to the environment.

Load Testing

Testing that the system can take as much load as it is supposed to take and testing how much load it can take and to what extent it can exceed its limit and where it breaks.

Performance Testing

Testing how the system performs like slow/fast and how it performs under certain workloads.

CHAPTER-7

Screenshots from Project

GUI – Main display window with name of restaurant name current time and date with calculator

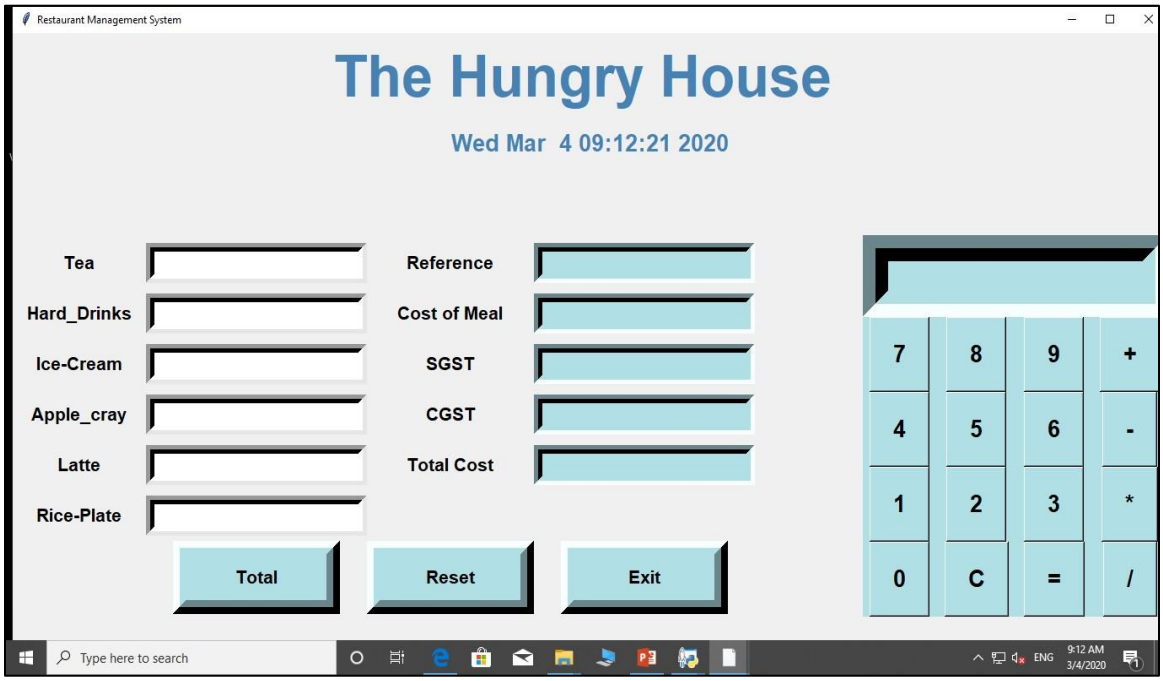


Fig 8.1.1 Home Page

We have added a calculator for instant calculation –

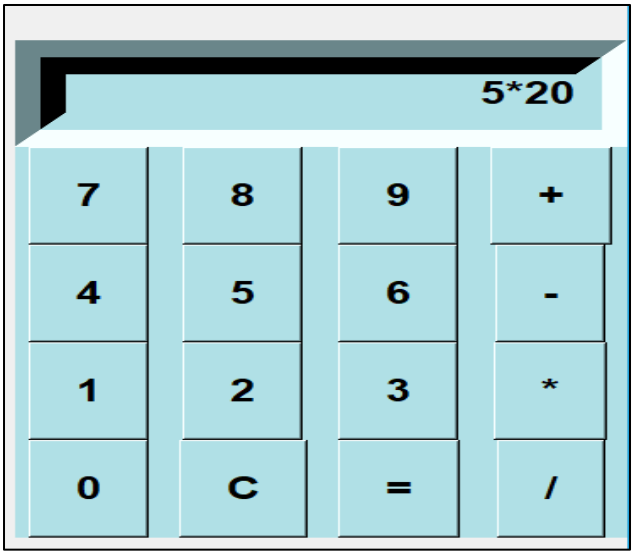


Fig 8.1.2 Calculation

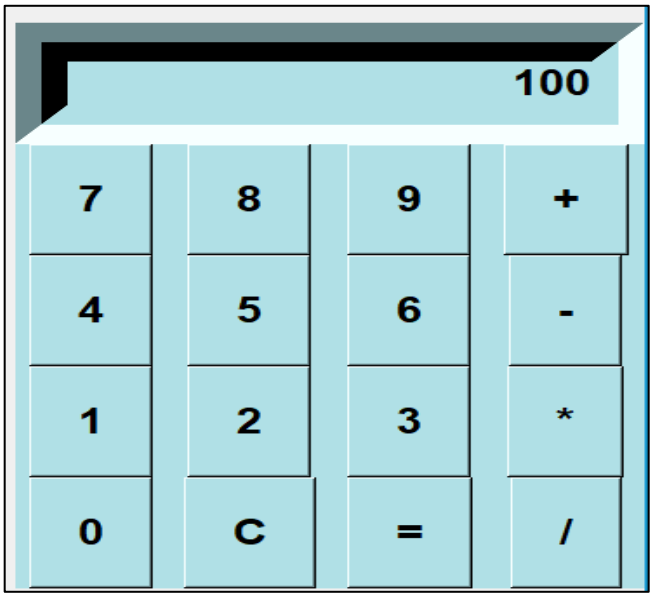


Fig 8.1.3 Calculation Result

Deciption for buttons used :

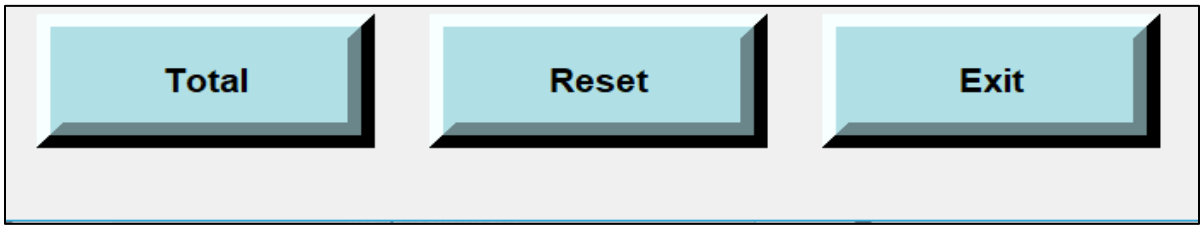


Fig 8.1.4 Buttons

Total button gives the calculation of Bill

Reset button resets all the values

Exit button closes the window

We need to insert number of menu items taken by customer ,after inserting no. of items

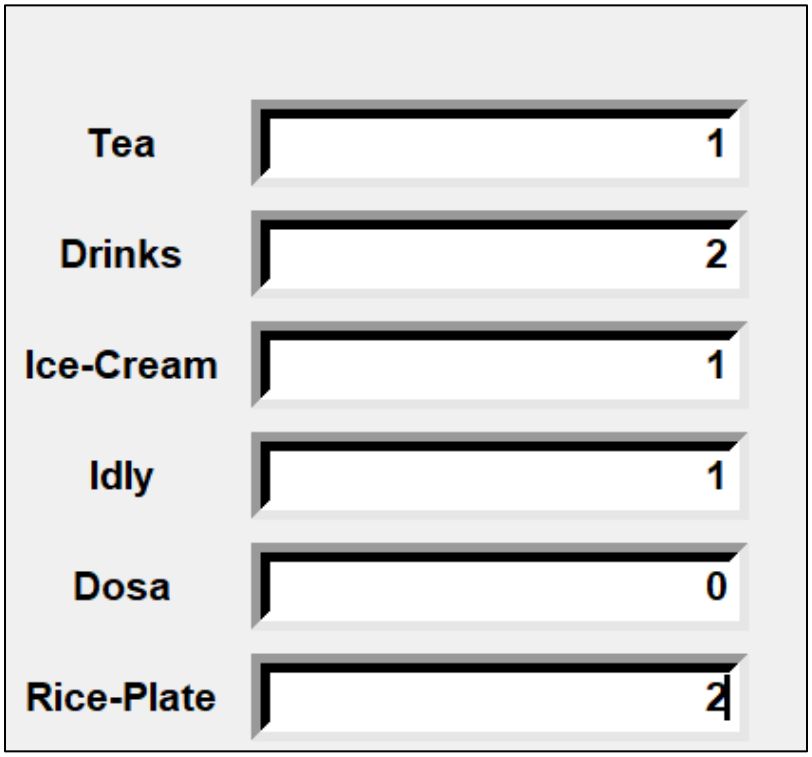
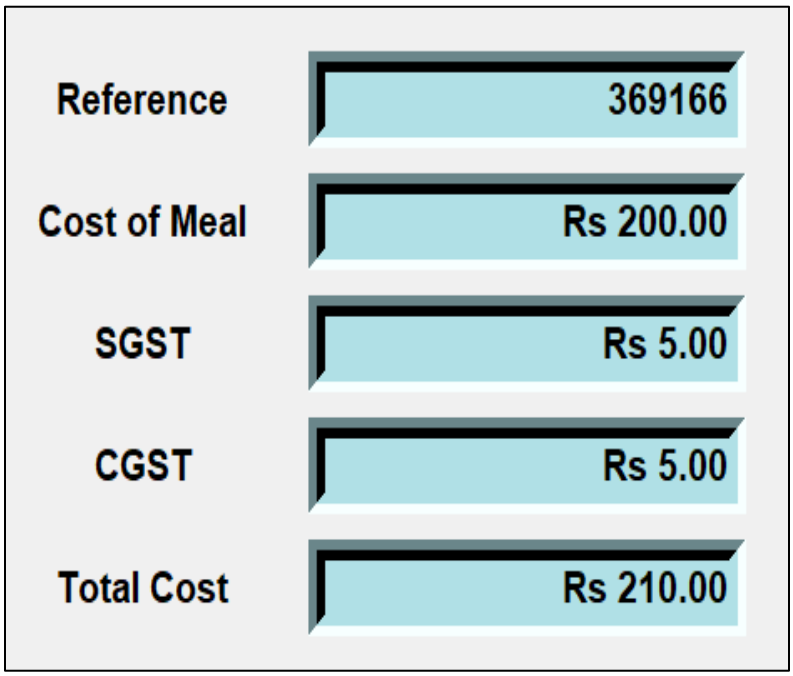


Fig 8.1.5 Quantity of items ordered

Generated Bill



➔ **Bill Number**

➔ **Total meal cost**

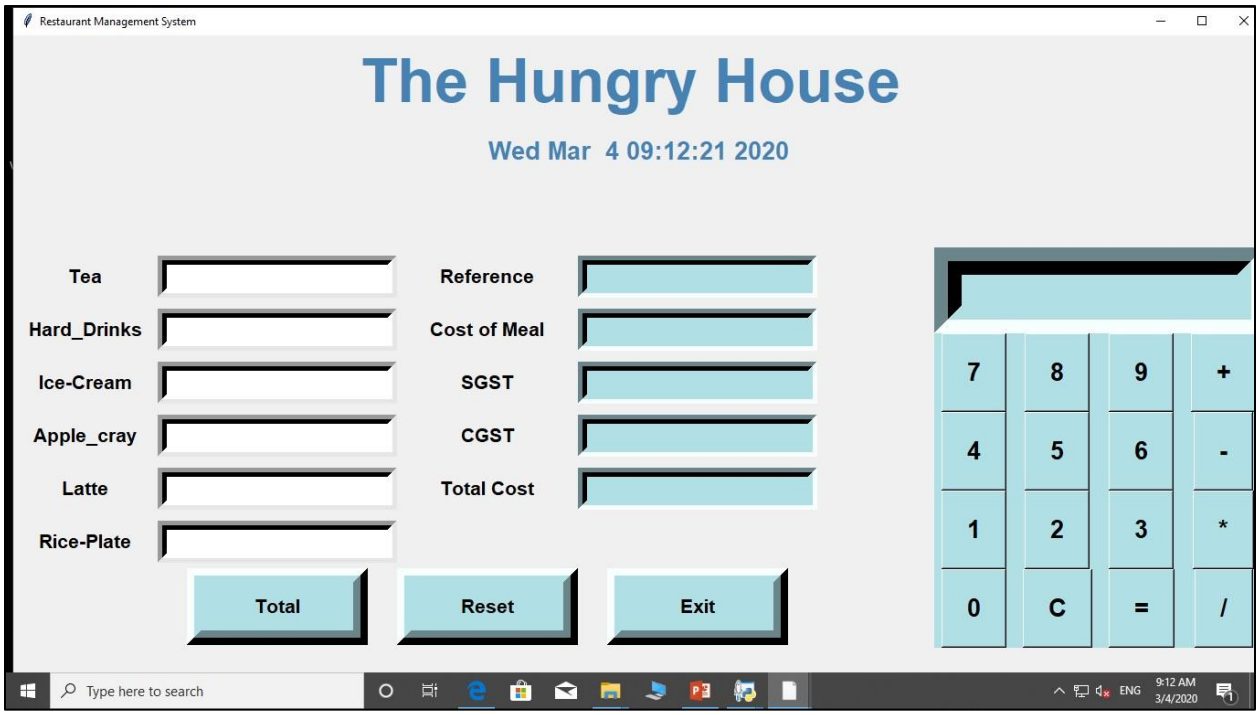
➔ **Total SGST**

➔ **Total CGST**

➔ **Total cost including GST**

Fig 8.1.6 Calculation Result

After clicking TOTAL button it will automatically generate the bill.



CHAPTER-8

Conclusion

This project has really been faithful and informative. It has made us learn and understand the many trivial concepts of Python Language. As we have used python Tkinter as a GUI it provides various controls, such as buttons, labels and text boxes to build a user friendly application.

The fast growing use of internet confirms the good future and scope of the proposed project.

Finally it has taught us a valuable lifelong lesson about the improvements and working and interacting in a group.

Limitation

The size of the database increases day-by-day, increasing the load on the database back up and data maintenance activity. Supported only in My SQL database. Direct Connection with expert are not available for everyone.

Future Scope

There is ample scope of enhancement and adding functionalities to this application. The application can have a meal box recommendation system based on the frequent search results of different users. The portal can also send email notifications to buyer about certain availabilities. The application can be more scalable by extending the search functionality based on country, city or area. While this application meets the basic requirements of a restaurant management system it can be extended to make the application more dynamic and robust. The User Interface can be made more attractive and user friendly.[1]

REFERENCES

1. https://core.ac.uk/download/pdf/132492064.pdf
2. https://docs.python.org/3/tutorial/
3. https://nairaproject.com/projects/3197.html
4. https://www.javatpoint.com/mysql-tutorial
5. https://www.tutorialspoint.com/mysql-tutorial
6. https://www.tutorialspoint.com/sdlc/sdlc_iterative_model.htm
7. https://www.academia.edu/38576785/testing
8. https://realpython.com/python-gui-tkinter/
9. https://www.w3schools.com/python/python_datetime.asp
10. https://www.slideshare.net/KrishnaRanjan/houseoffood