#### “EVENT MANAGEMENT SYSTEM”

#### A project Report

Submitted In Partial Fulfillment of the

Requirement for the of award of the Degree of

**BACHELOR OF COMPUTER APPLICATION**

**By**

**Miss. ISHA SANTOSH SAKPAL**

**Mr.SAHIL VILAS SHINDE**

Seat Number:

**Under the esteemed guidance of**

**Miss.Sifa Rawal Mam**

**Assistant professor**

****

**DEPARTMENT COMPUTER APPLICATION**

**KOKAN RATNA KALA MAHAVIDYALAY,BHARNE**

**(Affiliate University of Nagpur)**

**KHED,RATNAGIRI,**

**MAHARASHTRA**

**YEAR 2024-25**

**PERFORMAFOR THE APPROVAL PROJECT PROPOSAL**

PRN No: Roll No:

1. NAME OF THE STUDENT: ISHA SANTOSH SAKPAL AND

SAHIL VILAS SHINDE

2. TITLE OF THE PROJECT: EVENT MANAGEMENT SYSTEM

3. NAME OF THE GUIDE: SIFA RAWAL MAM

4. TEACHING EXPERIENCE OF THE GUIDE:

5. IS THIS YOUR FIRST PROJECT: YES NO

Signature of the Student Signature of the Guide

Date: ………….. Date: …………..

Signature of the Coordinator

Date: ……………

## KOKAN RATNA KALA MAHAVIDYALAY,BHARNE

***(Affiliate University of Nagpur)***

**KHED, RATNAGIRI, MAHARASHTRA**

## DEPARTMENT OF COMPUTER APPLICATION

****

**CERTIFICATE**

This is to certify that the project entitled, **“EVENT MANAGEMENT SYSTEM**”, is bonafied work of **Miss.ISHA SANTOSH SAKPAL** And **Mr.SAHIL VILAS SHINDE** Bearing Seat No:

& Submitted inpartial fulfilment of the requirements for the award of degree of **BACHELOR OF COMPUTER APPLICATION** From University of Nagpur.

**Internal Guide Coordinator**

**External Examiner**

**Date**: **college seal**

**EVENT MANAGEMENT**

**SYSTEM**



****

**ACKNOWLEDGEMENT**

It is my prime duty to offer my sincere gratitude to University of Mumbai to include the project work in the syllabus of Third Year Bachelor’s Degree so as to develope interest about research work among the students.

I would like to express my sincere thanks to Our Principal Dr. Sachin Ambekar Sir(Kokan Ratna Kala Mahavidyalay, Bharne Head of the Department of Computer Application for giving me the opportunity to complete the project work by providing facilities in the department and providing valuable encouragement to complete the task.

I am greatly obliged to Assist Prof Sifa Rawal Mam and lecturers in Computer Application who provided the valuable guidelines and conceptual guidance throughout the project work, also helped out in clearing concepts about the project.

Last but not the least my special thanks to my parents, my friends and all of those who have encouraged me, helped me to complete this course successfully.

# DECLARATION

I hereby declare that the project entitled, “**EVENET MANAGEMENT SYSTEM”** done at **KOKAN RATNA KALA MAHAVIDYALAY, BHARNE** has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfillment of the requirements for the award of degree of **BACHELOR OF COMPUTER APPLICATION** to be submitted as final semester project as part of our curriculum.

**Miss. Isha Santosh Sakpal.**

**Mr. Sahil Vilas Shinde.**

|  |  |  |
| --- | --- | --- |
| **Table Of Content** | | |
| **Sr. No.** | **Phase Name** | **Page No.** |
| 1 | Introduction | 9-12 |
| 1.1 | Background | 9 |
| 1.2 | Objectives | 10 |
| 1.3 | Purpose, Scope & Applicability | 11& 12 |
| 1.4 | Achievements | 12 |
| 1.5 | Organization of Report | 12 |
| 2 | Survey of Technologies | 13-18 |
| 2.1 | Existing System | 13 |
| 2.2 | Proposed System | 14 |
| 2.3 | Feasibility Study | 15 |
| 2.4 | Language Specification | 16 &17 |
| 2.5 | Fact Finding Techniques | 17 & 18 |
| 3 | Requirements and Analysis | 19-23 |
| 3.1 | Problem Definition | 19 |
| 3.2 | Solution Architecture | 19 |
| 3.3 | Software and Hardware Requirement | 20 |
| 3.4 | Functional And Requirement Specification | 20 & 21 |
| 3.5 | Planning And Scheduling | 22 |
| 3.6 | Conceptual Model | 22 & 23 |
| 4 | System Design | 24-38 |

|  |  |  |
| --- | --- | --- |
| 4.1 | Basic Modules | 24 & 25 |
| 4.2 | Data Design | 25 & 26 |
| 4.3 | Schema Design | 26 |
| 4.4 | ER Diagram | 26,27,28 |
| 4.5 | Data Flow Diagram | 29,30& 31 |
| 4.6 | Class Diagram | 32 |
| 4.7 | Use Case Diagram | 33 & 34 |
| 4.8 | Activity Diagram | 35 & 36 |
| 4.9 | Sequence Diagram | 37 & 38 |
| 4.10 | Component Diagram | 39 |
| **5** | **IMPLEMENTATION AND TESTING** |  |
| 5.1 | Implementation Approach |  |
| 5.2 | Coding Details |  |
| 5.3 | Testing Approach |  |
| 5.3.1 | Unit Testing |  |
| 5.3.2 | Integrated Testing |  |
| 5.4 | Modification and Improvements |  |
| **6** | **Results and Discussion** |  |
| 6.1 | Test Reports |  |
| 6.2 | User Documentation |  |
| **7** | **Conclusions** |  |
| 7.1 | Limitations of the System |  |
| 7.2 | Future Scope of the System |  |
|  | **References** |  |

PRELIMIN INVESTIGATION

**INTRODUCTION**

The Event Management System is complete solution for managing a Decor Company, in other words, an enhanced tool that assists in organizing the day to day activities of a Company.

There is the need of an application for efficient management handling customer orders.

This decor company management system keeps every record which is needed for admin & customer as well as the supplier (vendor) of that Company by reducing paperwork.

Today The Management of This Company is manually handled, because of which system requires more time; more manpower and it is complicated to manage. Hence I am solving this problem in this project By making All work computerized.

**CURRENT SYSTEM**

In the Current System there is no such facilities available which can make the running of a company smooth and easier.

The following points are includes in the current system:-

1. The existing system requires more time for processing.
2. The billing is done through rigid calculation.
3. Not any type of security is provided so it may loss data.
4. Difficult to search the previous records.
5. More man power is required.
6. This system is very slow & lengthy.
7. There are High chances of missing on dates.

### SCOPE & OBJECTIVE OF PROPOSED SYSTEM

##### SCOPE

* It provides user to enter data simple.
* Data storage will be faster and easy.
* It provides security.
* Updating data is easier

#### OBJECTIVE

* Keep distributors, customer’s as well as Employee’s information.
* Maintaining all bill records, which note the total transaction for each daily & for each monthly year.
* Handling of register was very complicated so in this system we are totally discarding it by computerizing system.
* It provide user to enter data easily with the help of interactive forms.

**ADVANTAGES OF THE PROPOSED SYSTEM**

* Easy to search customer details.
* Provides extra security.
* Does not need more manpower.
* Saves time.
* Easy to maintain records
* User Friendly

##### FEASIBILITY STUDY

###### The Feasibility is involved the following point:

1. **Technical feasibility:**

Technical consideration evaluates existing hardware & software. Technical feasibility centers on the existing computer System & to what extent it can support the proposed addition.

###### Behavioral feasibility:

Behavioral feasibility determines how much effort will go into educating, selling & training the user staff on a candidate system.

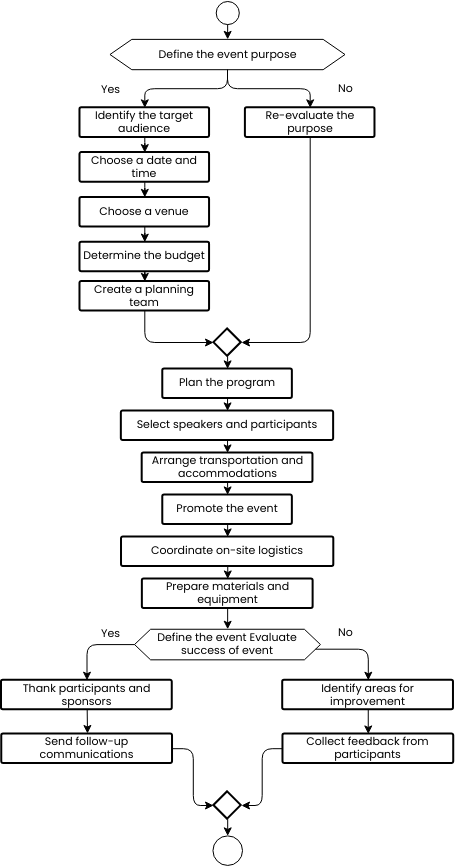
###### Economical feasibility:

More commonly known as cost / benefit analysis the procedure to determine the benefits & saving that are expected from a candidate System & compare them with cost.

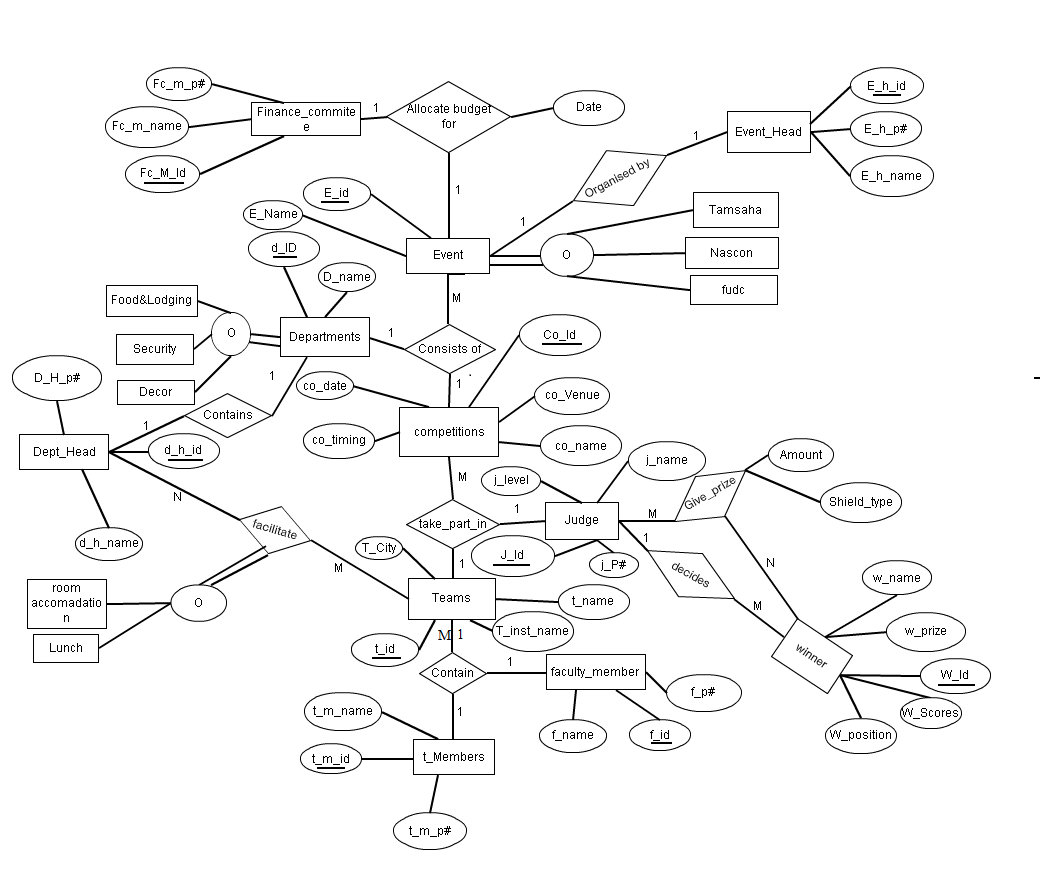
**GANTT CHART**

## PLANNING AND SCHEDULING: GANTT CHART:

|  |  |  |  |
| --- | --- | --- | --- |
| Name of the Phase | Expected Date of Completion | Actual Date of Completion | Signature of Guide |
| Project Search | 01-11-2024 | 03-11-2024 |  |
| Finalization and Allocation | 05-12-2024 | 20-12-2024 |  |
| Investigating of System Requirements | 10-01-2023 | 12-01-2023 |  |
| System Design | 20-02-2025 | 28-02-2025 |  |

**FLOW CHART FOR EVENT MANAGEMENT SYSTEM**

**ER DIAGRAM FOR EVENT MANAGEMNT SYSTEM:**



**DATA FLOW DIAGRAM :**

**ZERO LEVEL DFD:EVENT MANAGEMENT SYSTEM**

User

Services

Customer

Services

Booking event

User

Customer

Management

Event

Management

Enquiry

Management

System User

Management

Package

Management

Login

Management

# 

## UML DIAGRAMS:

**USE CASE DIAGRAM FOR ADMIN:**

# 

# ACTIVITY DIAGRAM FOR ADMIN

Login

Register event

Specify Requirements

Send mail to higher

Authorities for

confirmation

Send mail to event

organizers

reject

Admin issues requested

resources

Send mail/sms to organizers about status of event

resources

# 

# 

# 

# SYSTEM ANALYSIS

## FACT FINDING TECHNIQUES

The specific method analyst’s use for collecting data about requirement is collect fact

The tools used in information gathering or fact finding are Techniques commonly used are:-

1. **Interviewing.**
2. **Questionnaires.**
3. **Record inspection/Documentation.**
4. **Observation.**

###### Interviewing:-

An interview is a face to face interpersonal situation in which a person called the interview asks a person being interviewed .question designed to gather information about a problem.

The analyst or interview can schedule interview with key personal of the organization.

The analyst also needs to conduct detailed interview with all the people who will actually use the system.

###### Questionnaires:-

A questionnaire is a tool that has question to which individual respond.

###### A questionnaire has the following advantages –

* + - * it is economical & requires less skill than an interview
      * It can be used to gather data from large number of people simultaneously.
      * It is a uniform method in which all all question asked are the same to all people.

###### Record Review:-

In all organization documents such as form, records, report, manuals etc. are available. These help in determing how the present system runs.

The process of fact finding include collection of all possible documents & evaluating them.

Most manuals are not up to date& may not be readable.

###### On site Observation:-

The purpose of on-site observation is to get as possible to the real system begin studied.

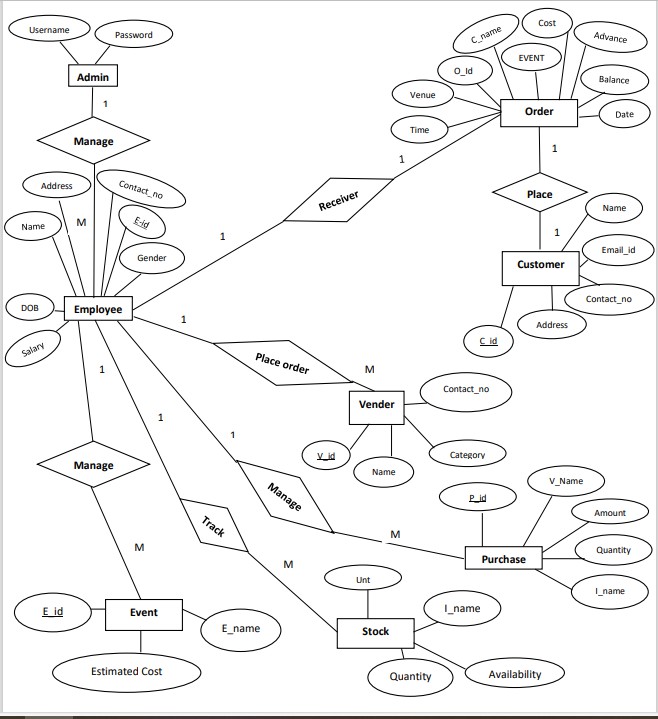
It is the process of recognizing & nothing people, objects & occurrence to obtain information.

**The following question can serve as a guide for on-site observation**

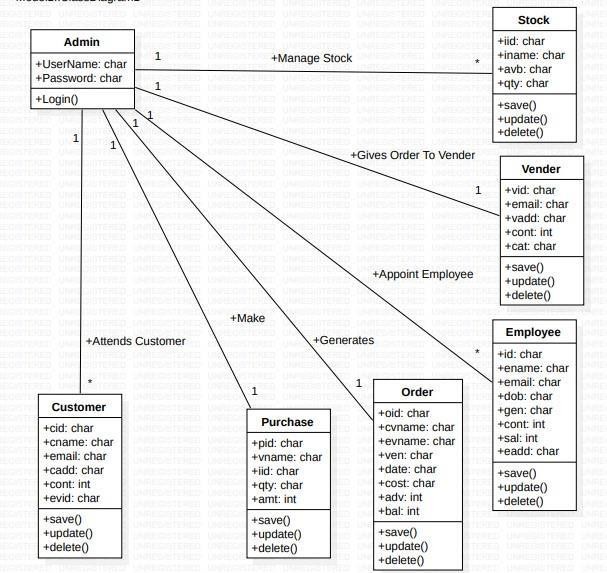
* + - * What kind of system is it? What does it do?
      * Who runs the system? Who are the important people in it?
      * What is the history of the system?

**DIAGRAMS**

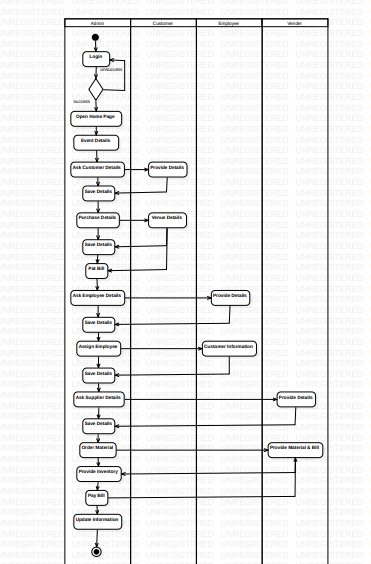
**Entity Relationship Diagram**

****

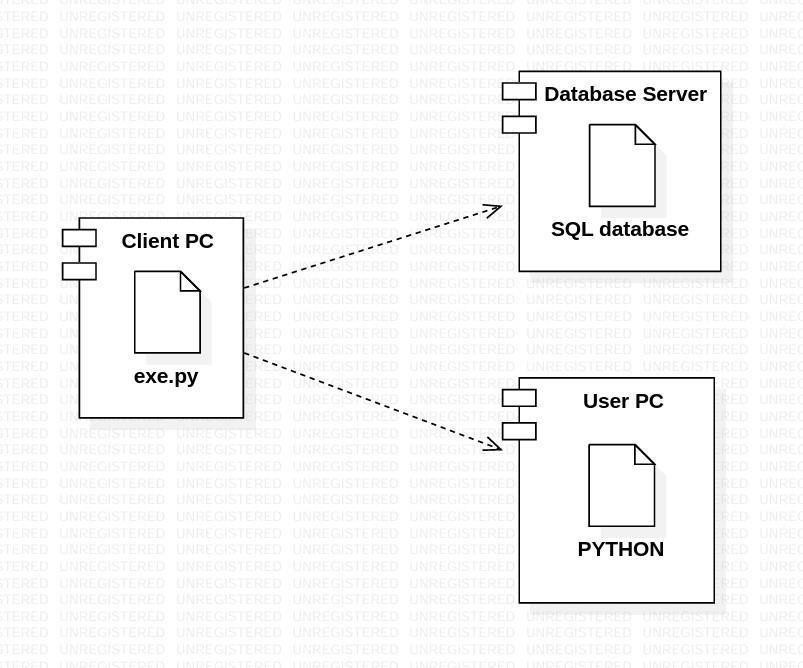
**CLASS- DIAGRAM**

****

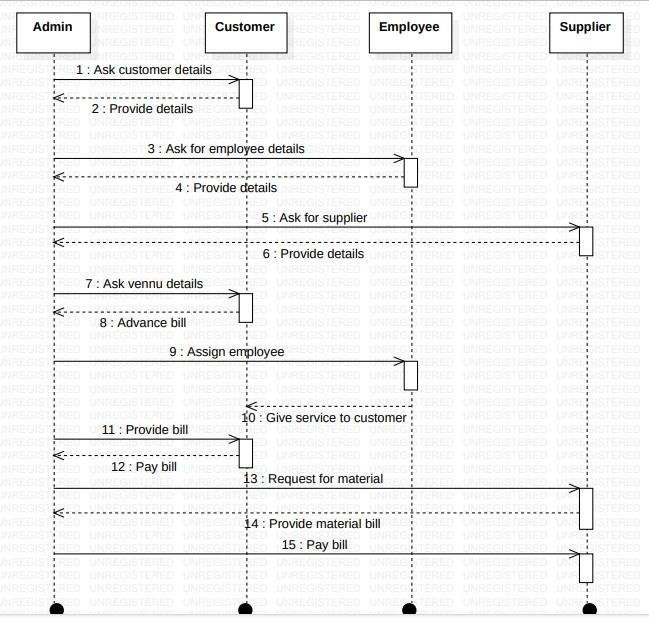
**ACTIVITY DAIGRAM**

****

**DEPLOYMENT DIAGRAM**

****

**SEQUENCE DAIGRAM**

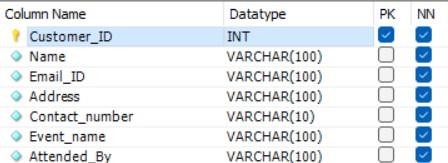
****

# SYSTEM DESIGN

##### LIST OF TABLES WITH PROGRAM & DESCRIPTIONS

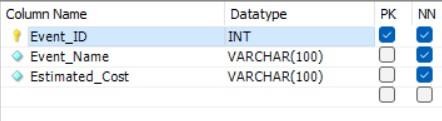
**CUSTOMER TABLE**

**Stores information of customers.**

****

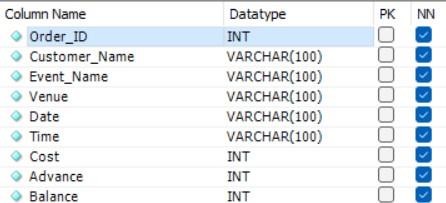
##### EVENT TABLE

**Stores Details Of Events.**

****

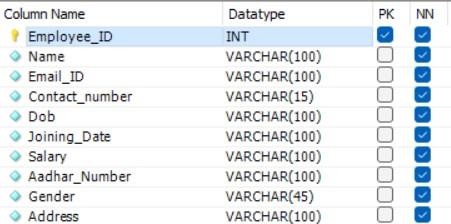
##### ORDER TABLE

**Stores Information About Orders**.



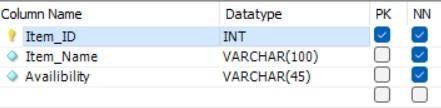
##### EMPLOYEE TABLE

**Stores information About Employees.**

****

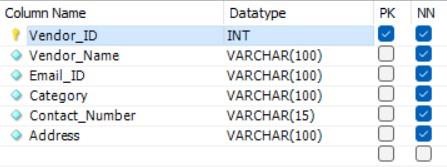
##### STOCK TABLE

**Store information About Stock.**

****

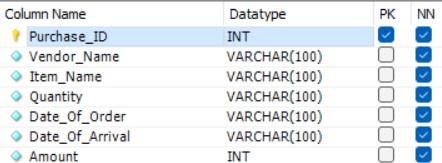
##### VENDOR TABLE

**Stores information of Vendors**

****

## PURCHASE TABLE

**Stores Purchase Details.**

****

# SYSTEM CODING

##### LOGIN FORM

from tkinter import \*

from PIL import Image, ImageTk from tkinter import messagebox from mdi\_form import MDI\_form

class Register:

def init (self, root): self.root = root

self.root.title("Register window") self.root.geometry("1350x700+0+0") self.root.config(bg="white")

self.bg = ImageTk.PhotoImage(file="wedding.jpg") bg\_label = Label(self.root, image=self.bg) bg\_label.place(x=1, y=0, relheight=1, relwidth=1)

self.var\_username = StringVar() self.var\_password = StringVar()

frame\_login = Frame(self.root,bg='white') frame\_login.place(x=540, y=150, width=400, height=400)

title = Label(frame\_login, text="Login", font=("times new roman", 25, "bold"),bg='white', fg="green")

title.place(x=70, y=30)

self.entry\_username = Entry(frame\_login, width=25, textvariable=self.var\_username, fg="black", border=0, bg='white', font=('Microsoft YaHei UI Light', 11, 'bold'))

self.entry\_username.place(x=30, y=80) self.entry\_username.insert(0, 'username') self.entry\_username.bind('<FocusIn>', self.on\_enter) self.entry\_username.bind('<FocusOut>', self.on\_leave)

Frame(frame\_login, width=295, height=2, bg='black').place(x=25, y=107) self.entry\_password = Entry(frame\_login, width=25,

textvariable=self.var\_password,show="\*", fg="black", border=0, bg='white', font=('Microsoft YaHei UI Light', 11, 'bold'))

self.entry\_password.place(x=30, y=150) self.entry\_password.insert(0, 'password') self.entry\_password.bind('<FocusIn>', self.on\_enter) self.entry\_password.bind('<FocusOut>', self.on\_leave)

Frame(frame\_login, width=295, height=2, bg='black').place(x=25, y=177) btn\_login = Button(frame\_login, width=39, pady=7, text='login', bg="purple",

fg='white', border=0, command=self.login)

btn\_login.place(x=35, y=204)

label\_register = Label(frame\_login, text='Don\'t have an account', fg='black', bg='white', font=('Microsoft YaHei UI Light', 11, 'bold') label\_register.place(x=75, y=270)

def login(self):

username = self.var\_username.get() password = self.var\_password.get()

if username == 'pooja' and password == 'vedika': messagebox.showinfo("Login Successful", "Welcome!") self.root.destroy()

from mdi\_form import MDI\_form mdi\_window = Tk()

mdi\_form = MDI\_form(mdi\_window)

elif username != 'pooja' and password != 'vedika': messagebox.showerror('Invalid', "Invalid username and password")

elif password != 'vedika': messagebox.showerror('Invalid', "Invalid password")

elif username != 'pooja': messagebox.showerror('Invalid', "Invalid username")

def on\_enter(self, e): entry = e.widget

if entry.get() == 'username' or entry.get() == 'password': entry.delete(0, 'end')

def on\_leave(self, e): entry = e.widget

if entry.get() == '':

if entry == self.entry\_username: entry.insert(0, 'username')

else:

entry.insert(0, 'password')

def main():

root = Tk()

obj = Register(root) root.mainloop()

if name == " main ": main()

##### CUSTOMER FORM

#from pydoc import render\_doc from tkinter import\*

from tkinter import ttk import tkinter as tk

from tkinter import ttk, messagebox import mysql.connector

import re

from PIL import Image ,ImageTk

class customer():

def init (self,root): self.root=root

self.root.title("Event management system") self.root.geometry("1390x700+0+0") root.configure(bg='#461257')

title = Label(self.root,text="Event management System",bd=9,relief=GROOVE,font=("times new roman",50,"bold"),bg='#461257',fg="#D7A1F9"title.pack(side=TOP,fill=X) backimg = Image.open("backB.png")

backimg = backimg.resize((35, 35), Image.ADAPTIVE) self.Photo\_img = ImageTk.PhotoImage(backimg)

buttonimg = Button(self.root, image=self.Photo\_img,command=self.backB, borderwidth=0, bg="#4DDCE6")

buttonimg.place(x=15, y=50, width=35, height=35) self.Customer\_ID\_var = StringVar()

self.Name\_var = StringVar()

self.Email\_ID\_var = StringVar() self.txt\_Address = StringVar() self.Contact\_number\_var = StringVar() self.Event\_name\_var = StringVar() self.Attended\_By\_var = StringVar() self.Search\_By = StringVar() self.search\_txt = StringVar()

Manage\_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="#D7A1F9") Manage\_Frame.place(x=28,y=100,width=650,height=650)

m\_title = Label(Manage\_Frame,text="Customer Details",bg="#D7A1F9",fg="black",font=("times new roman",50,"bold")) m\_title.pack(side=TOP,fill=X)

C\_ID = Label(root,text="Customer\_ID:",bg="#D7A1F9", fg="black",font=("times new roman",18,"bold")) C\_ID.place(x=100,y=250)

e1=Entry(root,textvariable=self.Customer\_ID\_var,font=('bold',13,'bold'), fg='black',width=30)

e1.place(x=300,y=250)

Name = Label(root,text="Name:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold")) Name.place(x=100,y=300)

e2=Entry(root,textvariable=self.Name\_var,font=('bold/Italic/red/GREEN\_BG', 13,'bold'),fg='black',width=30)

e2.place(x=300,y=300)

Email = Label(root,text="Email ID:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold")) Email.place(x=100,y=350) e3=Entry(root,textvariable=self.Email\_ID\_var,

font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='black',width=30) e3.place(x=300,y=350)

Address = Label(root,text="Address:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold")) Address.place(x=100,y=400)

self.txt\_Address=Text(root,width=30,height=3, font=('bold/Italic/red/GREEN\_BG',13,'bold')) self.txt\_Address.place(x=300,y=400)

Contact= Label(root,text="Contact number:",bg="#D7A1F9",fg="black",font=("times new roman",18,"bold")) Contact.place(x=100,y=500)

e5=Entry(root,textvariable=self.Contact\_number\_var, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='black',width=30) e5.place(x=300,y=500)

Event = Label(root,text="Event Name:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold"))

Event.place(x=100,y=550)

V=["wedding","Birthday","party","mehandi"] e6=ttk.Combobox(root,textvariable=self.Event\_name\_var,values=V,width=42) e6.place(x=300,y=550)

Attended = Label(root,text="Attended By:",bg="#D7A1F9",fg="black",font=("times new roman",18,"bold"))

Attended.place(x=100,y=600) e7=Entry(root,textvariable=self.Attended\_By\_var, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='black',width=30) e7.place(x=300,y=600)

button=Button(text="save",command=self.save\_customer, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3') button.place(x=160,y=650)

button=Button(text="Update",command=self.update\_data, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3') button.place(x=260,y=650)

button=Button(text="Delete",command=self.delete\_data, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3') button.place(x=360,y=650)

button=Button(text="Clear",command=self.clear\_customer, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3') button.place(x=460,y=650)

#==========================2ndFrame=====================#

Manage\_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="#D7A1F9") Manage\_Frame.place(x=720,y=100,width=800,height=650)

m\_title = Label(Manage\_Frame,

text="The Dream Maker",bg="#D7A1F9",fg="black", font=("times new roman",30,"bold")) m\_title.pack(side=TOP,fill=X)

m\_title = Label(Manage\_Frame,text="Here To make Your Special Days A Lil More Special",bg="#D7A1F9",fg='#461257',

font=("times new roman",20,"bold")) m\_title.pack(side=TOP,fill=X)

Name = Label(root,text="Search By:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold"))

Name.place(x=750,y=200)

#Name.place(x=100,y=450) V=["Customer\_ID","Name","Email\_ID","Address","Event Name"] combo=ttk.Combobox(root,textvariable=self.Search\_By ,values=V,width=20) combo.place(x=870,y=205)

NameEntry=Entry(root,textvariable=self.search\_txt, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='red',width=15) NameEntry.place(x=1030,y=205)

searchbtn = Button(text = 'Search', width=10,pady=5,command=self.search\_data).place(x=1190,y=205) showbtn = Button(text = 'Show All', width=10,pady=5,command=self.fetch\_data).place(x=1300,y=205)

Table\_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="pink") Table\_Frame.place(x=750,y=250,width=750,height=480) Scroll\_x = Scrollbar(Table\_Frame,orient=HORIZONTAL) Scroll\_Y = Scrollbar(Table\_Frame,orient=VERTICAL)

self.customer\_table = ttk.Treeview (Table\_Frame,

column =("Customer\_ID","Name","Email\_ID","Address","Contact Number", "Event name" ,

"Attended By"),xscrollcommand=Scroll\_Y.set,yscrollcommand=Scroll\_x.set) Scroll\_x.pack(side=BOTTOM,fill=X)

Scroll\_Y.pack(side=RIGHT,fill=Y)

Scroll\_x.config(command=self.customer\_table.xview) Scroll\_Y.config(command=self.customer\_table.yview)

# Rest of your code...

self.customer\_table.heading("Customer\_ID",text="Customer ID") self.customer\_table.heading("Name",text="Name") self.customer\_table.heading("Email\_ID",text="Email\_ID") self.customer\_table.heading("Address",text="Address") self.customer\_table.heading("Contact Number",text="Contact Number") self.customer\_table.heading("Event name",text="Event name") self.customer\_table.heading("Attended By",text="Attended By")

self.customer\_table['show']='headings' self.customer\_table.column("Customer\_ID", width=100) self.customer\_table.column("Name", width=100) self.customer\_table.column("Email\_ID",width=100)

self.customer\_table.column("Address", width=100) self.customer\_table.column("Contact Number", width=100) self.customer\_table.column("Event name", width=100) self.customer\_table.column("Attended By", width=100)

self.customer\_table.pack(fill=BOTH,expand=1) self.customer\_table.bind("<ButtonRelease-1>",self.get\_cursor) self.fetch\_data()

def validate\_contact\_number(self):

# Regular expression to match a 10-digit number pattern = r'^[0-9]{10}$'

if re.match(pattern, self.Contact\_number\_var.get()): return True

else:

return False

def validate\_email(self, email):

# Regular expression for basic email validation

pattern = r'^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$' if re.match(pattern, email):

return True else:

messagebox.showerror("Error", "Invalid email address.") return False

def save\_customer(self):

if (

self.Customer\_ID\_var.get() == "" or self.Name\_var.get() == ""

or self.Email\_ID\_var.get() == ""

or self.Contact\_number\_var.get() == "" or self.Event\_name\_var.get() == ""

or self.Attended\_By\_var.get() == ""

):

messagebox.showerror("Error", "All fields are required to fill")

elif not self.validate\_email(self.Email\_ID\_var.get()): return # Do nothing if email is invalid

else:

mysqldb = mysql.connector.connect(host='localhost', user='root', password='regal@123456789', database="sys") mycursor = mysqldb.cursor()

mycursor.execute(

"INSERT INTO customer\_details (Customer\_ID, Name, Email\_ID, Address, Contact\_number, Event\_name, Attended\_By) VALUES (%s, %s, %s, %s, %s, %s, %s)",

(

self.Customer\_ID\_var.get(), self.Name\_var.get(), self.Email\_ID\_var.get(), self.txt\_Address.get('1.0', END), self.Contact\_number\_var.get(), self.Event\_name\_var.get(), self.Attended\_By\_var.get()

)

)

mysqldb.commit() self.fetch\_data() self.clear\_customer() mysqldb.close()

messagebox.showinfo("Success", "Record has been inserted")

def fetch\_data(self):

mysqldb = mysql.connector.connect(host='localhost',user='root', password='regal@123456789',database="sys")

mycuror = mysqldb.cursor()

mycuror.execute("select \* from customer\_details") rows = mycuror.fetchall()

if len(rows)!=0: self.customer\_table.delete(\*self.customer\_table.get\_children()) for row in rows:

self.customer\_table.insert('',END,values=row) mysqldb.commit()

mysqldb.close()

def get\_cursor(self,ev):

curosor\_row = self.customer\_table.focus() contents = self.customer\_table.item(curosor\_row) row = contents['values'] self.Customer\_ID\_var.set(row[0]) self.Name\_var.set(row[1]) self.Email\_ID\_var.set(row[2]) self.txt\_Address.delete('1.0',END) self.txt\_Address.insert(END,row[3]) self.Contact\_number\_var.set(row[4]) self.Event\_name\_var.set(row[5]) self.Attended\_By\_var.set(row[6])

def clear\_customer(self): self.Customer\_ID\_var.set(" ") self.Name\_var.set(" ") self.Email\_ID\_var.set(" ") self.txt\_Address.delete('1.0',END) self.Contact\_number\_var.set(" ") self.Event\_name\_var.set(" ") self.Attended\_By\_var.set(" ")

def update\_data(self):

mysqldb = mysql.connector.connect(host='localhost',user='root', password='regal@123456789',database="sys")

mycuror = mysqldb.cursor()

mycuror.execute("update customer\_details

set Name=%s,Email\_ID=%s,Address=%s,Contact\_number=%s,Event\_name=%s, Attended\_By=%s where Customer\_ID= %s",

(self.Name\_var

.get(), self.Email\_ID\_var.get(),self.txt\_Address.get('1.0',END), self.Contact\_number\_var.get(), self.Event\_name\_var.get(),self.Attended\_By\_var.get(), self.Customer\_ID\_var.get()))

mysqldb.commit() self.fetch\_data() self.clear\_customer() mysqldb.close()

messagebox.showinfo("Success","Record has been updated") def delete\_data(self):

mysqldb = mysql.connector.connect(host='localhost', user='root', password='regal@123456789', database="sys") mycursor = mysqldb.cursor()

delete\_query = "DELETE FROM customer\_details WHERE Customer\_ID = %s" customer\_id = (self.Customer\_ID\_var.get(),) # Convert to a tuple mycursor.execute(delete\_query, customer\_id)

mysqldb.commit() mysqldb.close() self.fetch\_data() self.clear\_customer()

def search\_data(self):

mysqldb = mysql.connector.connect(host='localhost',user='root', password='regal@123456789',database="sys")

mycuror = mysqldb.cursor()

mycuror.execute("select \* from customer\_details where "

+ str(self.Search\_By.get()) +" Like '%"+str(self.search\_txt.get())+"%'") rows =mycuror.fetchall()

if len(rows) != 0: self.customer\_table.delete(\*self.customer\_table.get\_children()) for row in rows:

self.customer\_table.insert('', END,values=row) mysqldb.commit()

mysqldb.close() def backB(self):

self.root.destroy()

from mdi\_form import MDI\_form mdi\_win=Tk()

login\_app = MDI\_form(mdi\_win)

class customer(): pass

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

##### EVENT FORM

from tkinter import\* from tkinter import ttk import tkinter as tk

from tkinter import ttk, messagebox import mysql.connector

from PIL import Image ,ImageTk

class event():

def init (self,root): self.root=root

self.root.title("Event management system") self.root.geometry("1390x700+0+0") root.configure(bg='#461257')

title = Label(self.root,text="Event management System",bd=9,relief=GROOVE,

font=("times new roman",50,"bold"),bg='#461257',fg="#D7A1F9") title.pack(side=TOP,fill=X)

backimg = Image.open("backB.png")

backimg = backimg.resize((35, 35), Image.ADAPTIVE) self.Photo\_img = ImageTk.PhotoImage(backimg) buttonimg = Button(self.root, image=self.Photo\_img,

borderwidth=0, command=self.backB, bg="red") # Changed mdi to self.mdi buttonimg.place(x=15, y=50, width=35, height=35)

self.Event\_ID\_var = StringVar() self.Event\_Name\_var = StringVar() self.Estimated\_Cost = StringVar() self.Search\_By = StringVar() self.search\_txt = StringVar()

Manage\_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="#D7A1F9") Manage\_Frame.place(x=28,y=100,width=650,height=400)

m\_title = Label(Manage\_Frame,

text="Event Details",bg="#D7A1F9",fg="black", font=("times new roman",50,"bold")) m\_title.pack(side=TOP,fill=X)

E\_ID = Label(root,text="Event ID:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold")) E\_ID.place(x=100,y=250)

e1=Entry(root,textvariable=self.Event\_ID\_var,font=('bold',13,'bold'), fg='black',width=30)

e1.place(x=300,y=250)

Name = Label(root,text="Event Name:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold"))

Name.place(x=100,y=300) V=["wedding","Birthday","party","mehandi"]

e2=ttk.Combobox(root,textvariable=self.Event\_Name\_var,values=V,width=42) e2.place(x=300,y=300)

Cost = Label(root,text="Estimated Cost:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold"))

Cost.place(x=100,y=350) e3=Entry(root,textvariable=self.Estimated\_Cost, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='black',width=30) e3.place(x=300,y=350)

button=Button(text="save",command=self.add\_event, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3') button.place(x=160,y=400)

button=Button(text="Update",command=self.update\_data, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3') button.place(x=260,y=400)

button=Button(text="Delete",command=self.delete\_data, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3') button.place(x=360,y=400)

button=Button(text="Cancel",command=self.clear\_event, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3') button.place(x=460,y=400)

Manage\_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="#D7A1F9") Manage\_Frame.place(x=720,y=100,width=800,height=400)

m\_title = Label(Manage\_Frame,

text="The Dream Maker",bg="#D7A1F9",fg="black", font=("times new roman",30,"bold")) m\_title.pack(side=TOP,fill=X)

m\_title = Label(Manage\_Frame,text="Here To make Your Special Days A Lil More Special",bg="#D7A1F9",fg='#461257',

font=("times new roman",20,"bold")) m\_title.pack(side=TOP,fill=X)

Name = Label(root,text="Search By:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold"))

Name.place(x=750,y=200)

V=["Event\_ID","Event\_Name","Estimated\_Cost"] combo=ttk.Combobox(root,textvariable=self.Search\_By ,values=V,width=20) combo.place(x=870,y=205)

NameEntry=Entry(root,textvariable=self.search\_txt, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='black',width=15) NameEntry.place(x=1030,y=205)

searchbtn = Button(text = 'Search', width=10,pady=5, command=self.search\_data).place(x=1190,y=205)

showbtn = Button (text = 'Show All',

width=10,pady=5,command=self.fetch\_data).place(x=1300,y=205)

Table\_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="pink") Table\_Frame.place(x=750,y=250,width=750,height=240)

Scroll\_x = Scrollbar(Table\_Frame,orient=HORIZONTAL) Scroll\_Y = Scrollbar(Table\_Frame,orient=VERTICAL)

self.event\_table = ttk.Treeview(Table\_Frame,column=("Event\_ID","Event\_Name", "Estimated\_Cost"),xscrollcommand=Scroll\_Y.set,yscrollcommand=Scroll\_x.set) Scroll\_x.pack(side=BOTTOM,fill=X)

Scroll\_Y.pack(side=RIGHT,fill=Y)

Scroll\_x.config(command=self.event\_table.xview) Scroll\_Y.config(command=self.event\_table.yview)

self.event\_table.heading("Event\_ID",text="Event\_ID") self.event\_table.heading("Event\_Name",text="Event\_Name") self.event\_table.heading("Estimated\_Cost",text="Estimated\_Cost")

self.event\_table['show']='headings' self.event\_table.column("Event\_ID", width=100) self.event\_table.column("Event\_Name", width=100) self.event\_table.column("Estimated\_Cost",width=100) self.event\_table.pack(fill=BOTH,expand=1) self.event\_table.bind("<ButtonRelease-1>",self.get\_cursor) self.fetch\_data()

def add\_event(self):

if self.Event\_ID\_var.get()=="" or self.Event\_Name\_var.get()=="" or self.Estimated\_Cost.get()=="":

messagebox.showerror("error","all fields are required to fill") else:

mysqldb = mysql.connector.connect(host='localhost',user='root', password='regal@123456789',database="sys")

mycuror = mysqldb.cursor()

mycuror.execute("insert into event\_details values

(%s,%s,%s)", (self.Event\_ID\_var.get(), self.Event\_Name\_var.get(), self.Estimated\_Cost.get()))

mysqldb.commit() self.fetch\_data() self.clear\_event() mysqldb.close()

messagebox.showinfo("Success","Record has been inserted")

def fetch\_data(self):

mysqldb = mysql.connector.connect(host='localhost',user='root', password='regal@123456789',database="sys")

mycuror = mysqldb.cursor()

mycuror.execute("select \* from event\_details") rows = mycuror.fetchall()

if len(rows)!=0: self.event\_table.delete(\*self.event\_table.get\_children()) for row in rows:

self.event\_table.insert('',END,values=row) mysqldb.commit()

mysqldb.close()

def get\_cursor(self, ev):

cursor\_row = self.event\_table.focus() contents = self.event\_table.item(cursor\_row) row = contents['values']

if row: # Check if row is not empty

if len(row) >= 3: # Check if row has at least 3 elements self.Event\_ID\_var.set(row[0]) self.Event\_Name\_var.set(row[1]) self.Estimated\_Cost.set(row[2])

else:

messagebox.showerror("Error", "Selected row does not contain complete data")

else:

messagebox.showerror("Error", "No row selected")

def clear\_event(self): self.Event\_ID\_var.set("") self.Event\_Name\_var.set("") self.Estimated\_Cost.set("")

def update\_data(self):

if self.Event\_ID\_var.get()=="" or self.Event\_Name\_var.get()=="" or self.Estimated\_Cost.get()=="":

messagebox.showerror("error","fields are required to fill") else:

mysqldb = mysql.connector.connect(host='localhost',user='root', password='regal@123456789',

database="sys")

mycuror = mysqldb.cursor()

mycuror.execute("update event\_details set Event\_Name=%s,Estimated\_ Cost=%s where Event\_ID=%s",

(self.Event\_Name\_var.get(), self.Estimated\_Cost.get(), self.Event\_ID\_var.get()))

mysqldb.commit() self.fetch\_data() self.clear\_event()

mysqldb.close()

messagebox.showinfo("Success","Record has been updated")

def delete\_data(self):

event\_id = self.Event\_ID\_var.get() # Get the event ID if event\_id == "":

messagebox.showerror("Error", "Please select a record to delete") else:

mysqldb = mysql.connector.connect(host='localhost',user='root', password='regal@123456789',database="sys")

mycursor = mysqldb.cursor()

sql = "DELETE FROM event\_details WHERE Event\_ID = %s"

val = (event\_id,) # Note the comma to create a single-element tuple mycursor.execute(sql, val)

mysqldb.commit() self.fetch\_data() self.clear\_event() mysqldb.close()

messagebox.showinfo("Success", "Record has been deleted")

def search\_data(self):

mysqldb = mysql.connector.connect(host='localhost',user='root', password='regal@123456789',database="sys")

mycuror = mysqldb.cursor()

mycuror.execute("select \* from event\_details where "

+ str(self.Search\_By.get()) +" Like '%"+str(self.search\_txt.get())+"%'") rows =mycuror.fetchall()

if len(rows) != 0: self.event\_table.delete(\*self.event\_table.get\_children()) for row in rows:

self.event\_table.insert('', END,values=row) mysqldb.commit()

mysqldb.close()

def backB(self): self.root.destroy()

from mdi\_form import MDI\_form mdi\_win=Tk()

login\_app = MDI\_form(mdi\_win)

class event(): pass

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

1. **ORDER DETAILS FORM**

from tkinter import\* from tkinter import ttk import tkinter as tk

from tkinter import ttk, messagebox import mysql.connector

from PIL import Image ,ImageTk

from reportlab.lib.pagesizes import letter

from reportlab.platypus import SimpleDocTemplate, Table, TableStyle import os

class order():

def init (self,root): self.root=root

self.root.title("Event management system") self.root.geometry("1390x700+0+0") root.configure(bg='#461257')

title = Label(self.root,

text="Event management System",bd=9,relief=GROOVE, font=("times new roman",50,"bold"),bg='#461257',fg="#D7A1F9") title.pack(side=TOP,fill=X)

backimg = Image.open("backB.png")

backimg = backimg.resize((35, 35), Image.ADAPTIVE) self.Photo\_img = ImageTk.PhotoImage

(backimg)

buttonimg = Button(self.root, image=self.Photo\_img, borderwidth=0, command=self.backB, bg="#4DDCE6") buttonimg.place(x=15, y=50, width=35, height=35)

self.Order\_ID\_var = StringVar() self.Customer\_Name\_var = StringVar() self.Event\_Name\_var = StringVar() self.Venue\_var = StringVar() self.Date\_var = StringVar() self.Time\_var = StringVar() self.Cost\_var = StringVar() self.Advance\_var = StringVar() self.Balance\_var = StringVar() self.Search\_By = StringVar() self.search\_txt = StringVar()

Manage\_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="#D7A1F9") Manage\_Frame.place(x=28,y=100,width=650,height=650)

m\_title = Label(Manage\_Frame,

text="Order Details",bg="#D7A1F9",fg="black", font=("times new roman",50,"bold")) m\_title.pack(side=TOP,fill=X)

self.Order\_ID = Label(root,

text="Order ID:",bg="#D7A1F9",fg="black",font=("times new roman",18,"bold")) self. Order\_ID.place(x=100,y=250) self.e1=Entry(root,textvariable=self.Order\_ID\_var, font=('bold',13,'bold'),fg='black',width=30)

self.e1.place(x=300,y=250)

self.Name = Label(root, text="Customer Name:",bg="#D7A1F9",

fg="black",font=("times new roman",18,"bold")) self.Name.place(x=100,y=300) self.e2=Entry(root,textvariable=self.Customer\_Name\_var, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='black',width=30) self.e2.place(x=300,y=300)

self.EName = Label(root,

text="Event Name:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold")) self.EName.place(x=100,y=350) V=["wedding","Birthday","party","mehandi"]

e3=ttk.Combobox(root,textvariable=self.Event\_Name\_var,values=V,width=42) e3.place(x=300,y=350)

self.Venue = Label(root,text="Venue:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold")) self.Venue.place(x=100,y=400) self.e4=Entry(root,textvariable=self.Venue\_var, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='black',width=30) self.e4.place(x=300,y=400)

self.Date = Label(root,text="Date:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold"))

self. Date.place(x=100,y=450) self.e5=Entry(root,textvariable=self.Date\_var, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='black',width=30) self.e5.place(x=300,y=450)

self.Time = Label(root,text="Time:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold")) self.Time.place(x=100,y=500) self.e6=Entry(root,textvariable=self.Time\_var, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='black',width=30) self.e6.place(x=300,y=500)

self.Cost = Label(root,text="Cost:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold")) self.Cost.place(x=100,y=550) self.e7=Entry(root,textvariable=self.Cost\_var, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='black',width=30) self.e7.place(x=300,y=550)

self.Advance = Label(root,text="Advance:",bg="#D7A1F9", fg="black",font=("times new roman",18,"bold")) self.Advance.place(x=100,y=600) self.e8=Entry(root,textvariable=self.Advance\_var, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='black',width=30) self.e8.place(x=300,y=600)

self.Balace = Label(root,text="Balance:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold")) self.Balace.place(x=100,y=650) self.e9=Entry(root,textvariable=self.Balance\_var, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='black',width=30) self.e9.place(x=300,y=650)

button=Button(text="Save",command=self.add\_order, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3') button.place(x=60,y=700)

button=Button(text="Update",command=self.update\_data, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3') button.place(x=160,y=700)

button=Button(text="Delete",command=self.delete\_data, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3') button.place(x=260,y=700)

button=Button(text="Cancel",command=self.clear\_order, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3') button.place(x=360,y=700)

button=Button(text="Print",command=self.print, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3') button.place(x=460,y=700)

Manage\_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="#D7A1F9") Manage\_Frame.place(x=720,y=100,width=800,height=650)

m\_title = Label(Manage\_Frame,text=

"The Dream Maker",bg="#D7A1F9",fg="black",font=("times new roman",30,"bold")) m\_title.pack(side=TOP,fill=X)

m\_title = Label(Manage\_Frame,text="Here To make Your Special Days A Lil More Special",bg="#D7A1F9",fg="#461257",

font=("times new roman",20,"bold")) m\_title.pack(side=TOP,fill=X)

Name = Label(root,text="Search By:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold"))

Name.place(x=750,y=200)

#Name.place(x=100,y=450) V=["Order\_ID","Customer\_Name","Event\_Name","Venue","Date", "Time","Cost","Advance","Balance"] combo=ttk.Combobox(root,textvariable=self.Search\_By ,values=V,width=20) combo.place(x=870,y=205)

NameEntry=Entry(root,textvariable=self.search\_txt, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='red',width=15) NameEntry.place(x=1030,y=205)

searchbtn = Button(text =

'Search', width=10,pady=5,command=self.search\_data).place(x=1190,y=205) showbtn = Button(text =

'Show All', width=10,pady=5,command=self.fetch\_data).place(x=1300,y=205)

Table\_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="#D7A1F9") Table\_Frame.place(x=750,y=250,width=750,height=480)

Scroll\_x = Scrollbar(Table\_Frame,orient=HORIZONTAL) Scroll\_Y = Scrollbar(Table\_Frame,orient=VERTICAL)

self.order\_table = ttk.Treeview(Table\_Frame,column=("Order\_ID","Customer\_Name","Event\_Name",

"Venue","Date","Time", "Cost",

"Advance", "Balance"),xscrollcommand=Scroll\_Y.set,yscrollcommand=Scroll\_x.set) Scroll\_x.pack(side=BOTTOM,fill=X)

Scroll\_Y.pack(side=RIGHT,fill=Y)

Scroll\_x.config(command=self.order\_table.xview) Scroll\_Y.config(command=self.order\_table.yview)

self.order\_table.heading("Order\_ID",text="Order\_ID") self.order\_table.heading("Customer\_Name",text="Customer\_Name") self.order\_table.heading("Event\_Name",text="Event\_Name") self.order\_table.heading("Venue",text="Venue") self.order\_table.heading("Date",text="Date") self.order\_table.heading("Time",text="Time") self.order\_table.heading("Cost",text="cost") self.order\_table.heading("Advance",text="Advance") self.order\_table.heading("Balance",text="Balance") self.order\_table['show']='headings' self.order\_table.column("Order\_ID", width=100) self.order\_table.column("Customer\_Name", width=100) self.order\_table.column("Event\_Name",width=100)

self.order\_table.column("Venue", width=100) self.order\_table.column("Date", width=100) self.order\_table.column("Time", width=100) self.order\_table.column("Cost", width=100) self.order\_table.column("Advance", width=100) self.order\_table.column("Balance", width=100)

self.order\_table.pack(fill=BOTH,expand=1) self.order\_table.bind("<ButtonRelease-1>",self.get\_cursor) self.fetch\_data()

def add\_order(self): if (

self.Order\_ID\_var.get() == ""

or self.Customer\_Name\_var.get() == "" or self.Event\_Name\_var.get() == ""

or self.Venue\_var.get() == "" or self.Date\_var.get() == "" or self.Cost\_var.get() == ""

or self.Advance\_var.get() == "" or self.Balance\_var.get() == ""

):

messagebox.showerror("Error", "All fields are required to fill") else:

try:

Order\_ID = int(self.Order\_ID\_var.get()) except ValueError:

messagebox.showerror("Error", "Order ID must be an integer") return

if self.order\_id\_exists(Order\_ID): messagebox.showerror("Error", "Order ID already exists") return

try:

mysqldb = mysql.connector.connect(host='localhost', user='root', password='regal@123456789', database="sys") mycursor = mysqldb.cursor()

mycursor.execute(

"INSERT INTO order1\_details (Order\_ID, Customer\_Name, sEvent\_Name, Venue, Date, Time, Cost, Advance, Balance) " "VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)",

(

self.Order\_ID\_var.get(), self.Customer\_Name\_var.get(), self.Event\_Name\_var.get(), self.Venue\_var.get(), self.Date\_var.get(), self.Time\_var.get(), self.Cost\_var.get(), self.Advance\_var.get(), self.Balance\_var.get(),

),

)

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

self.clear\_order() mysqldb.close()

messagebox.showinfo("Success", "Record has been inserted") except Exception as e:

messagebox.showerror("Error", str(e)) def order\_id\_exists(self, order\_id):

# Check if Order\_ID exists in the database

mysqldb = mysql.connector.connect(host='localhost', user='root', password='regal@123456789', database="sys") mycursor = mysqldb.cursor()

mycursor.execute("SELECT \* FROM order1\_details WHERE Order\_ID =

%s", (order\_id,))

result = mycursor.fetchone() mysqldb.close()

return result is not None pass

def fetch\_data(self):

mysqldb = mysql.connector.connect(host='localhost',user='root', password='regal@123456789',database="sys")

mycuror = mysqldb.cursor() mycuror.execute("select \* from order1\_details")

rows = mycuror.fetchall() if len(rows)!=0:

self.order\_table.delete(\*self.order\_table.get\_children()) for row in rows:

self.order\_table.insert('',END,values=row) mysqldb.commit()

mysqldb.close()

def get\_cursor(self, ev):

cursor\_row = self.order\_table.focus() contents = self.order\_table.item(cursor\_row) row = contents.get('values')

if row:

self.Order\_ID\_var.set(row[0]) self.Customer\_Name\_var.set(row[1]) self.Event\_Name\_var.set(row[2]) self.Venue\_var.set(row[3]) self.Date\_var.set(row[4]) self.Time\_var.set(row[5]) self.Cost\_var.set(row[6]) self.Advance\_var.set(row[7]) self.Balance\_var.set(row[8])

def clear\_order(self): self.Order\_ID\_var.set(" ") self.Customer\_Name\_var.set(" ")

self.Event\_Name\_var.set(" ") self.Venue\_var.set(" ") self.Date\_var.set(" ") self.Time\_var.set(" ") self.Cost\_var.set(" ") self.Advance\_var.set(" ") self.Balance\_var.set(" ")

def update\_data(self):

mysqldb = mysql.connector.connect(host='localhost',user='root', password='regal@123456789',database="sys")

mycursor = mysqldb.cursor()

mycursor.execute("UPDATE order1\_details SET Customer\_Name=%s, Event\_Name=%s, Venue=%s, Date=%s, Time=%s, Cost=%s, Advance=%s, Balance=%s WHERE Order\_ID=%s",

(self.Customer\_Name\_var.get(), self.Event\_Name\_var.get(), self.Venue\_var.get(), self.Date\_var.get(), self.Time\_var.get(), self.Cost\_var.get(), self.Advance\_var.get(), self.Balance\_var.get(),self.Order\_ID\_var.get()))

mysqldb.commit() self.fetch\_data() self.clear\_order() mysqldb.close()

messagebox.showinfo("Success", "Record has been updated")

def delete\_data(self):

order\_id = (self.Order\_ID\_var.get(),)

mysqldb = mysql.connector.connect(host='localhost',user='root', password='regal@123456789',database="sys")

mycuror = mysqldb.cursor()

mycuror.execute("DELETE FROM order1\_details WHERE Order\_ID=%s", order\_id) mysqldb.commit()

self.fetch\_data() self.clear\_order() mysqldb.close()

def search\_data(self):

mysqldb = mysql.connector.connect(host='localhost',user='root', password='regal@123456789',database="sys")

mycuror = mysqldb.cursor()

mycuror.execute("select \* from order1\_details where "

+ str(self.Search\_By.get()) +" Like '%"+str(self.search\_txt.get())+"%'") rows =mycuror.fetchall()

if len(rows) != 0: self.order\_table.delete(\*self.order\_table.get\_children()) for row in rows:

self.order\_table.insert('', END,values=row) mysqldb.commit()

mysqldb.close()

def backB(self): self.root.destroy()

from mdi\_form import MDI\_form mdi\_win=Tk()

login\_app = MDI\_form(mdi\_win)

def print(self):

Order\_ID = self.Order\_ID\_var.get() if not Order\_ID:

messagebox.showerror("Error", "Order ID cannot be empty") return

try:

Order\_ID = int(Order\_ID) except ValueError:

messagebox.showerror("Error", "Order ID must be an integer") return

Order\_ID = self.Order\_ID\_var.get() Customer\_Name = self.Customer\_Name\_var.get() Event\_Name = self.Event\_Name\_var.get() Venue\_var = self.Venue\_var.get()

Date\_var = self.Date\_var.get() Time\_var = self.Time\_var.get() Cost\_var = self.Cost\_var.get() Advance\_var = self.Advance\_var.get() Balance\_var = self.Balance\_var.get()

mysqldb = mysql.connector.connect(host='localhost', user='root', password='regal@123456789', database="sys") mycursor = mysqldb.cursor()

query = "INSERT INTO order1\_details

(Order\_ID, Customer\_Name,Event\_Name,Venue,Date,Time, Cost,Advance,Balance) VALUES (%s, %s,%s, %s,%s, %s,%s, %s,%s)" values = (Order\_ID, Customer\_Name, Event\_Name, Venue\_var, Date\_var, Time\_var, Cost\_var, Advance\_var, Balance\_var) mycursor.execute(query, values)

mysqldb.commit() mysqldb.close()

print("Data submitted successfully.") self.generate\_pdf\_report()

def generate\_pdf\_report(self):

mysqldb = mysql.connector.connect(host='localhost', user='root', password='regal@123456789', database="sys") mycursor = mysqldb.cursor()

query = "SELECT \* FROM order1\_details"

mycursor.execute(query) data = mycursor.fetchall() mysqldb.close()

doc = SimpleDocTemplate("report.pdf", pagesize=letter) report\_data = [["Order\_ID", "Customer\_Name", "Event\_Name", "Venue", "Date", "Time", "Cost", "Advance", "Balance"]] # Header report\_data.extend(data) # Data from MySQL

|  |  |  |
| --- | --- | --- |
| table = Table(report\_data) |  | |
| table.setStyle(TableStyle([('BACKGROUND', | (0, | 0), (-1, 0), (0.8, 0.8, 0.8)), |
| ('TEXTCOLOR', | (0, | 0), (-1, 0), (0, 0, 0)), |
| ('ALIGN', (0, | 0), | (-1, -1), 'CENTER'), |

('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),

('BOTTOMPADDING', (0, 0), (-1, 0), 12),

('BACKGROUND', (0, 1), (-1, -1),

(0.8, 0.8, 0.8)),

('GRID', (0, 0), (-1, -1), 1, (0, 0, 0))]))

doc.build([table])

print("PDF report generated successfully.") os.startfile("report.pdf")

def main():

root = tk.Tk() app = order(root) root.mainloop()

if name == " main ": main()

##### 6 .EMPLOYEE DETAILS FORM

#from pydoc import render\_doc from tkinter import\*

from tkinter import ttk import tkinter as tk

from tkinter import ttk, messagebox import mysql.connector

import re

from PIL import Image ,ImageTk

class employee():

def init (self,root): self.root=root

self.root.title("Event management system") self.root.geometry("1390x700+0+0") root.configure(bg='#461257')

title = Label(self.root,text="Event management System",bd=9,relief=GROOVE,

font=("times new roman",50,"bold"),bg='#461257',fg="#D7A1F9") title.pack(side=TOP,fill=X)

backimg = Image.open("backB.png")

backimg = backimg.resize((35, 35), Image.ADAPTIVE) self.Photo\_img = ImageTk.PhotoImage(backimg) buttonimg = Button(self.root, image=self.Photo\_img, borderwidth=0, command=self.backB, bg="#4DDCE6") buttonimg.place(x=15, y=50, width=35, height=35)

self.Employee\_ID\_var=StringVar() self.Name\_var=StringVar() self.Email\_ID\_var=StringVar()

self.Contact\_number\_var=StringVar() self.Dob\_var=StringVar() self.Joining\_Date\_var=StringVar() self.Salary\_var=StringVar() self.Aadhar\_Number\_var=StringVar() self.Gender\_var=StringVar() self.txt\_Address=StringVar()

self.Search\_By = StringVar() self.search\_txt = StringVar()

Manage\_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="#D7A1F9") Manage\_Frame.place(x=28,y=100,width=650,height=650)

m\_title = Label(Manage\_Frame,

text="Employee Details",bg="#D7A1F9",fg="black", font=("times new roman",30,"bold"))

m\_title.pack(side=TOP,fill=X)

E\_ID = Label(root,text="Employee\_ID:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold"))

E\_ID.place(x=100,y=200) e1=Entry(root,textvariable=self.Employee\_ID\_var,font=('bold',13,'bold'), fg='black',width=30)

e1.place(x=300,y=200)

Name = Label(root,text="Name:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold")) Name.place(x=100,y=250) e2=Entry(root,textvariable=self.Name\_var,

font=('bold/Italic/red/GREEN\_BG',13, 'bold'),fg='black',width=30) e2.place(x=300,y=250)

Email = Label(root,text="Email\_ID:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold")) Email.place(x=100,y=300) e10=Entry(root,textvariable=self.Email\_ID\_var,

font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='blue',width=30) e10.place(x=300,y=300)

Contact = Label

(root,text="Contact Number:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold")) Contact.place(x=100,y=350) e4=Entry(root,textvariable=self.Contact\_number\_var,

font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='black',width=30) e4.place(x=300,y=350)

DOB = Label(root,text="DOB:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold")) DOB.place(x=100,y=400) e3=Entry(root,textvariable=self.Dob\_var,

font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='black',width=30) e3.place(x=300,y=400)

Joining = Label(root,

text="Joining Date:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold")) Joining.place(x=100,y=450) e6=Entry(root,textvariable=self.Joining\_Date\_var,

font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='black',width=30) e6.place(x=300,y=450)

Salary = Label(root,text="Salary:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold")) Salary.place(x=100,y=500) e7=Entry(root,textvariable=self.Salary\_var,

font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='black',width=30) e7.place(x=300,y=500)

Aadhar = Label(root,text="Aadhar Number:",bg="#D7A1F9", fg="black",font=("times new roman",18,"bold")) Aadhar.place(x=100,y=550) e8=Entry(root,textvariable=self.Aadhar\_Number\_var, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='black',width=30) e8.place(x=300,y=550)

Gender = Label(root,text="Gender:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold")) Gender.place(x=100,y=600) e9=Entry(root,textvariable=self.Gender\_var,

font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='black',width=30) e9.place(x=300,y=600)

Address = Label(root,text="Address:",bg="#D7A1F9",fg="black",

font=("times new roman",18,"bold")) Address.place(x=100,y=650) self.txt\_Address=Text(root,width=30,height=2, font=('bold/Italic/red/GREEN\_BG',13,'bold')) self.txt\_Address.place(x=300,y=650)

button=Button(text="save",command=self.save\_employee, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3') button.place(x=160,y=700)

button=Button(text="Update",command=self.update\_data, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3') button.place(x=260,y=700)

button=Button(text="Delete",command=self.delete\_data, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3') button.place(x=360,y=700)

button=Button(text="Cancel",command=self.clear\_employee, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3') button.place(x=460,y=700)

Manage\_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="#D7A1F9") Manage\_Frame.place(x=720,y=100,width=800,height=650)

m\_title = Label(Manage\_Frame,

text="The Dream Maker",bg="#D7A1F9",fg="black", font=("times new roman",30,"bold")) m\_title.pack(side=TOP,fill=X)

m\_title = Label(Manage\_Frame,text="Here To make Your Special Days A Lil More Special",bg="#D7A1F9",fg='#461257',

font=("times new roman",20,"bold"))

m\_title.pack(side=TOP,fill=X)

Name = Label(root,text="Search By:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold"))

Name.place(x=750,y=200)

#Name.place(x=100,y=450)

V=["Employee\_ID","Name","Email ID","Address","Event Name"] combo=ttk.Combobox(root,textvariable=self.Search\_By ,values=V,width=20) combo.place(x=870,y=205)

NameEntry=Entry(root,textvariable=self.search\_txt, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='red',width=15) NameEntry.place(x=1030,y=205)

searchbtn = Button(text = 'Search', width=10,pady=5,command=self.search\_data).place(x=1190,y=205)

showbtn = Button(text = 'Show All', width=10,pady=5,command=self.fetch\_data).place(x=1300,y=205)

Table\_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="pink") Table\_Frame.place(x=750,y=250,width=750,height=480)

Scroll\_x = Scrollbar(Table\_Frame,orient=HORIZONTAL) Scroll\_Y = Scrollbar(Table\_Frame,orient=VERTICAL)

self.employee\_table = ttk.Treeview(Table\_Frame,column=("Employee\_ID","Name","Email\_ID","Contact\_Number","DO B","Joinning\_Date" ,"Salary","Aadhar\_Number","Gender", "Address"),xscrollcommand=Scroll\_Y.set,yscrollcommand=Scroll\_x.set)

Scroll\_x.pack(side=BOTTOM,fill=X) Scroll\_Y.pack(side=RIGHT,fill=Y)

Scroll\_x.config(command=self.employee\_table.xview) Scroll\_Y.config(command=self.employee\_table.yview)

self.employee\_table.heading("Employee\_ID",text="Employee\_ID") self.employee\_table.heading("Name",text="Name") self.employee\_table.heading("Email\_ID",text="Email\_ID")

self.employee\_table.heading("Contact\_Number",text="Contact\_Number") self.employee\_table.heading("DOB",text="DOB") self.employee\_table.heading("Joinning\_Date",text="Joinning\_Date") self.employee\_table.heading("Salary",text="Salary") self.employee\_table.heading("Aadhar\_Number",text="Aadhar\_Number") self.employee\_table.heading("Gender",text="Gender")

self.employee\_table.heading("Address",text="Address")

self.employee\_table['show']='headings' self.employee\_table.column("Employee\_ID", width=100) self.employee\_table.column("Name", width=100)

self.employee\_table.column("Email\_ID", width=100) self.employee\_table.column("Contact\_Number", width=100) self.employee\_table.column("DOB",width=100) self.employee\_table.column("Joinning\_Date", width=100) self.employee\_table.column("Salary", width=100) self.employee\_table.column("Aadhar\_Number", width=100) self.employee\_table.column("Gender", width=100) self.employee\_table.column("Address", width=100) self.employee\_table.pack(fill=BOTH,expand=1) self.employee\_table.bind("<ButtonRelease-1>",self.get\_cursor) self.fetch\_data()

def validate\_contact\_number(self): pattern = r'^[0-9]{10}$'

if re.match(pattern, self.Contact\_number\_var.get()): return True

else:

return False

def validate\_email(self, email):

# Regular expression for basic email validation

pattern = r'^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$' if re.match(pattern, email):

return True else:

messagebox.showerror("Error", "Invalid email address.") return False

def save\_employee(self): if (

self.Employee\_ID\_var.get() == "" or self.Name\_var.get() == ""

or self.Email\_ID\_var.get() == ""

or self.Contact\_number\_var.get() == "" or self.Dob\_var.get() == ""

or self.Joining\_Date\_var.get() == "" or self.Salary\_var.get() == ""

or self.Aadhar\_Number\_var.get() == "" or self.Gender\_var.get() == ""

):

messagebox.showerror("Error", "All fields are required to fill") elif not self.validate\_email(self.Email\_ID\_var.get()):

return # Do nothing if email is invalid

else:

if self.validate\_contact\_number(): print("Valid contact number")

else:

print("Invalid contact number")

mysqldb = mysql.connector.connect(host='localhost', user='root', password='regal@123456789', database="sys") mycursor = mysqldb.cursor()

mycursor.execute("INSERT INTO employee1\_details (Employee\_ID, Name, Email\_ID, Address, Contact\_number, Dob, Joining\_Date, Salary, Aadhar\_Number, Gender) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)",

(self.Employee\_ID\_var.get(), self.Name\_var.get(), self.Email\_ID\_var.get(),self.txt\_Address.get('1.0', END),self.Contact\_number\_var.get(),self.Dob\_var.get(),self.Joining\_Date\_var.get(),

self.Salary\_var.get(),self.Aadhar\_Number\_var.get(),self.Gender\_va

r.get()))

mysqldb.commit() self.fetch\_data() self.clear\_employee() mysqldb.close()

messagebox.showinfo("Success", "Record has been inserted")

def fetch\_data(self): mysqldb =

mysql.connector.connect(host='localhost',user='root',password='regal@123456789',datab ase="sys")

mycuror = mysqldb.cursor()

mycuror.execute("select \* from employee1\_details") rows = mycuror.fetchall()

if len(rows)!=0:

self.employee\_table.delete(\*self.employee\_table.get\_children()) for row in rows:

self.employee\_table.insert('',END,values=row) mysqldb.commit()

mysqldb.close()

def get\_cursor(self,ev):

curosor\_row = self.employee\_table.focus() contents = self.employee\_table.item(curosor\_row) row = contents['values'] self.Employee\_ID\_var.set(row[0]) self.Name\_var.set(row[1]) self.Email\_ID\_var.set(row[2])

self.Contact\_number\_var.set(row[3]) self.Dob\_var.set(row[4]) self.Joining\_Date\_var.set(row[5]) self.Salary\_var.set(row[6]) self.Aadhar\_Number\_var.set(row[7]) self.Gender\_var.set(row[8]) self.txt\_Address.delete('1.0',END) self.txt\_Address.insert(END,row[9])

def clear\_employee(self): self.Employee\_ID\_var.set("") self.Name\_var.set(" ") self.Email\_ID\_var.set(" ")

self.Contact\_number\_var.set(" ") self.Dob\_var.set(" ") self.Joining\_Date\_var.set(" ") self.Salary\_var.set(" ") self.Aadhar\_Number\_var.set(" ") self.Gender\_var.set(" ") self.txt\_Address.delete('1.0',END)

def update\_data(self):

mysqldb = mysql.connector.connect(host='localhost',user='root', password='regal@123456789',database="sys")

mycuror = mysqldb.cursor()

mycuror.execute("update employee1\_details

set Name=%s,Email\_ID=%s,Contact\_number=%s,DOB=%s, Joining\_Date=%s,

Salary=%s,Aadhar\_Number=%s,Gender=%s,Address=%s where Employee\_ID=

%s",(self.Name\_var.get(),self.Email\_ID\_var.get(),self.Contact\_number\_var.get(),self.D ob\_var.get(),self.Joining\_Date\_var.get(),self.Salary\_var.get(),self.Aadhar\_Number\_var

.get(),self.Gender\_var.get(),self.txt\_Address.get('1.0',END),self.Employee\_ID\_var.get ()))

mysqldb.commit() self.fetch\_data() self.clear\_employee() mysqldb.close()

messagebox.showinfo("Success","Record has been inserted")

def delete\_data(self):

mysqldb = mysql.connector.connect(host='localhost',user='root', password='regal@123456789',database="sys")

mycuror = mysqldb.cursor()

mycuror.execute("delete from employee1\_details where Employee\_ID=%s", (self.Employee\_ID\_var.get(),))

mysqldb.commit() mysqldb.close() self.fetch\_data() self.clear\_employee()

def search\_data(self):

mysqldb = mysql.connector.connect(host='localhost',user='root', password='regal@123456789',database="sys")

mycuror = mysqldb.cursor()

mycuror.execute("select \* from employee1\_details where "

+ str(self.Search\_By.get()) +" Like '%"+str(self.search\_txt.get())+"%'") rows =mycuror.fetchall()

if len(rows) != 0: self.employee\_table.delete(\*self.employee\_table.get\_children()) for row in rows:

self.employee\_table.insert('', END,values=row) mysqldb.commit()

mysqldb.close()

def backB(self): self.root.destroy()

from mdi\_form import MDI\_form mdi\_win=Tk()

login\_app = MDI\_form(mdi\_win)

class employee(): pass

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

##### 7.STOCK DETAILS FORM

#from pydoc import render\_doc from tkinter import\*

from tkinter import ttk import tkinter as tk

from tkinter import ttk, messagebox import mysql.connector

from PIL import Image ,ImageTk

class stock():

def init (self,root): self.root=root

self.root.title("Event management system") self.root.geometry("1390x700+0+0") root.configure(bg='#461257')

title = Label(self.root,

text="Event management System",bd=9,relief=GROOVE, font=("times new roman",50,"bold"),bg='#461257',fg="#D7A1F9") title.pack(side=TOP,fill=X)

backimg = Image.open("backB.png")

backimg = backimg.resize((35, 35), Image.ADAPTIVE) self.Photo\_img = ImageTk.PhotoImage(backimg) buttonimg = Button(self.root, image=self.Photo\_img,

command=self.backB, borderwidth=0, bg="#4DDCE6") # Changed mdi to self.mdi buttonimg.place(x=15, y=50, width=35, height=35)

def validate\_Item\_ID(id): if id.isdigit():

return True else:

messagebox.showerror('Invalid','Invalid Item Entry') return False

def Item\_Name(name): if name.isalpha():

return True else:

messagebox.showerror('Invalid', 'Invalid Item Name Entry') return False

def Availibility(name): if id.isalpha():

return True else:

messagebox.showerror('Invalid','Invalid Avalibility Entry') return False

self.Item\_ID\_var = StringVar() self.Item\_Name\_var = StringVar() self.Availibility\_var = StringVar() self.Search\_By = StringVar() self.search\_txt = StringVar()

Manage\_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="#D7A1F9") Manage\_Frame.place(x=28,y=100,width=650,height=650)

m\_title = Label(Manage\_Frame,

text="Stock Details",bg="#D7A1F9",fg="black", font=("times new roman",50,"bold")) m\_title.pack(side=TOP,fill=X)

Item\_ID = Label(root,text="ItemID:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold")) Item\_ID.place(x=100,y=250)

e1=Entry(root,textvariable=self.Item\_ID\_var,

font=('bold',13,'bold'),fg='black',width=30) e1.place(x=300,y=250)

validate\_id =self.root.register(Item\_ID)

e1.config(validate = 'key',validatecommand=(validate\_id,'%s'))

Item\_Name = Label(root,text="Item\_Name:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold"))

Item\_Name.place(x=100,y=300) e2=Entry(root,textvariable=self.Item\_Name\_var, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='black',width=30) e2.place(x=300,y=300)

validate\_name =self.root.register(Item\_Name)

e1.config(validate = 'key',validatecommand=(validate\_name,'%p'))

Availibility = Label(root,text="Availibility:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold"))

Availibility.place(x=100,y=350) e3=Entry(root,textvariable=self.Availibility\_var, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='black',width=30) e3.place(x=300,y=350)

validate\_Availibility =self.root.register(Availibility) e1.config(validate = 'key',validatecommand=(validate\_Availibility,'%p'))

button=Button(text="save",command=self.add\_stock, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3') button.place(x=160,y=400)

button=Button(text="Update",command=self.update\_data, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3') button.place(x=260,y=400)

button=Button(text="Delete",command=self.delete\_data, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3') button.place(x=360,y=400)

button=Button(text="Cancel",command=self.clear\_stock, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3') button.place(x=460,y=400)

Manage\_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="#D7A1F9") Manage\_Frame.place(x=720,y=100,width=800,height=650)

m\_title = Label(Manage\_Frame,

text="The Dream Maker",bg="#D7A1F9",fg="black", font=("times new roman",30,"bold")) m\_title.pack(side=TOP,fill=X)

m\_title = Label(Manage\_Frame,text="Here To make Your Special Days A Lil More Special",bg="#D7A1F9",fg='#461257',

font=("times new roman",20,"bold"))

m\_title.pack(side=TOP,fill=X)

Name = Label(root,text="Search\_By:",bg="#D7A1F9",fg="black", font=("times new roman",18,"bold"))

Name.place(x=750,y=200)

V=["Item\_ID","Item\_Name","Availibility"] combo=ttk.Combobox(root,textvariable=self.Search\_By , values=V,width=20)

combo.place(x=870,y=205)

NameEntry=Entry(root,textvariable=self.search\_txt, font=('bold/Italic/red/GREEN\_BG',13,'bold'),fg='red',width=15) NameEntry.place(x=1030,y=205)

searchbtn = Button

(text = 'Search',width=10,pady=5,command=self.search\_data).place(x=1190,y=205)

showbtn = Button(text = 'Show All', width=10,pady=5,command=self.fetch\_data).place(x=1300,y=205)

Table\_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="pink") Table\_Frame.place(x=750,y=250,width=750,height=480)

Scroll\_x = Scrollbar(Table\_Frame,orient=HORIZONTAL) Scroll\_Y = Scrollbar(Table\_Frame,orient=VERTICAL)

self.stock\_table = ttk.Treeview(Table\_Frame,column=("Item\_ID","Item\_Name","Availibility"),xscrollcommand

=Scroll\_Y.set,yscrollcommand=Scroll\_x.set) Scroll\_x.pack(side=BOTTOM,fill=X) Scroll\_Y.pack(side=RIGHT,fill=Y)

Scroll\_x.config(command=self.stock\_table.xview) Scroll\_Y.config(command=self.stock\_table.yview)

self.stock\_table.heading("Item\_ID",text="Item\_ID") self.stock\_table.heading("Item\_Name",text="Item\_Name") self.stock\_table.heading("Availibility",text="Availibility")

self.stock\_table['show']='headings' self.stock\_table.column("Item\_ID", width=100) self.stock\_table.column("Item\_Name", width=100) self.stock\_table.column("Availibility",width=100)

self.stock\_table.pack(fill=BOTH,expand=1) self.stock\_table.bind("<ButtonRelease-1>",self.get\_cursor) self.fetch\_data()

def add\_stock(self):

if self.Item\_ID\_var.get()=="" or self.Item\_Name\_var.get()=="" or self.Availibility\_var.get()=="" :

messagebox.showerror("error","all fields are required to fill") else:

mysqldb = mysql.connector.connect(host='localhost',user='root', password='regal@123456789',database="sys")

mycuror = mysqldb.cursor() mycuror.execute("insert

into stock\_details values(%s,%s,%s)", (self.Item\_ID\_var.get (self.Item\_Name\_var.get(), self.Availibility\_var.get()

))

mysqldb.commit() self.fetch\_data() self.clear\_stock()

mysqldb.close()

messagebox.showinfo("Success","Record has been inserted")

def fetch\_data(self):

mysqldb = mysql.connector.connect(host='localhost',user='root', password='regal@123456789',database="sys")

mycuror = mysqldb.cursor()

mycuror.execute("select \* from stock\_details") rows = mycuror.fetchall()

if len(rows)!=0: self.stock\_table.delete(\*self.stock\_table.get\_children()) for row in rows:

self.stock\_table.insert('',END,values=row) mysqldb.commit()

mysqldb.close()

def get\_cursor(self, ev):

cursor\_row = self.stock\_table.focus() contents = self.stock\_table.item(cursor\_row) row = contents['values']

if row: self.Item\_ID\_var.set(row[0]) self.Item\_Name\_var.set(row[1]) self.Availibility\_var.set(row[2])

self.Item\_ID\_var.set(row[0]) self.Item\_Name\_var.set(row[1]) self.Availibility\_var.set(row[2])

def clear\_stock(self): self.Item\_ID\_var.set("") self.Item\_Name\_var.set(" ") self.Availibility\_var.set(" ")

def update\_data(self):

mysqldb = mysql.connector.connect(host='localhost',user='root', password='regal@123456789',database="sys")

mycuror = mysqldb.cursor()

mycuror.execute("update stock\_details set Item\_Name=%s, Availibility=%s where

Item\_ID= %s",(self.Item\_Name\_var.get(),self.Availibility\_var.get(), self.Item\_ID\_var.get()))

mysqldb.commit() self.fetch\_data() self.clear\_stock() mysqldb.close()

messagebox.showinfo("Success","Record has been inserted") def delete\_data(self):

mysqldb = mysql.connector.connect(host='localhost',user='root', password='regal@123456789',database="sys")

mycuror = mysqldb.cursor() mycuror.execute("delete from stock\_details where Item\_ID=%s", (self.Item\_ID\_var.get(),)) mysqldb.commit()

self.fetch\_data() self.clear\_stock() mysqldb.close()

def search\_data(self):

mysqldb = mysql.connector.connect(host='localhost', user='root', password='regal@123456789', database="sys") mycursor = mysqldb.cursor()

search\_column = self.Search\_By.get() search\_value = self.search\_txt.get()

if search\_column and search\_value:

sql\_query = "SELECT \* FROM stock\_details WHERE {} LIKE %s".format(search\_column)

mycursor.execute(sql\_query, ('%' + search\_value + '%',)) rows = mycursor.fetchall()

if len(rows) != 0:

self.stock\_table.delete(\*self.stock\_table.get\_children()) for row in rows:

self.stock\_table.insert('', END, values=row) mysqldb.commit()

mysqldb.close()

def backB(self): self.root.destroy()

from mdi\_form import MDI\_form mdi\_win=Tk()

login\_app = MDI\_form(mdi\_win)

class stock(): pass

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

##### VENDOR DETAILS FORM

from tkinter import\* from tkinter import ttk import tkinter as tk

from tkinter import ttk, messagebox import mysql.connector

from PIL import Image ,ImageTk import re

class vendor():

def init (self,root): self.root=root

self.root.title("Event management system") self.root.geometry("1390x700+0+0") root.configure(bg='#461257')

title = Label(self.root,

text="Event management System",bd=9,relief=GROOVE, font=("times new roman",50,"bold"),bg='#461257',fg="#D7A1F9") title.pack(side=TOP,fill=X)

backimg = Image.open("backB.png")

backimg = backimg.resize((35, 35), Image.ADAPTIVE)

self.Photo\_img = ImageTk.PhotoImage(backimg) # Changed mdi.Photo\_img to self.Photo\_img

buttonimg = Button(self.root, image=self.Photo\_img, borderwidth=0, command=self.backB, bg="#4DDCE6") # Changed mdi to self.mdi

buttonimg.place(x=15, y=50, width=35, height=35)

self.Vendor\_ID\_var = StringVar() self.Vendor\_Name\_var = StringVar() self.Email\_ID\_var = StringVar() self.Category\_var = StringVar() self.Contact\_number\_var = StringVar() self.Search\_By = StringVar() self.search\_txt = StringVar()

Manage\_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="#D7A1F9") Manage\_Frame.place(x=28,y=100,width=650,height=650)

m\_title = Label(Manage\_Frame,text="Vendor Details",bg="#D7A1F9",fg="black",font=("times new roman",50,"bold"))

m\_title.pack(side=TOP,fill=X)

C\_ID = Label(root,text="Vendor ID:",bg="#D7A1F9",fg="black",font=("times new roman",18,"bold"))

C\_ID.place(x=100,y=250) e1=Entry(root,textvariable=self.Vendor\_ID\_var,font=('bold',13,'bold'),fg='bla

ck',width=30)

e1.place(x=300,y=250)

Name = Label(root,text="Vendor Name:",bg="#D7A1F9",fg="black",font=("times new roman",18,"bold"))

Name.place(x=100,y=300) e2=Entry(root,textvariable=self.Vendor\_Name\_var,font=('bold/Italic/red/GREEN\_

BG',13,'bold'),fg='black',width=30) e2.place(x=300,y=300)

Email = Label(root,text="Email ID:",bg="#D7A1F9",fg="black",font=("times new roman",18,"bold"))

Email.place(x=100,y=350) e3=Entry(root,textvariable=self.Email\_ID\_var,font=('bold/Italic/red/GREEN\_BG'

,13,'bold'),fg='black',width=30) e3.place(x=300,y=350)

Event = Label(root,text="Category:",bg="#D7A1F9",fg="black",font=("times new roman",18,"bold"))

Event.place(x=100,y=400)

V=["flower Merchant","Entertainment","party Rental","Sommelier","private Caterers","Camera/production"]

e6=ttk.Combobox(root,textvariable=self.Category\_var,values=V,width=42) e6.place(x=300,y=400)

Contact = Label(root,text="Contact Number:",bg="#D7A1F9",fg="black",font=("times new roman",18,"bold"))

Contact.place(x=100,y=480) e5=Entry(root,textvariable=self.Contact\_number\_var,font=('bold/Italic/red/GRE

EN\_BG',13,'bold'),fg='black',width=30) e5.place(x=300,y=480)

Address = Label(root,text="Address:",bg="#D7A1F9",fg="black",font=("times new roman",18,"bold"))

Address.place(x=100,y=550) self.txt\_Address=Text(root,width=30,height=3,font=('bold/Italic/red/GREEN\_BG'

,13,'bold'))

self.txt\_Address.place(x=300,y=550)

button=Button(text="Save",command=self.add\_vendor,font=('bold/Italic/red/GREE N\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3')

button.place(x=160,y=650)

button=Button(text="Update",command=self.update\_data,font=('bold/Italic/red/G REEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3')

button.place(x=260,y=650)

button=Button(text="Delete",command=self.delete\_data,font=('bold/Italic/red/G REEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3')

button.place(x=360,y=650)

button=Button(text="Clear",command=self.clear\_vendor,font=('bold/Italic/red/G REEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3')

button.place(x=460,y=650)

Manage\_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="#D7A1F9") Manage\_Frame.place(x=720,y=100,width=800,height=650)

m\_title = Label(Manage\_Frame,text="The Dream Maker",bg="#D7A1F9",fg="black",font=("times new roman",30,"bold"))

m\_title.pack(side=TOP,fill=X)

m\_title = Label(Manage\_Frame,text="Here To make Your Special Days A Lil More Special",bg="#D7A1F9",fg='#461257',font=("times new roman",20,"bold"))

m\_title.pack(side=TOP,fill=X)

Name = Label(root,text="Search By:",bg="#D7A1F9",fg="black",font=("times new roman",18,"bold"))

Name.place(x=750,y=200)

V=["Vendor\_ID","Vendor\_Name","Email\_ID","Category","Address"] combo=ttk.Combobox(root,textvariable=self.Search\_By,values=V,width=20) combo.place(x=870,y=205)

NameEntry=Entry(root,textvariable=self.search\_txt,font=('bold/Italic/red/GREE N\_BG',13,'bold'),fg='black',width=15)

NameEntry.place(x=1030,y=205) searchbtn = Button(text = 'Search',

width=10,pady=5,command=self.search\_data).place(x=1190,y=205)

showbtn = Button(text = 'Show All', width=10,pady=5,command=self.fetch\_data).place(x=1300,y=205)

Table\_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="pink") Table\_Frame.place(x=750,y=250,width=750,height=480)

Scroll\_x = Scrollbar(Table\_Frame,orient=HORIZONTAL) Scroll\_Y = Scrollbar(Table\_Frame,orient=VERTICAL)

self.vendor\_table

=ttk.Treeview(Table\_Frame,column=("Vendor\_ID","Vendor\_Name","Email\_ID","Category","Co ntact Number","Address"),xscrollcommand=Scroll\_Y.set,yscrollcommand=Scroll\_x.set)

Scroll\_x.pack(side=BOTTOM,fill=X) Scroll\_Y.pack(side=RIGHT,fill=Y)

Scroll\_x.config(command=self.vendor\_table.xview) Scroll\_Y.config(command=self.vendor\_table.yview)

self.vendor\_table.heading("Vendor\_ID",text="Vendor\_ID") self.vendor\_table.heading("Vendor\_Name",text="Vendor\_Name") self.vendor\_table.heading("Email\_ID",text="Email\_ID") self.vendor\_table.heading("Category",text="Category") self.vendor\_table.heading("Contact Number",text="Contact Number") self.vendor\_table.heading("Address",text="Address") self.vendor\_table['show']='headings' self.vendor\_table.column("Vendor\_ID", width=100) self.vendor\_table.column("Vendor\_Name", width=100)

self.vendor\_table.column("Email\_ID",width=100)

self.vendor\_table.column("Category", width=100) self.vendor\_table.column("Contact Number", width=100) self.vendor\_table.column("Address", width=100)

self.vendor\_table.pack(fill=BOTH,expand=1) self.vendor\_table.bind("<ButtonRelease-1>",self.get\_cursor) self.fetch\_data()

def validate\_contact\_number(self): pattern = r'^[0-9]{10}$'

if re.match(pattern, self.Contact\_number\_var.get()): return True

else:

return False

def validate\_email(self, email):

pattern = r'^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$' if re.match(pattern, email):

return True else:

messagebox.showerror("Error", "Invalid email address.") return False

def add\_vendor(self):

if self.Vendor\_ID\_var.get()=="" or self.Vendor\_Name\_var.get()=="" or self.Email\_ID\_var.get()=="" or self.Category\_var.get()=="" or self.Contact\_number\_var.get()=="" :

messagebox.showerror("error","all fields are required to fill") elif not self.validate\_email(self.Email\_ID\_var.get()):

return # Do nothing if email is invalid

else:

if self.validate\_contact\_number(): print("Valid contact number")

else:

print("Invalid contact number")

mysqldb = mysql.connector.connect(host='localhost',user='root',password='regal@123456789',datab ase="sys")

mycuror = mysqldb.cursor() mycuror.execute("insert into vendor1\_details

values(%s,%s,%s,%s,%s,%s)",(self.Vendor\_ID\_var.get(),

Name\_var.get(), D\_var.get(),

y\_var.get(),

\_number\_var.get(),

self.Vendor\_ self.Email\_I self.Categor self.Contact

ress.get('1.0',END)

self.txt\_Add

))

mysqldb.commit() self.fetch\_data() self.clear\_vendor()

mysqldb.close()

messagebox.showinfo("Success","Record has been inserted")

def fetch\_data(self): mysqldb =

mysql.connector.connect(host='localhost',user='root',password='regal@123456789',datab ase="sys")

mycuror = mysqldb.cursor() mycuror.execute("select \* from vendor1\_details")

rows = mycuror.fetchall() if len(rows)!=0:

self.vendor\_table.delete(\*self.vendor\_table.get\_children())

for row in rows: self.vendor\_table.insert('',END,values=row)

mysqldb.commit() mysqldb.close()

def get\_cursor(self,ev):

curosor\_row = self.vendor\_table.focus() contents = self.vendor\_table.item(curosor\_row) row = contents['values'] self.Vendor\_ID\_var.set(row[0]) self.Vendor\_Name\_var.set(row[1]) self.Email\_ID\_var.set(row[2])

self.Category\_var.set(row[3]) self.Contact\_number\_var.set(row[4]) self.txt\_Address.delete('1.0',END) self.txt\_Address.insert(END,row[5])

def clear\_vendor(self): self.Vendor\_ID\_var.set("") self.Vendor\_Name\_var.set(" ") self.Email\_ID\_var.set(" ") self.Category\_var.set(" ") self.Contact\_number\_var.set(" ") self.txt\_Address.delete('1.0',END)

def update\_data(self): mysqldb =

mysql.connector.connect(host='localhost',user='root',password='regal@123456789',datab ase="sys")

mycuror = mysqldb.cursor() mycuror.execute("update vendor1\_details set

Vendor\_Name=%s,Email\_ID=%s,category=%s,Contact\_number=%s,address=%s where Vendor\_ID=%s",(self.Vendor\_Name\_var.get(), self.Email\_ID\_var.get(),self.Category\_var.get(),self.Contact\_number\_var.get(),self.tx t\_Address.get('1.0',END),self.Vendor\_ID\_var.get()))

mysqldb.commit() self.fetch\_data() self.clear\_vendor() mysqldb.close()

messagebox.showinfo("Success","Record has been inserted")

def delete\_data(self):

vendor\_id = (self.Vendor\_ID\_var.get(),) # Convert to tuple mysqldb = mysql.connector.connect(host='localhost', user='root',

password='regal@123456789', database="sys") mycursor = mysqldb.cursor()

mycursor.execute("delete from vendor1\_details where Vendor\_ID=%s", vendor\_id) mysqldb.commit()

self.fetch\_data() self.clear\_vendor() mysqldb.close()

def search\_data(self): mysqldb =

mysql.connector.connect(host='localhost',user='root',password='regal@123456789',datab ase="sys")

mycuror = mysqldb.cursor()

mycuror.execute("select \* from vendor1\_details where " + str(self.Search\_By.get()) +" Like '%"+str(self.search\_txt.get())+"%'")

rows =mycuror.fetchall() if len(rows) != 0:

self.vendor\_table.delete(\*self.vendor\_table.get\_children()) for row in rows:

self.vendor\_table.insert('', END,values=row) mysqldb.commit()

mysqldb.close()

def backB(self): self.root.destroy()

from mdi\_form import MDI\_form mdi\_win=Tk()

login\_app = MDI\_form(mdi\_win)

class vendor(): pass

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

### PURCHASE DETAILS FORM

#from pydoc import render\_doc from tkinter import\*

from tkinter import ttk import tkinter as tk

from tkinter import ttk, messagebox import mysql.connector

from PIL import Image ,ImageTk

class purchase():

def init (self,root): self.root=root

self.root.title("Event management system") self.root.geometry("1390x700+0+0") root.configure(bg='#461257')

title = Label(self.root,text="Event management System",bd=9,relief=GROOVE,font=("times new roman",50,"bold"),bg='#461257',fg="#D7A1F9")

title.pack(side=TOP,fill=X)

backimg = Image.open("backB.png")

backimg = backimg.resize((35, 35), Image.ADAPTIVE)

self.Photo\_img = ImageTk.PhotoImage(backimg) # Changed mdi.Photo\_img to self.Photo\_img

buttonimg = Button(self.root, image=self.Photo\_img, borderwidth=0, command=self.backB, bg="#4DDCE6") # Changed mdi to self.mdi

buttonimg.place(x=15, y=50, width=35, height=35)

self.Purchase\_ID\_var = StringVar() self.Vendor\_Name\_var = StringVar() self.Item\_Name\_var = StringVar() self.Quantity\_var = StringVar() self.Date\_Of\_Order\_var = StringVar() self.Date\_Of\_Arrival\_var = StringVar() self.Amount\_var = StringVar() self.Search\_By = StringVar() self.search\_txt = StringVar()

Manage\_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="#D7A1F9") Manage\_Frame.place(x=28,y=100,width=650,height=650)

m\_title = Label(Manage\_Frame,text="Purchase Details",bg="#D7A1F9",fg="black",font=("times new roman",50,"bold"))

m\_title.pack(side=TOP,fill=X)

P\_ID = Label(root, text="Purchase ID:", bg="#D7A1F9", fg="black", font=("times new roman", 18, "bold"))

P\_ID.place(x=100, y=250)

e1 = Entry(root,textvariable=self.Purchase\_ID\_var, font=('bold', 13, 'bold'), fg='black',width=30)

e1.place(x=300, y=250)

Name = Label(root,text="Vendor Name:",bg="#D7A1F9",fg="black",font=("times new roman",18,"bold"))

Name.place(x=100,y=300) e2=Entry(root,textvariable=self.Vendor\_Name\_var,font=('bold/Italic/red/GREEN\_

BG',13,'bold'),fg='black',width=30) e2.place(x=300,y=300)

Item = Label(root,text="Item Name:",bg="#D7A1F9",fg="black",font=("times new roman",18,"bold"))

Item.place(x=100,y=350) e2=Entry(root,textvariable=self.Item\_Name\_var,font=('bold/Italic/red/GREEN\_BG

',13,'bold'),fg='black',width=30) e2.place(x=300,y=350)

Quantity = Label(root,text="Quantity:",bg="#D7A1F9",fg="black",font=("times new roman",18,"bold"))

Quantity.place(x=100,y=400) e2=Entry(root,textvariable=self.Quantity\_var,font=('bold/Italic/red/GREEN\_BG'

,13,'bold'),fg='black',width=30) e2.place(x=300,y=400)

Order = Label(root,text="Date Of Order:",bg="#D7A1F9",fg="black",font=("times new roman",18,"bold"))

Order.place(x=100,y=450) e5=Entry(root,textvariable=self.Date\_Of\_Order\_var,font=('bold/Italic/red/GREE

N\_BG',13,'bold'),fg='black',width=30) e5.place(x=300,y=450)

Arrival = Label(root,text="Date Of Arrival:",bg="#D7A1F9",fg="black",font=("times new roman",18,"bold"))

Arrival.place(x=100,y=500) e6=Entry(root,textvariable=self.Date\_Of\_Arrival\_var,font=('bold/Italic/red/GR

EEN\_BG',13,'bold'),fg='black',width=30) e6.place(x=300,y=500)

Amount = Label(root,text="Amount:",bg="#D7A1F9",fg="black",font=("times new roman",18,"bold"))

Amount.place(x=100,y=550) e7=Entry(root,textvariable=self.Amount\_var,font=('bold/Italic/red/GREEN\_BG',1

3,'bold'),fg='black',width=30) e7.place(x=300,y=550)

button=Button(text="Add",command=self.add\_purchase,font=('bold/Italic/red/GRE EN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3')

button.place(x=60,y=650)

button=Button(text="Save",command='self.save',font=('bold/Italic/red/GREEN\_BG ',13,'bold'),fg='#F0FFFF',bg='#9400D3')

button.place(x=160,y=650)

button=Button(text="Update",command=self.update\_data,font=('bold/Italic/red/G REEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3')

button.place(x=260,y=650)

button=Button(text="Delete",command=self.delete\_data,font=('bold/Italic/red/G REEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3')

button.place(x=360,y=650)

button=Button(text="Cancel",command=self.clear\_purchase,font=('bold/Italic/re d/GREEN\_BG',13,'bold'),fg='#F0FFFF',bg='#9400D3')

button.place(x=460,y=650)

Manage\_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="#D7A1F9") Manage\_Frame.place(x=720,y=100,width=800,height=650)

m\_title = Label(Manage\_Frame,text="The Dream Maker",bg="#D7A1F9",fg="black",font=("times new roman",30,"bold"))

m\_title.pack(side=TOP,fill=X)

m\_title = Label(Manage\_Frame,text="Here To make Your Special Days A Lil More Special",bg="#D7A1F9",fg='#461257',font=("times new roman",20,"bold"))

m\_title.pack(side=TOP,fill=X)

Name = Label(root,text="Search By:",bg="#D7A1F9",fg="black",font=("times new roman",18,"bold"))

Name.place(x=750,y=200)

V=["Purchase\_ID:","Vendor Name:","Item Name","quantity","Date Of Order:","Date Of Arrival:","Amount:"]

combo=ttk.Combobox(root,textvariable=self.Search\_By ,values=V,width=20) combo.place(x=870,y=205)

NameEntry=Entry(root,textvariable=self.search\_txt,font=('bold/Italic/red/GREE N\_BG',13,'bold'),fg='red',width=15)

NameEntry.place(x=1030,y=205)

searchbtn = Button(text = 'Search', width=10,pady=5).place(x=1190,y=205)

showbtn = Button(text = 'Show All', width=10,pady=5).place(x=1300,y=205) #=====================TableFrame========================#

Table\_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="pink") Table\_Frame.place(x=750,y=250,width=750,height=480)

Scroll\_x = Scrollbar(Table\_Frame,orient=HORIZONTAL) Scroll\_Y = Scrollbar(Table\_Frame,orient=VERTICAL)

self.purchase\_table = ttk.Treeview(Table\_Frame,column=("Purchase ID","Vendor Name","Item Name","Quantity","Date Of Order","Date Of Arrival","Amount"),xscrollcommand=Scroll\_Y.set,yscrollcommand=Scroll\_x.set)

Scroll\_x.pack(side=BOTTOM,fill=X)

Scroll\_Y.pack(side=RIGHT,fill=Y)

Scroll\_x.config(command=self.purchase\_table.xview) Scroll\_Y.config(command=self.purchase\_table.yview)

self.purchase\_table.heading("Purchase ID",text="Purchase\_ID") self.purchase\_table.heading("Vendor Name",text="Vendor\_Name") self.purchase\_table.heading("Item Name",text="Item Name") self.purchase\_table.heading("Quantity",text="Quantity") self.purchase\_table.heading("Date Of Order",text="Date Of Order") self.purchase\_table.heading("Date Of Arrival",text="Date Of Arrival") self.purchase\_table.heading("Amount",text="Amount")

self.purchase\_table['show']='headings' self.purchase\_table.column("Purchase ID", width=100) self.purchase\_table.column("Vendor Name", width=100) self.purchase\_table.column("Item Name", width=100) self.purchase\_table.column("Quantity",width=100) self.purchase\_table.column("Date Of Order", width=100) self.purchase\_table.column("Date Of Arrival", width=100) self.purchase\_table.column("Amount", width=100) self.purchase\_table.pack(fill=BOTH,expand=1) self.purchase\_table.bind("<ButtonRelease-1>",self.get\_cursor) self.fetch\_data()

def add\_purchase(self):

if self.Purchase\_ID\_var.get()=="" or self.Vendor\_Name\_var.get()=="" or self.Item\_Name\_var.get()=="" or self.Quantity\_var.get()=="" or self.Date\_Of\_Order\_var.get()=="" or self.Date\_Of\_Arrival\_var.get()=="" or self.Amount\_var.get()=="":

messagebox.showerror("error","all fields are required to fill") else:

mysqldb = mysql.connector.connect(host='localhost',user='root',password='regal@123456789',datab ase="sys")

mycuror = mysqldb.cursor()

mycuror.execute("insert into purchase1\_details values (%s, %s, %s, %s, %s,

%s, %s)",

(self.Purchase\_ID\_var.get(), self.Vendor\_Name\_var.get(),

self.Item\_Name\_var.get(),

self.Quantity\_var.get(), self.Date\_Of\_Order\_var.get(), self.Date\_Of\_Arrival\_var.get(),

self.Amount\_var.get())) mysqldb.commit() self.fetch\_data() self.clear\_purchase() mysqldb.close()

messagebox.showinfo("Success","Record has been inserted")

def fetch\_data(self):

mysqldb = mysql.connector.connect(host='localhost',user='root',password='regal@123456789',datab ase="sys")

mycuror = mysqldb.cursor()

mycuror.execute("select \* from purchase1\_details") rows = mycuror.fetchall()

if len(rows)!=0:

self.purchase\_table.delete(\*self.purchase\_table.get\_children()) for row in rows:

self.purchase\_table.insert('',END,values=row) mysqldb.commit()

mysqldb.close()

def get\_cursor(self,ev):

curosor\_row = self.purchase\_table.focus() contents = self.purchase\_table.item(curosor\_row) row = contents['values'] self.Purchase\_ID\_var.set(row[0]) self.Vendor\_Name\_var.set(row[1]) self.Item\_Name\_var.set(row[2])

self.Quantity\_var.set(row[3]) self.Date\_Of\_Order\_var.set(row[4]) self.Date\_Of\_Arrival\_var.set(row[5]) self.Amount\_var.set(row[6])

def clear\_purchase(self): self.Purchase\_ID\_var.set("") self.Vendor\_Name\_var.set(" ") self.Item\_Name\_var.set(" ") self.Quantity\_var.set(" ") self.Date\_Of\_Order\_var.set(" ") self.Date\_Of\_Arrival\_var.set(" ") self.Amount\_var.set(" ")

def update\_data(self):

if self.Purchase\_ID\_var.get() == "" or self.Vendor\_Name\_var.get() == "" or self.Item\_Name\_var.get() == "" or self.Quantity\_var.get() == "" or self.Date\_Of\_Order\_var.get() == "" or self.Date\_Of\_Arrival\_var.get() == "" or self.Amount\_var.get() == "":

messagebox.showerror("Error", "All fields are required to fill") else:

mysqldb = mysql.connector.connect(host='localhost', user='root', password='regal@123456789', database="sys")

mycursor = mysqldb.cursor()

query = "UPDATE purchase1\_details SET `Vendor Name`=%s, `Item Name`=%s,

`Quantity`=%s, `Date Of Order`=%s, `Date Of Arrival`=%s, `Amount`=%s WHERE

`Purchase\_ID`=%s"

values = (self.Vendor\_Name\_var.get(), self.Item\_Name\_var.get(), self.Quantity\_var.get(), self.Date\_Of\_Order\_var.get(), self.Date\_Of\_Arrival\_var.get(), self.Amount\_var.get(), self.Purchase\_ID\_var.get())

mycursor.execute(query, values)

mysqldb.commit() self.fetch\_data() self.clear\_purchase() mysqldb.close()

messagebox.showinfo("Success", "Record has been updated")

def delete\_data(self):

mysqldb = mysql.connector.connect(host='localhost', user='root', password='regal@123456789', database="sys")

mycuror = mysqldb.cursor()

# Correct the column name in the WHERE clause

mycuror.execute("delete from purchase1\_details where Purchase\_ID=%s", (self.Purchase\_ID\_var.get(),))

mysqldb.commit() mysqldb.close() self.fetch\_data() self.clear\_purchase()

def search\_data(self): mysqldb =

mysql.connector.connect(host='localhost',user='root',password='regal@123456789',datab ase="sys")

mycuror = mysqldb.cursor()

mycuror.execute("select \* from purchase1\_details where " + str(self.Search\_By.get()) +" Like '%"+str(self.search\_txt.get())+"%'")

rows =mycuror.fetchall() if len(rows) != 0:

self.purchase\_table.delete(\*self.purchase\_table.get\_children()) for row in rows:

self.purchase\_table.insert('', END,values=row) mysqldb.commit()

mysqldb.close()

def search\_data(self): mysqldb =

mysql.connector.connect(host='localhost',user='root',password='regal@123456789',datab ase="sys")

mycuror = mysqldb.cursor()

mycuror.execute("select \* from purchase1\_details where " + str(self.Search\_By.get()) +" Like '%"+str(self.search\_txt.get())+"%'")

rows =mycuror.fetchall() if len(rows) != 0:

self.purchase\_table.delete(\*self.purchase\_table.get\_children())

for row in rows:

self.purchase\_table.insert('', END,values=row) mysqldb.commit()

mysqldb.close()

def backB(self): self.root.destroy()

from mdi\_form import MDI\_form mdi\_win=Tk()

login\_app = MDI\_form(mdi\_win)

class purchase(): pass

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

**TEST CASE VALIDATION**

###### Test Cases

It is the process of dividing the I/P domain into the different classes (valid & invalid) & for valid I/P classes make the equal partition so that it will reduce the test cases.

In equivalence portioning, we will make a set of similar test cases & if a single test for particular set is passed we can consider that the entire test from particular set are passed.

###### Test Data

The system functionality for each of the component was tested successful.

The testing was carried out on a computer running on Intel core- i3, 8 GB RAM, 512SSD.

Software is working successfully .all the data is entered with satisfactory results.

Finally all the Input validation & controls are checked for their successful firing by providing wrong input data.

###### Test Results

1. The system was able to successful accept the test result.
2. All the results are successfully entered in the database.
3. All the report of the test results were generated successfully.
4. Output data of various reports printed was examined and was found to be accurate.
5. All the input controls fired properly as and when required.
6. Each of the aspect of the system as functioning successfully with satisfactory results.

User can generate a report as per their requirements.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **C**  **as e no** | **Scenario** | **Sr.no** | **Action** | **Expected output** | **Actual output** | **Result** |
| **1** | **Login form** | **A** | **User enter wrong user name** | **Invalid username**  **/password** | **Invalid username**  **/password** | **Pass** |
| **B** | **User enter wrong password** | **Invalid username**  **/password** | **Invalid username**  **/password** | **Pass** |
| **2** | **Employee Detials** | **A** | **User enters numerical char in name text box** | **Only Characte rs Are Allowed Allowed** | **Only Characte rs Are Allowed Allowed** | **Pass** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **B** | **User enters char in contact No** | **Invalid Contact Number** | **Invalid Contact Number** | **Pass** |
| **C** | **User enter wrong email ID** | **Invalid Email ID** | **Invalid Email ID** | **Pass** |
| **D** | **User leaves Required Field Empty** | **Please Fill The \* Required Fields(s)** | **Please Fill The \* Required Fields(s)** | **Pass** |
| **E** | **User Search For A Record Which Doesn’t Exist** | **Record Not Found!!** | **Record Not Found!!** | **Pass** |
| **F** | **Enters DOB**  **Greater than Current Date** | **Birth Date should Not be Greater Than current date** | **Birth Date should Not be Greater Than current date** | **Pass** |
| **G** | **User Enters Salary as Characte rs or 0** | **Please Enter Valid input** | **Please Enter Valid input** | **Pass** |
| **3** |  | **A** | **User** | **Only** | **Only** | **Pass** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Customr Details** |  | **enters numerical char in name text box** | **Characte rs Are Allowed** | **Characte rs Are Allowed** |  |
| **B** | **User enters char in contact No** | **Invalid Contact Number** | **Invalid Contact Number** | **Pass** |
| **C** | **User enter wrong email ID** | **Invalid Email ID** | **Invalid Email ID** | **Pass** |
| **D** | **User leaves Required Field Empty** | **Please Fill The \* Required Fields(s)** | **Please Fill The \* Required Fields(s)** | **Pass** |
| **E** | **User Search For A Record Which Doesn’t Exist** | **Record Not Found!!** | **Record Not Found!!** | **Pass** |
| **4** | **Event Details** | **A** | **User enters numerical char in name text box** | **Only Characte rs Are Allowed Allowed** | **Only Characte rs Are Allowed Allowed** | **Pass** |
| **B** | **User Enters cost as Characte rs or 0** | **Please Enter Valid input** | **Please Enter Valid input** | **Pass** |
| **C** | **User leaves** | **Please Fill The \*** | **Please Fill The \*** | **Pass** |

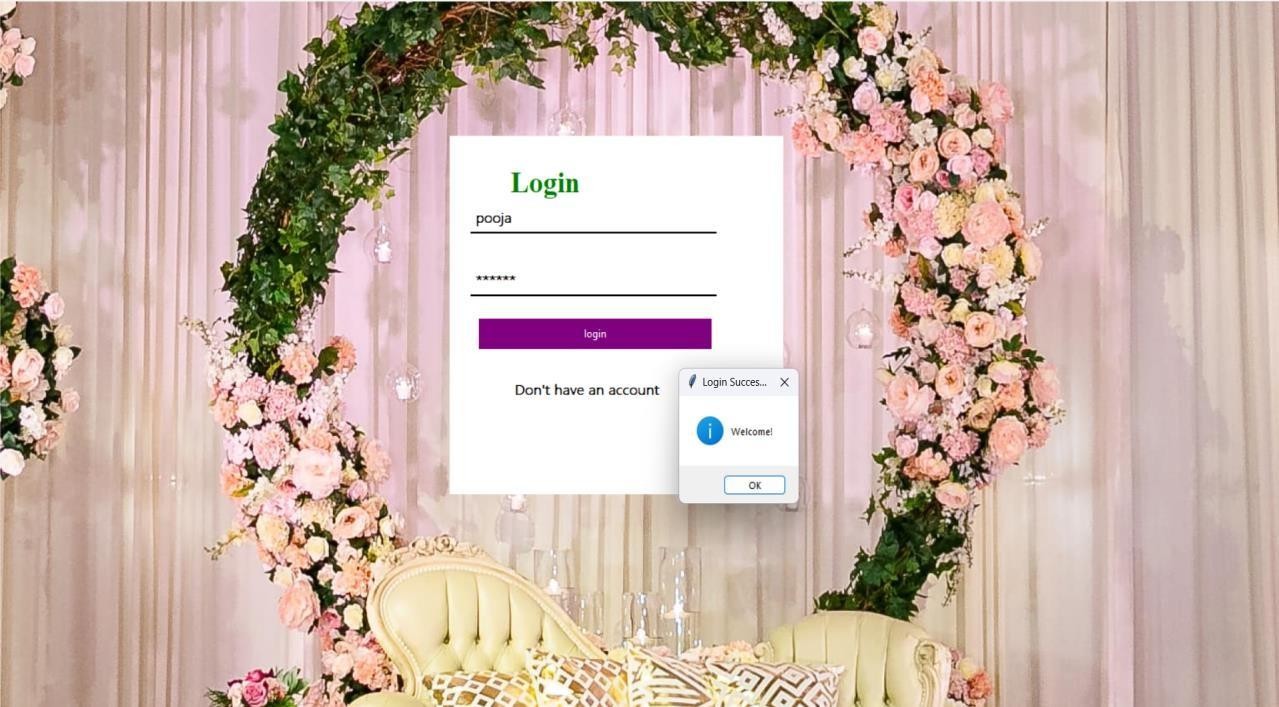
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Required Field Empty** | **Required Fields(s)** | **Required Fields(s)** |  |
| **D** | **User Search For A Record Which Doesn’t Exist** | **Record Not Found!!** | **Record Not Found!!** | **Pass** |
| **5** | **Order Details** | **A** | **User leaves Required Field Empty** | **Please Fill The \* Required Fields(s)** | **Please Fill The \* Required Fields(s)** | **Pass** |
| **B** | **User Search For A Record Which Doesn’t Exist** | **Record Not Found!!** | **Record Not Found!!** | **Pass** |
| **C** | **Enters Event date**  **Greater than Current Date** | **Date should Not be Greater Than current date** | **Date should Not be Greater Than current date** | **Pass** |
| **D** | **User Enters**  **advance as Characte rs or 0** | **Please Enter Valid input** | **Please Enter Valid input** | **Pass** |
| **6** | **Vendor Details** | **A** | **User enters numerical char in**  **name text** | **Only Characte rs Are Allowed** | **Only Characte rs Are Allowed** | **Pass** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **box** | **Allowed** | **Allowed** |  |
| **B** | **User enters char in contact No** | **Invalid Contact Number** | **Invalid Contact Number** | **Pass** |
| **C** | **User enter wrong email ID** | **Invalid Email ID** | **Invalid Email ID** | **Pass** |
| **D** | **User leaves Required Field Empty** | **Please Fill The \* Required Fields(s)** | **Please Fill The \* Required Fields(s)** | **Pass** |
| **E** | **User Search For A Record Which Doesn’t Exist** | **Record Not Found!!** | **Record Not Found!!** | **Pass** |
| **7** | **Stock details** | **A** | **User enters numerical char in name text box** | **Only Characte rs Are Allowed Allowed** | **Only Characte rs Are Allowed Allowed** | **Pass** |
| **B** | **User leaves Required Field Empty** | **Please Fill The \* Required Fields(s)** | **Please Fill The \* Required Fields(s)** | **Pass** |

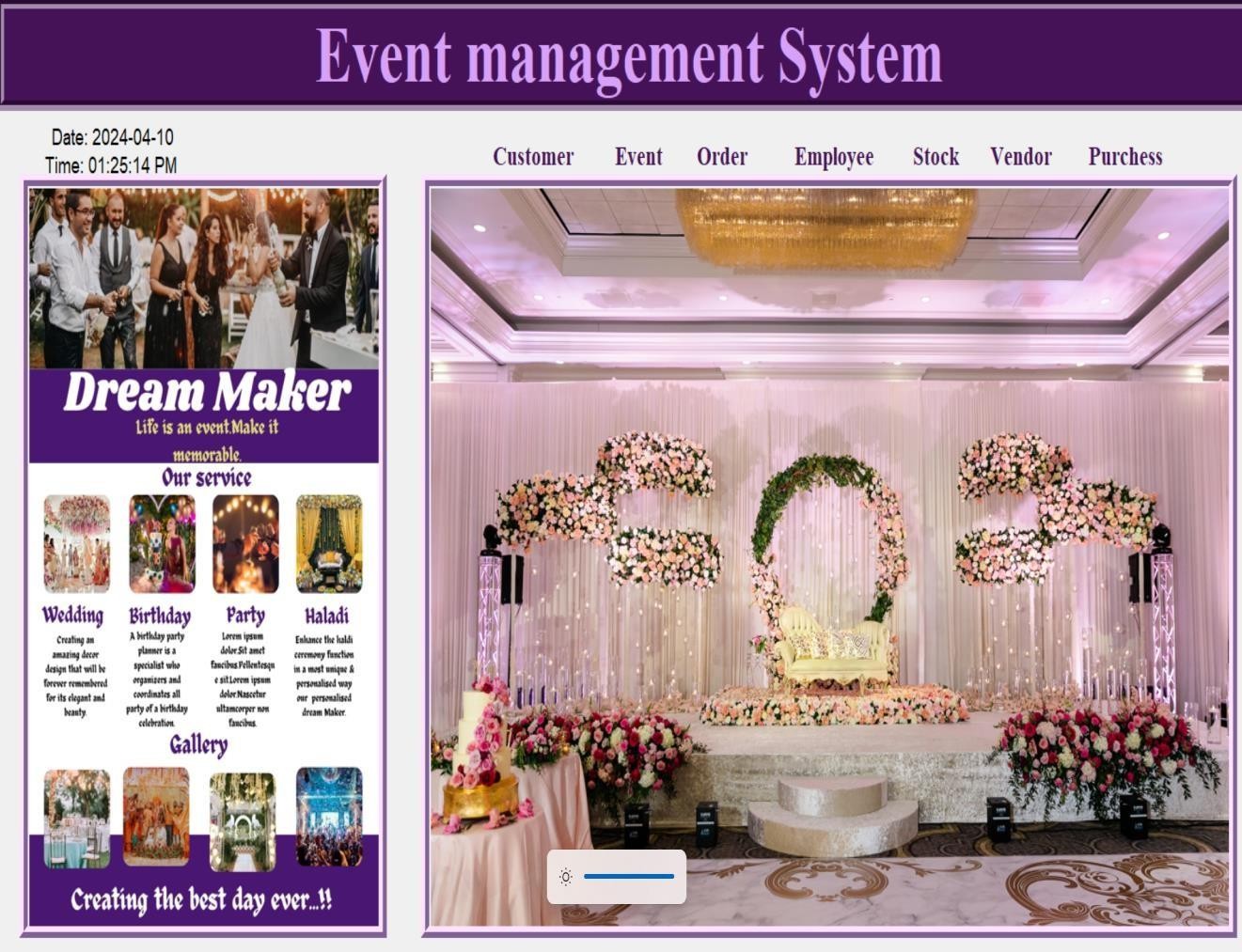
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **C** | **User Search For A Record Which**  **Doesn’t**  **Exist** | **Record Not Found!!** | **Record Not Found!!** | **Pass** |
| **8** | **Purchase Details** | **A** | **User enters numerical char in name text box** | **Only Characte rs Are Allowed Allowed** | **Only Characte rs Are Allowed Allowed** | **Pass** |
| **B** | **User leaves Required Field Empty** | **Please Fill The \* Required Fields(s)** | **Please Fill The \* Required Fields(s)** | **Pass** |
| **C** | **User Search For A Record Which Doesn’t Exist** | **Record Not Found!!** | **Record Not Found!!** | **Pass** |
| **D** | **Enters Amount As 0 or as String** | **Please enter valid input** | **Please enter valid input** | **Pass** |

# SCREEN LAYOUT

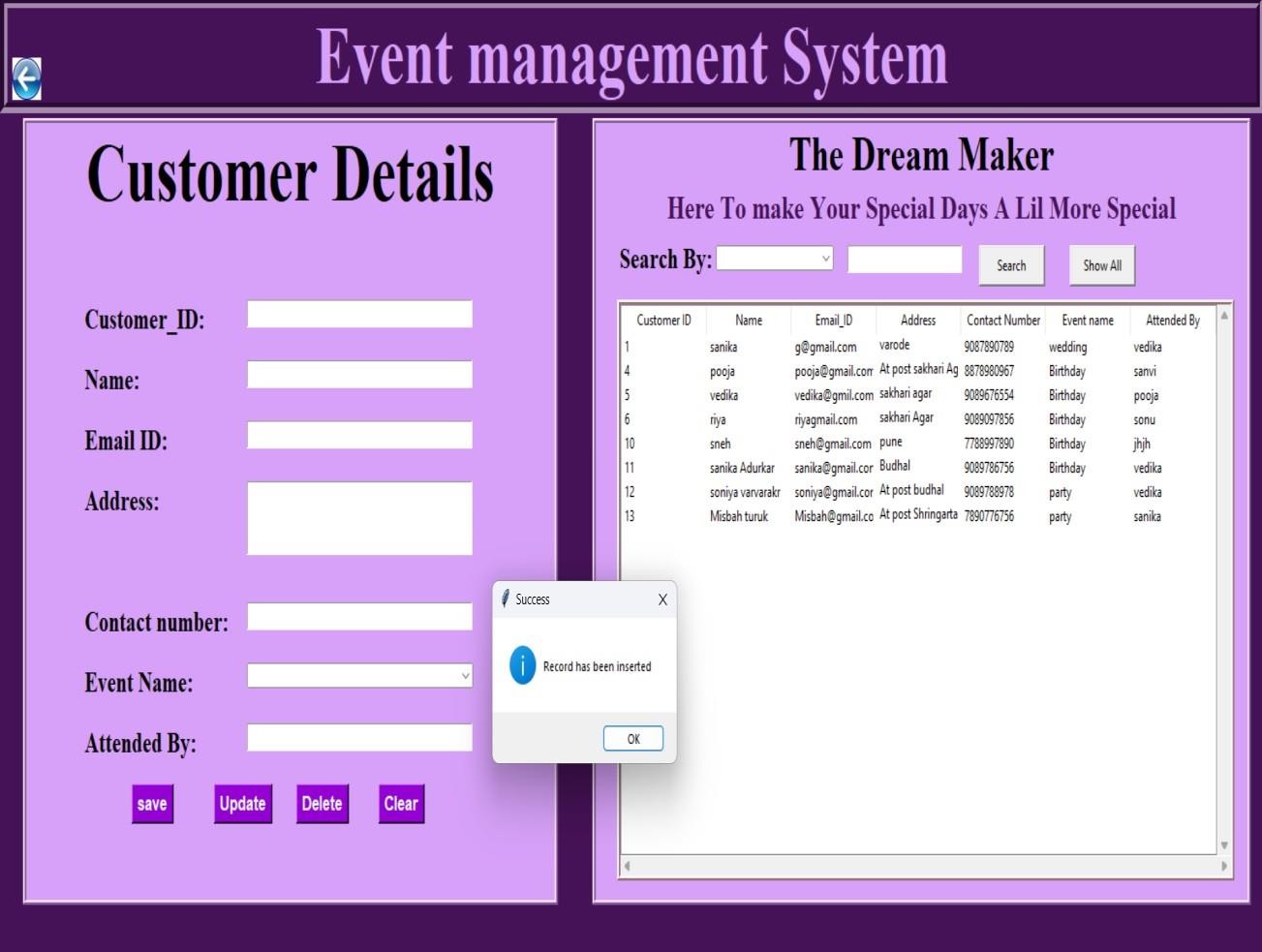
**LOGIN FORM**

****

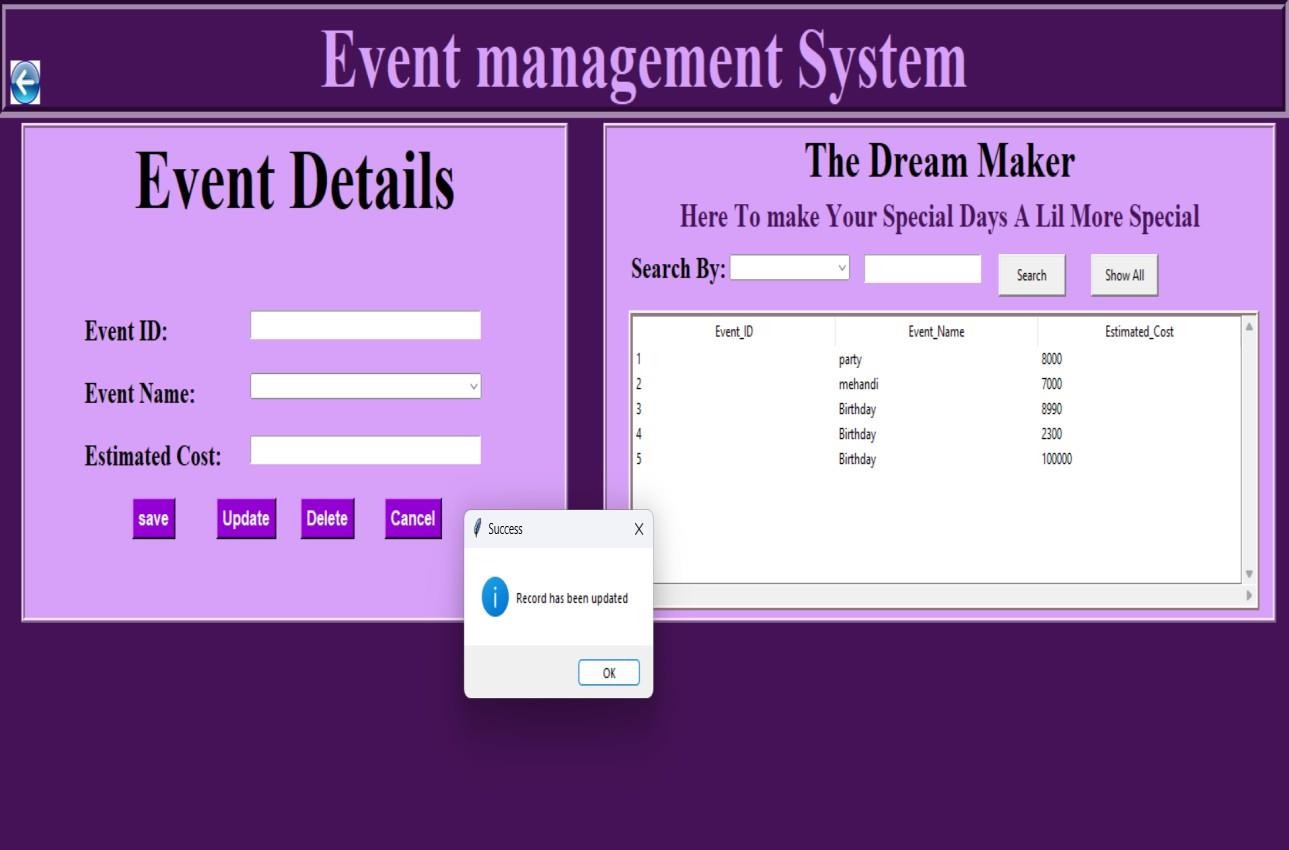
**MDI FORM**

****

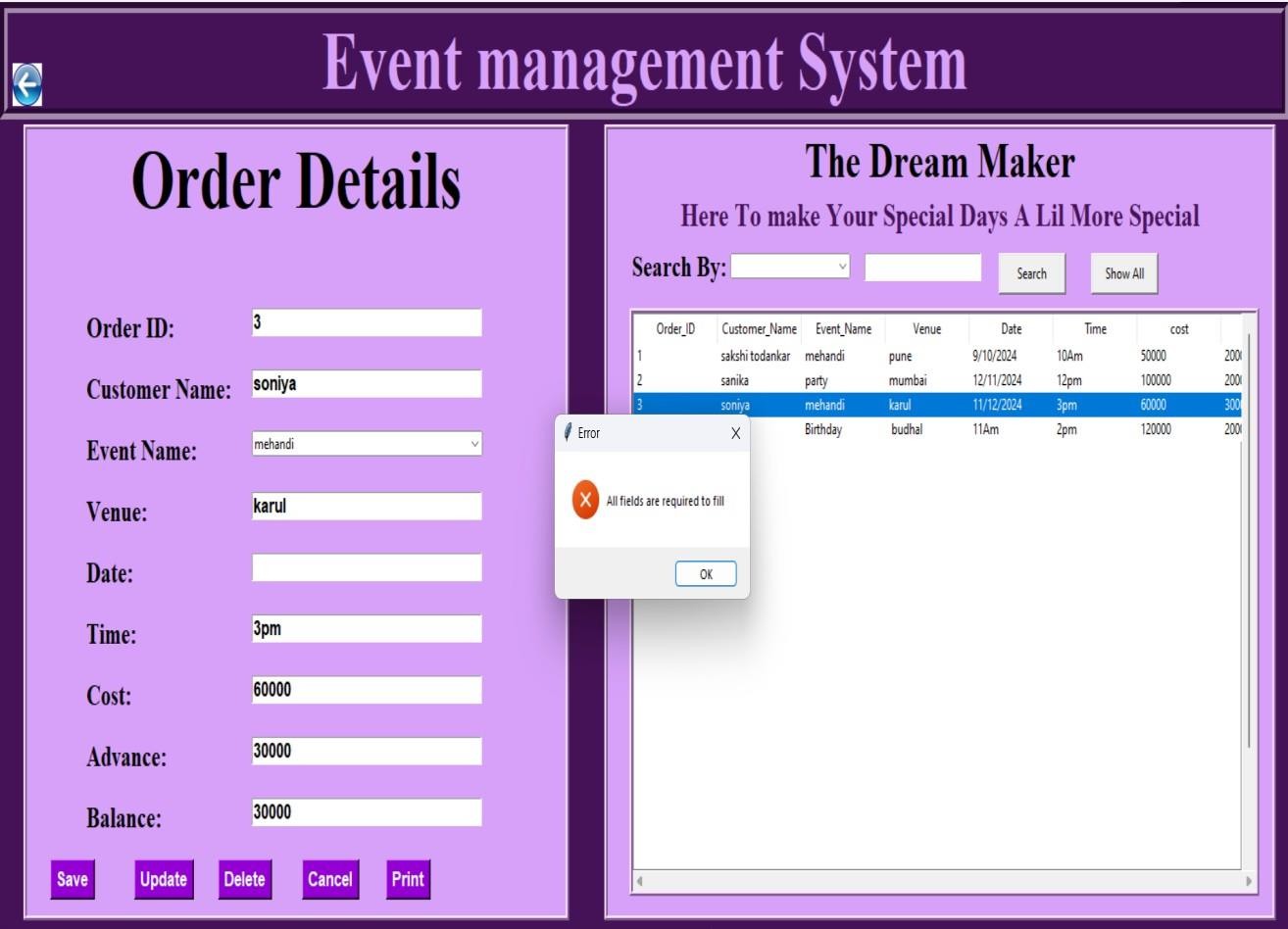
**CUSTOMER DETAILS FORM**

****

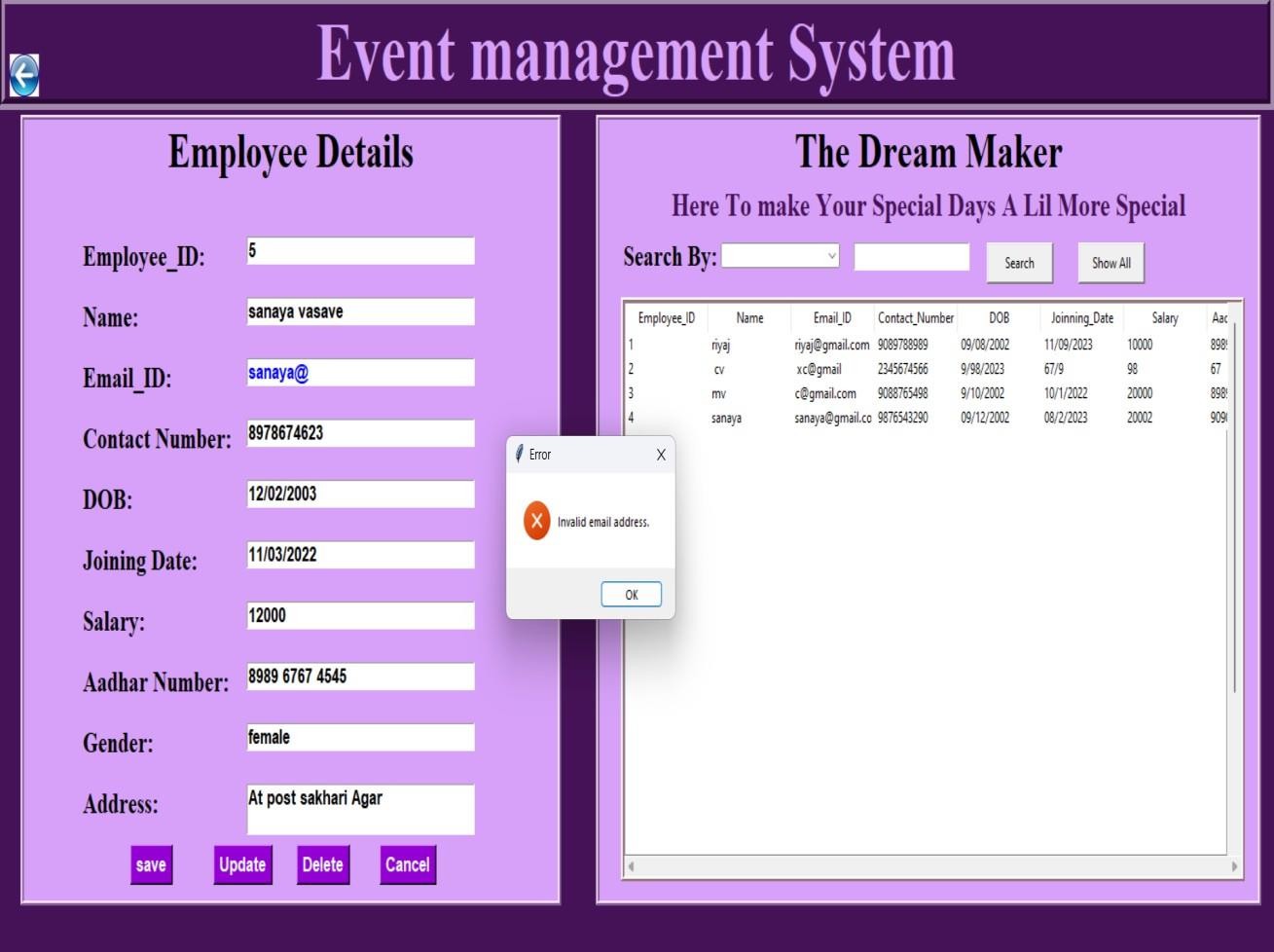
**EVENT DETAILS FORM**

****

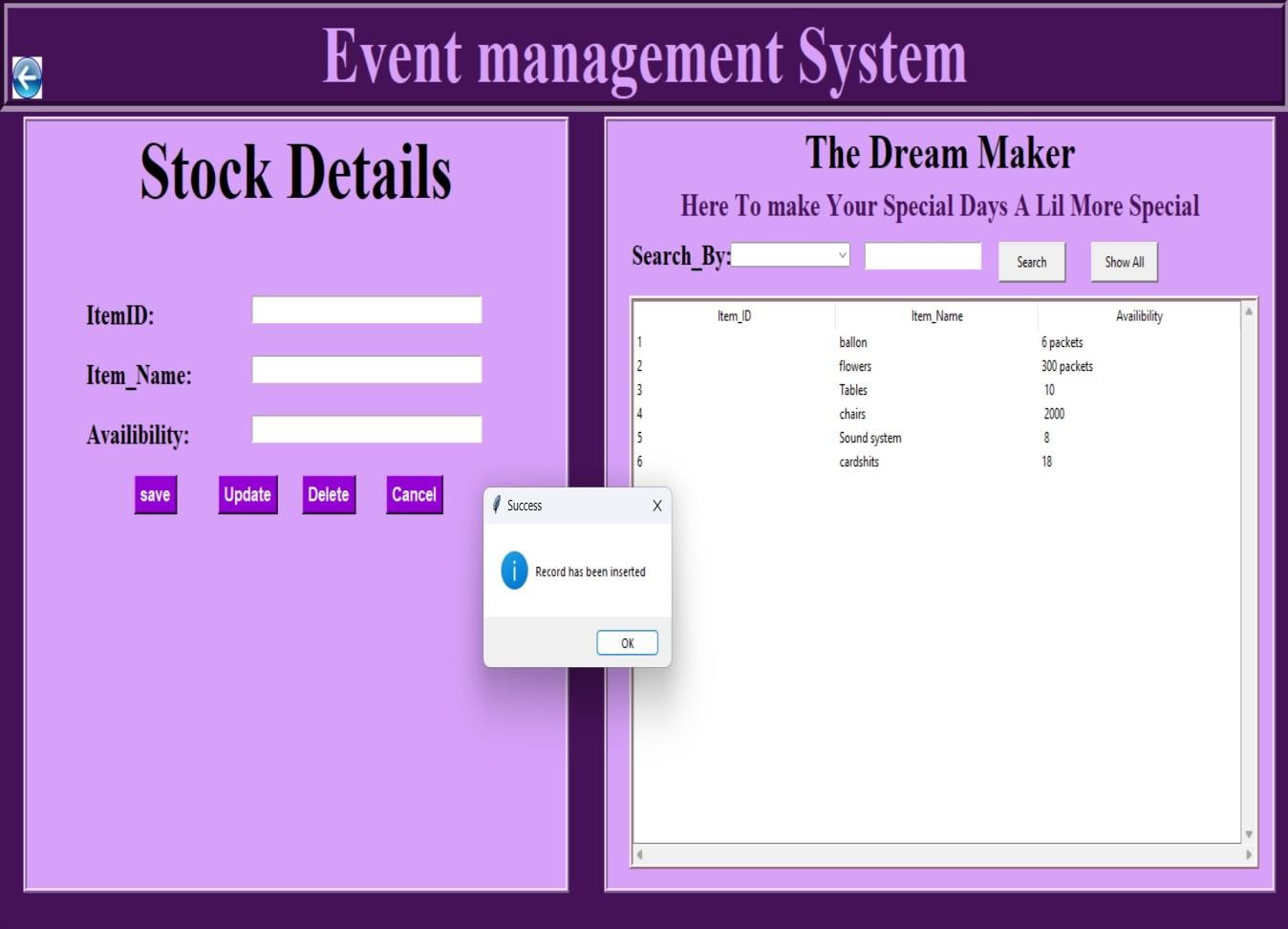
**ORDER DETAILS FORM**

****

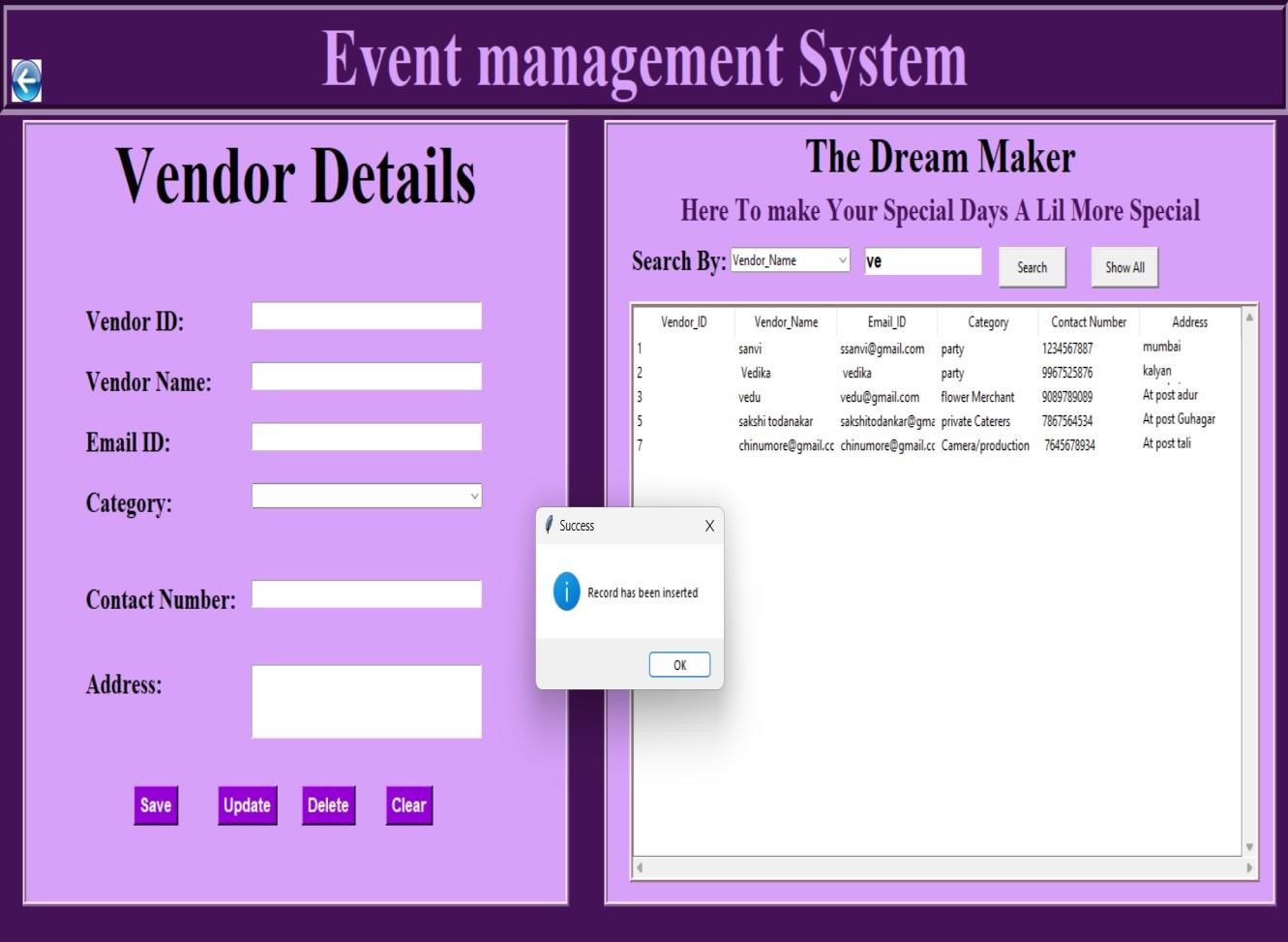
**EMPLOYEE DETAILS FORM**

****

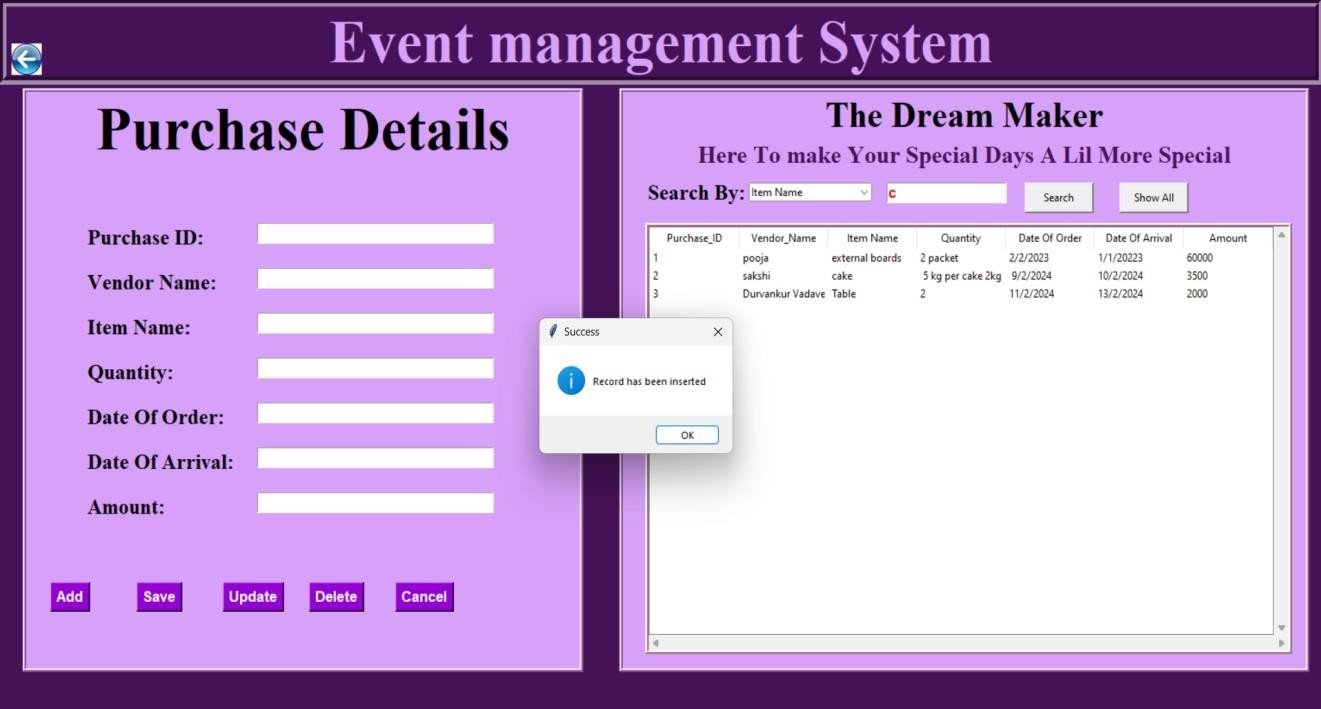
**STOCK DETAILS**

****

**VENDOR DETAILS FORM**

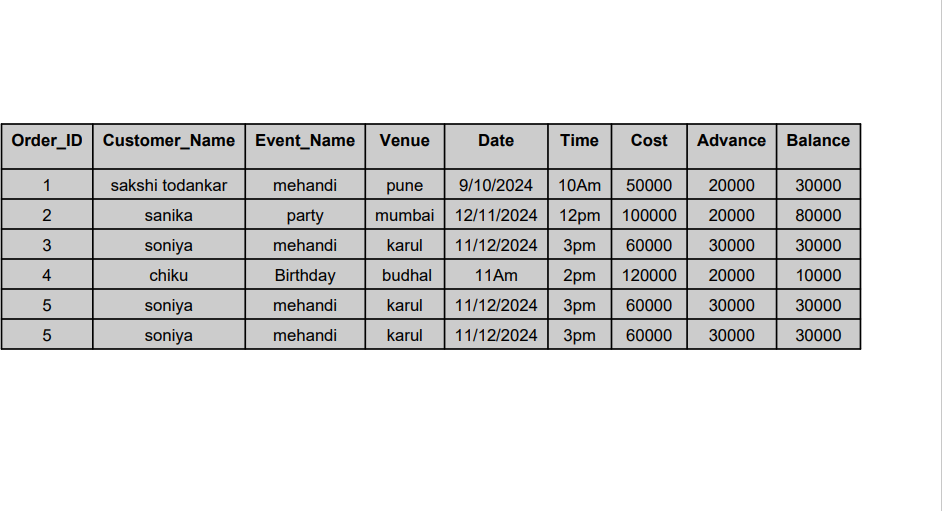
****

**PURCHASE DETAILS FORM**

****

**REPORT LAYOUT**

**ORDER REPORT**

****

# SYSTEM IMPLEMENTATION

**HARDWARE AND SOFTWARE REQUIRMENT**

The software is used for the development of proposed system is:-

#### The hardware requirement:-

Intel Core i-3 4GB RAM

500GB HD

#### The software requirement:-

* + **Back End**: SQL EXPRESS SERVER-2008
  + **Front End**: Visual studio 2010-VB.NET

**FUTURE ENHANCEMENT**

* I will add the facility to keep information about the regular and non regular customer.
* I will Work on the user’s feedback
* In future we will give this facility to access system to multiple users.
* In future we will make web based system.
* We will implement more impressive transaction.

**REFERENCE BIBLIOGRAPHY**

* + System Analysis and Design in a Changing World.
  + PYTHON 3.11 Programming Black Book
  + Software Engineering a Practitioners Approach.
  + CodeWithWajid(YouTube)
  + Zero To Master In Python Programming, VayuEducation Of India