**VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, PUNE**



**PROJECT REPORT**

**ON**

**Zoo Management System**

**THE PBL OF IT WORKSHOP**

**IN**

**COMPUTER ENGINEERING DEPARTMENT**

**BY**

Saurabh Raut - [saurabh.22110178viit.ac.in](mailto:tushar.22110166@viit.ac.in)

Rahul Shelke - [rahul.22110481@viit.ac.in](mailto:rahul.22110481@viit.ac.in)

Darshan Vora – darshan.22110769@viit.ac.in

Renuka Sarse -  [@viit.ac.in](mailto:shreyash.22110438@viit.ac.in)

Class: SY B-Tech Division: C Batch: C3

Batch Teacher-Vikas Maral Sir

INDEX

|  |  |  |
| --- | --- | --- |
| Sr. No. | Contents | Page No. |
| 1 | ABSTRACT | 3 |
| 2 | INTRODUCTION AND THEORY | 4 |
| 3 | EXPERIMENTAL DETAILS  Algorithm  Flowchart  Program | 5  5  6  8 |
| 4 | RESULTS | 11 |
| 5 | CONCLUSION | 12 |
| 6 | REFERENCE | 13 |

**ABSTRACT**:

Zoo management system is a comprehensive software application that helps zookeepers manage their daily operations effectively. It is designed to streamline the process of managing animals, staff, and visitors in a zoo environment. This system provides a centralized platform for managing animal health records, feeding schedules, enclosure maintenance, and other important tasks. With the help of this system, zookeepers can ensure the safety and well-being of animals while providing an enjoyable experience for visitors.

One of the key features of the zoo management system is animal management. This module allows zookeepers to keep track of each animal's health, diet, and behavior patterns. They can also schedule medical checkups and vaccinations for the animals.

allows managers to set goals and objectives for each staff member and track their progress.

**INTRODUCTION AND THEORY:**

Additionally, this module helps zookeepers monitor breeding programs and track the lineage of each animal. With this information, they can make informed decisions about which animals should be paired for breeding to maintain genetic diversity

Managing staff is another important aspect of running a zoo The zoo management system provides tools for scheduling shifts, tracking attendance, and managing payroll. It also.

**A. Aims and Objectives**

To increase interest in wild life, to provide recreation and education, and to conserve endangered species.

Build up scientific knowledge on wildlife resources. Train personnel at various levels for conservation and management of wildlife.

Carry out research relevant to management including the development of techniques appropriate to Indian conditions.

**Objectives:**

Today most modern zoos have four main objectives: Conservation, Education, Research and Recreation. CONSERVATION: To be involved in programs which assist the survival of wild populations of animals.

Algorithm:

Step 1: Display a login window for user/admin login.

Step 2: Authenticate user/admin credentials using MySQL connector.

Step 3: If user logs in, display a ticket booking window.

Step 4: If admin logs in, display a management window.

Step 5: In ticket booking window, display current Animal table in zoo.

Step 6: Allow user to select Date and No.of Tickets, and book tickets.

Step 7: Update Animals in database using MySQL connector.

Step 8: In management window, display current Information about animal.

Step 9: Allow admin to add new Animal and update their information.

Step 10: Terminate program on exit button click.

Program:

For Login Page:

from tkinter import \*

import tkinter as tk

from PIL import Image, ImageTk

from tkinter import ttk

import mysql.connector

from tkinter import messagebox

from Ticket\_zoo import ZoomanagementSystem

from Ticket\_zoo3 import Ticket\_page

import mysql.connector

from tkinter import Toplevel, Label, Entry, Button, messagebox

class Login\_Page:

    def \_\_init\_\_(self, root1):

        self.username\_var = StringVar()

        self.password\_var = StringVar()

        self.root1 = root1

        self.root1.title("Ticket Booking")

        self.root1.geometry("1550x800+0+0")

        self.root1.configure(bg="#F1DBBF", relief=FLAT)

        # self.root1.attributes('-fullscreen', True)

        self.background = PhotoImage(file="zoo25.png")

        label1 = Label(self.root1, image=self.background)

        label1.place(x=0, y=0,relheight=1,relwidth=1)

        lbltitle=Label(self.root1,text="WELCOME   TO    ZOO  ",bd=5,relief=FLAT,

                        bg='#A9907E',fg="#F8C4B4",font=("times new roman",50,"bold"),padx=2,pady=4)

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

        lbltitle.config(highlightbackground="#420b07")

        DataFrame=LabelFrame(self.root1,bd=10,relief=FLAT,bg="#97DEFF",padx=100,font=("times new roman",15,"bold"),labelanchor="n")

        DataFrame.place(x=150,y=200,width=500,height=360)

        lblID = Label(DataFrame, font=('times new roman', 17, "bold"),

        text="LOGIN HERE",

        padx=2, bg="#97DEFF",fg="#3E54AC", anchor='nw', justify='left')

        lblID.place(x=60,y=20)

        lblID.config(width=50, height=10)

        self.label\_username = Label(DataFrame, text="Username:",fg="#3E54AC",bg="#97DEFF",font=("times new roman",13,"bold"))

        self.label\_username.place(x=30,y=60)

        self.entry\_username = Entry(DataFrame, textvariable=self.username\_var,width=27)

        self.entry\_username.place(x=120,y=60)

        self.label\_password = Label(DataFrame, text="Password:",fg="#3E54AC",bg="#97DEFF",font=("times new roman",13,"bold"))

        self.label\_password.place(x=30,y=110)

        self.entry\_password = Entry(DataFrame,show="\*", textvariable=self.password\_var,width=27)

        self.entry\_password.place(x=120,y=110)

        style = ttk.Style()

        style.configure("TButton", padding=2, relief="flat", backgroundcolor="red", borderwidth=0, bordercolor="",font=("TImes new roman", 12), foreground="#9E4784", focuscolor="#46C2CB", focusthickness=0, highlightthickness=0, highlightcolor="#46C2CB", borderradius=250)

        self.btn\_signup = ttk.Button(DataFrame, text="Signup", command=self.signup,style="TButton")

        self.btn\_signup.place(x=100,y=230)

        self.admin\_log = ttk.Button(DataFrame, text="Admin Login", command=self.admin\_login,style="TButton")

        self.admin\_log.place(x=20,y=180)

        self.user\_login = ttk.Button(DataFrame, text="User Login", command=self.User\_login,style="TButton")

        self.user\_login.place(x=170,y=180)

        self.exit\_btn = ttk.Button(DataFrame, text="Exit", command=self.root1.destroy,style="TButton")

        self.exit\_btn.place(x=100,y=285)

    def signup(self):

        # Create a new top-level window for the sign-up dialog

        signup\_window = Toplevel(self.root1)

        signup\_window.configure(bg="#FDD36A", relief=FLAT)

        signup\_window.title("Sign up")

        x = self.root1.winfo\_rootx()

        y = self.root1.winfo\_rooty()

        offset\_x = 280

        offset\_y = 240

        signup\_window.geometry(f"+{x + offset\_x}+{y + offset\_y}")

        # Add a label and entry widget for the username

        username\_label = Label(signup\_window, text="Username:",bg="#FDD36A",font=("times new roman",12))

        username\_label.grid(row=0, column=0, padx=5, pady=5)

        username\_entry = Entry(signup\_window)

        username\_entry.grid(row=0, column=1, padx=5, pady=5)

        # Add a label and entry widget for the password

        password\_label = Label(signup\_window, text="Password:",bg="#FDD36A",font=("times new roman",12))

        password\_label.grid(row=1, column=0, padx=5, pady=5)

        password\_entry = Entry(signup\_window, show="\*")

        password\_entry.grid(row=1, column=1, padx=5, pady=5)

        # Add a label and entry widget for the confirm password

        confirm\_label = Label(signup\_window, text="Confirm Password:",bg="#FDD36A",font=("times new roman",12))

        confirm\_label.grid(row=2, column=0, padx=5, pady=5)

        confirm\_entry = Entry(signup\_window, show="\*")

        confirm\_entry.grid(row=2, column=1, padx=5, pady=5)

        # Add a button to submit the sign-up form

        signup\_button = ttk.Button(signup\_window, text="Sign up", command=lambda: self.process\_signup(username\_entry.get(), password\_entry.get(), confirm\_entry.get(), signup\_window),style="TButton")

        signup\_button.grid(row=3, column=1, padx=5, pady=5)

    def process\_signup(self, username, password, confirm\_password, signup\_window):

        if not username or not password:

            messagebox.showerror("Error", "Username and password are required.")

        elif password != confirm\_password:

            messagebox.showerror("Error", "Passwords do not match.")

        else:

            conn = mysql.connector.connect(host="localhost", user="root", password="9206", database="zoo")

            my\_cursor = conn.cursor()

            my\_cursor.execute("INSERT INTO user\_data (username, password) VALUES (%s, %s)", (

                username,

                password

            ))

            conn.commit()

            messagebox.showinfo("Signup", "Signup Successful. Please login with your credentials.")

            signup\_window.destroy()

            conn.close()

    def admin\_login(self):

        username = self.entry\_username.get()

        password = self.entry\_password.get()

        # user\_role = self.combobox\_user\_role.get()

        if username == "admin" and password == "password":

            self.new\_window = Toplevel(self.root1)

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

            self.root1.withdraw()

        else:

            messagebox.showerror("Error ","Wrong Admin Credentials")

    def User\_login(self):

        username = self.entry\_username.get()

        password = self.entry\_password.get()

        # user\_role = self.combobox\_user\_role.get()

        if self.username\_var.get() == "" or self.password\_var.get() == "":

                messagebox.showerror("Error", "All fields required")

        else:

            conn = mysql.connector.connect(host="localhost", user="root", password="9206", database="zoo")

            my\_cursor = conn.cursor()

            my\_cursor.execute("SELECT \* FROM user\_data WHERE username = %s AND password = %s", (

                self.username\_var.get(),

                self.password\_var.get()

            ))

            row = my\_cursor.fetchone()

            if row is not None:

                messagebox.showinfo("Login", "Login Succcesfull")

                self.new\_window = Toplevel(self.root1)

                self.app = Ticket\_page(self.new\_window)

                self.root1.withdraw()

            else:

                messagebox.showerror("Error", "Username and Password not found")

            conn.close()

        # else:

        #     self.label\_error.config(text="Invalid username or password")

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

    root1 = Tk()

    obj = Login\_Page(root1)

    root1.mainloop()

For Admin Page:

from tkinter import\*

import tkinter as tk

from PIL import Image,ImageTk

from tkinter import ttk

import mysql.connector

from tkinter import messagebox

class ZoomanagementSystem:

        def \_\_init\_\_(self,root):

                self.root=root

                self.root.title("Zoo Management System")

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

                self.root.configure(bg="#F3DEBA",relief=FLAT)

                self.Aname\_var=StringVar()

                self.breed\_var=StringVar()

                self.cageno\_var=StringVar()

                self.type\_var=StringVar()

                self.gender\_var=StringVar()

                self.age\_var=StringVar()

                lbltitle=Label(self.root,text="ZOO MANAGEMENT SYSTEM",bd=5,relief=FLAT,

                        bg='#A9907E',fg="#E9E8E8",font=("times new roman",50,"bold"),padx=2,pady=4)

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

                lbltitle.config(highlightbackground="#420b07")

        #DataFrames

                DataFrame = Frame(self.root,bd=15,relief=FLAT,bg="#F3DEBA",padx=20)

                DataFrame.place(x=0,y=100,width=1530,height=400)

                DataFrameLeft=LabelFrame(DataFrame,bd=10,relief=FLAT,padx=140,text="Animal Information",bg="#F3DEBA",font=("times new roman",15,"bold"),labelanchor="n")

                DataFrameLeft.place(x=0,y=5,width=800,height=350)

                DataFrameRight=LabelFrame(DataFrame,bd=10,relief=FLAT,padx=100,bg="#F3DEBA",font=("times new roman",15,"bold"),labelanchor="n")

                DataFrameRight.place(x=810,y=0,width=800,height=400)

        # Images

                img1=Image.open("zoo11.png")

                img1=img1.resize((250,300),Image.LANCZOS)

                self.photoimg1=ImageTk.PhotoImage(img1)

                b1=Button(DataFrame,image=self.photoimg1,borderwidth=0,relief=FLAT,bd=0)

                b1.config(highlightbackground="#95C690")

                b1.place(x=640,y=3)

                # self.audio\_file = AudioSegment.from\_file("zoo12\_audio.mp3")

                # def play\_audio(self):

                #         play(self.audio\_file)

                img2=Image.open("zoo12.png")

                img2=img2.resize((130,90),Image.LANCZOS)

                self.photoimg2=ImageTk.PhotoImage(img2)

                b2=Button(self.root,image=self.photoimg2,borderwidth=0,relief=FLAT,bd=0)

                b2.config(highlightbackground="#95C690")

                b2.place(x=70,y=3)

                img3=Image.open("tiger1.jpg")

                img3=img3.resize((120,90),Image.LANCZOS)

                self.photoimg3=ImageTk.PhotoImage(img3)

                b3=Button(self.root,image=self.photoimg3,borderwidth=0)

                b3.place(x=120,y=427)

                img4=Image.open("jiraffe.jpg")

                img4=img4.resize((120,90),Image.LANCZOS)

                self.photoimg4=ImageTk.PhotoImage(img4)

                b4=Button(self.root,image=self.photoimg4,borderwidth=0)

                b4.place(x=270,y=427)

                img6=Image.open("chimps.jpg")

                img6=img6.resize((120,90),Image.LANCZOS)

                self.photoimg6=ImageTk.PhotoImage(img6)

                b6=Button(self.root,image=self.photoimg6,borderwidth=0)

                b6.place(x=420,y=427)

                img7=Image.open("chitah.jpg")

                img7=img7.resize((120,90),Image.LANCZOS)

                self.photoimg7=ImageTk.PhotoImage(img7)

                b7=Button(self.root,image=self.photoimg7,borderwidth=0)

                b7.place(x=580,y=427)

                img8=Image.open("eagle.jpg")

                img8=img8.resize((120,90),Image.LANCZOS)

                self.photoimg8=ImageTk.PhotoImage(img8)

                b8=Button(self.root,image=self.photoimg8,borderwidth=0)

                b8.place(x=730,y=427)

                img10=Image.open("meerkat.jpg")

                img10=img10.resize((120,90),Image.LANCZOS)

                self.photoimg10=ImageTk.PhotoImage(img10)

                b10=Button(self.root,image=self.photoimg10,borderwidth=0)

                b10.place(x=890,y=427)

                img11=Image.open("panda.jpg")

                img11=img11.resize((120,90),Image.LANCZOS)

                self.photoimg11=ImageTk.PhotoImage(img11)

                b11=Button(self.root,image=self.photoimg11,borderwidth=0)

                b11.place(x=1050,y=427)

                img12=Image.open("elephant.jpg")

                img12=img12.resize((120,90),Image.LANCZOS)

                self.photoimg12=ImageTk.PhotoImage(img12)

                b12=Button(self.root,image=self.photoimg12,borderwidth=0)

                b12.place(x=1200,y=427)

        #buttonsFrame

                ButtonFrame = Frame(self.root,bd=1.4,relief=RIDGE,padx=300,pady=10,bg="#ABC4AA")

                ButtonFrame.place(x=0,y=520,width=1530,height=55)

        # Main buttons

                btnAddData =Button(ButtonFrame,command=self.add\_data,text="ADD ANIMAL",font=("times new roman",12,"bold"),bg="#159895",fg="white",relief=FLAT)

                btnAddData.grid(row=0,column=0)

                btnUPdateAni =Button(ButtonFrame,command=self.Update,text="UPDATE",font=("times new roman",12,"bold"),bg="#4f4d39",fg="white",relief=FLAT)

                btnUPdateAni.grid(row=0,column=1)

                btnDeleteAni =Button(ButtonFrame,command=self.Delete,text="DELETE",font=("times new roman",12,"bold"),bg="#E74646",fg="white",relief=FLAT)

                btnDeleteAni.grid(row=0,column=2)

                btnRestAni =Button(ButtonFrame,command=self.Reset,text="RESET",font=("times new roman",12,"bold"),bg="#767356",fg="white",relief=FLAT)

                btnRestAni.grid(row=0,column=3)

                btnExit =Button(ButtonFrame,command=root.destroy,text="EXIT",font=("times new roman",12,"bold"),bg="#8a8764",fg="white",relief=FLAT)

                btnExit.grid(row=0,column=4)

        #SEARCH BY

                lblsearch=Label(ButtonFrame,font=('times new roman',17,"bold"),text="Search By",padx=2,bg="#F3DEBA",fg="white",relief=FLAT)

                lblsearch.grid(row=0,column=5,sticky=W)

                #variable

                self.search\_var=StringVar()

                search\_combo=ttk.Combobox(ButtonFrame,textvariable=self.search\_var,width=14,font=("arial",13,"bold"))

                search\_combo["values"]=("ID","AnimalName","Breed")

                search\_combo.grid(row=0,column=6)

                search\_combo.current(0)

                self.searchTxt\_var=StringVar()

                txtSearch=Entry(ButtonFrame,textvariable=self.searchTxt\_var,bd=3,relief=RIDGE,width=14,font=("arial",13,"bold"))

                txtSearch.grid(row=0,column=7)

                searchBtn =Button(ButtonFrame,command=self.search\_data,text="SEARCH",font=("times new roman",12,"bold"),bg="#b2ad81",fg="white",relief=FLAT)

                searchBtn.grid(row=0,column=8)

                showAll =Button(ButtonFrame,command=self.fetch\_data,text="SHOW ALL",font=("times new roman",12,"bold"),bg="#c6c190",fg="white",relief=FLAT)

                showAll.grid(row=0,column=9)

                # Login =Button(self.root1,command=self.login,text="LOGIN",font=("times new roman",12,"bold"),bg="#c6c190",fg="white",relief=FLAT)

                # Login.place(x=700,y=700)

        # label And Entry

                lblID = Label(DataFrameRight, font=('times new roman', 17, "bold"),

                text="Rules: \n \n1) Respect the animals \n2) Stay on designated paths and areas \n3) Do not feed the animals \n4) Do not litter \n5) Follow the rules and guidelines \n6) Do not climb fences or enter restricted areas \n7) Do not disturb or touch the exhibits \n\n ENJOY AND VISIT AGAIN !!!!",

                padx=2, bg="#F3DEBA",fg="#A61F69", anchor='nw', justify='left')

                lblID.place(x=0,y=0)

                lblID.pack(fill='both', expand=True)

                lblID.config(width=150, height=150)

                lblName=Label(DataFrameLeft,font=('times new roman',17,"bold"),text="Animal Name",padx=2,bg="#F3DEBA")

                lblName.grid(row=2,column=0,sticky=W)

                txtName=Entry(DataFrameLeft,bd=3,textvariable=self.Aname\_var,relief=RIDGE,width=30,font=("times new roman",13,"bold"))

                txtName.grid(row=2,column=1)

                lblBreed=Label(DataFrameLeft,font=('times new roman',17,"bold"),text="Breed",padx=2,bg="#F3DEBA")

                lblBreed.grid(row=3,column=0,sticky=W)

                txtBreed=Entry(DataFrameLeft,bd=3,textvariable=self.breed\_var,relief=RIDGE,width=30,font=("times new roman",13,"bold"))

                txtBreed.grid(row=3,column=1)

                lblCage=Label(DataFrameLeft,font=('times new roman',17,"bold"),text="Cage No.",padx=2,bg="#F3DEBA")

                lblCage.grid(row=4,column=0,sticky=W)

                txtCage=Entry(DataFrameLeft,bd=3,textvariable=self.cageno\_var,relief=RIDGE,width=30,font=("times new roman",13,"bold"))

                txtCage.grid(row=4,column=1)

                lbltype=Label(DataFrameLeft,font=('times new roman',17,"bold"),text="Type of Animal",padx=2,bg="#F3DEBA")

                lbltype.grid(row=5,column=0,sticky=W)

                Type\_Combo=ttk.Combobox(DataFrameLeft,textvariable=self.type\_var,width=27,font=("times new roman",13,"bold"))

                Type\_Combo["values"]=("LAND","WATER","SKY")

                Type\_Combo.grid(row=5,column=1)

                Type\_Combo.current(0)

                lblGender=Label(DataFrameLeft,font=('times new roman',17,"bold"),text="Gender of Animal",padx=2,bg="#F3DEBA")

                lblGender.grid(row=6,column=0,sticky=W)

                Gender\_Combo=ttk.Combobox(DataFrameLeft,textvariable=self.gender\_var,width=27,font=("times new roman",13,"bold"))

                Gender\_Combo["values"]=("MALE","FEMALE")

                Gender\_Combo.grid(row=6,column=1)

                Gender\_Combo.current(0)

                lblAge=Label(DataFrameLeft,font=('times new roman',17,"bold"),text="Age of Animal",padx=2,bg="#F3DEBA")

                lblAge.grid(row=7,column=0,sticky=W)

                txtAge=Entry(DataFrameLeft,bd=3,textvariable=self.age\_var,relief=RIDGE,width=30,font=("times new roman",13,"bold"))

                txtAge.grid(row=7,column=1)

        # Frame Details

                Framedetails=Frame(self.root,bd=15,relief=FLAT,bg="#675D50")

                Framedetails.place(x=0,y=580,width=1530,height=210)

        # MAin table and Scrollbar

                Table\_Frame=Frame(Framedetails,bd=0,relief=FLAT,bg="#675D50")

                Table\_Frame.place(x=0,y=1,width=1500,height=180)

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

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

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

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

                self.zoo\_table=ttk.Treeview(Table\_Frame,column=("ID","AnimalName","Breed","Cage","Type","Gender","Age"),xscrollcommand=scroll\_x.set,yscrollcommand=scroll\_y.set)

                scroll\_x.config(command=self.zoo\_table.xview)

                scroll\_y.config(command=self.zoo\_table.yview)

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

                self.zoo\_table.heading("ID",text="ID")

                self.zoo\_table.heading("AnimalName",text="Animal Name")

                self.zoo\_table.heading("Breed",text="Breed")

                self.zoo\_table.heading("Cage",text="Cage")

                self.zoo\_table.heading("Type",text="Type")

                self.zoo\_table.heading("Gender",text="Gender")

                self.zoo\_table.heading("Age",text="Age")

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

                self.zoo\_table.column("ID",width=100)

                self.zoo\_table.column("AnimalName",width=100)

                self.zoo\_table.column("Breed",width=100)

                self.zoo\_table.column("Cage",width=100)

                self.zoo\_table.column("Type",width=100)

                self.zoo\_table.column("Gender",width=100)

                self.zoo\_table.column("Age",width=100)

                self.fetch\_data()

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

        # def login(self):

        #         self.root1.withdraw()

        #         self.root.deiconify()

#Main Table

        def add\_data(self):

                if self.Aname\_var.get()=="" or self.breed\_var.get()=="" or self.cageno\_var.get()=="" or self.type\_var.get()=="" or self.gender\_var.get()=="" or self.age\_var.get()=="":

                        messagebox.showerror("Error","All Fields are Required")

                        return

                conn = mysql.connector.connect(host="localhost",username="root",password="9206",database="zoo")

                my\_cursor = conn.cursor()

                my\_cursor.execute("INSERT INTO animal1 (AnimalName, Breed, Cage, Type, Gender, Age) VALUES (%s, %s, %s, %s, %s, %s)",

                                (self.Aname\_var.get(), self.breed\_var.get(), self.cageno\_var.get(), self.type\_var.get(), self.gender\_var.get(), self.age\_var.get()))

                conn.commit()

                conn.close()

                self.fetch\_data()

                messagebox.showinfo("Success","Data Has been Inserted")

        def fetch\_data(self):

                conn=mysql.connector.connect(host="localhost",username="root",password="9206",database="zoo")

                my\_cursor=conn.cursor()

                my\_cursor.execute("select \* from animal1")

                row=my\_cursor.fetchall()

                if len(row)!=0:

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

                    for i in row:

                        self.zoo\_table.insert("",END,values=i)

                    conn.commit()

                conn.close()

        def get\_cursor(self,ev=""):

                cursor\_row=self.zoo\_table.focus()

                content=self.zoo\_table.item(cursor\_row)

                row=content["values"]

                #self.ref\_var.set(row[0]),

                self.Aname\_var.set(row[1]),

                self.breed\_var.set(row[2]),

                self.cageno\_var.set(row[3]),

                self.type\_var.set(row[4]),

                self.gender\_var.set(row[5]),

                self.age\_var.set(row[6])

        def Update(self):

                if self.Aname\_var.get()=="":

                        messagebox.showerror("Error","All fields are Required")

                else:

                        conn=mysql.connector.connect(host="localhost",username="root",password="9206",database="zoo")

                        my\_cursor=conn.cursor()

                        my\_cursor.execute("update animal1 set AnimalName = %s,Breed=%s,Cage=%s,Type=%s,Gender=%s,Age=%s where AnimalName=%s",(

                self.Aname\_var.get(),

                self.breed\_var.get(),

                self.cageno\_var.get(),

                self.type\_var.get(),

                self.gender\_var.get(),

                self.age\_var.get(),

                self.Aname\_var.get()

                #self.ref\_var.get()

                        ))

                        conn.commit()

                        self.fetch\_data()

                        conn.close()

                        messagebox.showinfo("success","Animal has been Updated")

        def Delete(self):

                conn=mysql.connector.connect(host="localhost",username="root",password="9206",database="zoo")

                my\_cursor=conn.cursor()

                sql="delete from animal1 where AnimalName=%s"

                val=(self.Aname\_var.get(),)

                my\_cursor.execute(sql,val)

                conn.commit()

                self.fetch\_data()

                conn.close()

                messagebox.showinfo("Delete","Info deleted successfully")

        def Reset(self):

                #self.ref\_var.set(" "),

                self.Aname\_var.set(" "),

                self.breed\_var.set(" "),

                self.cageno\_var.set(" "),

                #self.type\_var.set(" "),

                #self.gender\_var.set(" "),

                self.age\_var.set(" "),

        def search\_data(self):

                conn=mysql.connector.connect(host="localhost",username="root",password="9206",database="zoo")

                my\_cursor=conn.cursor()

                my\_cursor.execute("SELECT \* FROM animal1 WHERE " + str(self.search\_var.get()) + " LIKE '" + str(self.searchTxt\_var.get()) + "%'")

                rows=my\_cursor.fetchall()

                if len(rows)!=0:

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

                        for i in rows:

                                self.zoo\_table.insert("",END,values=i)

                        conn.commit()

                else:

                        messagebox.showerror("Error","Record not Found")

                        self.searchTxt\_var.set(" ")

                conn.close()

        def exit(self):

               root.destroy()

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

    root = Tk()

    obj = ZoomanagementSystem(root)

    root.mainloop()

For Ticket Window

from tkinter import \*

import tkinter as tk

from PIL import Image, ImageTk

from tkinter import ttk

import mysql.connector

from tkinter import messagebox

from reportlab.lib.pagesizes import letter

from reportlab.pdfgen import canvas

from tkinter import LabelFrame

import mysql.connector

import random

import pytz

from tkcalendar import Calendar

from datetime import datetime

class Ticket\_page:

    def \_\_init\_\_(self, root2):

        self.root2 = root2

        self.root2.title("Ticket Booking")

        self.root2.geometry("1550x800+0+0")

        self.root2.configure(bg="#E96479", relief=FLAT)

        lbltitle=Label(self.root2,text="TICKET BOOKING",bd=5,relief=FLAT,

                        bg='#4D455D',fg="#E9E8E8",font=("times new roman",50,"bold"),padx=2,pady=4)

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

        lbltitle.config(highlightbackground="#420b07")

        DataFrameLeft=LabelFrame(self.root2,bd=10,relief=FLAT,padx=10,font=("times new roman",15,"bold"),labelanchor="n",bg="#BCCEF8")

        DataFrameLeft.place(x=10,y=100,width=400,height=310)

        DataFrameRight=LabelFrame(self.root2,bd=10,relief=FLAT,font=("times new roman",15,"bold"),labelanchor="n",bg="#BCCEF8")

        DataFrameRight.place(x=710,y=100,width=800,height=310)

        DownFrame=LabelFrame(self.root2,bd=10,relief=FLAT,font=("times new roman",15,"bold"),labelanchor="n",bg="#E96479")

        DownFrame.place(x=710,y=450,width=800,height=310)

        BottomFrame=LabelFrame(self.root2,bd=10,relief=FLAT,font=("times new roman",15,"bold"),labelanchor="n",bg="#C0DBEA")

        BottomFrame.place(x=10,y=450,width=400,height=310)

        lbl\_ticket\_info = Label(BottomFrame, font=('times new roman', 17, "bold"),

        text="\n             \* \* \* Rules: \* \* \*\n \n \n1) Purchase tickets in advance \n2) Check for discounts \n3) Read the fine print \n4) Keep your tickets safe \n5) Follow the rules and guidelines \n6) Enjoy your visit!!!!!",

        padx=2, bg="#C0DBEA",fg="#A61F69", anchor='nw', justify='left')

        lbl\_ticket\_info.place(x=0,y=0)

        lbl\_ticket\_info.pack(fill='both', expand=True)

        lbl\_ticket\_info.config(width=150, height=150)

        self.tree = ttk.Treeview(DownFrame, columns=("ID", "Name", "Breed", "Cage No.", "Type"))

        self.tree.heading("#0", text="No.")

        self.tree.heading("ID", text="ID")

        self.tree.heading("Name", text="Name")

        self.tree.heading("Breed", text="Breed")

        self.tree.heading("Cage No.", text="Cage No.")

        self.tree.heading("Type", text="Type")

        self.tree.pack()

        self.tree.place(x=125,y=50)

        self.tree.column("#0", width=50)

        self.tree.column("ID",width=50)

        self.tree.column("Name",width=120)

        self.tree.column("Breed",width=170)

        self.tree.column("Cage No.",width=70)

        self.tree.column("Type",width=70)

        img11=Image.open("zoo28.jpeg")

        img11=img11.resize((280,310),Image.LANCZOS)

        self.photoimg11=ImageTk.PhotoImage(img11)

        b11=Button(self.root2,image=self.photoimg11,borderwidth=0,relief=FLAT)

        b11.place(x=420,y=100)

        img12=Image.open("zoo21.png")

        img12=img12.resize((380,310),Image.LANCZOS)

        self.photoimg12=ImageTk.PhotoImage(img12)

        b12=Button(self.root2,image=self.photoimg12,borderwidth=0,relief=FLAT)

        b12.place(x=420,y=450)

        lbl\_animal\_rightnow = Label(DownFrame, font=('times new roman', 17, "bold"),

        text="\* \* \* ANIMAL INFO \* \* \*",

        padx=2, bg="#E96479",fg="#FBFFB1", anchor='nw', justify='left')

        lbl\_animal\_rightnow.place(x=235,y=0)

        lbl\_animal\_rightnow.config(width=20, height=0)

        lbl\_ticket\_details = Label(DataFrameRight, font=('times new roman', 17, "bold"),

        text="\* \* \* YOUR TICKETS DETAILS WILL BE PRINTED HERE \* \* \*",

        padx=2, bg="#BCCEF8",fg="#10A19D", anchor='nw', justify='left')

        lbl\_ticket\_details.place(x=40,y=0)

        lbl\_ticket\_details.config(width=100, height=0)

        self.label\_ticket\_details = Label(DataFrameRight, font=('times new roman', 17, "bold"),

        padx=2,fg="#A61F69",bg="#BCCEF8", anchor='nw', justify='left')

        self.label\_ticket\_details.place(x=20,y=60)

        self.label\_ticket\_details.config(width=50, height=15)

        self.conn = mysql.connector.connect(host="localhost", user="root", password="9206", database="zoo")

        self.cursor = self.conn.cursor()

        self.cursor.execute("SELECT \* FROM animal1")

        rows = self.cursor.fetchall()

        for i, row in enumerate(rows):

            self.tree.insert("", "end", text=str(i+1), values=row)

        self.label = Label(DataFrameLeft, text="Book Your Ticket",bg="#BCCEF8",fg="#9D3C72", font=('times new roman', 15, "bold"))

        self.label.place(x=80,y=25)

        self.label\_date = Label(DataFrameLeft, text="Date of Visit:",bg="#BCCEF8",fg="#9D3C72", font=('times new roman', 12, "bold"))

        self.label\_date.place(x=10,y=60)

        #self.label\_date.pack(pady=10)

        self.entry\_date = StringVar()

        self.entry\_date\_field = Entry(DataFrameLeft, textvariable=self.entry\_date,width=25)

        self.entry\_date\_field.place(x=150,y=60)

        self.btn\_calendar = ttk.Button(DataFrameLeft, text="Select Date", command=self.select\_date,style="TButton")

        self.btn\_calendar.place(x=175,y=90)

        self.Noticket\_var=StringVar()

        self.label\_tickets = Label(DataFrameLeft, text="Number of Tickets:",bg="#BCCEF8",fg="#9D3C72", font=('times new roman', 12, "bold"))

        self.label\_tickets.place(x=0,y=140)

        style = ttk.Style()

        style.configure("TButton", padding=2, relief="flat", backgroundcolor="red", borderwidth=0, bordercolor="",font=("TImes new roman", 12), foreground="#9E4784", focuscolor="#46C2CB", focusthickness=0, highlightthickness=0, highlightcolor="#46C2CB", borderradius=200)

        self.combobox\_tickets = ttk.Combobox(DataFrameLeft, values=[1, 2, 3, 4, 5],textvariable=self.Noticket\_var,width=23)

        self.combobox\_tickets.place(x=140,y=140)

        self.btn\_book\_tickets = ttk.Button(DataFrameLeft, text="Book Tickets", command=self.add\_data,style="TButton")

        self.btn\_book\_tickets.place(x=175,y=170)

        self.btn\_book\_tickets\_pdf = ttk.Button(DataFrameRight, text="Print Tickets", command=self.print\_ticket\_pdf,style="TButton")

        self.btn\_book\_tickets\_pdf.place(x=30,y=250)

        self.btn\_exit = ttk.Button(DataFrameRight, text="Exit", command=self.root2.destroy,style="TButton")

        self.btn\_exit.place(x=150,y=250)

    def select\_date(self):

        self.calendar\_window = Toplevel(self.root2)

        self.calendar\_window.geometry("+{}+{}".format(420, 100))

        timezone = pytz.timezone('Asia/Kolkata')

        current\_time = datetime.now(timezone)

        current\_date = current\_time.date()

        self.calendar = Calendar(self.calendar\_window, selectmode="day", year=current\_date.year,

                                  month=current\_date.month, day=current\_date.day,

                                  mindate=current\_date, date\_format='y-mm-dd HH:MM:SS')

        self.calendar.pack(padx=20, pady=10)

        self.btn\_confirm\_date = ttk.Button(self.calendar\_window, text="Confirm Date", command=self.confirm\_date,style="TButton")

        self.btn\_confirm\_date.pack(pady=10)

    def confirm\_date(self):

        # get the selected date and time in the Asia/Kolkata timezone

        selected\_date = self.calendar.selection\_get()

        timezone = pytz.timezone('Asia/Kolkata')

        current\_time = datetime.now(timezone)

        selected\_date = datetime(

            year=selected\_date.year,

            month=selected\_date.month,

            day=selected\_date.day,

            hour=current\_time.hour,

            minute=current\_time.minute,

            second=current\_time.second,

            tzinfo=timezone

        )

        self.entry\_date.set(selected\_date.strftime("%Y-%m-%d %H:%M:%S"))

        self.calendar\_window.destroy()

    def add\_data(self):

        if self.entry\_date.get()=="" or self.combobox\_tickets.get()=="" :

                messagebox.showerror("Error","All Fields are Required")

                return

        conn = mysql.connector.connect(host="localhost",username="root",password="9206",database="zoo")

        my\_cursor = conn.cursor()

        my\_cursor.execute("INSERT INTO tickets (Date, NoTickets) VALUES (%s, %s)",

                        (self.entry\_date.get(), self.Noticket\_var.get()))

        conn.commit()

        conn.close()

        messagebox.showinfo("Success","Ticket Booked Successfully !!!")

        self.print\_ticket()

    def calculate\_ticket\_price(self,num\_tickets):

        ticket\_price = 10

        total\_price = num\_tickets \* ticket\_price

        return total\_price

    def print\_ticket(self):

        self.ticket\_no = random.randint(1, 100)

        date = datetime.strptime(self.entry\_date.get(), '%Y-%m-%d %H:%M:%S')

        self.date\_str = date.strftime('%A, %B %d, %Y booked at %I:%M %p')

        self.num\_tickets = int(self.combobox\_tickets.get())

        self.perticket =10

        self.ticket\_price = self.calculate\_ticket\_price(self.num\_tickets)

        ticket\_details = f"Ticket\_No.:      {self.ticket\_no}\nDate of Visit:    {self.date\_str}\nNo.of Tickets:   {self.num\_tickets}\nPrice per Ticket: ₹{self.perticket}\nPrice of Ticket: ₹{self.ticket\_price}"

        self.label\_ticket\_details.configure(text="Your Ticket Details:\n-----------------------------------------\n"+ticket\_details)

    def print\_ticket\_pdf(self):

        # Set up canvas

        c = canvas.Canvas("ticket.pdf", pagesize=letter)

        # Draw ticket content

        c.setFont("Helvetica", 16)

        c.drawString(200, 750,"Zoo management System")

        c.drawString(100, 700, "Ticket ID:  " + str(self.ticket\_no))

        c.drawString(100, 650, "------------------------------------------------------------------------------------")

        c.drawString(100, 600, "Date of Visit:  " + self.date\_str)

        c.drawString(100, 550, "Ticket Price: Rs.  " + str(self.ticket\_price)+"/-")

        c.drawString(100, 500, "No. of Tickets:  " + str(self.num\_tickets))

    # Save PDF file

        c.save()

        messagebox.showinfo("Success","Ticket Downloaded Successfully!!")

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

    root2 = Tk()

    obj = Ticket\_page(root2)

    root2.mainloop()

flowchart:

**RESULTS**:

1) Efficient and accurate record-keeping for animal care and veterinary needs.

2) Streamlined ticketing and admission processes for visitors.

3) Simplified membership and donor management.

4) Enhanced staff scheduling and task assignment capabilities.

5) Improved inventory tracking and procurement for supplies and equipment.

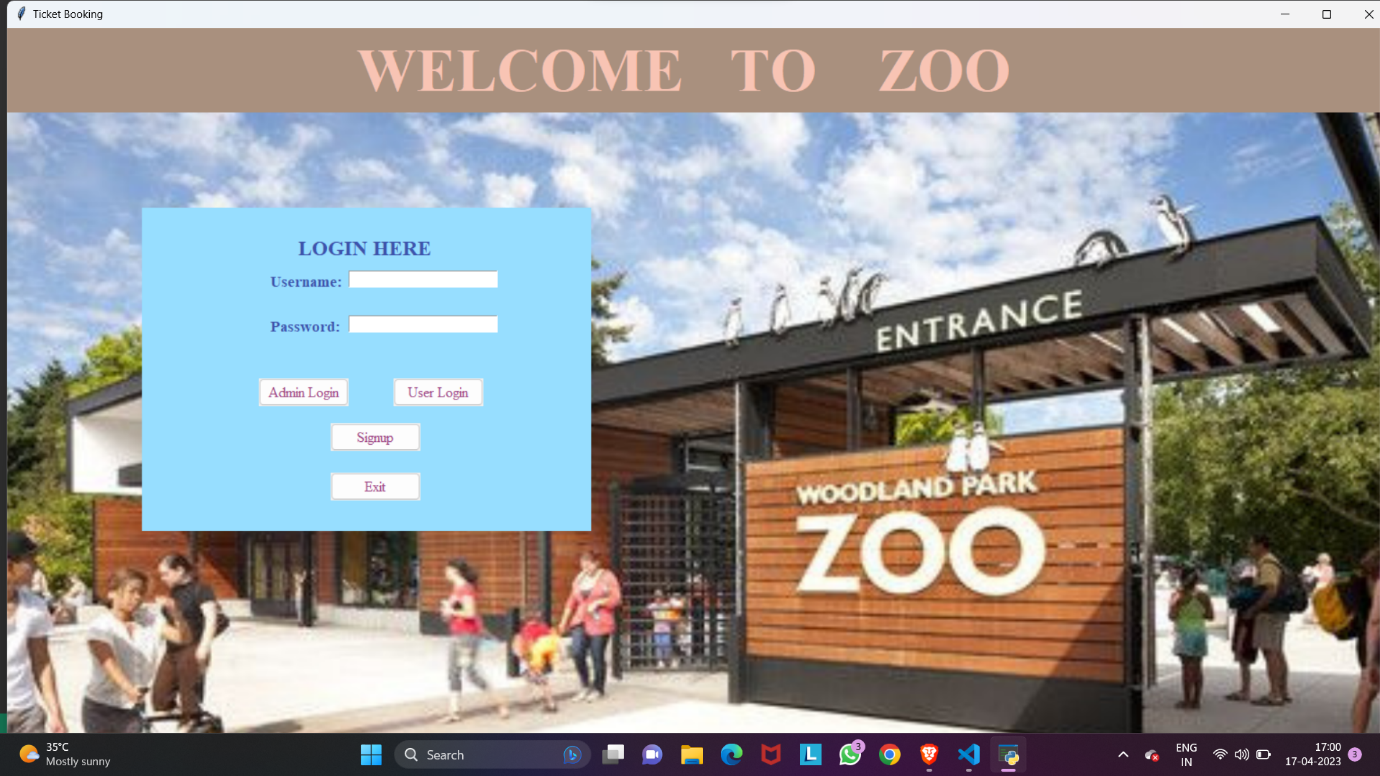
6) Better financial reporting and forecasting.

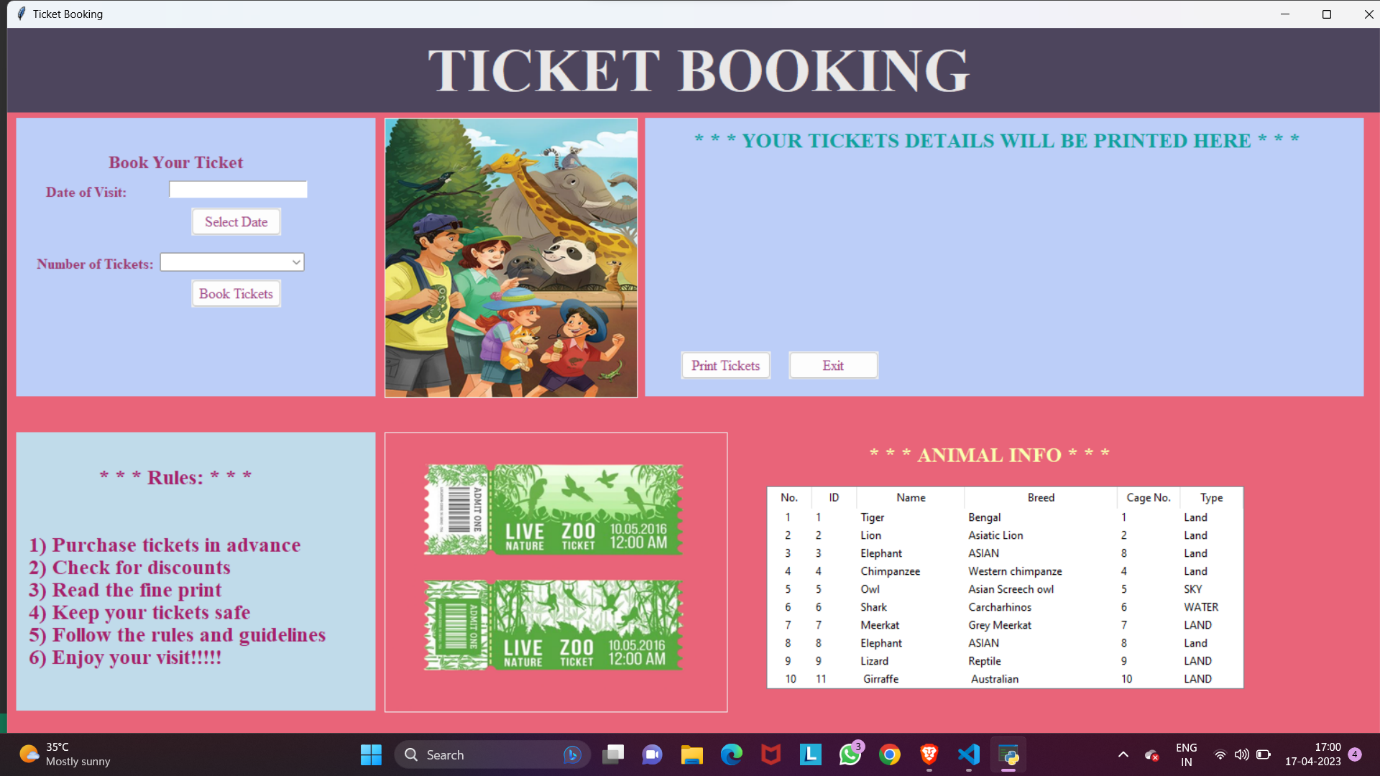
7) Increased operational visibility and control.

8) Improved communication between departments and staff.

9) Greater ability to analyze and act on visitor data and feedback.

10) Enhanced security and safety measures for both staff and animals.

**OUTPUT**: 





**CONCLUSION**:

A well-designed zoo management system is essential for the efficient and effective management of a zoo. It enables zoos to maintain high standards of animal care, ensure visitor safety and satisfaction, and streamline administrative and operational processes.

A comprehensive zoo management system should include modules for visitor management, animal management, staff management, facility management, and reporting and analytics. These modules should be interconnected to form a flowchart that allows for smooth data flow, decision-making, and action-taking.

Some of the key benefits of a zoo management system include improved animal welfare, enhanced visitor experience, increased operational efficiency, better financial management, and improved reporting and analytics. Overall, a zoo management system can help zoos to achieve their goals and objectives while ensuring the long-term sustainability of the organization

**REFERENCE**:

<https://www.researchgate.net/publication/309459560_Zoo_Information_Management_System_ZIMS_for_Anna_Zoological_Park_Chennai_India>

https://www.scribd.com/document/510655264/Zoo-Management-System-1#

<https://www.lovelycoding.org/zoo-management-system/>