SAVITRIBAI PHULE PUNE UNIVERSITY

A PRELIMINARY PROJECT REPORT ON

"TEMPLE MANAGEMENT SYSTEM"

SUBMITTED TOWARDS THE PARTIAL FULFILMENT OF THE REQUIREMENTS

OF

BACHELOR OF ENGINEERING (TE COMPUTER ENGINEERING)

Academic Year: 2018-19

By:

1. **Aditi Deshpande (BECOA125)**
2. **Anand Kulkarni (BECOA160)**

Under The Guidance of

Prof. Anand Birajdar

DEPARTMENT OF COMPUTER ENGINEERING,

PIMPRI CHINCHWAD COLLEGE OF ENGINEERING

SECTOR 26, NIGDI, PRADHIKARAN

PIMPRI CHINCHWAD COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that, the project entitled

"TEMPLE MANAGEMENT SYSTEM"

successfully submitted by following students of "PCET's Pimpri Chinchwad College of Engineering, Nigdi, Pune-44" as a part of mini project

Under the guidance of Prof. Rajesh Lomte.

In the partial fulfillment of the requirements for the T.E. (Computer Engineering)

1. Aditi Deshpande BECOA125
2. Anand Kulkarni BECOA160

Prof. Anand Birajdar

Project Guide

ABSTRACT

Temples are also being an important part of Indian Culture. The mob in the temples is increasing so there should be efficient management of mob in the temples to avoid stampedes and life loss.

Temple Management System is a highly scalable product for Hindu Temples. It offers a complete automation in billing and accounts, inventory, staff management and administration sections. It helps temple administrators to assure effective devotee service.

Temple Management solution is bundled with an excellent accounting package which enables easy operation and efficient reporting. It's available with desktop version support to allow devotees to make online offerings via through wallets or by cash, seva bookings, and many more useful features.

INDEX

INTRODUCTION

PROBLEM STATEMENT

To develop a software application to manage challenges faced by the temple administration of keeping track of all activities in the temple and providing reliable services to the devotees.

PROJECT IDEA

The idea is to automate the activities that are being carried out in the temples and to keep track of them.

MOTIVATION

It was our general observation that the temples are always overcrowded and there is lack of management. Also there is need to manage the resources to its fullest. Thus we came up with an idea to develop a software application to manage challenges faced by the temple administration of keeping track of all activities in the temple and providing reliable services to the devotees.

SCOPE

Temple Management & Tracking System is a software application for temples. It is designed to cater complete management requirements and automation of temples.

The application is made for streamlining all the activities associated with the temple under a common umbrella and cut down on the time and effort required to run the temple.

The system is ideal for all small/big temples having large number of devotees, employees(workers),accounts and payroll and a huge amount of donations.

LITERATURE SURVEY/ REQUIREMENT ANALYSIS

- Problems in traditional way:-

The traditional method that is used in temples is the use of paper receipts, tokens for darshan, seva details, etc and also registerbook for the salary and payroll, salary, attendance, etc documentation.

- Overcoming the problem:-

To tackle the above scenario the application will help the store the data on a database that can be accessed at any moment of time and manage the resources efficiently and effectively.

PROJECT DESIGN

SYSTEM REQUIREMENTS

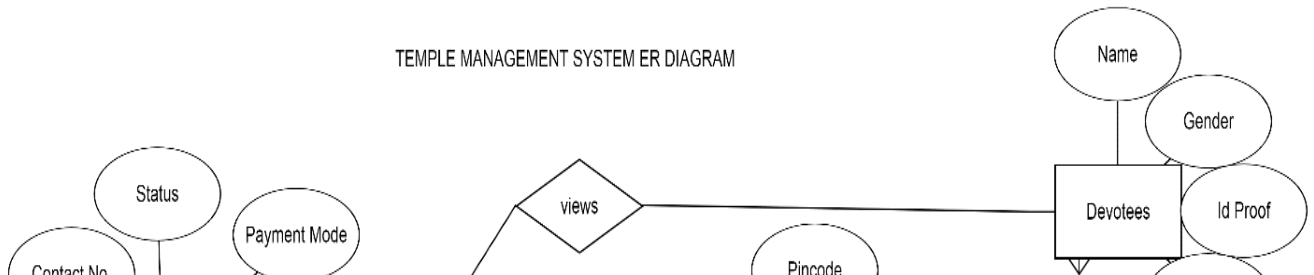
1). Hardware requirements :-

- Intel core i3 processor.
- Memory at least 1 GB ram is used.
- Hard disk space minimum 1 GB for database usage.

2). Software requirements :-

- Operating System Windows.
- Python programming language support.
- Database Oracle-SQL
- Python packages pandas, matplotlib, numpy, tkinter

ER MODEL/DIAGRAM



SCHEMA OF MODULES

- Schema of Devotee Module

SQL> describe devotee;

Name	Null?	Type
ENTRY		VARCHAR2(10)
DEVOTEENAME		VARCHAR2(30)
AGE		NUMBER(3)
GENDER		VARCHAR2(12)
IDPROOF		NUMBER(12)

- Schema of Account and Payroll Module

SQL> describe accountandpayroll;

Name	Null?	Type
PAYMENT_DATE		DATE
ACCOUNTHOLDERNAME		VARCHAR2(30)
CONTACTNO		NUMBER(10)
EMAILID		VARCHAR2(15)
PAYMODE		VARCHAR2(6)
STATUS		VARCHAR2(8)

- Schema of Seva module

SQL> describe seva;

Name	Null?	Type
SEVA_NAME		VARCHAR2(30)
SEVA_DESCRIPTION		VARCHAR2(20)
DURATION		VARCHAR2(15)
FROM_DATE		DATE
TO_DATE		DATE

- Schema of Inventory Module

SQL> describe stock;

Name	Null?	Type
STOCK_DATE		VARCHAR2(10)
TIMEOF		VARCHAR2(16)
QUANTITY		VARCHAR2(10)
UNITS		NUMBER(10)
EXPINCOME		VARCHAR2(20)

- Schema of Add User Module

SQL> describe user_registration;

Name	Null?	Type
ACCOUNTCREATION		DATE
FIRSTNAME		VARCHAR2(10)
LASTNAME		VARCHAR2(10)
DATEOFBIRTH		VARCHAR2(10)
GENDER		VARCHAR2(12)
USERNAME		VARCHAR2(15)
PASSWORD		VARCHAR2(15)
ADDRESS		VARCHAR2(30)
MOBILENO		NUMBER(10)
CITY		VARCHAR2(15)
PINCODE		NUMBER(10)
STATE		VARCHAR2(15)
COUNTRY		VARCHAR2(15)

AADHARNO NUMBER(12)

- Schema of Donation Module

SQL>describe donation;

Name	Null?	Type
PAY_DATE		DATE
NAME		VARCHAR2(15)
AMOUNT		NUMBER(10)
PAYMENT_MODE		VARCHAR2(15)

NORMALISATION

A database is in second normal form if it satisfies the following conditions:

a) It is in first normal form

b) It is in second normal form

- All non-key attributes are fully functional dependent on the primary key

In a table, if attribute B is functionally dependent on A, but is not functionally dependent on a proper subset of A, then B is considered fully functional dependent on A. Hence, in a 2NF table, all non-key attributes cannot be dependent on a subset of the primary key. Note that if the primary key is not a composite key, all non-key attributes are always fully functional dependent on the primary key. A table that is in 1st normal form and contains only a single key as the primary key is automatically in 2nd normal form.

HOURS ESTIMATION

The time required to complete our project was around 1 week.

MODULE DESCRIPTION

- **Devotee Management Module**

  Devotees of Temple can be managed with the help of this module as, Entry of new devotee with their detail profile & contact Information can be done through this module, and information will be stored in data base for further use & avoid duplication. Service features are as;

  - Add New Devotees
  - Information of all Devotees
  - Information of all activities of devotees

- **Accounts & Payroll Management Module**
  - Payroll Module includes the all the details regarding the employee. It includes employee details, their salary/wage details, advance payment details and attendance details of employees.
  - Account module includes the details of all accounting transactions in the Temple. It includes journal book, voucher, bank details, bhandaram details and receipts and payment.

- **Donation Management Module**
  - Module includes name of donar, amount and payment mode details.

- **Inventory Management Module**
  - Inventory Module includes prasad quantity ,units ,total expected income and stock adjustment details.

- **Seva Module**
  - SEVA of temple can be managed using this module in which different kind of features is available as;
  - Add new SEVA and charge of that SEVA
  - No. of Person allowed in SEVA
  - Schedule of SEVA

- **Add User Module**
  - Administration will register the user who are going to actually use the system so that no unauthorized user can access and manipulate information.

RESULTS AND DISCUSSION

SOURCE CODE

```
from tkinter import *
from tkinter import messagebox as ms
from PIL import ImageTk, Image
import cx_Oracle as cx
import datetime as dt
import time
import tkinter.font as tkFont
import tkinter.ttk as ttk

# ======================Main Window=======================================#

root = Tk()
width = 830
height = 510
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()
x = (screen_width / 2) - (width / 2)
y = (screen_height / 2) - (height / 2)
root.resizable(0, 0)
root.geometry("%dx%d+%d+%d" % (width, height, x, y))


# ==============================Background Image================================#
img = ImageTk.PhotoImage(Image.open("temple1.jpg"))
bg_image = Label(root, image=img)
bg_image.pack()

img1 = ImageTk.PhotoImage(Image.open("admin.png"))
bg_image1 = Label(root, image=img1, width=55, height=55)

bg_image1.place(x=560, y=150)
img2 = ImageTk.PhotoImage(Image.open("user.png"))
bg_image2 = Label(root, image=img2, width=54, height=52)

bg_image2.place(x=561, y=207)
Top = Frame(root, bd=4, relief=FLAT, bg='white', width=100)
Top.place(x=330, y=15)
Form = Frame(root, height=800, width=200)
Form.place(x=620, y=150)


#
================================================HomPage=============================
================#
def main_form():
    lbl_title = Label(Top, text="WELCOME", font=("Helvetica", 20), width=15)
    lbl_title.grid(row=0,column=2)
    btn_admin = Button(Form, text="Admin", font=('arial',
19),pady=5,width=10,command=admin_form)
```

```
    btn_admin.grid(row=0)
    btn_user = Button(Form, text="User", font=('arial',
19),pady=5,width=10,command=user_form)
    btn_user.grid(row=2)

# ==================================End of
HomePage=============================================================#


# ==================================Admin
Login=============================================================#
def admin_login():
    if emailid.get() == "admin" and password.get() == "admin":
        emailid.delete("0", END)
        password.delete("0", END)
        Admin_First_Page()
    else:
        ms.showerror("Invalid EmailId or password")
        emailid.delete("0", END)
        password.delete("0", END)


def backtoroot():
    Admin_pg.destroy()
    root.deiconify()


def admin_form():
    root.withdraw()
    global Admin_pg
    Admin_pg = Toplevel(root)

    img3 = ImageTk.PhotoImage(Image.open("temple1.jpg"))
    bg_image3 = Label(Admin_pg, image=img3)
    bg_image3.place(x=0,y=0)

    width = 830
    height = 511
    screen_width = Admin_pg.winfo_screenwidth()
    screen_height = Admin_pg.winfo_screenheight()
    x = (screen_width / 2) - (width / 2)
    y = (screen_height / 2) - (height / 2)
    Admin_pg.resizable(0, 0)
    Admin_pg.geometry("%dx%d+%d+%d" % (width, height, x, y))

    Label(Admin_pg, text="Admin Login", justify=CENTER, font=("Showcard Gothic",
20)).place(x=320, y=60)

    global emailid
    Label(Admin_pg, text="UserName: ", font=('', 16)).place(x=200, y=150)
    emailid = Entry(Admin_pg, bd=3, font=('', 16))
    emailid.place(x=320, y=145)

    global password
    Label(Admin_pg, text="Password: ", font=('', 16)).place(x=200, y=210)
    password = Entry(Admin_pg, bd=3, font=('', 16), show='*')
    password.place(x=320, y=205)

    Button(Admin_pg, text="Login", font=('', 16), width=10, bg="purple1",
fg='white', command=admin_login).place(x=250, y=270)
    Button(Admin_pg, text="Back", font=('', 16), width=10, bg="purple1",
fg='white', command=backtoroot).place(x=420, y=270)
    Admin_pg.mainloop()
# ==========================================Admin Login
End=============================================================#


# ================================Page After Admin
```

```
Login=============================================================#
def Backto_adminlogin():
    admin_first_page.destroy()
    Admin_pg.deiconify()

def Admin_First_Page():
    Admin_pg.withdraw()
    global admin_first_page

    admin_first_page = Toplevel(root)

    img3 = ImageTk.PhotoImage(Image.open("temple1.jpg"))
    bg_image3 = Label(admin_first_page, image=img3)
    bg_image3.place(x=0, y=0)

    width = 830
    height = 511
    screen_width = admin_first_page.winfo_screenwidth()
    screen_height = admin_first_page.winfo_screenheight()
    x = (screen_width / 2) - (width / 2)
    y = (screen_height / 2) - (height / 2)
    admin_first_page.resizable(0, 0)
    admin_first_page.geometry("%dx%d+%d+%d" % (width, height, x, y))

    Button(admin_first_page, text="Add User", bg='white', font=('Helvetica', '12',
"bold italic"),
           command=adduser).place(x=320, y=60, width=220, height=50)
    Button(admin_first_page, text="View Devotees", bg='white', font=('Helvetica',
'12', "bold italic"),
           command=displaydevotees).place(x=320, y=120, width=220, height=50)
    Button(admin_first_page, text="Accounts And Payroll", bg='white',
font=('Helvetica', '12', "bold italic"),
           command=displayAccount).place(x=320, y=180, width=220, height=50)
    Button(admin_first_page, text="Inventory Status", bg='white',
font=('Helvetica', '12', "bold italic"),
           command=displayinventory).place(x=320, y=240, width=220, height=50)
    Button(admin_first_page, text="Donation Details", bg='white',
font=('Helvetica', '12', "bold italic"),
           command=displayDonation).place(x=320, y=300, width=220, height=50)
    Button(admin_first_page, text="Seva Details", bg='white', font=('Helvetica',
'12', "bold italic"),
           command=displaySeva).place(x=320, y=360, width=220, height=50)
    Button(admin_first_page, text="Back", bg='white', font=('Helvetica', '12',
"bold italic"),
           command=Backto_adminlogin).place(x=320, y=420, width=220, height=50)
    admin_first_page.mainloop()
# ==================================End of Admin
=====================================================================#


# ====================================User
Login=============================================================================#
def user_login():
    userloginconn = cx.connect("System", "oracle", "XE")
    userlogincursor = userloginconn.cursor()
    UserLoginUsername = entry_userpg_username.get()
    UserLoginPassword = entry_userpg_password.get()
    userlogincursor.execute("select password from user_registration where
username=:k", {'k': UserLoginUsername})
    Userpassword = userlogincursor.fetchone()
    if UserLoginPassword == Userpassword[0]:
        ms.showinfo("Login Success", "Login Successful" + "\nGood Day")
        entry_userpg_username.delete("0", END)
        entry_userpg_password.delete("0", END)
        User_First_Page()
    else:
        ms.showinfo("Login Failed", "Enter Valid Username and Password" + "\nTry
Again...!")
```

```
        entry_userpg_username.delete("0", END)
        entry_userpg_password.delete("0", END)
        userloginconn.close()


def Backtoroot():
    User_pg.destroy()
    root.deiconify()


def user_form():
    root.withdraw()
    global User_pg
    User_pg = Toplevel(root)

    img3 = ImageTk.PhotoImage(Image.open("temple1.jpg"))
    bg_image3 = Label(User_pg, image=img3)
    bg_image3.place(x=0,y=0)

    width = 830
    height = 511
    screen_width = User_pg.winfo_screenwidth()
    screen_height = User_pg.winfo_screenheight()
    x = (screen_width / 2) - (width / 2)
    y = (screen_height / 2) - (height / 2)
    User_pg.resizable(0, 0)
    User_pg.geometry("%dx%d+%d+%d" % (width, height, x, y))

    Label(User_pg, text="User Login", justify=CENTER, font=("Showcard Gothic",
20)).place(x=320, y=60)

    global entry_userpg_username
    Label(User_pg, text="UserName: ", font=('', 16)).place(x=200, y=150)
    entry_userpg_username = Entry(User_pg, bd=3, font=('', 16))
    entry_userpg_username.place(x=320, y=145)

    global entry_userpg_password
    Label(User_pg, text="Password: ", font=('', 16)).place(x=200, y=210)
    entry_userpg_password = Entry(User_pg, bd=3, font=('', 16), show='*')
    entry_userpg_password.place(x=320, y=205)

    Button(User_pg, text="Login", font=('', 16), width=10, bg="purple1",
fg='white', command=user_login).place(x=250, y=270)
    Button(User_pg, text="Back", font=('', 16), width=10, bg="purple1", fg='white',
command=Backtoroot).place(x=420, y=270)
    User_pg.mainloop()
# ============================================End User
Login============================================================#


# ============================================User First
Page============================================================#
def backto_userlogin():
    user_first_page.destroy()
    User_pg.deiconify()

def User_First_Page():
    User_pg.withdraw()
    global user_first_page
    user_first_page = Toplevel(root)

    img3 = ImageTk.PhotoImage(Image.open("temple1.jpg"))
    bg_image3 = Label(user_first_page, image=img3)
    bg_image3.place(x=0,y=0)

    width = 830
    height = 511
    screen_width = user_first_page.winfo_screenwidth()
```

```
    screen_height = user_first_page.winfo_screenheight()
    x = (screen_width / 2) - (width / 2)
    y = (screen_height / 2) - (height / 2)
    user_first_page.resizable(0, 0)
    user_first_page.geometry("%dx%d+%d+%d" % (width, height, x, y))

    Button(user_first_page, text="Add Devotees", bg='white', font=('Helvetica',
'12', "bold italic"),
           command=addDevotees).place(x=320, y=60, width=220, height=50)
    Button(user_first_page, text="Enter Accounts And Payroll", bg='white',
font=('Helvetica', '12', "bold italic"),
           command=account_and_payroll).place(x=320, y=120, width=220, height=50)
    Button(user_first_page, text="Enter Inventory Details", bg='white',
font=('Helvetica', '12', "bold italic"),
           command=inventory).place(x=320, y=180, width=220, height=50)
    Button(user_first_page, text="Enter Donation Details", bg='white',
font=('Helvetica', '12', "bold italic"),
           command=donation).place(x=320, y=240, width=220, height=50)
    Button(user_first_page, text="Enter Seva Details", bg='white',
font=('Helvetica', '12', "bold italic"),
           command=seva).place(x=320, y=300, width=220, height=50)
    Button(user_first_page, text="Back", bg='white', font=('Helvetica', '12', "bold
italic"),
           command=backto_userlogin).place(x=320, y=360, width=220, height=50)
    user_first_page.mainloop()
# ================================================Endo of User First
Page=============================================#


# ========================================Add Devotees
Page=========================================================#
def submitAddDevotee():
    devoteeconn = cx.connect("System", "oracle", "XE")
    devoteecursor = devoteeconn.cursor()
    devotee_name = entry_devotee_name.get()
    devotee_age = entry_devotee_age.get()
    devotee_gender = entry_devotee_gender.get()
    devotee_idproof = entry_devotee_idproof.get()
    devoteecursor.execute("insert into devotee values(:ed,:dn,:da,:dg,:di)", {'ed':
dt.datetime.today(),
                  'dn': devotee_name,'da': devotee_age, 'dg': devotee_gender,
'di': devotee_idproof})
    devoteeconn.commit()
    devoteeconn.close()


def devoteesto_userfirstpage():
    devotee_pg.destroy()
    user_first_page.deiconify()


def resetDevotees():
    entry_devotee_name.delete("0", END)
    entry_devotee_age.delete("0", END)
    entry_devotee_gender.delete("0", END)
    entry_devotee_idproof.delete("0", END)


def addDevotees():
    user_first_page.withdraw()
    global devotee_pg
    devotee_pg = Toplevel(root)

    img3 = ImageTk.PhotoImage(Image.open("temple1.jpg"))
    bg_image3 = Label(devotee_pg, image=img3)
    bg_image3.place(x=0,y=0)

    width = 830
```

```
    height = 511
    screen_width = devotee_pg.winfo_screenwidth()
    screen_height = devotee_pg.winfo_screenheight()
    x = (screen_width / 2) - (width / 2)
    y = (screen_height / 2) - (height / 2)
    devotee_pg.resizable(0, 0)
    devotee_pg.geometry("%dx%d+%d+%d" % (width, height, x, y))

    Label(devotee_pg, text="DEVOTEES MODULE", font=("Showcard Gothic",
20)).place(x=320, y=70)

    global entry_devotee_name
    Label(devotee_pg, text='DEVOTEE NAME :', font=('', 13)).place(x=230, y=140)
    entry_devotee_name = Entry(devotee_pg, font=('', 14))
    entry_devotee_name.place(x=400, y=140)

    global entry_devotee_age
    Label(devotee_pg, text='AGE :', font=('', 13)).place(x=230, y=170)
    entry_devotee_age = Entry(devotee_pg, font=('', 14))
    entry_devotee_age.place(x=400, y=170)

    global entry_devotee_gender
    Label(devotee_pg, text='GENDER :', font=('', 13)).place(x=230, y=200)
    entry_devotee_gender = Entry(devotee_pg, font=('', 14))
    entry_devotee_gender.place(x=400, y=200)

    global entry_devotee_idproof
    Label(devotee_pg, text='ID PROOF :', font=('', 13)).place(x=230, y=230)
    entry_devotee_idproof = Entry(devotee_pg, font=('', 14))
    entry_devotee_idproof.place(x=400, y=230)

    Button(devotee_pg, text='SUBMIT', width=10, font=('', 13),
command=submitAddDevotee).place(x=270, y=290)
    Button(devotee_pg, text='CANCEL', width=10, font=('', 13),
command=devoteesto_userfirstpage).place(x=370, y=290)
    Button(devotee_pg, text='RESET', width=10, font=('', 13),
command=resetDevotees).place(x=470, y=290)
    devotee_pg.mainloop()
# ======================================End of Add Devotees
Page=========================================================#


# ======================================Display Devotees
List=========================================================#
def dispdevoteeto_adminfirstpage():
    dispdovotee_pg.destroy()
    admin_first_page.deiconify()

def displaydevotees():
    admin_first_page.withdraw()
    global dispdovotee_pg
    dispdovotee_pg = Toplevel(root)

    width = 830
    height = 511
    screen_width = dispdovotee_pg.winfo_screenwidth()
    screen_height = dispdovotee_pg.winfo_screenheight()
    x = (screen_width / 2) - (width / 2)
    y = (screen_height / 2) - (height / 2)
    dispdovotee_pg.resizable(0, 0)
    dispdovotee_pg.geometry("%dx%d+%d+%d" % (width, height, x, y))

    devotee_header = ['Visit Date', 'Devotee Name', 'Age', 'Gender', 'Id Proof']

    tree = None
    s = "\Click on header to sort by that column to change width of column drag
boundary"
```

```
    msg = ttk.Label(dispdovotee_pg, justify="left", anchor="n", padding=(10, 2, 10,
6), text=s)
    msg.pack(fill='x')

    container = ttk.Frame(dispdovotee_pg, width=180)
    container.pack(fill='both', expand=True)

    # create a treeview with dual scrollbars
    tree = ttk.Treeview(dispdovotee_pg, columns=devotee_header, show="headings",
height=20)
    vsb = ttk.Scrollbar(dispdovotee_pg, orient="vertical", command=tree.yview)
    hsb = ttk.Scrollbar(dispdovotee_pg, orient="horizontal", command=tree.xview)
    tree.configure(yscrollcommand=vsb.set, xscrollcommand=hsb.set)
    tree.grid(column=0, row=0, sticky='nsew', in_=container)
    vsb.grid(column=1, row=0, sticky='ns', in_=container)
    hsb.grid(column=0, row=1, sticky='ew', in_=container)
    container.grid_columnconfigure(0, weight=1)
    container.grid_rowconfigure(0, weight=1)

    for col in devotee_header:
        tree.heading(col, text=col.title(), command=lambda c=col: sortby(tree, c,
0))
        # adjust the column's width to the header string
        tree.column(col, width=tkFont.Font().measure(col.title()))

    dispdevoteeconn = cx.connect("System", "oracle", "XE")
    dispdevoteecursor = dispdevoteeconn.cursor()
    dispdevoteecursor.execute("select * from devotee")
    devotee_list = dispdevoteecursor.fetchall()
    for item in devotee_list:
        tree.insert('', 'end', values=item)
        # adjust column's width if necessary to fit each value
        for ix, val in enumerate(item):
            col_w = tkFont.Font().measure(val)
            if tree.column(devotee_header[ix], width=None) < col_w:
                tree.column(devotee_header[ix], width=col_w)

    Button(dispdovotee_pg, text="Back", bg='purple1', fg='white', font=(" ", 15),
command=dispdevoteeto_adminfirstpage).pack(fill='x')
    dispdovotee_pg.mainloop()
# ====================================End of Display Devotees
List=========================================================#


#=========================================Enter Accounts And Payroll
Page=========================================#
def reset_account_and_payroll():
    entry_acc_name.delete("0", END)
    entry_contactno.delete("0", END)
    entry_email_id.delete("0", END)


def account_and_payroll_to_userfirstpage():
    AccPayroll_pg.destroy()
    user_first_page.deiconify()


def account_and_payroll():
    user_first_page.withdraw()
    global AccPayroll_pg
    AccPayroll_pg = Toplevel(root)

    img3 = ImageTk.PhotoImage(Image.open("temple1.jpg"))
    bg_image3 = Label(AccPayroll_pg, image=img3)
    bg_image3.place(x=0,y=0)

    screen_width = AccPayroll_pg.winfo_screenwidth()
    screen_height = AccPayroll_pg.winfo_screenheight()
```

```
    x = (screen_width / 2) - (width / 2)
    y = (screen_height / 2) - (height / 2)
    AccPayroll_pg.resizable(0, 0)
    AccPayroll_pg.geometry("%dx%d+%d+%d" % (width, height, x, y))

    def accountsandpayroll():
        accountsandpayrollconn = cx.connect("System", "oracle", "XE")
        accountsandpayrollcursor = accountsandpayrollconn.cursor()
        acc_name = entry_acc_name.get()
        contactno = entry_contactno.get()
        mailid = entry_email_id.get()
        date = dt.datetime.today()
        if v.get() == 1 and v1.get() == 1:
            accountsandpayrollcursor.execute("insert into accountandpayroll
values(:apd,:acn,:cno,:mid,:pam,:stat)",
                                             {'apd': date, 'acn': acc_name, 'cno':
contactno, 'mid': mailid,
                                              'pam': "CASH", 'stat': "PAYED"})
            accountsandpayrollconn.commit()
            accountsandpayrollconn.close()
        elif v.get() == 1 and v1.get() == 2:
            accountsandpayrollcursor.execute("insert into accountandpayroll
values(:apd,:acn,:cno,:mid,:pam,:stat)",
                                             {'apd': date, 'acn': acc_name, 'cno':
contactno, 'mid': mailid,
                                              'pam': "CASH", 'stat': "UNPAYED"})
            accountsandpayrollconn.commit()
            accountsandpayrollconn.close()
        elif v.get() == 2 and v1.get() == 1:
            accountsandpayrollcursor.execute("insert into accountandpayroll
values(:apd,:acn,:cno,:mid,:pam,:stat)",
                                             {'apd': date,
                                              'acn': acc_name, 'cno': contactno,
'mid': mailid, 'pam': "CHEQUE",
                                              'stat': "PAYED"})
            accountsandpayrollconn.commit()
            accountsandpayrollconn.close()
        else:
            accountsandpayrollcursor.execute("insert into accountandpayroll
values(:apd,:acn,:cno,:mid,:pam,:stat)",
                                             {'apd': date, 'acn': acc_name, 'cno':
contactno, 'mid': mailid,
                                              'pam': "CHEQUE", 'stat': "UNPAYED"})
            accountsandpayrollconn.commit()
            accountsandpayrollconn.close()

    Label(AccPayroll_pg, text="ACCOUNTS AND PAYROLL\nMODULE", font=("Showcard
Gothic", 20)).place(x=250, y=20)

    global entry_acc_name
    Label(AccPayroll_pg, text='NAME ', font=('', 13)).place(x=220, y=115)
    entry_acc_name = Entry(AccPayroll_pg, font=('', 13), width=25)
    entry_acc_name.place(x=350, y=115)

    global entry_contactno
    Label(AccPayroll_pg, text='CONTACT NO ', font=('', 13)).place(x=220, y=150)
    entry_contactno = Entry(AccPayroll_pg, font=('', 13), width=25)
    entry_contactno.place(x=350, y=150)

    global entry_email_id
    Label(AccPayroll_pg, text='EMAIL ID ', font=('', 13)).place(x=220, y=185)
    entry_email_id = Entry(AccPayroll_pg, font=('', 13), width=25)
    entry_email_id.place(x=350, y=185)

    v = IntVar()
    Label(AccPayroll_pg, text="MODE OF \nPAYMENT", font=('', 13)).place(x=220,
y=220)
    Radiobutton(AccPayroll_pg, text='CASH', variable=v, value=1, font=('',
```

```
13)).place(x=350, y=230)
    Radiobutton(AccPayroll_pg, text='CHEQUE', variable=v, value=2, font=('',
13)).place(x=450, y=230)

    v1 = IntVar()
    Label(AccPayroll_pg, text='STATUS', font=('', 13)).place(x=220, y=275)
    Radiobutton(AccPayroll_pg, text='PAYED', variable=v1, value=1, font=('',
13)).place(x=350, y=275)
    Radiobutton(AccPayroll_pg, text='UNPAYED', variable=v1, value=2, font=('',
13)).place(x=450, y=275)
    Button(AccPayroll_pg, text='SUBMIT', width=12, font=('', 13),
command=accountsandpayroll).place(x=230, y=320)
    Button(AccPayroll_pg, text='BACK', font=('', 13), width=12,
command=account_and_payroll_to_userfirstpage).place(x=340, y=320)
    Button(AccPayroll_pg, text='RESET', width=12, font=('', 13),
command=reset_account_and_payroll).place(x=450, y=320)
    AccPayroll_pg.mainloop()
# ====================================End of Accounts And Payroll
Page================================================#


# ====================================Inventory
Page=========================================================================#
def submit_stock():
    stockconn = cx.connect("System", "oracle", "XE")
    stockcursor = stockconn.cursor()
    stockdate = entry_stockdate.get()
    quantity = entry_qty.get()
    units = entry_unit.get()
    expincome = entry_expincome.get()
    time_string = time.strftime('%H:%M:%S')
    stockcursor.execute("insert into stock values(:stk,:tm,:qty,:uts,:exi)",
                        {'stk': stockdate, 'tm': time_string, 'qty': quantity,
'uts': units, 'exi': expincome})
    stockconn.commit()
    stockconn.close()
    ms.showinfo("Success","Entry Done")


def inventory_to_userfirstpage():
    inventory_pg.destroy()
    user_first_page.deiconify()


def reset_inventory():
    entry_stockdate.delete("0", END)
    entry_qty.delete("0", END)
    entry_unit.delete("0", END)
    entry_expincome.delete("0", END)


def inventory():
    user_first_page.withdraw()
    global inventory_pg
    inventory_pg = Toplevel(root)

    img3 = ImageTk.PhotoImage(Image.open("temple1.jpg"))
    bg_image3 = Label(inventory_pg, image=img3)
    bg_image3.place(x=0,y=0)

    screen_width = inventory_pg.winfo_screenwidth()
    screen_height = inventory_pg.winfo_screenheight()
    x = (screen_width / 2) - (width / 2)
    y = (screen_height / 2) - (height / 2)
    inventory_pg.resizable(0, 0)
    inventory_pg.geometry("%dx%d+%d+%d" % (width, height, x, y))

    Label(inventory_pg, text='INVENTORY MODULE',font=("Showcard Gothic",
```

```
20)).place(x=280, y=40)

    global entry_stockdate
    Label(inventory_pg, text='STOCK DATE', font=('', 13)).place(x=180, y=115)
    entry_stockdate = Entry(inventory_pg, font=('', 15))
    entry_stockdate.place(x=360, y=115)

    global entry_qty
    Label(inventory_pg, text='QUANTITY', font=('', 13)).place(x=180, y=150)
    entry_qty = Entry(inventory_pg, font=('', 15))
    entry_qty.place(x=360, y=150)

    global entry_unit
    Label(inventory_pg, text='UNIT', font=('', 13)).place(x=180, y=185)
    entry_unit = Entry(inventory_pg, font=('', 15))
    entry_unit.place(x=360, y=185)

    global entry_expincome
    Label(inventory_pg, text='EXPECTED INCOME', font=('', 13)).place(x=180, y=220)
    entry_expincome = Entry(inventory_pg, font=('', 15))
    entry_expincome.place(x=360, y=220)

    Button(inventory_pg, text='SUBMIT', font=('', 13), width=10,
command=submit_stock).place(x=250,y=280)
    Button(inventory_pg, text='BACK', font=('', 13), width=10,
command=inventory_to_userfirstpage).place(x=350,y=280)
    Button(inventory_pg, text='RESET', font=('', 13), width=10,
command=reset_inventory).place(x=450,y=280)
    inventory_pg.mainloop()
# ============================End of Inventory
Page=========================================================================#


#====================================Display Inventory
Page=========================================================#
def dispinvent_to_adminfirstpage():
    dispinvent_pg.destroy()
    admin_first_page.deiconify()


def sortby(tree, col, descending):
    data = [(tree.set(child, col), child) for child in tree.get_children('')]
    data.sort(reverse=descending)
    for ix, item in enumerate(data):
        tree.move(item[1], '', ix)
    tree.heading(col, command=lambda col=col: sortby(tree, col, int(not
descending)))


def displayinventory():
    admin_first_page.withdraw()
    global dispinvent_pg
    dispinvent_pg = Tk()
    width = 830
    height = 510
    screen_width = dispinvent_pg.winfo_screenwidth()
    screen_height = dispinvent_pg.winfo_screenheight()
    x = (screen_width / 2) - (width / 2)
    y = (screen_height / 2) - (height / 2)
    dispinvent_pg.resizable(0, 0)
    dispinvent_pg.geometry("%dx%d+%d+%d" % (width, height, x, y))

    seva_header = ['Stock Date', 'Time Of Entry', 'Quantity', 'Units', 'Expected
Income']

    tree = None
    s = "\Click on header to sort by that column to change width of column drag
boundary"
```

```
    msg = ttk.Label(dispinvent_pg, justify="left", anchor="n", padding=(10, 2, 10,
6), text=s)
    msg.pack(fill='x')

    container = ttk.Frame(dispinvent_pg,width=180)
    container.pack(fill='both', expand=True)

    # create a treeview with dual scrollbars
    tree = ttk.Treeview(dispinvent_pg,columns=seva_header, show="headings",
height=20)
    vsb = ttk.Scrollbar(dispinvent_pg,orient="vertical", command=tree.yview)
    hsb = ttk.Scrollbar(dispinvent_pg,orient="horizontal", command=tree.xview)
    tree.configure(yscrollcommand=vsb.set, xscrollcommand=hsb.set)
    tree.grid(column=0, row=0, sticky='nsew', in_=container)
    vsb.grid(column=1, row=0, sticky='ns', in_=container)
    hsb.grid(column=0, row=1, sticky='ew', in_=container)
    container.grid_columnconfigure(0, weight=1)
    container.grid_rowconfigure(0, weight=1)

    for col in seva_header:
        tree.heading(col, text=col.title(), command=lambda c=col: sortby(tree, c,
0))
        tree.column(col, width=tkFont.Font().measure(col.title()))

    dispinvent_pgconn = cx.connect("System", "oracle", "XE")
    dispinvent_pgcursor = dispinvent_pgconn.cursor()
    dispinvent_pgcursor.execute("select * from stock")
    car_list = dispinvent_pgcursor.fetchall()
    dispinvent_pgconn.commit()
    dispinvent_pgconn.close()
    for item in car_list:
        tree.insert('', 'end', values=item)
        for ix, val in enumerate(item):
            col_w = tkFont.Font().measure(val)
            if tree.column(seva_header[ix], width=None) < col_w:
                tree.column(seva_header[ix], width=col_w)

    Button(dispinvent_pg, text="Back", bg='purple1', fg='white', font=(" ",
15),command=dispinvent_to_adminfirstpage).pack(fill='x')


# ============================End Of Display of Inventory
Page=========================================================#

#====================================Display Donation
Page=============================================================#

def dispdonation_to_adminfirstpage():
    dispdonation_pg.destroy()
    admin_first_page.deiconify()

def displayDonation():
    admin_first_page.withdraw()
    global dispdonation_pg
    dispdonation_pg = Tk()
    width = 830
    height = 510
    screen_width = dispdonation_pg.winfo_screenwidth()
    screen_height = dispdonation_pg.winfo_screenheight()
    x = (screen_width / 2) - (width / 2)
    y = (screen_height / 2) - (height / 2)
    dispdonation_pg.resizable(0, 0)
    dispdonation_pg.geometry("%dx%d+%d+%d" % (width, height, x, y))

    donation_header = ['Pay Date', 'Name', 'Amount', 'Payment Mode']

    tree = None
```

```
    s = "\Click on header to sort by that column to change width of column drag
boundary"

    msg = ttk.Label(dispdonation_pg, justify="left", anchor="n", padding=(10, 2,
10, 6), text=s)
    msg.pack(fill='x')

    container = ttk.Frame(dispdonation_pg, width=180)
    container.pack(fill='both', expand=True)

    # create a treeview with dual scrollbars
    tree = ttk.Treeview(dispdonation_pg, columns=donation_header, show="headings",
height=20)
    vsb = ttk.Scrollbar(dispdonation_pg, orient="vertical", command=tree.yview)
    hsb = ttk.Scrollbar(dispdonation_pg, orient="horizontal", command=tree.xview)
    tree.configure(yscrollcommand=vsb.set, xscrollcommand=hsb.set)
    tree.grid(column=0, row=0, sticky='nsew', in_=container)
    vsb.grid(column=1, row=0, sticky='ns', in_=container)
    hsb.grid(column=0, row=1, sticky='ew', in_=container)
    container.grid_columnconfigure(0, weight=1)
    container.grid_rowconfigure(0, weight=1)

    for col in donation_header:
        tree.heading(col, text=col.title(), command=lambda c=col: sortby(tree, c,
0))
        # adjust the column's width to the header string
        tree.column(col, width=tkFont.Font().measure(col.title()))

    conn = cx.connect("System", "oracle", "XE")
    cursor = conn.cursor()
    cursor.execute("select * from donation")
    car_list = cursor.fetchall()
    for item in car_list:
        tree.insert('', 'end', values=item)
        # adjust column's width if necessary to fit each value
        for ix, val in enumerate(item):
            col_w = tkFont.Font().measure(val)
            if tree.column(donation_header[ix], width=None) < col_w:
                tree.column(donation_header[ix], width=col_w)

    Button(dispdonation_pg, text="Back", bg='purple1', fg='white', font=(" ",
15),command=dispdonation_to_adminfirstpage).pack(fill='x')
# ============================End Of Display of Donation
Page=========================================================#


#====================================Display Account And Payroll
Page=========================================================#

def dispAccount_to_adminfirstpage():
    dispaccount_pg.destroy()
    admin_first_page.deiconify()

def displayAccount():
    admin_first_page.withdraw()
    global dispaccount_pg
    dispaccount_pg = Tk()
    width = 830
    height = 510
    screen_width = dispaccount_pg.winfo_screenwidth()
    screen_height = dispaccount_pg.winfo_screenheight()
    x = (screen_width / 2) - (width / 2)
    y = (screen_height / 2) - (height / 2)
    dispaccount_pg.resizable(0, 0)
    dispaccount_pg.geometry("%dx%d+%d+%d" % (width, height, x, y))

    account_header = ['Payment Date', 'Account Holder Name', 'Account Number',
'Email Id', 'Pay Mode', 'Status']
```

```
    tree = None
    s = "\Click on header to sort by that column to change width of column drag
boundary"

    msg = ttk.Label(dispaccount_pg, justify="left", anchor="n", padding=(10, 2, 10,
6), text=s)
    msg.pack(fill='x')

    container = ttk.Frame(dispaccount_pg, width=180)
    container.pack(fill='both', expand=True)

    # create a treeview with dual scrollbars
    tree = ttk.Treeview(dispaccount_pg, columns=account_header, show="headings",
height=20)
    vsb = ttk.Scrollbar(dispaccount_pg, orient="vertical", command=tree.yview)
    hsb = ttk.Scrollbar(dispaccount_pg, orient="horizontal", command=tree.xview)
    tree.configure(yscrollcommand=vsb.set, xscrollcommand=hsb.set)
    tree.grid(column=0, row=0, sticky='nsew', in_=container)
    vsb.grid(column=1, row=0, sticky='ns', in_=container)
    hsb.grid(column=0, row=1, sticky='ew', in_=container)
    container.grid_columnconfigure(0, weight=1)
    container.grid_rowconfigure(0, weight=1)

    for col in account_header:
        tree.heading(col, text=col.title(), command=lambda c=col: sortby(tree, c,
0))
        # adjust the column's width to the header string
        tree.column(col, width=tkFont.Font().measure(col.title()))

    conn = cx.connect("System", "oracle", "XE")
    cursor = conn.cursor()
    cursor.execute("select * from accountandpayroll")
    car_list = cursor.fetchall()
    for item in car_list:
        tree.insert('', 'end', values=item)
        # adjust column's width if necessary to fit each value
        for ix, val in enumerate(item):
            col_w = tkFont.Font().measure(val)
            if tree.column(account_header[ix], width=None) < col_w:
                tree.column(account_header[ix], width=col_w)

    Button(dispaccount_pg, text="Back", bg='purple1', fg='white', font=(" ",
15),command=dispAccount_to_adminfirstpage).pack(fill='x')

# ============================End Of Display of Account And Payroll
Page=========================================================#


#====================================Display Seva
Page=============================================================#
def dispSeva_to_adminfirstpage():
    dispseva_pg.destroy()
    admin_first_page.deiconify()

def displaySeva():
    admin_first_page.withdraw()
    global dispseva_pg
    dispseva_pg = Tk()
    width = 830
    height = 510
    screen_width = dispseva_pg.winfo_screenwidth()
    screen_height = dispseva_pg.winfo_screenheight()
    x = (screen_width / 2) - (width / 2)
    y = (screen_height / 2) - (height / 2)
    dispseva_pg.resizable(0, 0)
    dispseva_pg.geometry("%dx%d+%d+%d" % (width, height, x, y))
```

```
    seva_header = ['Seva Name', 'Seva Description', 'Duration', 'From Date', 'To
Date']

    tree = None
    s = "\Click on header to sort by that column to change width of column drag
boundary"

    msg = ttk.Label(dispseva_pg, justify="left", anchor="n", padding=(10, 2, 10,
6), text=s)
    msg.pack(fill='x')

    container = ttk.Frame(dispseva_pg, width=180)
    container.pack(fill='both', expand=True)

    # create a treeview with dual scrollbars
    tree = ttk.Treeview(dispseva_pg, columns=seva_header, show="headings",
height=20)
    vsb = ttk.Scrollbar(dispseva_pg, orient="vertical", command=tree.yview)
    hsb = ttk.Scrollbar(dispseva_pg, orient="horizontal", command=tree.xview)
    tree.configure(yscrollcommand=vsb.set, xscrollcommand=hsb.set)
    tree.grid(column=0, row=0, sticky='nsew', in_=container)
    vsb.grid(column=1, row=0, sticky='ns', in_=container)
    hsb.grid(column=0, row=1, sticky='ew', in_=container)
    container.grid_columnconfigure(0, weight=1)
    container.grid_rowconfigure(0, weight=1)

    for col in seva_header:
        tree.heading(col, text=col.title(), command=lambda c=col: sortby(tree, c,
0))
        # adjust the column's width to the header string
        tree.column(col, width=tkFont.Font().measure(col.title()))

    conn = cx.connect("System", "oracle", "XE")
    cursor = conn.cursor()
    cursor.execute("select * from seva")
    car_list = cursor.fetchall()
    for item in car_list:
        tree.insert('', 'end', values=item)
        # adjust column's width if necessary to fit each value
        for ix, val in enumerate(item):
            col_w = tkFont.Font().measure(val)
            if tree.column(seva_header[ix], width=None) < col_w:
                tree.column(seva_header[ix], width=col_w)

    Button(dispseva_pg, text="Back", bg='purple1', fg='white', font=(" ",
15),command=dispSeva_to_adminfirstpage).pack(fill='x')

# ====================================Donation
Page=========================================================================#
def paydonation():
    payconn = cx.connect("System", "oracle", "XE")
    paycursor = payconn.cursor()
    name = entry_name.get()
    amount = entry_amount.get()
    date = dt.datetime.today()
    online = "Online"
    cash = "Cash"
    if v.get() == 1:
        paycursor.execute("insert into donation values(:d,:n,:a,:p)", {'d': date,
'n': name, 'a': amount, 'p': cash})
        payconn.commit()
    else:
        paycursor.execute("insert into donation values(:d,:n,:a,:p)", {'d': date,
'n': name, 'a': amount,'p': online})
        payconn.commit()


def reset_donation():
```

```
    entry_name.delete("0", END)
    entry_amount.delete("0", END)


def donation_to_userfirstpage():
    donation_pg.destroy()
    user_first_page.deiconify()

def donation():
    user_first_page.withdraw()
    global donation_pg
    donation_pg = Toplevel(root)

    img3 = ImageTk.PhotoImage(Image.open("temple1.jpg"))
    bg_image3 = Label(donation_pg, image=img3)
    bg_image3.place(x=0,y=0)

    screen_width = donation_pg.winfo_screenwidth()
    screen_height = donation_pg.winfo_screenheight()
    x = (screen_width / 2) - (width / 2)
    y = (screen_height / 2) - (height / 2)
    donation_pg.resizable(0, 0)
    donation_pg.geometry("%dx%d+%d+%d" % (width, height, x, y))

    L1 = Label(donation_pg, text="Donation Module", font=("Showcard Gothic", 20),
               justify=CENTER, height=2, fg="black").place(x=280, y=100)

    global entry_name
    Label(donation_pg, text='NAME: ', font=('', 13)).place(x=220, y=200)
    entry_name = Entry(donation_pg, font=('', 15))
    entry_name.place(x=360, y=200)

    global entry_amount
    Label(donation_pg, text='AMOUNT: ', font=('', 13)).place(x=220, y=235)
    entry_amount = Entry(donation_pg, font=('', 15))
    entry_amount.place(x=360, y=235)

    global v
    v = IntVar()
    Radiobutton(donation_pg, text='Online', variable=v, value=2, font=('',
13)).place(x=300, y=290)
    Radiobutton(donation_pg, text='Offline/Cash', variable=v, value=1, font=('',
13)).place(x=440, y=290)

    Button(donation_pg, text='PAY', width=10, font=('', 13),
command=paydonation).place(x=270, y=360)
    Button(donation_pg, text='BACK', width=10, font=('', 13),
command=donation_to_userfirstpage).place(x=370, y=360)
    Button(donation_pg, text='RESET', width=10, font=('', 13),
command=reset_donation).place(x=470, y=360)

    donation_pg.mainloop()
# ============================================End of Donation
module=========================================================#


# ============================================Seva
Page=================================================================#
def sevasubmit():
    sevaconn = cx.connect("System", "oracle", "XE")
    sevacursor = sevaconn.cursor()
    seva_name = entry_seva_name.get()
    seva_description = entry_seva_description.get("1.0", END)
    seva_duration = entry_seva_duration.get()
    from_date = entry_from_date.get()
    to_date = entry_to_date.get()
    if var1.get() == 1:
        sevacursor.execute("insert into seva values(:sn,:sd,:sdu,:fd,:td)",
```

```
        {'sn': seva_name, 'sd': seva_description, 'sdu': seva_duration, 'fd':
from_date,
                                'td': to_date})
        sevaconn.commit()
        sevaconn.close()
    else:
        ms.showinfo(" ","Please agree terms and conditions!!!");


def seva_to_userfirstpage():
    seva_pg.destroy()
    user_first_page.deiconify()


def sevareset():
    entry_seva_name.delete("0", END)
    entry_seva_description.delete("1.0", END)
    entry_seva_duration.delete("0", END)
    entry_from_date.delete("0", END)
    entry_to_date.delete("0", END)

def seva():
    user_first_page.withdraw()
    global seva_pg
    seva_pg = Toplevel(root)

    img3 = ImageTk.PhotoImage(Image.open("temple1.jpg"))
    bg_image3 = Label(seva_pg, image=img3)
    bg_image3.place(x=0,y=0)

    screen_width = seva_pg.winfo_screenwidth()
    screen_height = seva_pg.winfo_screenheight()
    x = (screen_width / 2) - (width / 2)
    y = (screen_height / 2) - (height / 2)
    seva_pg.resizable(0, 0)
    seva_pg.geometry("%dx%d+%d+%d" % (width, height, x, y))

    global var1
    var1 = IntVar()
    Label(seva_pg, text="SEVA MODULE", font=("Showcard Gothic", 20)).place(x=350,
y=20)

    global entry_seva_name
    Label(seva_pg, text='SEVA NAME :', font=('', 13)).place(x=200, y=75)
    entry_seva_name = Entry(seva_pg, font=('', 14))
    entry_seva_name.place(x=400, y=75)

    global entry_seva_description
    Label(seva_pg, text='DESCRIPTION :', font=('', 13)).place(x=200, y=105)
    entry_seva_description = Text(seva_pg, height=5, width=20)
    entry_seva_description.place(x=400, y=105)

    global entry_seva_duration
    Label(seva_pg, text='DURATION :', font=('', 13)).place(x=200, y=200)
    entry_seva_duration = Entry(seva_pg, font=('', 14))
    entry_seva_duration.place(x=400, y=200)

    global entry_from_date
    Label(seva_pg, text='FROM DATE :', font=('', 13)).place(x=200, y=230)
    entry_from_date = Entry(seva_pg, font=('', 14))
    entry_from_date.place(x=400, y=230)

    global entry_to_date
    Label(seva_pg, text='TO DATE :', font=('', 13)).place(x=200, y=260)
    entry_to_date = Entry(seva_pg, font=('', 14))
    entry_to_date.place(x=400, y=260)

    Label(seva_pg, text='I PERSONALLY AGREE TO ' + "\n" + 'ALL TERMS AND CONDITIONS
```

```
APPLIED', font=('', 13)).place(x=70, y=290)
    Checkbutton(seva_pg, text='Agree', variable=var1, font=('', 14)).place(x=400,
y=295)
    Button(seva_pg, text='SUBMIT', width=10, font=('', 13),
command=sevasubmit).place(x=270, y=360)
    Button(seva_pg, text='CANCEL', width=10, font=('', 13),
command=seva_to_userfirstpage).place(x=370, y=360)
    Button(seva_pg, text='RESET', width=10, font=('', 13),
command=sevareset).place(x=470, y=360)
    seva_pg.mainloop()
# ========================================End of Seva
Page============================================================#


# ========================================Add User
Page=========================================================================#
def createaccount():
    createaccountconn = cx.connect("System", "oracle", "XE")
    createaccountcursor = createaccountconn.cursor()
    firstname = entry_first_name.get()
    lastname = entry_last_name.get()
    dateofbirth = entry_date_of_birth.get()
    gender = entry_gender.get()
    uname = entry_username.get()
    upassword = entry_user_password.get()
    address = entry_address.get()
    mobno = entry_mob_no.get()
    city = entry_city.get()
    pincode = entry_pincode.get()
    state = entry_state.get()
    cntry = entry_country.get()
    adharno = entry_aadharno.get()
    date = dt.datetime.today()
    createaccountcursor.execute("insert into user_registration
values(:acd,:fn,:ln,:dob,:g,:un,:up,:ad,:mn,:ci,:pin,:st,:co,:adn)",{'acd': date,
'fn': firstname, 'ln': lastname, 'dob': dateofbirth, 'g': gender, 'un': uname,
'up': upassword, 'ad': address, 'mn': mobno,'ci': city, 'pin': pincode, 'st':
state, 'co': cntry, 'adn': adharno})
    createaccountconn.commit()
    createaccountconn.close()


def adduser_to_adminfirstpage():
    adduser_pg.destroy()
    admin_first_page.deiconify()


def adduser():
    admin_first_page.withdraw()
    global adduser_pg
    adduser_pg = Toplevel(root)

    img3 = ImageTk.PhotoImage(Image.open("temple1.jpg"))
    bg_image3 = Label(adduser_pg, image=img3)
    bg_image3.place(x=0,y=0)

    screen_width = adduser_pg.winfo_screenwidth()
    screen_height = adduser_pg.winfo_screenheight()
    x = (screen_width / 2) - (width / 2)
    y = (screen_height / 2) - (height / 2)
    adduser_pg.resizable(0, 0)
    adduser_pg.geometry("%dx%d+%d+%d" % (width, height, x, y))

    Frame(adduser_pg, padx=10, pady=10)

    global entry_first_name
    Label(adduser_pg, text='First Name: ', font=('', 15), pady=5,
padx=3).grid(sticky=W)
```

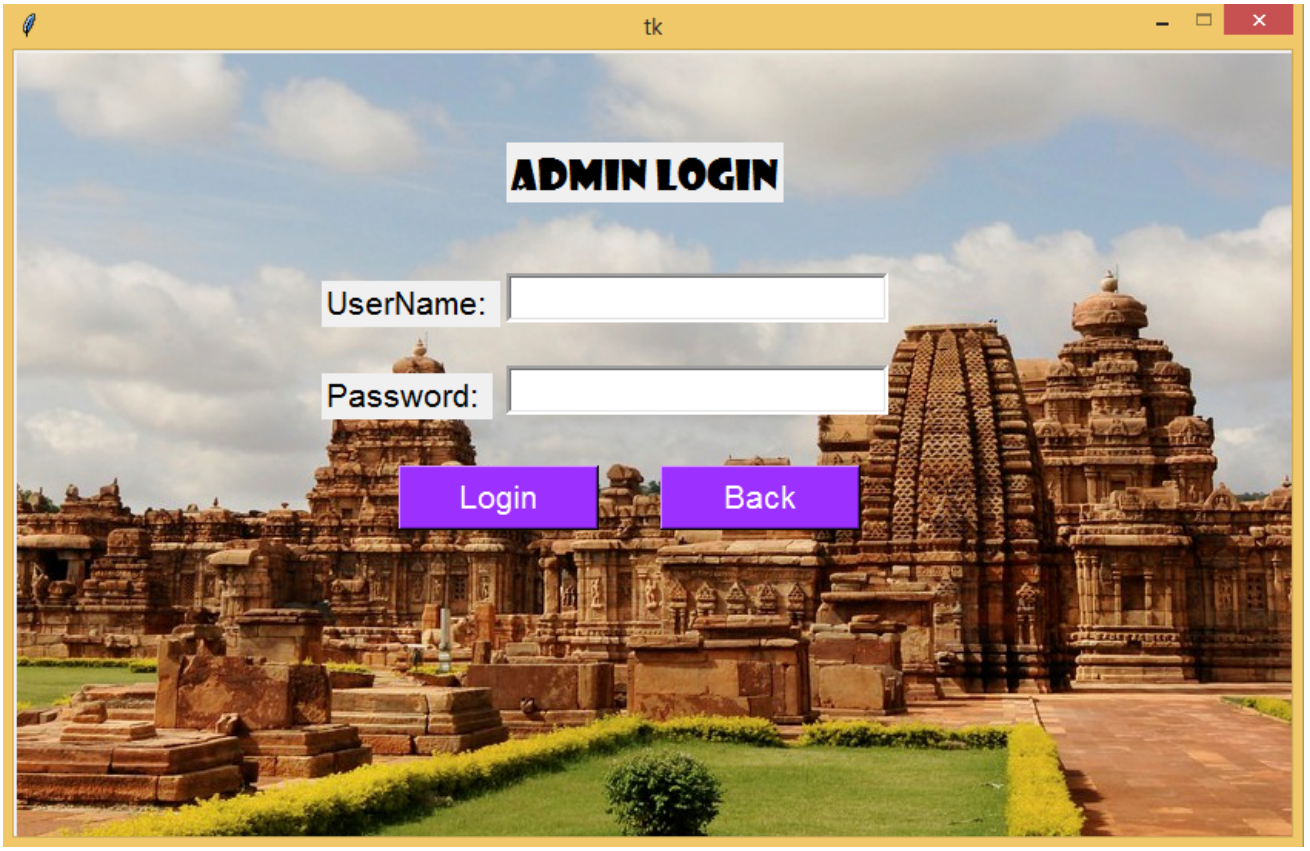
```
    entry_first_name = Entry(adduser_pg, bd=3, font=('', 15))
    entry_first_name.grid(row=0, column=1)

    global entry_last_name
    Label(adduser_pg, text='Last Name: ', font=('', 15), pady=5,
padx=3).grid(sticky=W, row=0, column=3)
    entry_last_name = Entry(adduser_pg, bd=3, font=('', 15))
    entry_last_name.grid(row=0, column=4)

    global entry_date_of_birth
    Label(adduser_pg, text='Date Of Birth: ', font=('', 15), pady=5,
padx=3).grid(sticky=W, row=1, column=0)
    entry_date_of_birth = Entry(adduser_pg, bd=3, font=('', 15))
    entry_date_of_birth.grid(row=1, column=1)

    global entry_gender
    Label(adduser_pg, text='Gender: ', font=('', 15), pady=5,
padx=3).grid(sticky=W, row=1, column=3)
    entry_gender = Entry(adduser_pg, bd=3, font=('', 15))
    entry_gender.grid(row=1, column=4)

    global entry_username
    Label(adduser_pg, text='Username: ', font=('', 15), pady=5,
padx=5).grid(sticky=W, row=2, column=0)
    entry_username = Entry(adduser_pg, bd=3, font=('', 15))
    entry_username.grid(row=2, column=1)

    global entry_user_password
    Label(adduser_pg, text='Password: ', font=('', 15), pady=5,
padx=5).grid(sticky=W, row=3, column=0)
    entry_user_password = Entry(adduser_pg, bd=3, font=('', 15), show='*')
    entry_user_password.grid(row=3, column=1)

    global entry_address
    Label(adduser_pg, text='Address: ', font=('', 15), pady=5,
padx=3).grid(sticky=W, row=4, column=0)
    entry_address = Entry(adduser_pg, bd=3, font=('', 15))
    entry_address.grid(row=4, column=1)

    global entry_mob_no
    Label(adduser_pg, text='Mobile No: ', font=('', 15), pady=5,
padx=3).grid(sticky=W, row=4, column=3)
    entry_mob_no = Entry(adduser_pg, bd=3, font=('', 15))
    entry_mob_no.grid(row=4, column=4)

    global entry_city
    Label(adduser_pg, text='City: ', font=('', 15), pady=5, padx=3).grid(sticky=W,
row=5, column=0)
    entry_city = Entry(adduser_pg, bd=3, font=('', 15))
    entry_city.grid(row=5, column=1)

    global entry_pincode
    Label(adduser_pg, text='Pincode: ', font=('', 15), pady=5,
padx=3).grid(sticky=W, row=5, column=3)
    entry_pincode = Entry(adduser_pg, bd=3, font=('', 15))
    entry_pincode.grid(row=5, column=4)

    global entry_state
    Label(adduser_pg, text='State: ', font=('', 15), pady=5, padx=3).grid(sticky=W,
row=6, column=0)
    entry_state = Entry(adduser_pg, bd=3, font=('', 15))
    entry_state.grid(row=6, column=1)

    global entry_country
    Label(adduser_pg, text='Country: ', font=('', 15), pady=5,
padx=3).grid(sticky=W, row=6, column=3)
    entry_country = Entry(adduser_pg, bd=3, font=('', 15))
    entry_country.grid(row=6, column=4)
```

```
    global entry_aadharno
    Label(adduser_pg, text='Adhar No: ', font=('', 15), pady=5,
padx=3).grid(sticky=W, row=7, column=0)
    entry_aadharno = Entry(adduser_pg, bd=3, font=('', 15))
    entry_aadharno.grid(row=7, column=1)

    Button(adduser_pg, text='Create Account', bd=3, font=('', 15), padx=5, pady=5,
command=createaccount).grid(column=1)
    Button(adduser_pg, text='Back', bd=3, font=('', 15), padx=5, pady=5,
command=adduser_to_adminfirstpage).grid(row=9, column=2)
    adduser_pg.mainloop()
# ======================================End of User
Page=================================================================#


main_form()
root.mainloop()
```

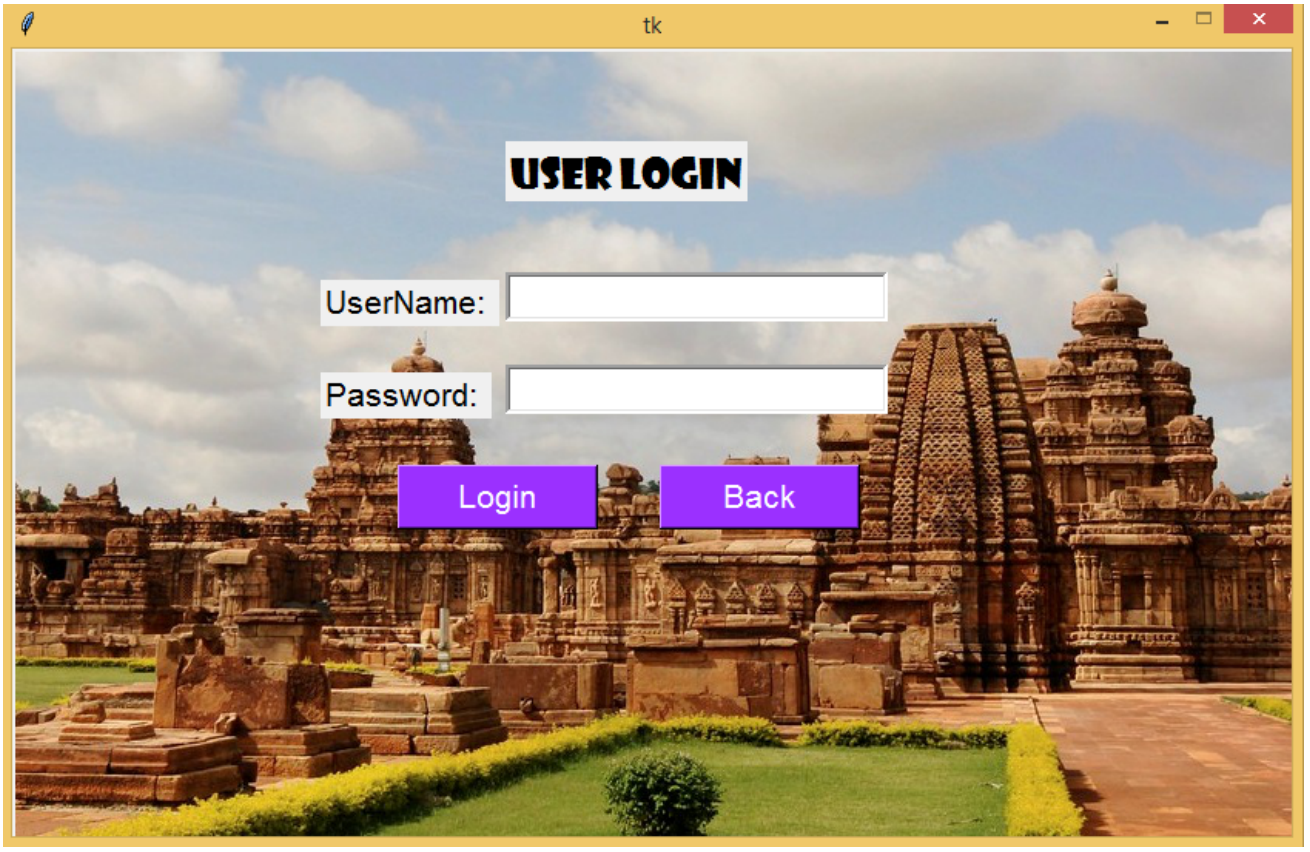
GUI SCREENSHOTS

1.Homepage

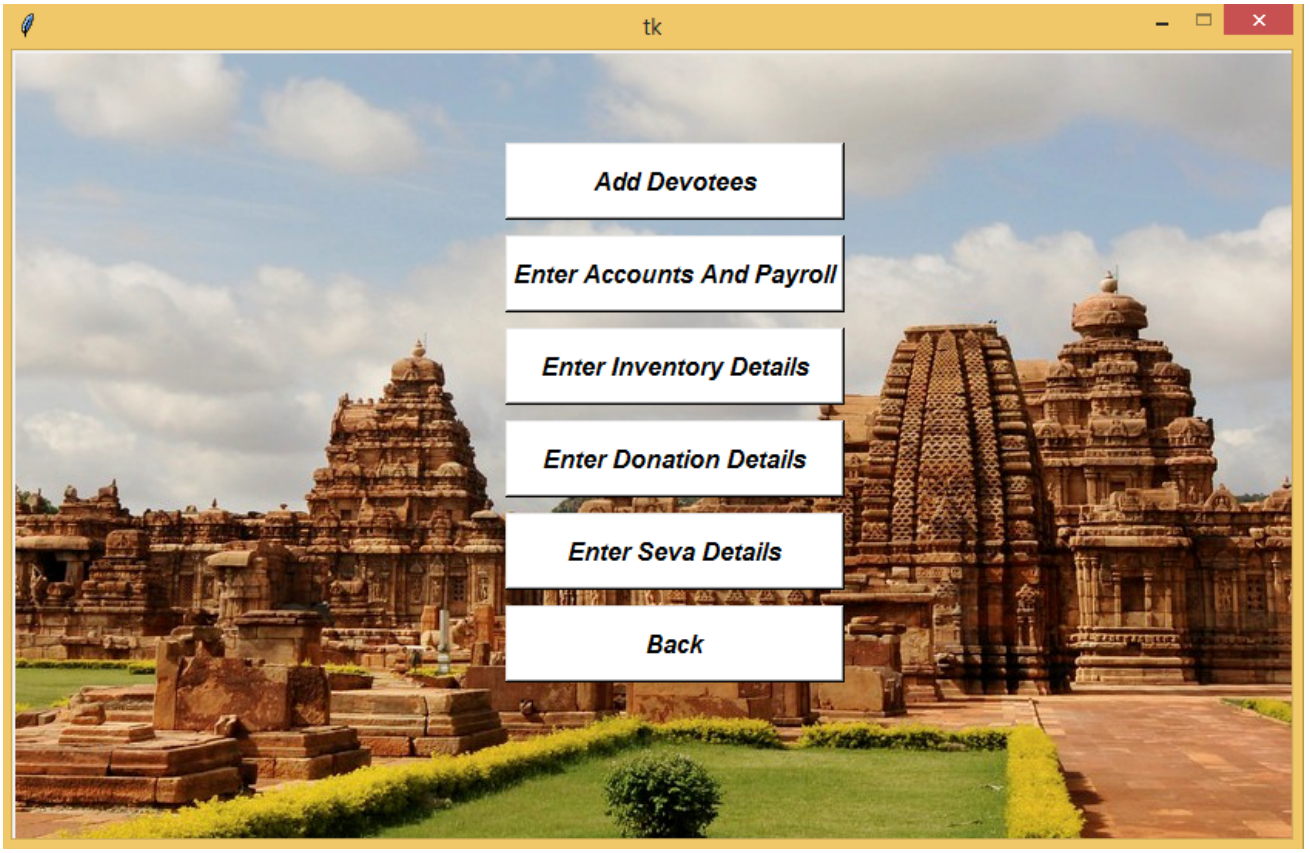
2.Admin Login



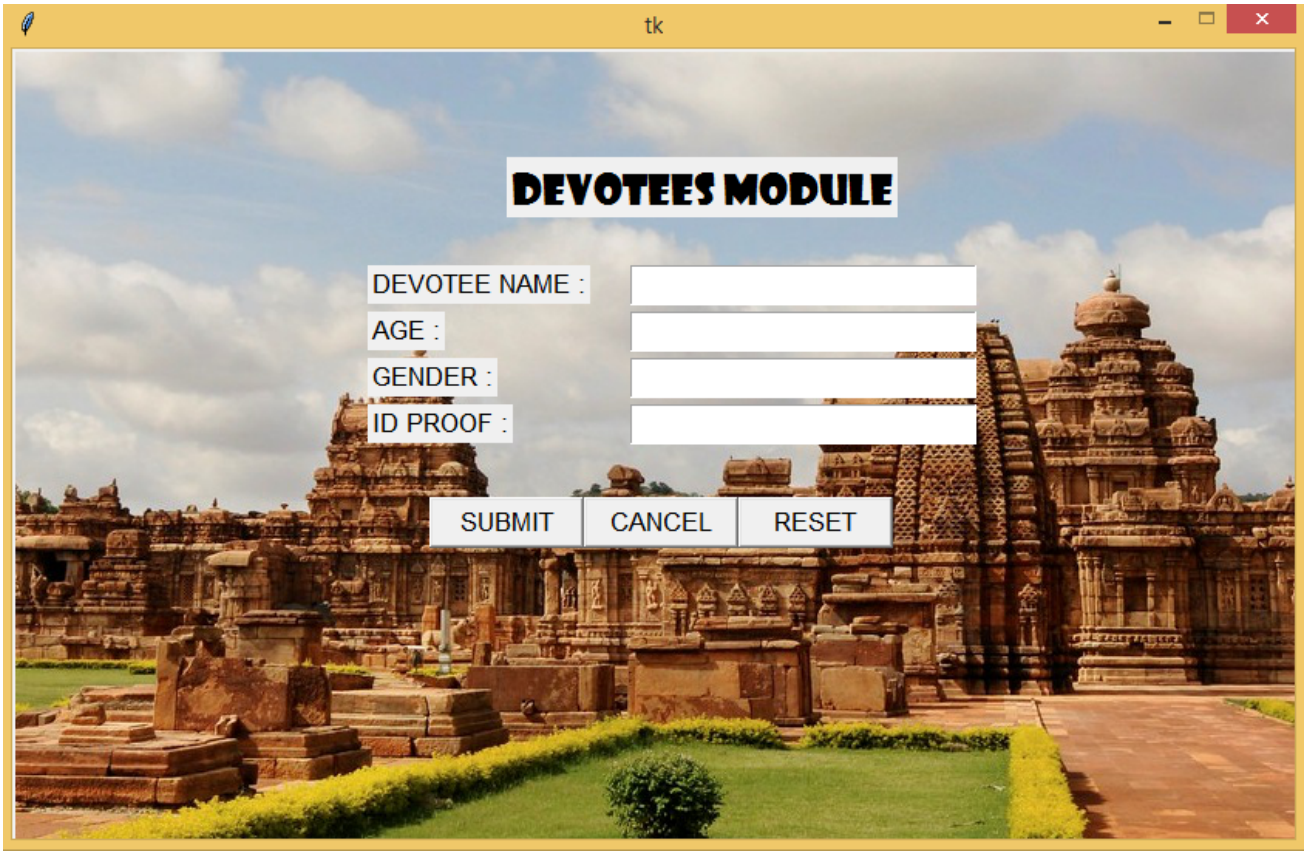
3. User Login



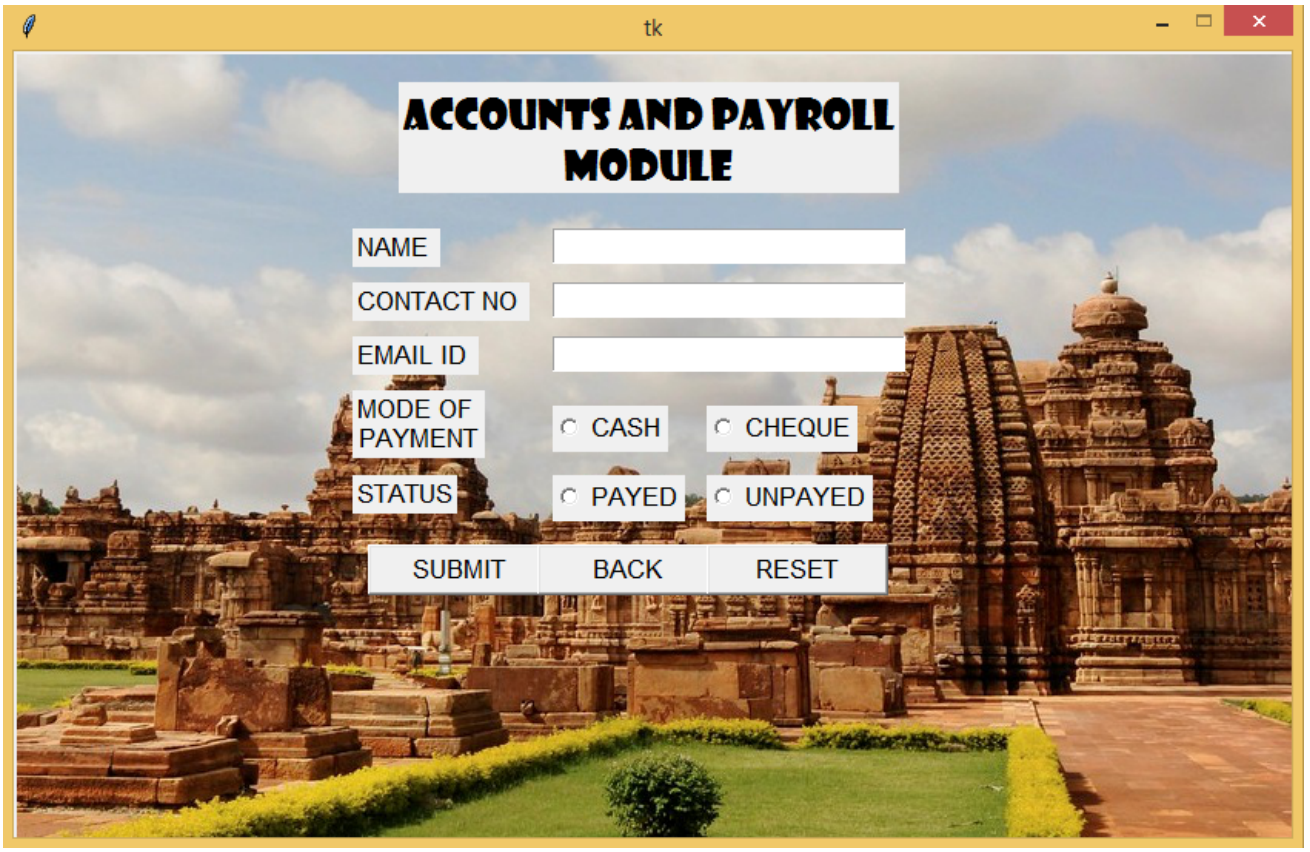
4.Page After User Login



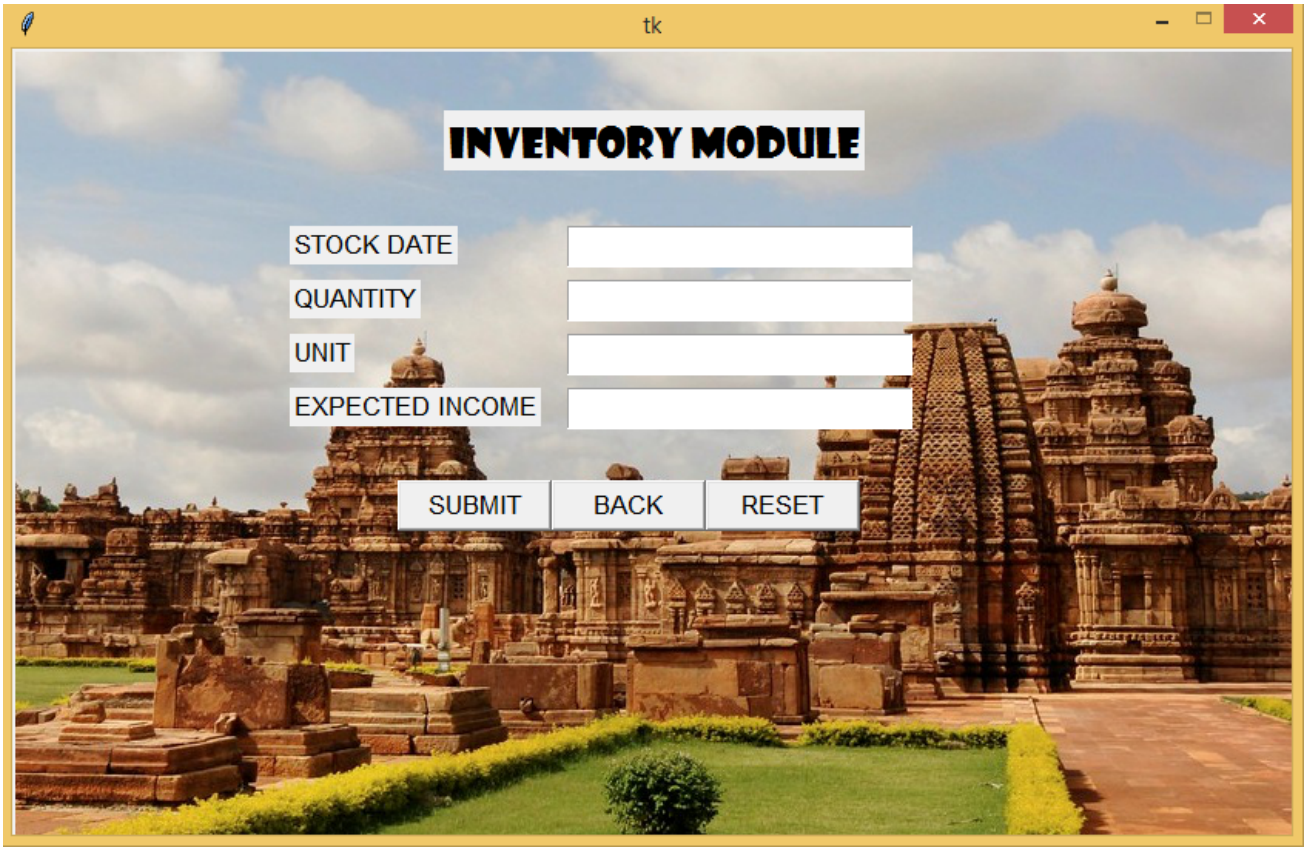
5.Add Devotee Module



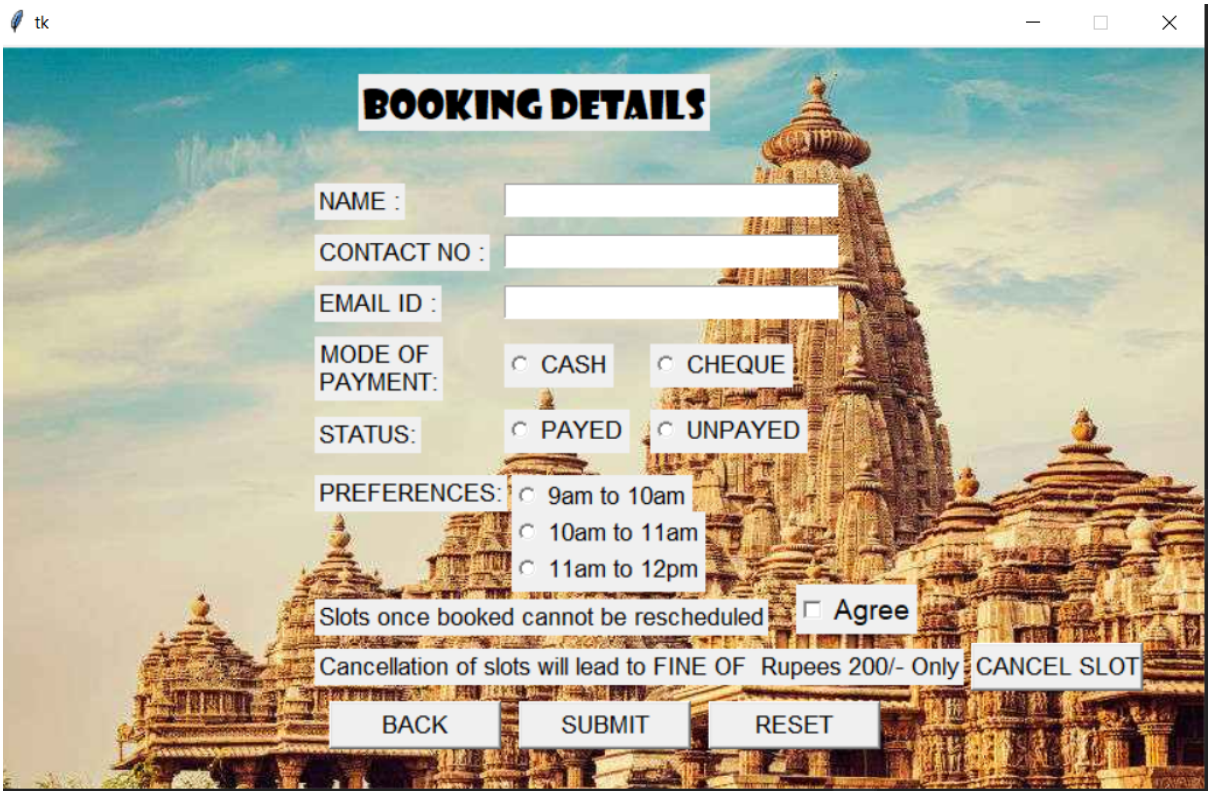
6. Accounts and Payroll Module



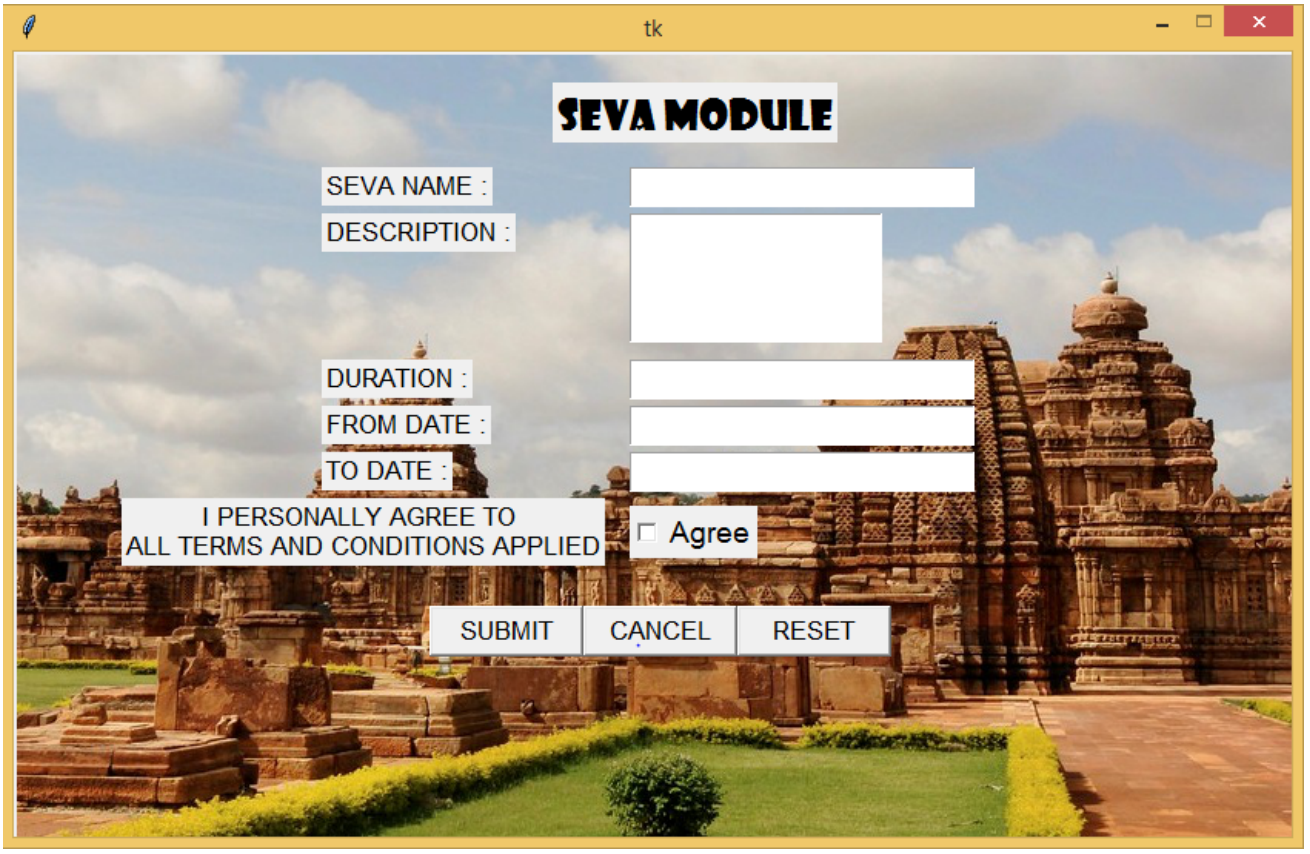
7. Inventory Module



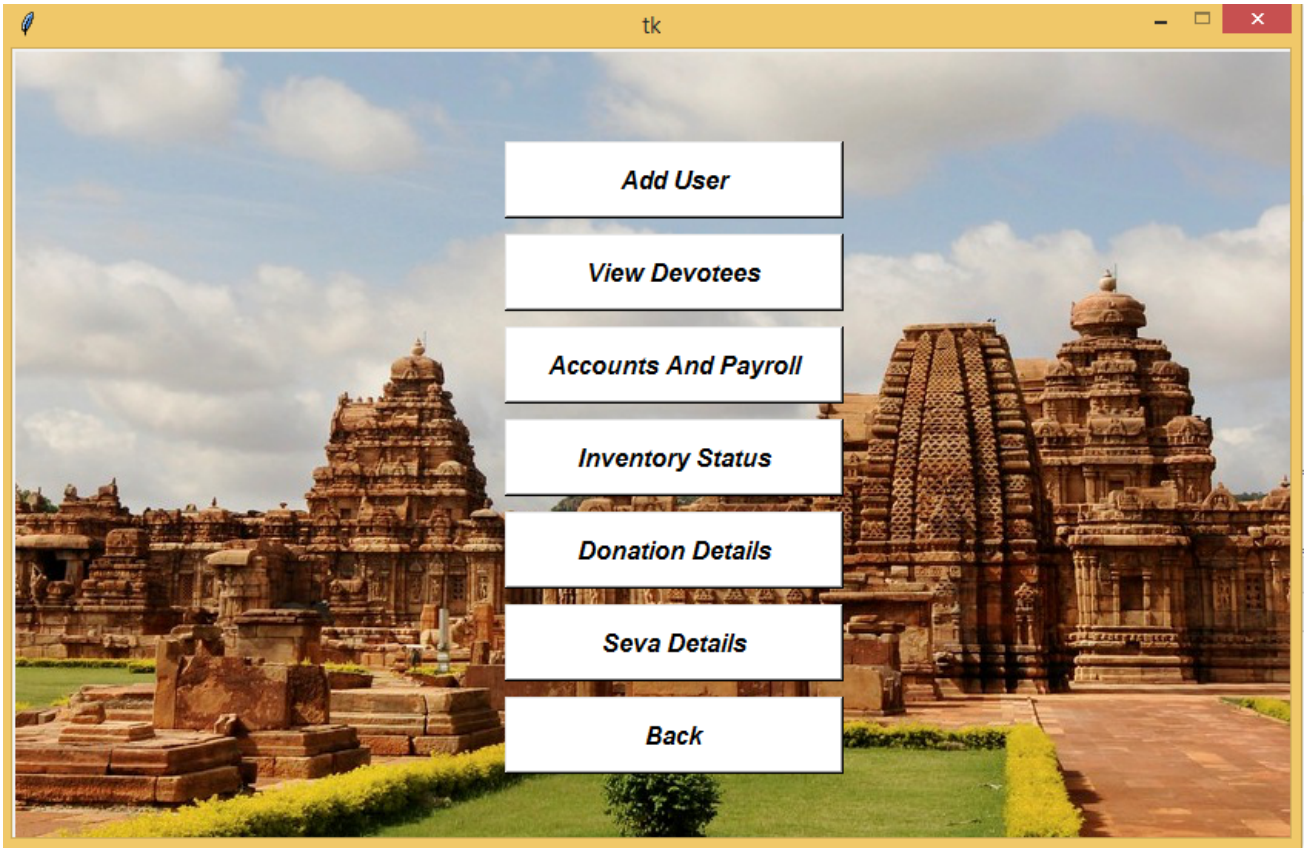
8. Booking Module



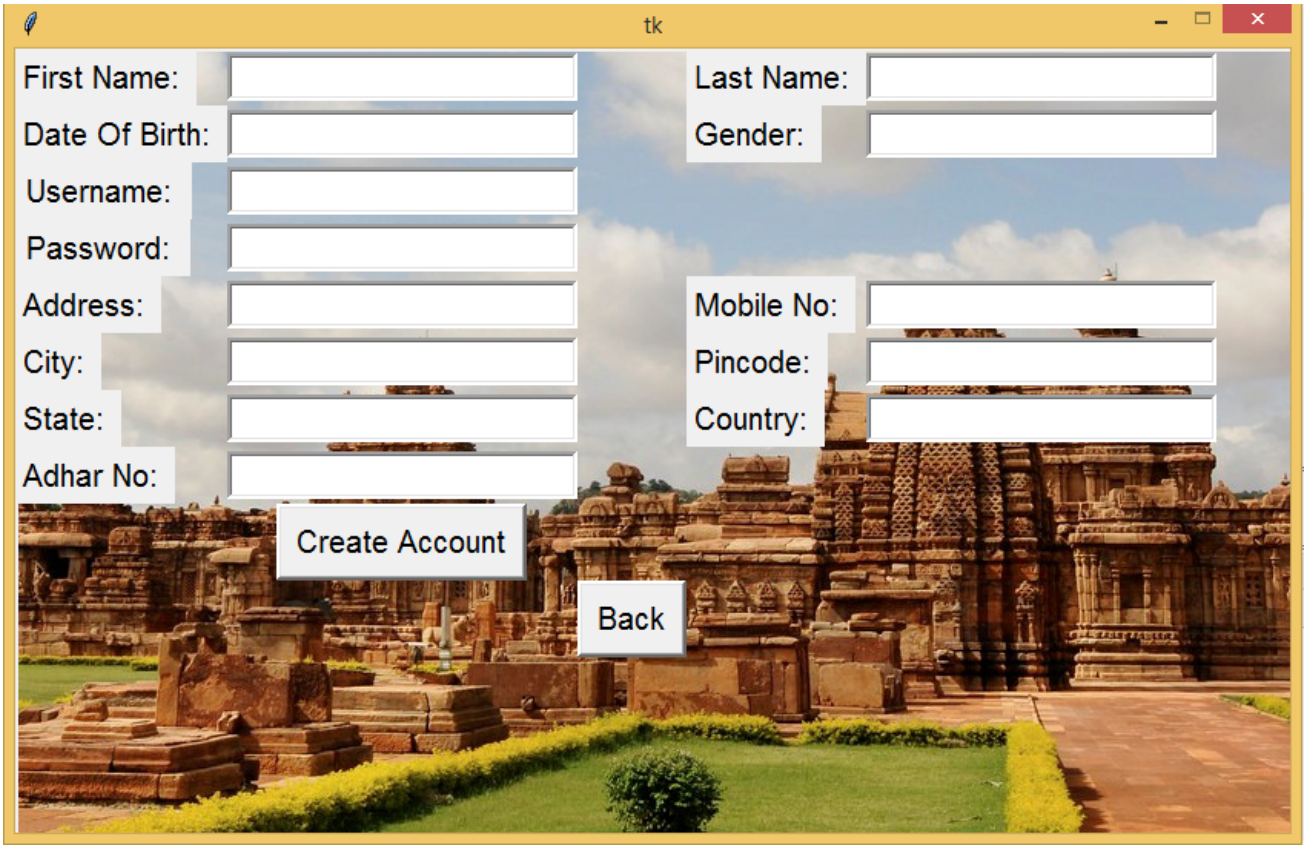
9. Seva Module



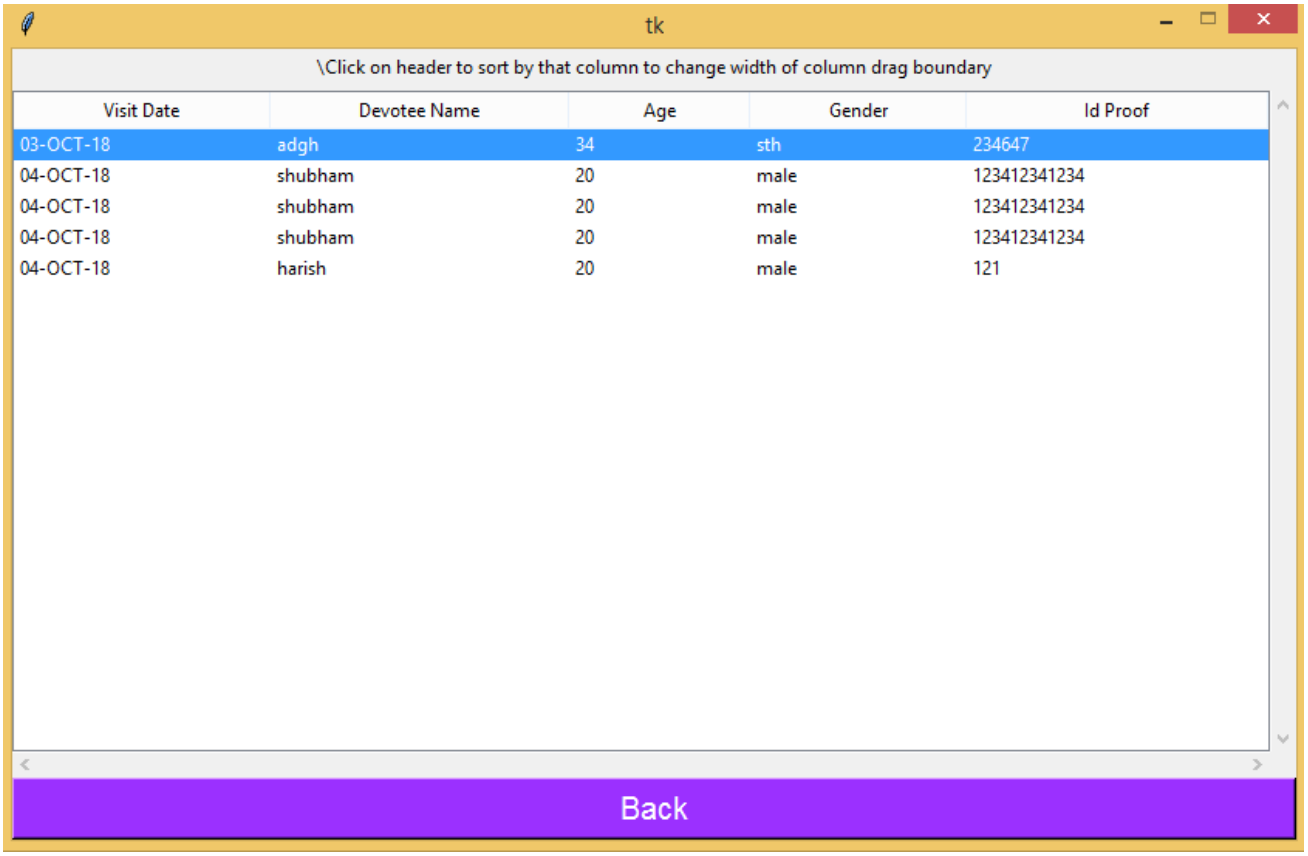
10. Page After Admin Login



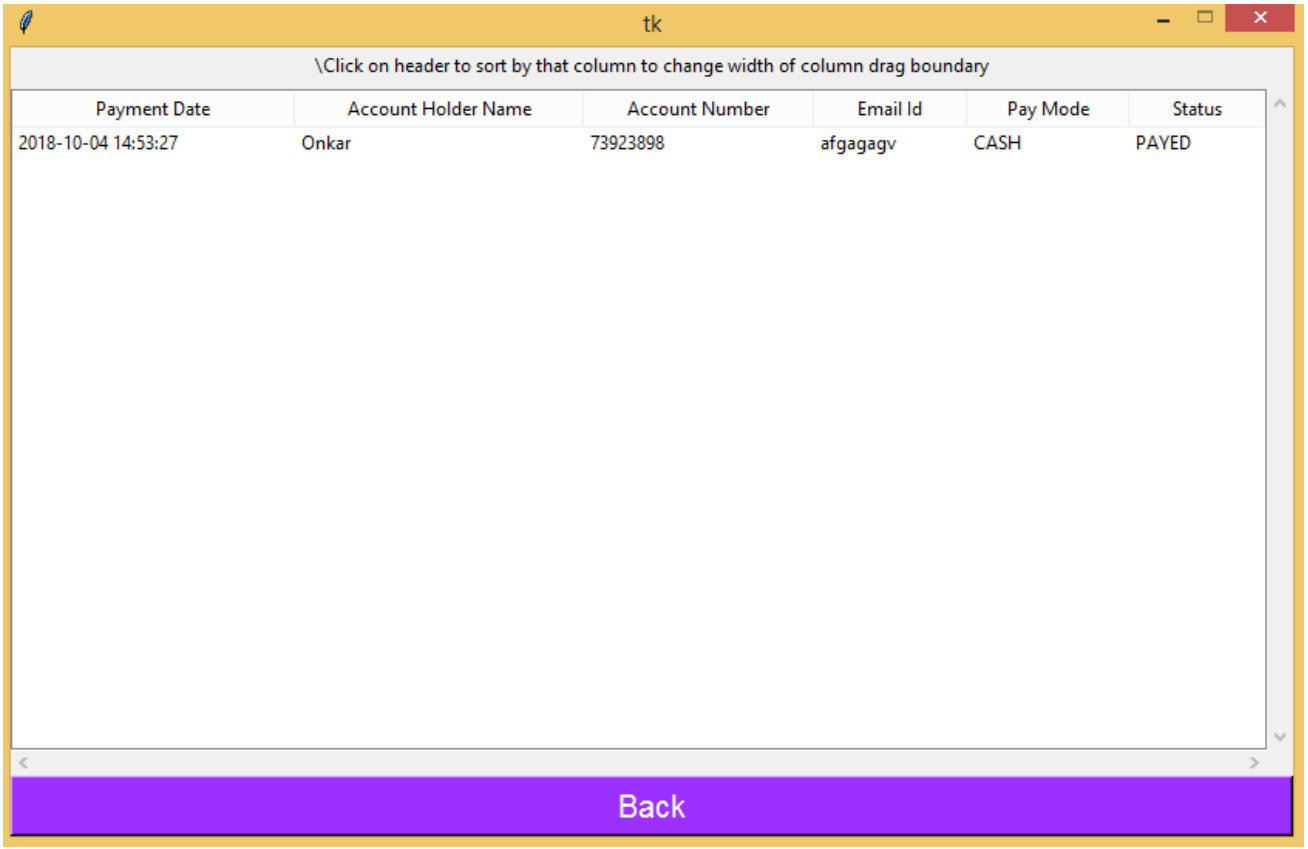
11. Add User Module



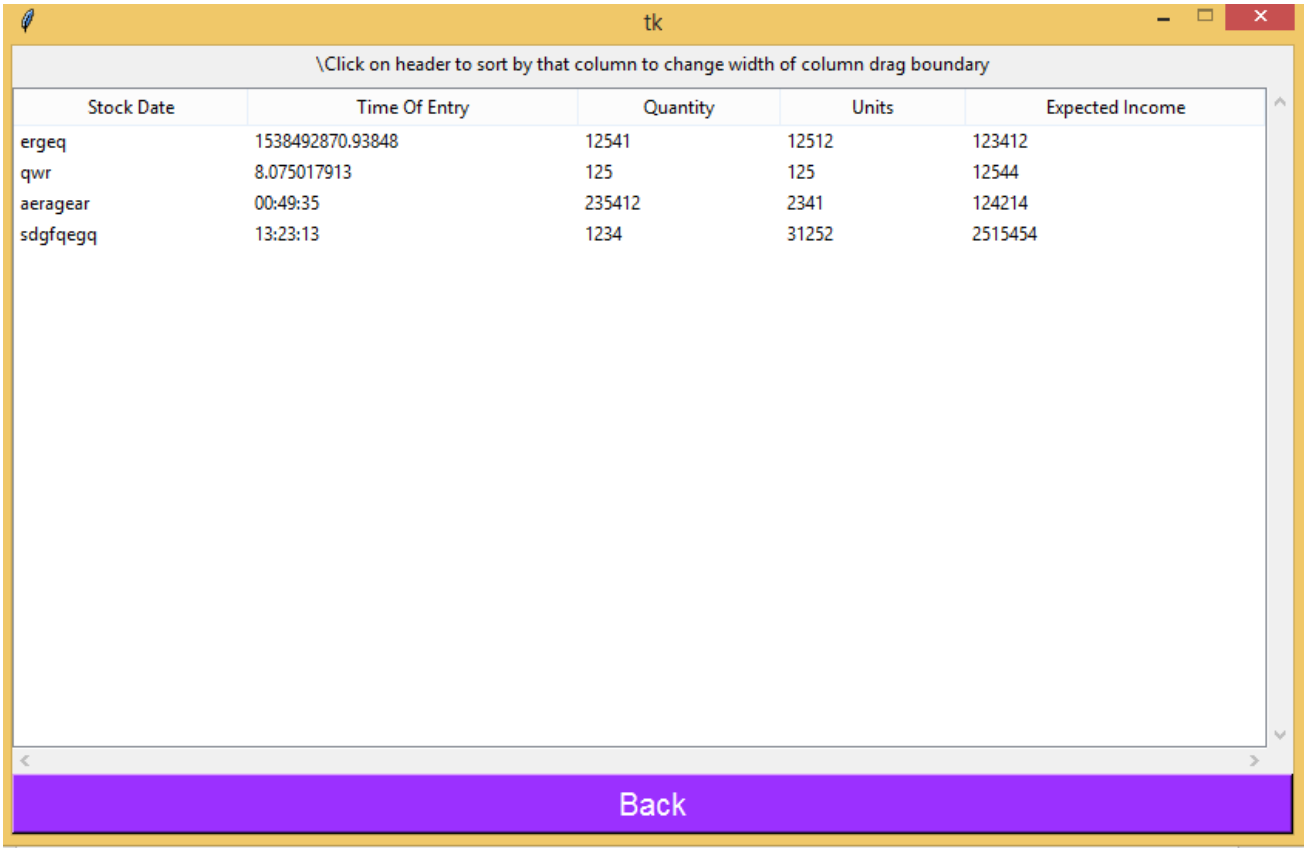
12. Display Devotee Present in Temple



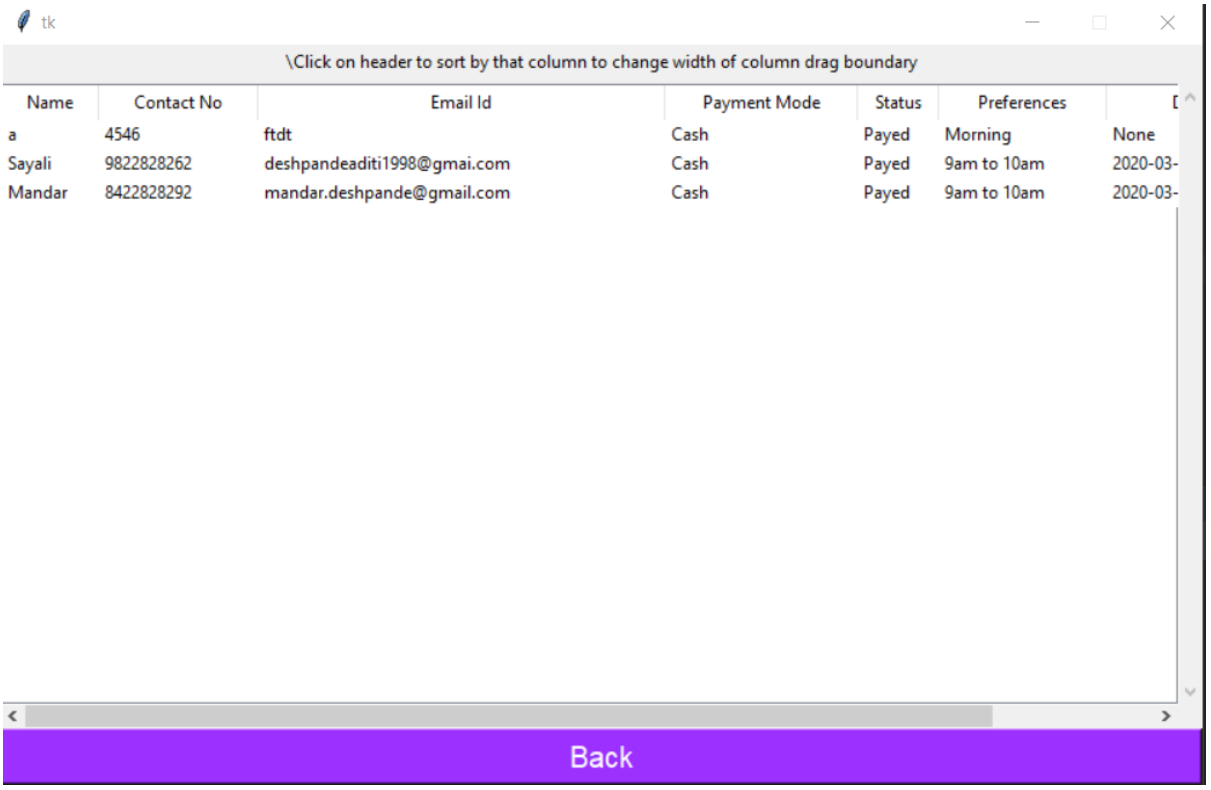
13. Display Account and Payroll Module



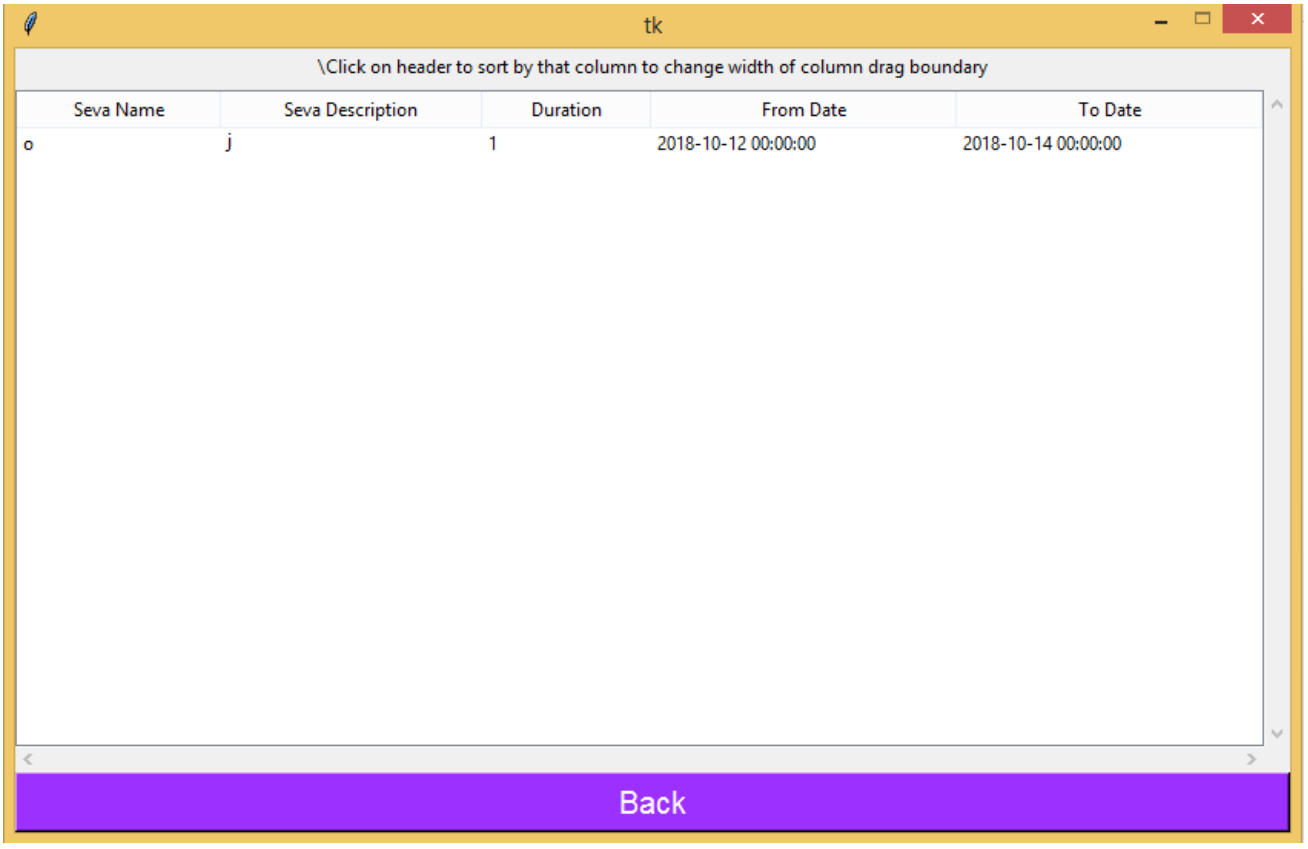
14. Display Inventory Module



15. Display Donation Module



16. Display Seva Module



TEST CASES

Test cases Id	**Objective**	**Steps**	**Expected O/P**	**Observed O/P**	**Result Pass/Fail**
1	Login button checking for admin login	Username and password enter correct	Login successfully and display the home window.	Correctly login in Ok button	Pass
2	Login button checking for admin login	Username is correct but password is not correct	Invalid password Please enter correct password	Login not successful	Fail
3	Login button checking for admin login	Username is incorrect but password is correct	Username is not valid	Login is not successful	Fail
4	Login button checking for admin login	Username and password is incorrect	This is not valid username and password	Not valid admin user	fail
5	Login button checking for user login	Username and password enter correct	Login successfully and display the home window.	Correctly login in Ok button	Pass
6	Login button checking for user login	Username is correct but password is not correct	Invalid password Please enter correct password	Login not successful	Fail
7	Login button checking for user login	Username is incorrect but password is correct	Username is not valid and	Login is not successful	Fail
8	Login button checking for user login	Username and password is incorrect	This is not valid username and password	Not valid user	fail
9	clicked back button	Press back button	Go to previous	Back button work successfully and go to	pass

				previous window	
10	clicked reset button	Press reset button	Same window	Resets Entry Fields	pass
11	clicked Add User button	Press Add User button	Go to next window	Add User button work successfully and go to next window	pass
12	clicked view devotees button	Press view devotees button	Go to next window	View Devotees button work successfully and go to next window	pass
13	clicked View Donation button	Press View Donation button	Go to next window	View Donation button work successfully and go to next window	pass
14	clicked view inventory button	Press view inventory button	Go to next window	view inventory button work successfully and go to next window	pass
15	clicked view seva button	Press view seva button	Go to next window	view seva button work successfully and go to next window	pass
16	clicked account and payroll button	Press account and payroll button	Go to next window	account and payroll button work successfully and go to next window	pass

17	clicked enter account and payroll button	Press enter account and payroll button	Go to next window	Enter account and payroll button work successfully and go to next window	pass
18	clicked Enter devotees button	Press Enter devotees button	Go to next window	Enter Devotees button work successfully and go to next window	pass
19	clicked Enter Donation button	Press Enter Donation button	Go to next window	Enter Donation button work successfully and go to next window	pass
20	clicked Enter inventory button	Press Enter inventory button	Go to next window	Enter inventory button work successfully and go to next window	pass
21	clicked Enter seva button	Press Enter seva button	Go to next window	Enter seva button work successfully and go to next window	pass

CONCLUSION

Thus we have automated the temples to make the temple management reduce their efforts of performing tedious operations.

REFRENCES

- Use of Tkinter in python by tutorials point.
- Oracle SQL insert, retrieve, create syntax from internet.