ELECTRICITY DEPARTMENT MANAGEMENT SYSTEM

Computer Science Project (083)

ELECTRICITY DEPARTMENT MANAGEMENT SYSTEM

A GUI-based application to manage bills, complaints and user data of an electricity board with an email-based communication automated system.

Hardik Kathuria
Class 12th-C

Board Roll No.-

Certificate

This is to certify that **Hardik Kathuria** of class **XII-C** of **DPSG, Palam Vihar** has done his project on **Electricity Department Management System** under my supervision. He has taken interest and has shown utmost sincerity in completion of this project.

I certify that this Project is up to my expectation & as per guidelines issued by **CBSE.**

Internal Examiner **External Examiner**

Acknowledgement

I would like to express a deep sense of thanks & gratitude to my project guide Ms. Monica Yadav for guiding me immensely through the course of the project.

She always evinced keen interest in my work. Her constructive advice & constant motivation have been responsible for the successful completion of this project.

My sincere thanks also goes to Mr.Vidhukesh Vimal, Principal for his co-ordination in extending every possible support for the completion of this project.

Overview :

This program is a Tkinter-based GUI application for managing a social media platform's consumer and employee interactions, including complaint management, bill payments, and user authentication.

Features

User Authentication:

Login: Users/Employees can log in with their credentials. Successful login directs them to their respective dashboards (consumer or employee).

New Application: Users can apply for a new account by filling out a form with their details.

Consumer Dashboard:

View Pending Bills: Consumers can check their pending bills.

Register a Complaint: Consumers can register complaints regarding various issues.

Change Password: Consumers can change their account password.

Pay Bills: Consumers can pay their bills using debit/credit cards or UPI. A QR code is displayed for UPI payments.

View Complaints: Consumers can view their complaints and mark them as resolved, which deletes the complaint from the database.

Employee Dashboard:

Add Users: Employees can review the user applications and can add them as users in the database.

Billing: Employees can input the monthly usage in energy units which is then computed into payable amount and the user is billed.

Manage Complaints: Employees can view and manage consumer complaints. They can see complaint details and respond to them via email.

Complaint Response: Employees can send responses to consumer complaints, which are emailed to the consumers.

Complaint Management:

View Complaints: Both consumers and employees can view complaints. Employees can respond to complaints, and consumers can mark them as resolved.

Complaint Description: Detailed descriptions of complaints are displayed when selected from the list.

Email Integration:

An automated electronic mail system is used to notify the users and employees for almost all updates such as billing, complaints, responses, applications.

Dummy Payment Interface:

A dummy payment interface is made which either takes card details or shows a qr code for payment and marks the bills as paid after submission of the payment.

Libraries Utilised:

Tkinter and related libraries:

1. **Tkinter**: Used to create the graphical user interface (GUI) for the application.Provides various widgets like buttons, labels, frames, and entry fields.
2. **tkinter.messagebox**: Used to display message boxes for alerts, errors, and information dialogs.
3. **tkinter.ttk**: Provides themed widgets for a more modern look and feel.
4. **sv_ttk**: Third party library used to apply additional themes to tkinter widgets.
5. **PIL (Python Imaging Library)**: Specifically, the ImageTk and Image modules from this are used to handle and display images in the GUI.

MySQL Connector:

Used to connect to the MySQL database and execute SQL queries.

SMTP and related libraries:

1. smtplib:

   Used to send emails using the Simple Mail Transfer Protocol (SMTP).

2. email.mime.text:

   Used to create MIME (Multipurpose Internet Mail Extensions) text objects for email content.

3. email.mime.multipart:

   Used to create MIME multipart objects, allowing the inclusion of multiple parts (e.g., text and attachments) in an email.

Others:

1. os:

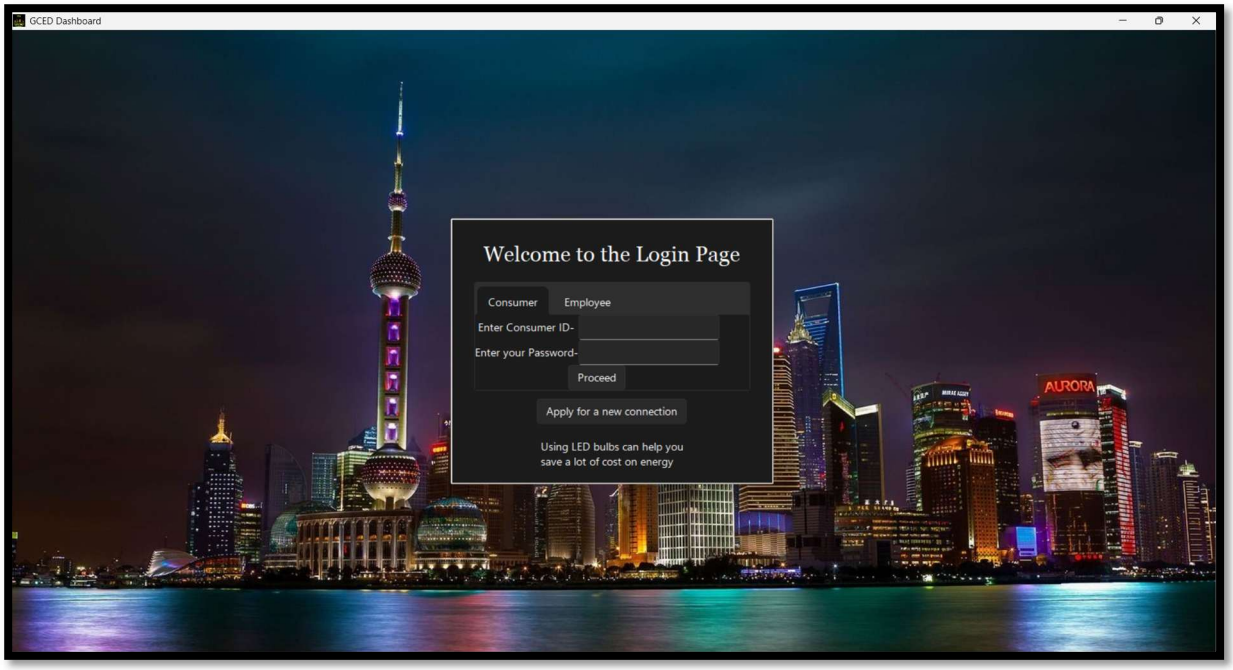
   Used to handle file paths and directories, especially for resource files.

2. sys:

   Used to handle paths for bundled applications created with tools like PyInstaller.

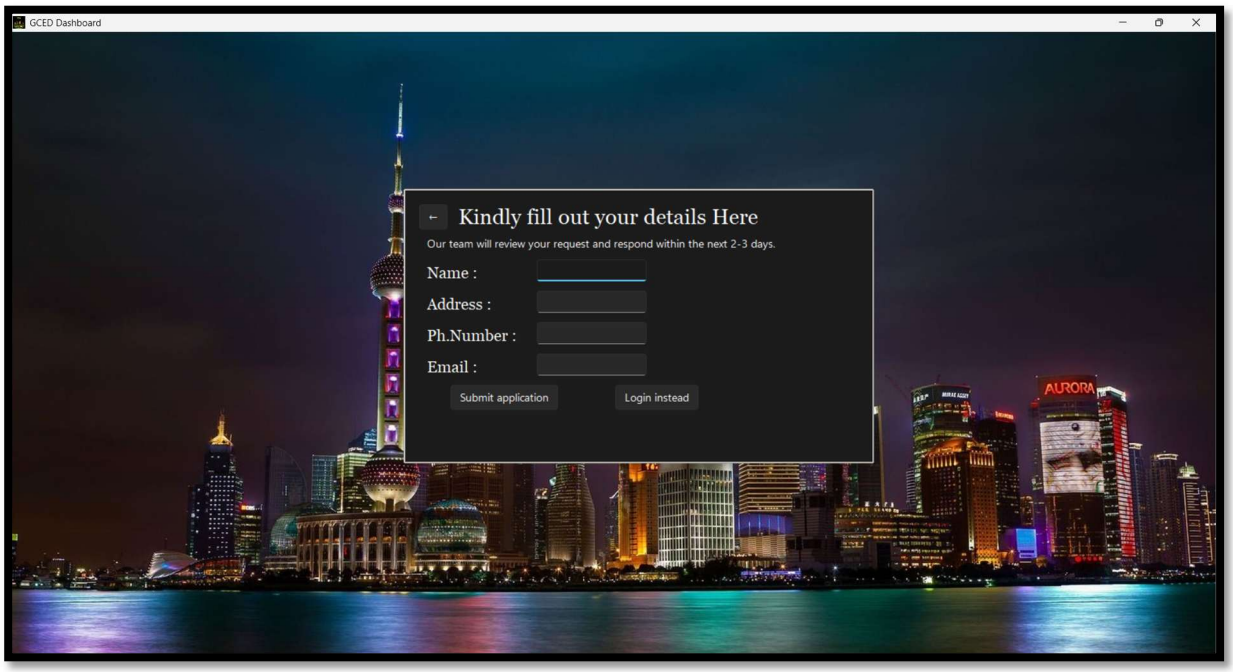
Application Usage

1.Login Window

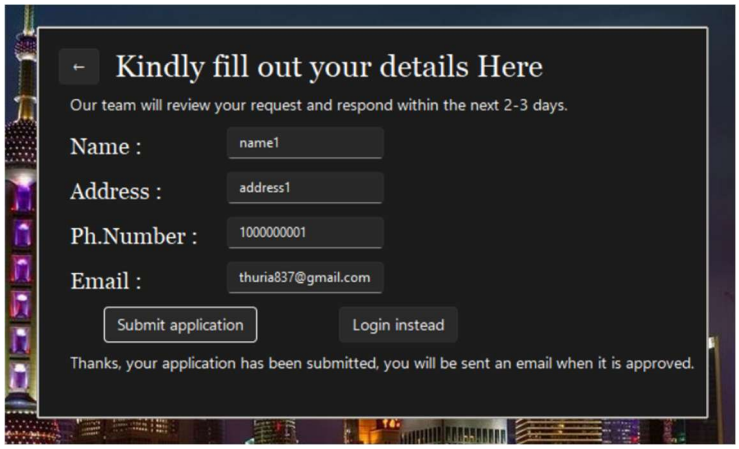


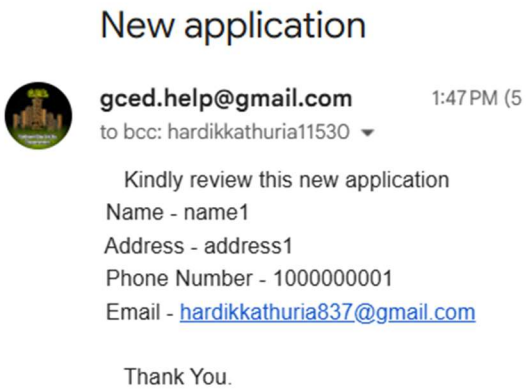
2.Applying for a new connection

After clicking on the new application button, the following window opens:

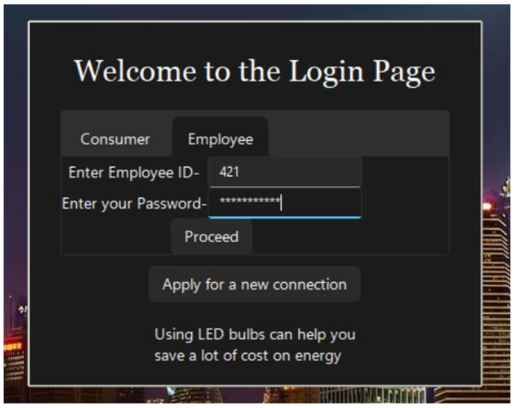


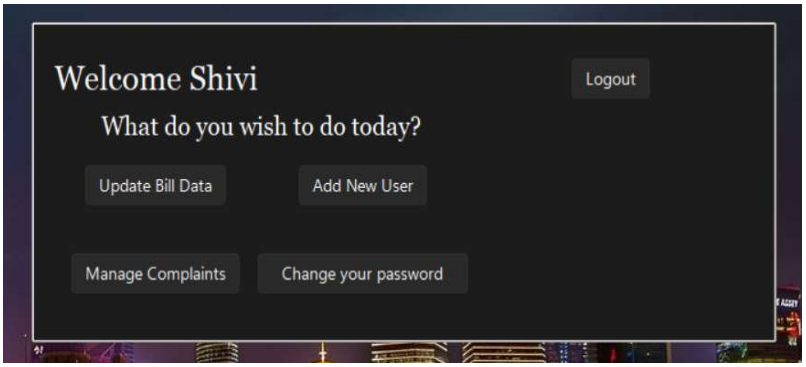
Then, after enter details and submitting the application, the application manager is sent an email to notify of this application.

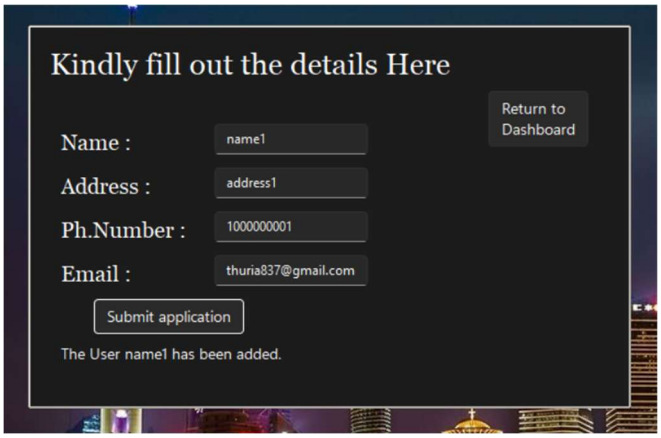


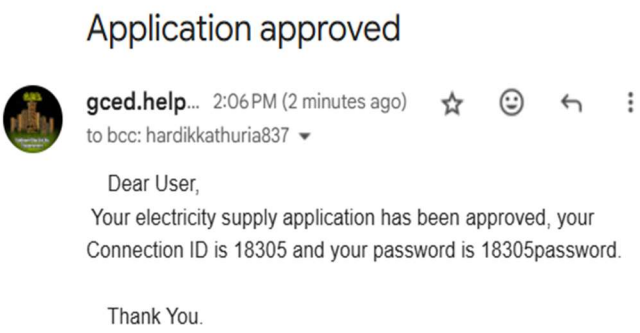


Then, the employee can verify the credentials, login to their dashboard and add the user, and the user is notified of their account creation and sent the account details.



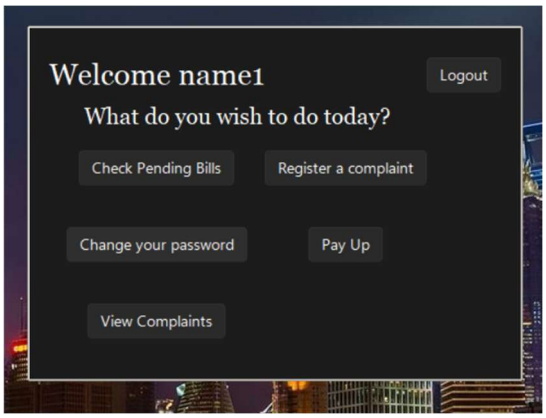


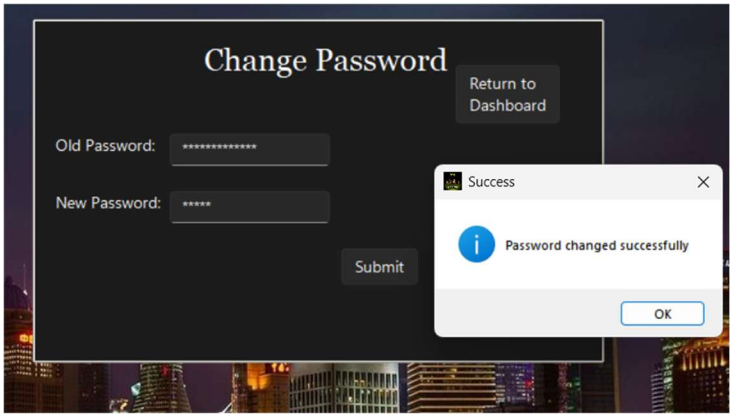




3. User Login and password change

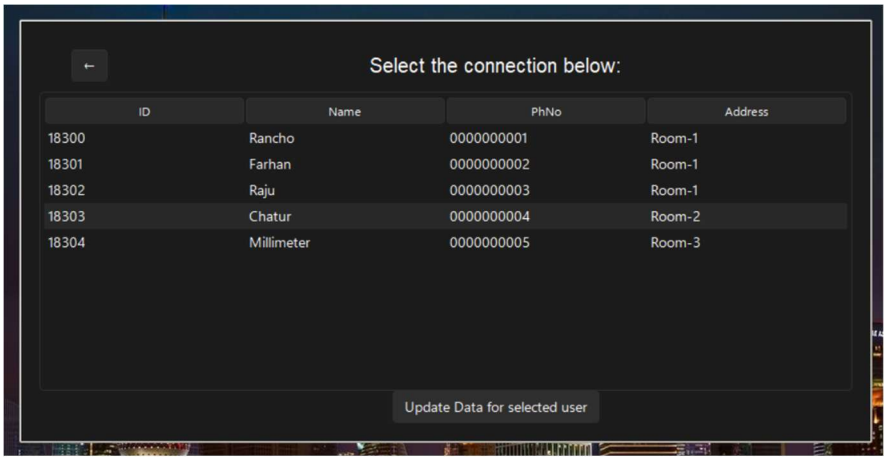
The user can login with the given credentials and then change their password to their own password.

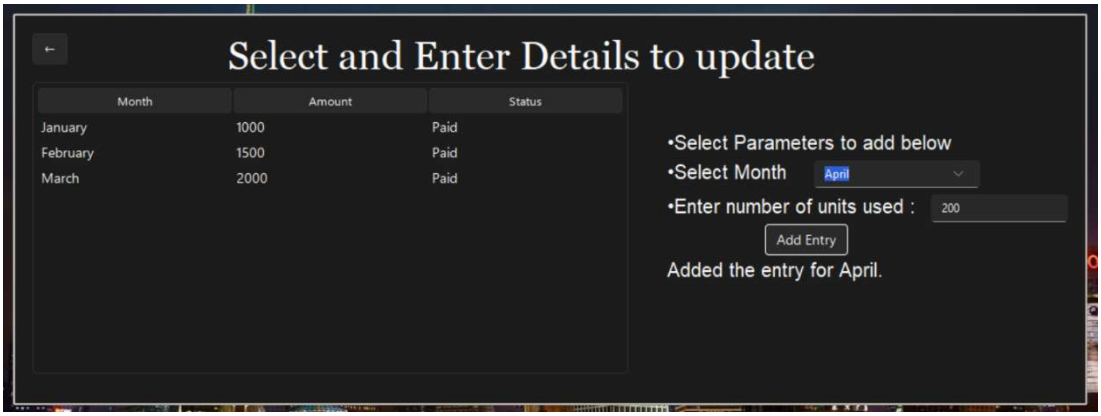




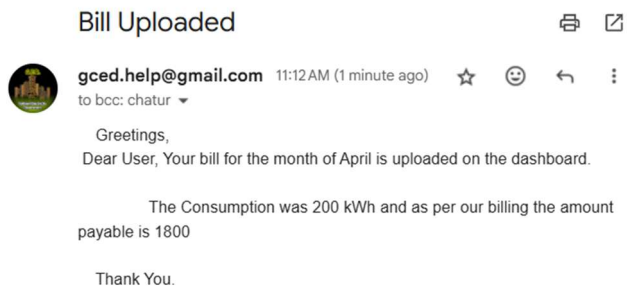
4. Bill Updation

An employee can login and input the usage data which is then uploaded as payable amount.



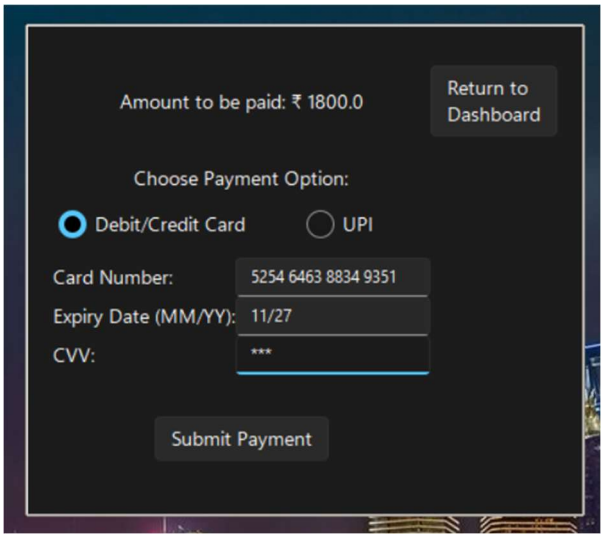


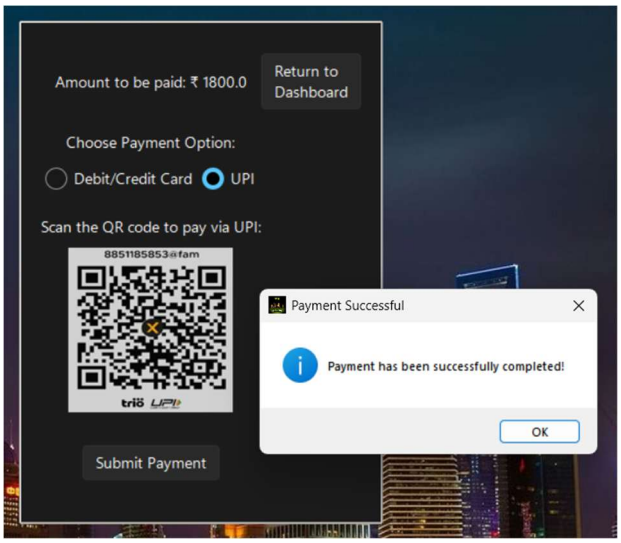
Subsequently, the user is sent a notification email for the bill.

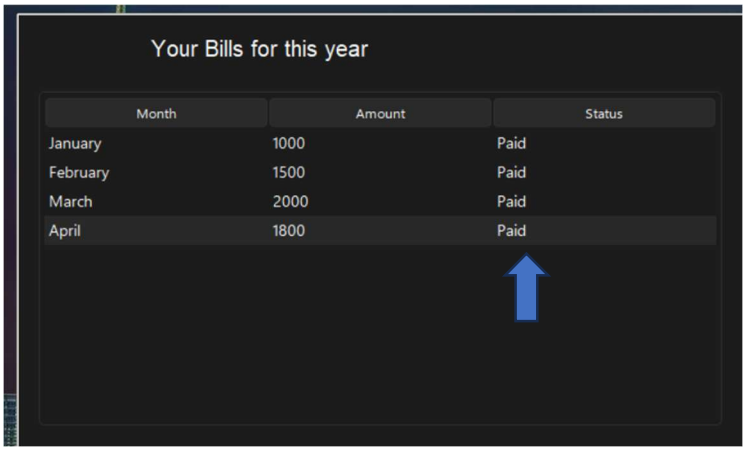


5. Payment

A dummy payment interface is added in the consumer dashboard, which sums up all the pending amounts and after submission marks them as paid.

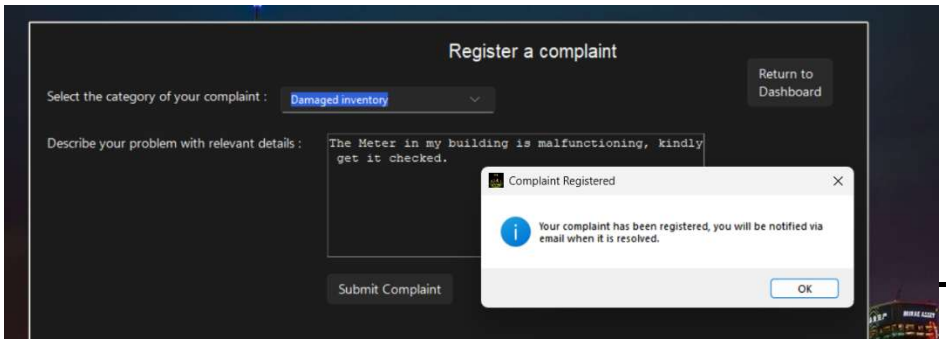






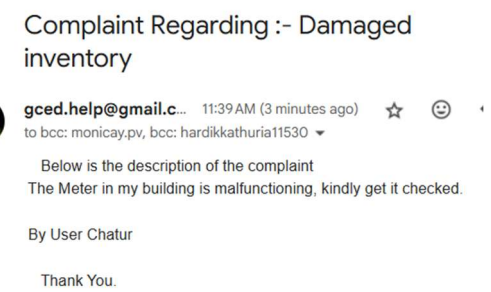
6.Complaints Management

The user can register their complaints with the categories and description of the complaint

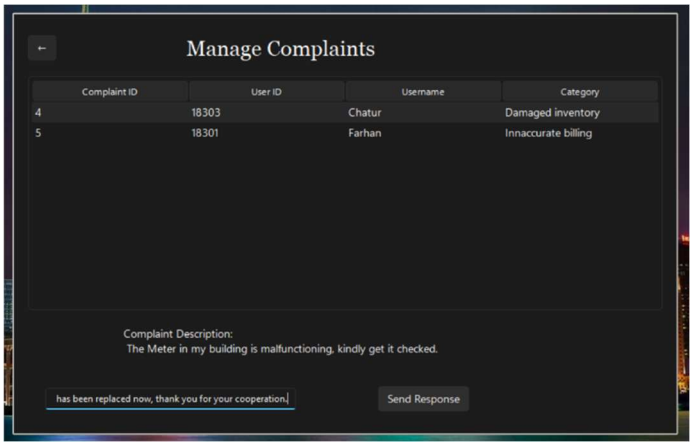


Based on the category of the complaint, the complaints manager and the other responsible manager is notified.

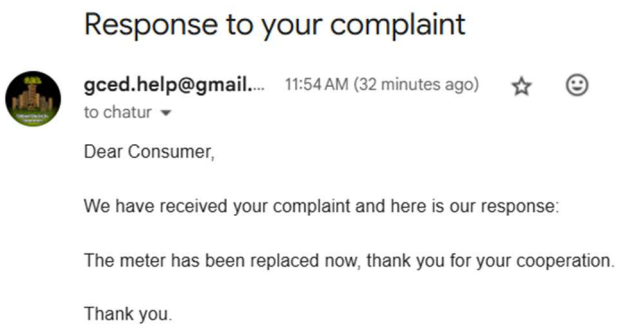
Email	Role
hardikkathuria837@gmail.com	Bills Manager
hardikkathuria11530@gmail.com	New application Manager
hardikkathuria11530@gmail.com	Inventory Manager
monicay.pv@dpsgs.org	Complaints Manager
hardik.kathuria837@outlook.com	Workers Manager

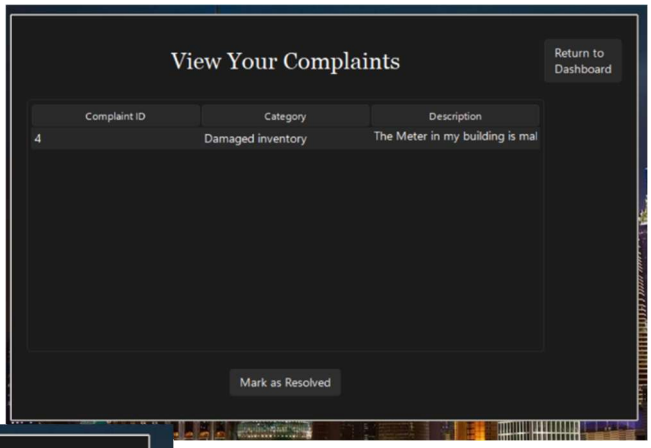


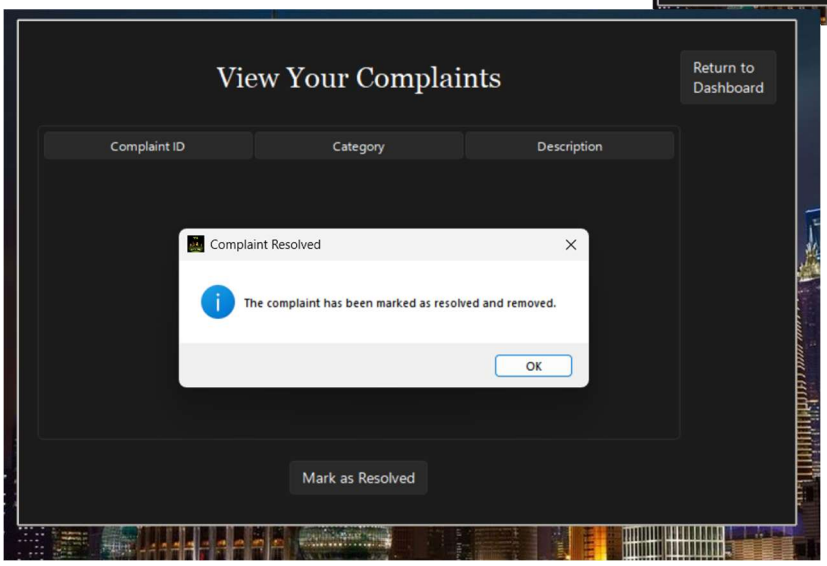
The Employee can then login and view the complaints and respond accordingly



The response is then emailed to the user and the user can mark the complaint as resolved in the app.







Program Code :

```
import mysql.connector as sql
import tkinter
from tkinter import messagebox,ttk
import sv_ttk
from PIL import ImageTk,Image
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
import os
import sys

try:
    database=sql.connect(host='127.0.0.1',
                         user='newuser',
                         password='newuserpassword')
    cursor=database.cursor(buffered=True)
except:
    connectionwin=tkinter.Tk()
    connectionwin.title("Connection Error")
    connectionwin.geometry("550x250")
    connectionlabel=ttk.Label(connectionwin,text="Kindly enter mysql username and password here.",font=('Georgia',17))
    usern=ttk.Label(connectionwin,text="Username :",font=('Georgia',15))
    passw=ttk.Label(connectionwin,text="Password :",font=('Georgia',15))
    username=tkinter.StringVar()
    password=tkinter.StringVar()
    mysqlusername=ttk.Entry(connectionwin,textvariable=username)
    mysqlpassword=ttk.Entry(connectionwin,textvariable=password)
    connectionlabel.place(x=0,y=0)
    usern.place(x=0,y=50)
    passw.place(x=0,y=100)
    mysqlusername.place(x=130,y=50)
    mysqlpassword.place(x=130,y=100)
    def connect():
        global database
        try:
            database=sql.connect(host='127.0.0.1',
                                 user=username.get(),
                                 password=password.get())
            connectionwin.destroy()
        except:
```

```
        messagebox.showerror("Error","Could not connect to database")
        exit()

    sv_ttk.set_theme("dark")
    connectbutton=ttk.Button(connectionwin,text="Connect",command=connect)
    connectbutton.place(x=200,y=150)
    connectionwin.mainloop()
cursor=database.cursor(buffered=True)
cursor.execute("SHOW DATABASES LIKE 'schoolprojects'")
result = cursor.fetchone()

if result:
    cursor.execute("USE schoolprojects")
else:
    cursor.execute("CREATE DATABASE schoolprojects")
    cursor.execute("USE schoolprojects")

    commands_list=[
    "CREATE TABLE employeedata(EmployeeID varchar(255) primary key,Name varchar(255),PhNumber varchar(255),Email varchar(255),password varchar(255),Role varchar(255))",
    "CREATE TABLE consumerdata(ConnectionID int primary key,Name varchar(255),Phno varchar(255),Address varchar(255),Password varchar(255),Email varchar(255))",
    "CREATE TABLE complaints(compid varchar(255),userid varchar(255),username varchar(255),category varchar(255),description text)",
    "CREATE TABLE billsdata2025(ID varchar(255),Month varchar(255),Amount varchar(255),Status varchar(255))",
    "insert into employeedata values('420','Aarudh','9876543210','hardikkathuria837@gmail.com','420password','Bills Manager')",
    "insert into employeedata values('421','Shivi','9876543211','hardikkathuria11530@gmail.com','421password','New application Manager')",
    "insert into employeedata values('422','Udit','9876543212','hardikkathuria11530@gmail.com','422password','Inventory Manager')",
    "insert into employeedata values('423','Palak','9876543213','monicay.pv@dpsgs.org','423password','Complaints Manager')",
    "insert into employeedata values('424','Mayank','9876543210','hardik.kathuria837@outlook.com','424password','Workers Manager')",
    "insert into consumerdata values(18300,'Rancho','0000000001','Room-1','18300password','hardikkathuria837@gmail.com')",
    "insert into consumerdata values(18301,'Farhan','0000000002','Room-1','18301password','hardikkathuria11530@gmail.com')",
    "insert into consumerdata values(18302,'Raju','0000000003','Room-1','18302password','raju@hotmail.com')",
    "insert into consumerdata values(18303,'Chatur','0000000004','Room-2','18303password','chatur@hotmail.com')",
    "insert into consumerdata values(18304,'Millimeter','0000000005','Room-3','18304password','mm@hotmail.com')",
    "insert into billsdata2025 values('18300','January','1000','Paid')",
    "insert into billsdata2025 values('18300','February','1500','Paid')",
    "insert into billsdata2025 values('18300','March','2000','Paid')",
    "insert into billsdata2025 values('18301','January','1000','Paid')",
    "insert into billsdata2025 values('18301','February','1500','Paid')",
    "insert into billsdata2025 values('18301','March','2000','Pending')",
    "insert into billsdata2025 values('18302','January','1000','Paid')",
    "insert into billsdata2025 values('18302','February','1500','Pending')",
    "insert into billsdata2025 values('18302','March','2000','Pending')",
```

```
    "insert into billsdata2025 values('18303','January','1000','Paid')",

    "insert into billsdata2025 values('18303','February','1500','Paid')",

    "insert into billsdata2025 values('18303','March','2000','Paid')",

    "insert into billsdata2025 values('18304','January','1000','Paid')",

    "insert into billsdata2025 values('18304','February','1500','Paid')",

    "insert into billsdata2025 values('18304','March','2000','Pending')",

    "insert into complaints values('1','18300','Rancho','Power Cuts','Power cuts are frequent in my area. Kindly look into the matter.')",

    "insert into complaints values('2','18301','Farhan','Damaged inventory','The meter is damaged and needs replacement. Please send someone to fix it.')",

    "insert into complaints values('3','18302','Raju','Payment problems','I have paid my bill but it is still showing as pending. Please update the status.')"

]

    for i in commands_list:

        cursor.execute(i)

        database.commit()

    print('database created')


def add_user():

    EmployeeID=idvar.get()

    employee_dashboard_frame.destroy()

    add_user_frame=ttk.Frame(main,relief='raised',border=20,height=350,width=550)

    add_user_framelabel=ttk.Label(add_user_frame,text='Kindly fill out the details Here',font=("Georgia",20))

    add_user_framelabel.place(x=0,y=0)

    def add_user_return_to_employee_dashboard():

        add_user_frame.destroy()

        employee_dashboard(EmployeeID)

    return_button=ttk.Button(add_user_frame,text="Return to \nDashboard",command=add_user_return_to_employee_dashboard)

    return_button.place(x=400,y=40)


    Namevar=tkinter.StringVar()

    Addressvar=tkinter.StringVar()

    phnovar=tkinter.StringVar()

    emailvar=tkinter.StringVar()


    labelone=ttk.Label(add_user_frame,text="Name :",font=('Georgia',15))

    labelone.place(x=10,y=75)

    entryone=ttk.Entry(add_user_frame,textvariable=Namevar)

    entryone.place(x=150,y=70)


    labeltwo=ttk.Label(add_user_frame,text="Address :",font=('Georgia',15))

    labeltwo.place(x=10,y=115)

    entrytwo=ttk.Entry(add_user_frame,textvariable=Addressvar)

    entrytwo.place(x=150,y=110)
```

```
    labelthree=ttk.Label(add_user_frame,text="Ph.Number :",font=('Georgia',15))
    labelthree.place(x=10,y=155)
    entrythree=ttk.Entry(add_user_frame,textvariable=phnovar)
    entrythree.place(x=150,y=150)

    labelfour=ttk.Label(add_user_frame,text="Email :",font=('Georgia',15))
    labelfour.place(x=10,y=195)
    entryfour=ttk.Entry(add_user_frame,textvariable=emailvar)
    entryfour.place(x=150,y=190)

    cursor.execute("select connectionid from consumerdata")
    idslist_=cursor.fetchall()
    idlist=[]
    for i in range(len(idslist_)):
        id=int(idslist_.pop()[0])
        idlist.append(id)
    idlist.sort()
    idu=str(idlist[-1]+1)
    password=idu+"password"

    def add_now():
        name,address,phno,emailu=Namevar.get(),Addressvar.get(),phnovar.get(),emailvar.get()
        accepted_label=ttk.Label(add_user_frame,text=f"The User {name} has been added.")
        accepted_label.place(x=10,y=270)
        add_consumer(idu,name,phno,address,password,emailu)
        subject="Application approved"
        content=f"Dear User, \n Your electricity supply application has been approved, your Connection ID is {idu} and your password is
{password}."
        send_email(emailu,subject,content)
    submitbutton=ttk.Button(add_user_frame,text="Submit application",command=add_now)
    submitbutton.place(x=40,y=230)
    add_user_frame.place(x=400,y=150)

def send_email(To,Subject,Content):
    HOST = "smtp.gmail.com"
    PORT = 587

    FROM_EMAIL = "gced.help@gmail.com"
    TO_EMAIL = To
    PASSWORD = "xqte qigp gseh dodc"

    MESSAGE = f"""Subject: {Subject} \n
```

```
    {Content}

    Thank You."""

    smtp = smtplib.SMTP(HOST, PORT)

    smtp.ehlo()
    smtp.starttls()
    smtp.login(FROM_EMAIL, PASSWORD)

    smtp.sendmail(FROM_EMAIL, TO_EMAIL, MESSAGE)
    print("Sent Email")
    smtp.quit()

def register_complaint():
    consumer_dashboard_frame.destroy()
    connectionid=idvar.get()
    username=cursor.execute(f"select Name from consumerdata where connectionid='{connectionid}'")
    regcomplaint_frame=ttk.Frame(main,relief='raised',border=20,height=450,width=900)
    regcomplaintheading=ttk.Label(regcomplaint_frame,text="Register a complaint",font=(35))

    def complaint_return_to_consumer_dashboard():
        regcomplaint_frame.destroy()
        consumer_dashboard(connectionid)
    return_button=ttk.Button(regcomplaint_frame,text="Return to \nDashboard",command=complaint_return_to_consumer_dashboard)
    return_button.place(x=750,y=20)

    categoryvar=tkinter.StringVar()
    category=ttk.Combobox(regcomplaint_frame,width=30,textvariable=categoryvar)
    category['values']=('Innaccurate billing','Uncooperative workers','Payment problems','Power Cuts','Damaged inventory','Other')
    categorylabel=ttk.Label(regcomplaint_frame,text="Select the category of your complaint :")
    categorylabel.place(x=0,y=50)
    describelabel=ttk.Label(regcomplaint_frame,text="Describe your problem with relevant details :")
    describelabel.place(x=0,y=100)
    category.place(x=250,y=50)
    regcomplaintheading.place(x=430,y=0)
    describe=tkinter.Text(regcomplaint_frame,width=50,height=8)
    describe.place(x=300,y=100)

    def submit_complaint():
        complaint_info=describe.get("1.0",'end')
        connectionid=idvar.get()
```

```
    cursor.execute(f"select Name from consumerdata where connectionid='{connectionid}'")
    username=cursor.fetchone()[0]
    emailsto=[]
    cursor.execute("select compid from complaints")
    idslist_=cursor.fetchall()
    idlist=[]
    for i in range(len(idslist_)):
        id=int(idslist_.pop()[0])
        idlist.append(id)
    idlist.sort()
    if len(idlist)==0:
        compid='1'
    else:
        compid=str(idlist[-1]+1)
    cmd=f"insert into complaints values('{compid}','{connectionid}','{username}','{str(categoryvar.get())}','{complaint_info}')"
    cursor.execute(cmd)
    database.commit()
    cursor.execute(f"select Email from employeedata where Role='Complaints Manager'")
    email=cursor.fetchall()[0][0]
    emailsto.append(email)
    if str(categoryvar.get())=='Damaged inventory'or str(categoryvar.get())=='Power Cuts':
        cursor.execute(f"select Email from employeedata where Role='Inventory Manager'")
        email=cursor.fetchall()[0][0]
        emailsto.append(email)
    elif str(categoryvar.get())=='Innaccurate billing' or str(categoryvar.get())=='Payment problems':
        cursor.execute(f"select Email from employeedata where Role='Bills Manager'")
        email=cursor.fetchall()[0][0]
        emailsto.append(email)
    elif str(categoryvar.get())=='Uncooperative workers':
        cursor.execute(f"select Email from employeedata where Role='Workers Manager'")
        email=cursor.fetchall()[0][0]
        emailsto.append(email)
    subject=f"Complaint Regarding :- {str(categoryvar.get())}"
    content=f"Below is the description of the complaint \n {complaint_info} \n By User {username}"
    send_email(emailsto,subject,content)
    messagebox.showinfo("Complaint Registered","Your complaint has been registered, you will be notified via email when it is resolved.")

  submitbutton=ttk.Button(regcomplaint_frame,text="Submit Complaint",command=submit_complaint)
  submitbutton.place(x=300,y=250)
  regcomplaint_frame.place(x=250,y=150)

def showyearbills_c():
```

```
    consumer_dashboard_frame.destroy()

    connectionid=idvar.get()
    showyearbills_frame=ttk.Frame(main,relief='raised',border=20,height=450,width=900)
    showyearbills_heading=ttk.Label(showyearbills_frame,text="Your Bills for this year",font=(35))
    showyearbills_heading.place(x=100,y=0)
    def showyear_return_to_consumer_dashboard():
        showyearbills_frame.destroy()
        consumer_dashboard(connectionid)
    return_button=ttk.Button(showyearbills_frame,text="Return to \nDashboard",command=showyear_return_to_consumer_dashboard)
    return_button.place(x=750,y=20)
    def destroyandpay():
        showyearbills_frame.destroy()
        do_payment_frame()
    pay_button=ttk.Button(showyearbills_frame,text="Clear Dues",command=destroyandpay)
    pay_button.place(x=750,y=90)
    showuserbills=ttk.Treeview(showyearbills_frame,columns=["Month","Amount","Status"])
    showuserbills.column("#0",width=0,stretch='NO')
    showuserbills.heading("Month", text="Month")
    showuserbills.heading("Amount", text="Amount")
    showuserbills.heading("Status", text="Status")

    getbillscmd=f"select Month,Amount,Status from billsdata2025 where ID='{connectionid}'"
    cursor.execute(getbillscmd)
    n=0
    for i in cursor:
        showuserbills.insert(parent="",index="end",iid=n,text=str(connectionid),values=i)
        n+=1
    showuserbills.place(x=0,y=50)
    showyearbills_frame.place(x=400,y=200)

def updatebilldata_e():
    EmployeeID=idvar.get()
    employee_dashboard_frame.destroy()
    updatebillwindow=ttk.Frame(main,relief='raised',border=20)

    def return_to_employee_dashboard():
        updatebillwindow.destroy()
        employee_dashboard(EmployeeID)
    return_button=ttk.Button(updatebillwindow,text="←",command=return_to_employee_dashboard)
    return_button.grid(row=0,column=0,pady=10)
```

```
selectidlabel=ttk.Label(updatebillwindow,text="Select the connection below:",font=(30))
selectid=ttk.Treeview(updatebillwindow, selectmode ='browse',columns=["ID",'Name','PhNo','Address'])
selectid.column("#0",width=0,stretch='NO')
selectid.heading("ID",text="ID")
selectid.heading("Name",text="Name")
selectid.heading("PhNo",text="PhNo")
selectid.heading("Address",text="Address")

consumerlist=[]
getdatacmd='select ConnectionID,Name,Phno,Address from consumerdata'
cursor.execute(getdatacmd)
for i in cursor:
    consumerlist.append(i)
n=0
for i in consumerlist:
    selectid.insert(parent="",index="end",iid=n,text=str(18300+n),values=i)
    n+=1
    def select_and_update():
        selectandupdatewindow=ttk.Frame(main,height=400,width=1100,relief='raised',border=20)
        selectandupdateheading=ttk.Label(selectandupdatewindow,text="Select and Enter Details to update",font=('Georgia',30))
        selected_item=selectid.focus()
        details=selectid.item(selected_item)
        selected_id=details["text"]
        def return_to_update_bill_window():
            selectandupdatewindow.destroy()
            updatebilldata_e()
        return_button=ttk.Button(selectandupdatewindow,text="←",command=return_to_update_bill_window)
        return_button.place(x=0,y=0)
        updatebillwindow.destroy()
        showuserbills=ttk.Treeview(selectandupdatewindow,columns=["Month","Amount","Status"])
        showuserbills.column("#0",width=0,stretch='NO')
        showuserbills.heading("Month", text="Month")
        showuserbills.heading("Amount", text="Amount")
        showuserbills.heading("Status", text="Status")

        getbillscmd=f"select Month,Amount,Status from billsdata2025 where ID='{selected_id}'"
        cursor.execute(getbillscmd)
        n=0
        for i in cursor:
            showuserbills.insert(parent="",index="end",iid=n,text=str(selected_id),values=i)
            n+=1
```

```
    addinglabel=ttk.Label(selectandupdatewindow,text='•Select Parameters to add below',font=(20))
    selectmonthlabel=ttk.Label(selectandupdatewindow,text="•Select Month",font=(20))
    monthvar=tkinter.StringVar()
    unitsvar=tkinter.StringVar()
    selectmonth=ttk.Combobox(selectandupdatewindow,values=["January", "February", "March", "April", "May", "June",
"July", "August", "September", "October", "November", "December"],textvariable=monthvar)

    enterunitslabel=ttk.Label(selectandupdatewindow,text="•Enter number of units used :",font=(20))
    enterunits=ttk.Entry(selectandupdatewindow,textvariable=unitsvar)

    def enter_values():
      month=monthvar.get()
      units=unitsvar.get()
      amount=calculate_electricity_bill(units=units)
      cmd=f"insert into billsdata2025 values('{selected_id}','{month}','{amount}','Pending')"
      cursor.execute(cmd)
      database.commit()
      donelabel=ttk.Label(selectandupdatewindow,text=f"Added the entry for {month}.",font=(20))
      donelabel.place(x=650,y=230)
      cmd1=f"select email from consumerdata where ConnectionID={selected_id}"
      cursor.execute(cmd1)
      for i in cursor:
        To=i
      Subject="Bill Uploaded"
      Content=f"""Greetings, \n Dear User, Your bill for the month of {month} is uploaded on the dashboard.\n
      The Consumption was {units} kWh and as per our billing the amount payable is {amount} """
      send_email(To,Subject,Content)
    addentry=ttk.Button(selectandupdatewindow,text="Add Entry",command=enter_values)

    selectandupdateheading.place(x=200,y=0)
    showuserbills.place(x=0,y=50)
    addinglabel.place(x=650,y=100)
    selectmonthlabel.place(x=650,y=130)
    selectmonth.place(x=800,y=130)
    enterunitslabel.place(x=650,y=165)
    enterunits.place(x=920,y=165)
    addentry.place(x=750,y=195)

    selectandupdatewindow.place(x=250,y=200)
proceedbutton=ttk.Button(updatebillwindow,text="Update Data for selected user",command=select_and_update)

selectidlabel.grid(row=0,column=4)
```

```
    selectid.grid(row=1,column=0,columnspan=8)
    proceedbutton.grid(row=2,column=4)
    updatebillwindow.place(x=350,y=150)

def change_consumer_password():
    consumer_dashboard_frame.destroy()

    def submit_password_change():
        connectionid = idvar.get()
        old_password = old_password_entry.get()
        new_password = new_password_entry.get()

        cursor.execute(f"select Password from consumerdata where connectionid='{connectionid}'")
        current_password = cursor.fetchone()[0]

        if current_password == old_password:
            cursor.execute(f"update consumerdata set Password='{new_password}' where connectionid='{connectionid}'")
            database.commit()
            messagebox.showinfo("Success", "Password changed successfully")
        else:
            messagebox.showerror("Error", "Old password is incorrect")
    connectionid=idvar.get()
    change_password_frame = ttk.Frame(main,relief='raised',border=20,height=300,width=500)
    change_password_frame.place(x=600,y=250)
    def return_to_consumer_dashboard():
        change_password_frame.destroy()
        consumer_dashboard(connectionid)
    return_button=ttk.Button(change_password_frame,text="Return to \nDashboard",command=return_to_consumer_dashboard)
    return_button.place(x=350,y=20)

    change_password_heading = ttk.Label(change_password_frame, text="Change Password",font=("Georgia",20))
    change_password_heading.place(x=130, y=0)

    old_password_label = ttk.Label(change_password_frame, text="Old Password:")
    old_password_label.place(x=0, y=80)
    old_password_entry = ttk.Entry(change_password_frame, show="*")
    old_password_entry.place(x=100, y=80)

    new_password_label = ttk.Label(change_password_frame, text="New Password:")
    new_password_label.place(x=0, y=130)
    new_password_entry = ttk.Entry(change_password_frame, show="*")
    new_password_entry.place(x=100, y=130)
```

```
    submit_button = ttk.Button(change_password_frame, text="Submit", command=submit_password_change)
    submit_button.place(x=250, y=180)

def change_employee_password():
    employee_dashboard_frame.destroy()
    def submit_password_change():
        employeeid = idvar.get()
        old_password = old_password_entry.get()
        new_password = new_password_entry.get()

        cursor.execute(f"select Password from employeedata where employeeid='{employeeid}'")
        current_password = cursor.fetchone()[0]

        if current_password == old_password:
            cursor.execute(f"update employeedata set Password='{new_password}' where employeeid='{employeeid}'")
            database.commit()
            messagebox.showinfo("Success", "Password changed successfully")
        else:
            messagebox.showerror("Error", "Old password is incorrect")

    change_password_frame = ttk.Frame(main,relief='raised',border=20,height=300,width=450)
    change_password_frame.place(x=600,y=250)
    EmployeeID=idvar.get()
    def return_to_employee_dashboard():
        change_password_frame.destroy()
        employee_dashboard(EmployeeID)
    return_button=ttk.Button(change_password_frame,text="←",command=return_to_employee_dashboard)
    return_button.place(x=0,y=0)

    change_password_heading = ttk.Label(change_password_frame, text="Change Password",font=("Georgia",20))
    change_password_heading.place(x=130, y=0)

    old_password_label = ttk.Label(change_password_frame, text="Old Password:")
    old_password_label.place(x=0, y=80)
    old_password_entry = ttk.Entry(change_password_frame, show="*")
    old_password_entry.place(x=100, y=80)

    new_password_label = ttk.Label(change_password_frame, text="New Password:")
    new_password_label.place(x=0, y=130)
    new_password_entry = ttk.Entry(change_password_frame, show="*")
    new_password_entry.place(x=100, y=130)
```

```
    submit_button = ttk.Button(change_password_frame, text="Submit", command=submit_password_change)
    submit_button.place(x=250, y=180)

def resource_path(relative_path):
    try:
        base_path = sys._MEIPASS
    except Exception:
        base_path = os.path.abspath(".")

    return os.path.join(base_path, relative_path)

def do_payment_frame():
    consumer_dashboard_frame.destroy()
    consumerid= idvar.get()
    cursor.execute(f"select sum(Amount) from billsdata2025 where ID='{consumerid}' and Status='Pending'")
    amount = cursor.fetchone()[0]
    payment_frame = ttk.Frame(main, relief='raised', border=20, width=700, height=1000)
    payment_frame.place(x=600, y=100)

    def return_to_consumer_dashboard():
        payment_frame.destroy()
        consumer_dashboard(consumerid)
    return_button=ttk.Button(payment_frame,text="Return to \nDashboard",command=return_to_consumer_dashboard)
    return_button.grid(row=0, column=2, pady=10)

    amount_label = ttk.Label(payment_frame, text=f"Amount to be paid: ₹ {amount}")
    amount_label.grid(row=0, column=0, columnspan=2, pady=10)

    payment_options_label = ttk.Label(payment_frame, text="Choose Payment Option:")
    payment_options_label.grid(row=1, column=0, columnspan=2, pady=10)

    card_option = ttk.Radiobutton(payment_frame, text="Debit/Credit Card", value="card")
    card_option.grid(row=2, column=0, sticky="w")

    upi_option = ttk.Radiobutton(payment_frame, text="UPI", value="upi")
    upi_option.grid(row=2, column=1, sticky="w")

    card_frame = ttk.Frame(payment_frame)
    card_frame.grid(row=3, column=0, columnspan=2, pady=10)

    card_number_label = ttk.Label(card_frame, text="Card Number:")
```

```
card_number_label.grid(row=0, column=0, sticky="w")
card_number_entry = ttk.Entry(card_frame)
card_number_entry.grid(row=0, column=1)

expiry_date_label = ttk.Label(card_frame, text="Expiry Date (MM/YY):")
expiry_date_label.grid(row=1, column=0, sticky="w")
expiry_date_entry = ttk.Entry(card_frame)
expiry_date_entry.grid(row=1, column=1)

cvv_label = ttk.Label(card_frame, text="CVV:")
cvv_label.grid(row=2, column=0, sticky="w")
cvv_entry = ttk.Entry(card_frame, show="*")
cvv_entry.grid(row=2, column=1)

upi_frame = ttk.Frame(payment_frame)
upi_frame.grid(row=4, column=0, columnspan=2, pady=10)

qr_code_label = ttk.Label(upi_frame, text="Scan the QR code to pay via UPI:")
qr_code_label.grid(row=0, column=0, columnspan=2, pady=10)

qr_code_image = Image.open(resource_path("D:\\Laptop Transfer\\sqlschlproject\\qr_code.jpg"))
qr_code_image = qr_code_image.resize((150, 150))
qr_code_photo = ImageTk.PhotoImage(qr_code_image)

qr_code_display = ttk.Label(upi_frame, image=qr_code_photo)
qr_code_display.image = qr_code_photo
qr_code_display.grid(row=1, column=0, columnspan=2)

def submitted():
    messagebox.showinfo("Payment Successful", "Payment has been successfully completed!")
    cursor.execute(f"update billsdata2025 set Status='Paid' where ID='{consumerid}' and Status='Pending'")
    database.commit()
    payment_frame.destroy()
    consumer_dashboard(consumerid)
submit_button = ttk.Button(payment_frame, text="Submit Payment",command=submitted)
submit_button.grid(row=5, column=0, columnspan=2, pady=20)

def show_payment_option():
    if card_option.instate(['selected']):
        card_frame.grid()
        upi_frame.grid_remove()
    elif upi_option.instate(['selected']):
```

```
        upi_frame.grid()
        card_frame.grid_remove()

    card_option.config(command=show_payment_option)
    upi_option.config(command=show_payment_option)

    card_frame.grid_remove()
    upi_frame.grid_remove()

def calculate_electricity_bill(units):
    units=int(units)
    rate_1 = 5
    rate_2 = 8
    rate_3 = 10

    if units <= 100:
        bill = units * rate_1
    elif units <= 200:
        bill = (100 * rate_1) + ((units - 100) * rate_2)
    else:
        bill = (100 * rate_1) + (100 * rate_2) + ((units - 200) * rate_3)

    if bill < 1000:
        bill = 1000
    else:
        bill+=500
    return bill

def manage_complaints():
    EmployeeID=idvar.get()

    employee_dashboard_frame.destroy()
    manage_complaints_frame = ttk.Frame(main, relief='raised', border=20)
    manage_complaints_frame.place(x=400,y=200)

    cursor.execute("SELECT compid, userid, username, category FROM complaints")
    complaints = cursor.fetchall()
    complaints_heading = ttk.Label(manage_complaints_frame, text="Manage Complaints", font=("Georgia", 20))
    complaints_heading.grid(row=0, column=0, columnspan=3, pady=10)

    def return_to_employee_dashboard():
        manage_complaints_frame.destroy()
```

```
    employee_dashboard(EmployeeID)
return_button=ttk.Button(manage_complaints_frame,text="←",command=return_to_employee_dashboard)
return_button.grid(row=0,column=0,sticky='w')

complaints_tree = ttk.Treeview(manage_complaints_frame, columns=("complaintid", "userid","username", "category"), show='headings')
complaints_tree.heading("complaintid", text="Complaint ID")
complaints_tree.heading("userid", text="User ID")
complaints_tree.heading("username", text="Username")
complaints_tree.heading("category", text="Category")
complaints_tree.grid(row=1, column=0, columnspan=4, pady=10)

for complaint in complaints:
    complaints_tree.insert("", "end", values=complaint)

def show_complaint_description(event):
    selected_item = complaints_tree.selection()[0]
    complaintid = complaints_tree.item(selected_item, "values")[0]

    cursor.execute(f"SELECT description FROM complaints WHERE compid={complaintid}")
    description = cursor.fetchone()[0]

    description_label.config(text=f"Complaint Description:\n {description}")
def enable_response_and_show(event):
    show_complaint_description(event)
    response_entry.config(state='normal')
    send_response_button.config(state='normal')
complaints_tree.bind("<<TreeviewSelect>>", enable_response_and_show)

description_label = ttk.Label(manage_complaints_frame, text="Complaint Description: ")
description_label.grid(row=2, column=0, columnspan=3, pady=10)
response_entry = ttk.Entry(manage_complaints_frame, width=50, state='disabled')
response_entry.grid(row=3, column=0, columnspan=2, pady=10)

def send_response():
    selected_item = complaints_tree.selection()[0]
    complaintid = complaints_tree.item(selected_item, "values")[0]
    response = response_entry.get()

    cursor.execute(f"SELECT email FROM consumerdata WHERE connectionid=(SELECT userid FROM complaints WHERE
compid={complaintid})")
    user_email = cursor.fetchone()[0]

    sender_email = "gced.help@gmail.com"
```

```
        sender_password = "xqte qigp gseh dodc"
        subject = "Response to your complaint"
        body = f"Dear Consumer,\n\nWe have received your complaint and here is our response:\n\n{response}\n\nThank you."

        msg = MIMEMultipart()
        msg['From'] = sender_email
        msg['To'] = user_email
        msg['Subject'] = subject
        msg.attach(MIMEText(body, 'plain'))

        server = smtplib.SMTP('smtp.gmail.com', 587)
        server.starttls()
        server.login(sender_email, sender_password)
        text = msg.as_string()
        server.sendmail(sender_email, user_email, text)
        server.quit()
        messagebox.showinfo("Email Sent", "Response email has been sent to the user")
    send_response_button = ttk.Button(manage_complaints_frame, text="Send Response", command=send_response, state='disabled')
    send_response_button.grid(row=3, column=2, pady=10)

def view_consumer_complaints():
    consumer_dashboard_frame.destroy()
    consumer_id=idvar.get()
    view_complaints_frame = ttk.Frame(main, relief='raised', border=20)
    view_complaints_frame.place(x=250,y=150)
    view_complaints_heading = ttk.Label(view_complaints_frame, text="View Your Complaints", font=("Georgia", 20))
    view_complaints_heading.grid(row=0, column=0, columnspan=3, pady=10)
    def return_to_consumer_dashboard():
        view_complaints_frame.destroy()
        consumer_dashboard(consumer_id)
    return_button=ttk.Button(view_complaints_frame,text="Return to \nDashboard",command=return_to_consumer_dashboard)
    return_button.grid(row=0, column=3, pady=10,sticky='n')

    cursor.execute(f"SELECT compid, category, description FROM complaints WHERE userid={consumer_id}")
    complaints = cursor.fetchall()

    complaints_tree = ttk.Treeview(view_complaints_frame, columns=("compid", "category", "description"), show='headings')
    complaints_tree.heading("compid", text="Complaint ID")
    complaints_tree.heading("category", text="Category")
    complaints_tree.heading("description", text="Description")
    complaints_tree.grid(row=1, column=0, columnspan=3, pady=10)
```

```
    for complaint in complaints:
        complaints_tree.insert("", "end", values=complaint)

    def mark_as_resolved():
        selected_item = complaints_tree.selection()[0]
        complaintid = complaints_tree.item(selected_item, "values")[0]

        cursor.execute(f"DELETE FROM complaints WHERE compid={complaintid}")
        database.commit()

        complaints_tree.delete(selected_item)

        messagebox.showinfo("Complaint Resolved", "The complaint has been marked as resolved and removed.")

    resolve_button = ttk.Button(view_complaints_frame, text="Mark as Resolved", command=mark_as_resolved)
    resolve_button.grid(row=2, column=0, columnspan=3, pady=10)

def employee_dashboard(EmployeeID):
    global employee_dashboard_frame

    namearg="select Name from employeedata where EmployeeID="+str(EmployeeID)
    cursor.execute(namearg)
    for i in cursor:
        nametuple=i
    loginwindow.destroy()
    employee_dashboard_frame=ttk.Frame(main,relief='raised',border=20)
    employee_dashboard_frame.columnconfigure(2,minsize=250)

    def logout():
        employee_dashboard_frame.destroy()
        login_window()
    logoutbutton=ttk.Button(employee_dashboard_frame,text="Logout",command=logout)
    logoutbutton.grid(row=0,column=2)
    employee_dashboard_frame.rowconfigure(0,minsize=50)
    employee_dashboard_frame.rowconfigure(2,minsize=70)
    employee_dashboard_frame.rowconfigure(3,minsize=70)

    welcomelabel=ttk.Label(employee_dashboard_frame,text=str("Welcome "+nametuple[0]),font=("Georgia",20),justify="left")
    welcomelabel.grid(row=0,column=0)

    anotherlabel=ttk.Label(employee_dashboard_frame,text="What do you wish to do today?",font=("Georgia",16))
    anotherlabel.grid(row=1,column=0,columnspan=2)
```

```
    updatebill=ttk.Button(employee_dashboard_frame,text="Update Bill Data",command=updatebilldata_e)
    updatebill.grid(row=2,column=0)

    adduser=ttk.Button(employee_dashboard_frame,text="Add New User",command=add_user)
    adduser.grid(row=2,column=1)

    manage_complaints_button = ttk.Button(employee_dashboard_frame, text="Manage Complaints", command=manage_complaints)
    manage_complaints_button.grid(row=3, column=0, pady=10)

    changepwdbutton=ttk.Button(employee_dashboard_frame,text="Change your password",command=change_employee_password,width=20)
    changepwdbutton.grid(row=3,column=1)

    employee_dashboard_frame.place(x=550,y=240)

def consumer_dashboard(ConnectionID):
    global consumer_dashboard_frame

    namearg="select Name from consumerdata where ConnectionID="+str(ConnectionID)
    cursor.execute(namearg)
    for i in cursor:
        nametuple=i
    loginwindow.destroy()
    consumer_dashboard_frame=ttk.Frame(main,relief='raised',border=20)
    def logout():
        consumer_dashboard_frame.destroy()
        login_window()
    logoutbutton=ttk.Button(consumer_dashboard_frame,text="Logout",command=logout)
    logoutbutton.grid(row=0,column=2)
    consumer_dashboard_frame.rowconfigure(0,minsize=50)
    consumer_dashboard_frame.rowconfigure(2,minsize=70)
    consumer_dashboard_frame.rowconfigure(3,minsize=70)
    consumer_dashboard_frame.rowconfigure(4,minsize=70)
    welcomelabel=ttk.Label(consumer_dashboard_frame,text=str("Welcome "+nametuple[0]),font=("Georgia",20),justify="left")
    welcomelabel.grid(row=0,column=0)

    anotherlabel=ttk.Label(consumer_dashboard_frame,text="What do you wish to do today?",font=("Georgia",16))
    anotherlabel.grid(row=1,column=0,columnspan=2)
    showbill=ttk.Button(consumer_dashboard_frame,text="Check Pending Bills",command=showyearbills_c)
    showbill.grid(row=2,column=0)
    complaintbutton=ttk.Button(consumer_dashboard_frame,text="Register a complaint",command=register_complaint)
    complaintbutton.grid(row=2,column=1)
```

```
    changepwdbutton=ttk.Button(consumer_dashboard_frame,text="Change your password",command=change_consumer_password)
    changepwdbutton.grid(row=3,column=0)
    paymentbutton=ttk.Button(consumer_dashboard_frame,text="Pay Up",command=do_payment_frame)
    paymentbutton.grid(row=3,column=1)

    view_complaints_button = ttk.Button(consumer_dashboard_frame, text="View Complaints", command=view_consumer_complaints)
    view_complaints_button.grid(row=4, column=0, pady=10)

    consumer_dashboard_frame.place(x=550,y=240)

def add_consumer(connectionid,name,phno,address,email,password):
    values=str(connectionid)+",'"+str(name)+"','"+str(phno)+"','"+str(address)+"','"+str(email)+"','"+str(password)+"'"
    insert="insert into consumerdata values("+str(values)+")"
    cursor.execute(insert)
    database.commit()
    print(name,"has been added.")

def verify_consumer_login():
    ConnectionID=idvar.get()
    password=passwvar.get()
    argument="select password from consumerdata where ConnectionID="+str(ConnectionID)
    password="('"+str(password)+"',)"
    cursor.execute(argument)
    for x in cursor:
        if password==str(x):
            consumer_dashboard(ConnectionID)
        else:
            messagebox.showerror("Error","Wrong ID or Password")

def verify_employee_login():
    EmployeeID=idvar.get()
    password=passwvar.get()
    argument="select password from employeedata where EmployeeID="+str(EmployeeID)
    password="('"+str(password)+"',)"
    cursor.execute(argument)
    for x in cursor:
        if password==str(x):
            employee_dashboard(EmployeeID)
        else:
            messagebox.showerror("Error","Wrong ID or Password")

def apply_new():
```

```
loginwindow.destroy()
newappln=ttk.Frame(main,relief='raised',border=20,width=600,height=350)
newapplnlabel=ttk.Label(newappln,text='Kindly fill out your details Here',font=("Georgia",20))
newapplnlabel2=ttk.Label(newappln,text='Our team will review your request and respond within the next 2-3 days.')
newapplnlabel.place(x=50,y=0)
newapplnlabel2.place(x=10,y=40)

def login_instead():
    newappln.destroy()
    login_window()
login_instead_button=ttk.Button(newappln,text="Login instead",command=login_instead)
login_instead_button.place(x=250,y=230)
back_button=ttk.Button(newappln,text="←",command=login_instead)
back_button.place(x=0,y=0)

Namevar=tkinter.StringVar()
Addressvar=tkinter.StringVar()
phnovar=tkinter.StringVar()
emailvar=tkinter.StringVar()

labelone=ttk.Label(newappln,text="Name :",font=('Georgia',15))
labelone.place(x=10,y=75)
entryone=ttk.Entry(newappln,textvariable=Namevar)
entryone.place(x=150,y=70)

labeltwo=ttk.Label(newappln,text="Address :",font=('Georgia',15))
labeltwo.place(x=10,y=115)
entrytwo=ttk.Entry(newappln,textvariable=Addressvar)
entrytwo.place(x=150,y=110)

labelthree=ttk.Label(newappln,text="Ph.Number :",font=('Georgia',15))
labelthree.place(x=10,y=155)
entrythree=ttk.Entry(newappln,textvariable=phnovar)
entrythree.place(x=150,y=150)

labelfour=ttk.Label(newappln,text="Email :",font=('Georgia',15))
labelfour.place(x=10,y=195)
entryfour=ttk.Entry(newappln,textvariable=emailvar)
entryfour.place(x=150,y=190)

def submit_appln():
    name,address,phno,emailu=Namevar.get(),Addressvar.get(),phnovar.get(),emailvar.get()
```

```
        cursor.execute("select email from employeedata where Role='New application Manager'")
        for i in cursor:
            email=i[0]
            content=f"Kindly review this new application \n Name - {name} \n Address - {address} \n Phone Number - {phno} \n Email - {emailu}"
            send_email(email,"New application",content)
            accepted_label=ttk.Label(newappln,text="Thanks, your application has been submitted, you will be sent an email when it is approved.")
            accepted_label.place(x=10,y=270)

    submitbutton=ttk.Button(newappln,text="Submit application",command=submit_appln)
    submitbutton.place(x=40,y=230)
    newappln.place(x=500,y=200)

def login_window():
    global idvar
    global loginwindow
    global passwvar
    loginwindow=ttk.Frame(main,relief='raised',border=20)
    loginlabel=ttk.Label(loginwindow,text="Welcome to the Login Page",font=("Georgia",20))
    loginnotebook=ttk.Notebook(loginwindow,width=350)
    tiplabelone=ttk.Label(loginwindow,text="Using LED bulbs can help you\nsave a lot of cost on energy")
    idvar=tkinter.StringVar()
    passwvar=tkinter.StringVar()
    consumerlogin=ttk.Frame(loginnotebook)
    useridtext=ttk.Label(master=consumerlogin,text="Enter Consumer ID-")
    enterid=ttk.Entry(consumerlogin,textvariable=idvar)
    passwordtext=ttk.Label(consumerlogin,text="Enter your Password-")
    enterpassword=ttk.Entry(consumerlogin,textvariable=passwvar,show="*")
    proceed=ttk.Button(consumerlogin,text="Proceed",command=verify_consumer_login)

    useridtext.grid(row=0,column=0)
    enterid.grid(row=0,column=1)
    passwordtext.grid(row=1,column=0)
    enterpassword.grid(row=1,column=1)
    proceed.grid(row=2,column=0,columnspan=2)

    employeelogin=ttk.Frame(loginnotebook)
    useridtext=ttk.Label(master=employeelogin,text="Enter Employee ID-")
    enterid=ttk.Entry(employeelogin,textvariable=idvar)
    passwordtext=ttk.Label(employeelogin,text="Enter your Password-")
    enterpassword=ttk.Entry(employeelogin,show="*",textvariable=passwvar)
    proceed=ttk.Button(employeelogin,text="Proceed",command=verify_employee_login)
    useridtext.grid(row=0,column=0)
```

```
    enterid.grid(row=0,column=1)
    passwordtext.grid(row=1,column=0)
    enterpassword.grid(row=1,column=1)
    proceed.grid(row=2,column=0,columnspan=2)

    loginnotebook.add(consumerlogin,text="Consumer")
    loginnotebook.add(employeelogin,text="Employee")

    loginlabel.grid(row=0,column=0,padx=10,pady=10)
    loginnotebook.grid(row=1,column=0,padx=10,pady=10)
    tiplabelone.grid(row=4,column=0)
    emptygaplabel=ttk.Label(loginwindow,text='')
    emptygaplabel.grid(row=3,column=0)
    newuserbutton=ttk.Button(loginwindow,text='Apply for a new connection',command=apply_new)
    newuserbutton.grid(row=2,column=0)

    loginwindow.place(x=560,y=240)

main=tkinter.Tk()
main.configure(height=1200,width=1200)
main.title("GCED Dashboard")
main.state('zoomed')
icon = ImageTk.PhotoImage(file = resource_path("D:\\Laptop Transfer\\sqlschlproject\\icon.jpg"))
main.iconphoto(False, icon)
bg=ImageTk.PhotoImage(Image.open(resource_path("D:\\Laptop Transfer\\sqlschlproject\\bgpic.jpg")),width=1200,height=1200)
bglabel=ttk.Label(main,image=bg)
bglabel.place(x=0,y=0)

sv_ttk.set_theme("dark")

login_window()

main.mainloop()
```