```
import smtplib
import os
import string
import random
import bcrypt
import logging
from sqlalchemy import update
from flask_login import LoginManager, login_user, login_required, current_user,
logout_user
from flask import Flask,url_for,render_template,redirect,request,session,jsonify
from datetime import datetime
from sqlalchemy import event
from database import
db,EmergencyContact,Employee,Item,CheckOut,CheckIn,Location,Category,Subcategory
,Unit,Currency,Department,Sales,Customer,PurchaseOrder,Vendor,UtilityCost,Budget
from email.message import EmailMessage
from flask_limiter import Limiter
from flask_limiter.util import get_remote_address
from flask_wtf.csrf import CSRFProtect,generate_csrf
from flask_caching import Cache

logging.basicConfig(
    filename='application.log',
    level=logging.ERROR,
    format='%(asctime)s - %(levelname)s - %(message)s'
)

app=Flask(__name__)

app.config["SQLALCHEMY_DATABASE_URI"] = "sqlite:///database.db"
app.config['SECRET_KEY']=os.getenv("SECRET_KEY")
app.config["SQLALCHEMY_TRACK_MODIFICATIONS"] = False

app.config['CACHE_TYPE'] = 'redis'
app.config['CACHE_REDIS_HOST'] = 'localhost'
app.config['CACHE_REDIS_PORT'] = 6379
app.config['CACHE_REDIS_DB'] = 0

# CRITICAL: Use SQLALCHEMY_ENGINE_OPTIONS to define a connection listener
app.config["SQLALCHEMY_ENGINE_OPTIONS"] = {
    # Recommended for SQLite with Flask's multi-threaded server
    "connect_args": {"check_same_thread": False}
}

csrf = CSRFProtect(app)

cache = Cache(app)

company_email=os.getenv("company_email")
company_email_password=os.getenv("company_email_password")

salt = bcrypt.gensalt()

limiter = Limiter(
    get_remote_address,
    app=app,
    storage_uri="memory://sqlite:///database.db",
)

db.init_app(app)
login_manager=LoginManager(app)
login_manager.login_view = "login"
characters = string.ascii_letters + string.digits + string.punctuation
```

```
with app.app_context():

    @event.listens_for(db.engine, "connect")
    def set_sqlite_pragma(dbapi_connection, connection_record):
        if app.config["SQLALCHEMY_DATABASE_URI"].startswith("sqlite"):
            # Execute the necessary SQL command to enable foreign key
enforcement
            cursor = dbapi_connection.cursor()
            cursor.execute("PRAGMA foreign_keys=ON")
            cursor.close()

    db.create_all()

    # add info for the first admin and remove
    # admin_emergencyContact=EmergencyContact(
    #                 firstname="Yabsera",
    #                 lastname="Yabsera",middlename="Yabsera",
    #                 phonenumber="92020201161",location_name="Addis Ababa",
    #                 fyida_id="321",gender="Male",
    #                 email="yabserabgl@gmail.com")

    # db.session.add(admin_emergencyContact)
    # db.session.commit()

    # password_to_send = ''.join(random.choice(characters) for i in range(15))
    # print(password_to_send)
    # password=(password_to_send).encode("utf-8")
    # admin=Employee(
    #                 emergency_contact_fyida_id="321",
    #
firstname="Yabsera",lastname="Yabsera",middlename="Yabsera",phonenumber="9202020
1161",
    #
gender="Male",email="yabserapython@gmail.com",date_of_employement=datetime.today
(),fyida_id="123",
    #                 employee_tin_number="123",currency_name="ETH",
    #                 position="IT",location_name="Addis Ababa",
    #
department_name="Administration",job_description="Administration",
    #                 bank_account_number="123456",salary="12333",
    #                 password=bcrypt.hashpw(password,salt))
    # db.session.add(admin)
    # db.session.commit()

    db_location=Location.location_array
    db_unit=Unit.unit_array
    db_currency=Currency.currency_array
    db_utility=UtilityCost.utility_type_array
    db_department=Department.department_array
    db_category=Category.category_array
    db_subcategory=Subcategory.subcategory_array
    db_vendor_name=db.session.query(Vendor.vendor_name).all()
    item_name_list=db.session.query(Item.item_name).all()

@login_manager.user_loader
def load_user(employee_tin_number):
    return db.session.get(Employee, employee_tin_number)

@app.before_request
def logout_if_not_active():
    if current_user.is_authenticated:
        employee=db.session.get(Employee, current_user.employee_tin_number)
        if not employee or employee.employment_status!="Active":
            logout_user()
```

```
            session.clear()
            return redirect(url_for('login'))

@app.route("/employee_registeration",methods=["GET","POST"])
@login_required
def employee_registeration():
    try:
        if session["department_name"]=="Human Resources" or
session["department_name"]=="Administration":
            if request.method=="POST":

emergency_contact_fyida_id=request.form["emergency_contact_fyida_id"]

emergency_contact_firstname=request.form["emergency_contact_firstname"]

emergency_contact_lastname=request.form["emergency_contact_lastname"]

emergency_contact_middlename=request.form["emergency_contact_middlename"]

emergency_contact_gender=request.form["emergency_contact_gender"]
                emergency_contact_phonenumber="+251
"+request.form["emergency_contact_phonenumber"]
                emergency_contact_email=request.form["emergency_contact_email"]

emergency_contact_location=request.form["emergency_contact_location"]

                emergency_contact=EmergencyContact(
                    firstname=emergency_contact_firstname,

lastname=emergency_contact_lastname,middlename=emergency_contact_middlename,

phonenumber=emergency_contact_phonenumber,location_name=emergency_contact_locati
on,

fyida_id=emergency_contact_fyida_id,gender=emergency_contact_gender,
                    email=emergency_contact_email)
                db.session.add(emergency_contact)
                db.session.commit()

                firstname=request.form["firstname"]
                lastname=request.form["lastname"]
                middlename=request.form["middlename"]
                gender=request.form["gender"]
                phonenumber="+251 "+request.form["phonenumber"]
                email=request.form["email"]
                date_of_employement=request.form["date_of_employement"]
                date_of_employement = datetime.strptime(date_of_employement,
"%Y-%m-%d").date()
                fyida_id=request.form["fyida_id"]
                position=request.form["position"]
                location=request.form["location"]
                department=request.form["department"]
                job_description=request.form["job_description"]
                tin_number=request.form["tin_number"]
                bank_account_number=request.form["bank_account_number"]
                currency=request.form["currency"]
                salary=request.form["salary"]
                password_to_send = ''.join(random.choice(characters) for i in
range(15))

                password=(password_to_send).encode("utf-8")
                employee=Employee(
                    emergency_contact_fyida_id=emergency_contact_fyida_id,
```

```
firstname=firstname,lastname=lastname,middlename=middlename,phonenumber=phonenum
ber,

gender=gender,email=email,date_of_employement=date_of_employement,fyida_id=fyida
_id,
                    employee_tin_number=tin_number,currency_name=currency,
                    position=position,location_name=location,
                    department_name=department,job_description=job_description,
                    bank_account_number=bank_account_number,salary=salary,
                    password=bcrypt.hashpw(password,salt))

                db.session.add(employee)
                db.session.commit()

employee=db.session.query(Employee).filter(Employee.email==email).first()
                subject="Well Come to Comapny Name"
                body=f"This sent by bot for Comapny Name password. Employee id:
{employee.employee_tin_number}  Your password: {password_to_send}"
                msg = EmailMessage()
                msg['subject']=subject
                msg['From']=company_email
                msg['To'] = email
                msg.set_content(body)
                try:

                    with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp: # For
Gmail, use SMTP_SSL and port 465
                        smtp.login(company_email, company_email_password)
                        smtp.send_message(msg)
                        return render_template("employee_registeration.html")

                except Exception as e:
                    db.session.rollback()
                    return render_template("404.html")
                return redirect("/dashboard")
            csrf_token = generate_csrf()
            return render_template("employee_registeration.html",
                                   csrf_token=csrf_token,
                                   db_location=db_location,
                                   db_unit=db_unit,
                                   db_currency=db_currency,
                                   db_department=db_department)
        return render_template("404.html")

    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/employee_termination",methods=["GET","POST"])
@login_required
def employee_termination():
    try:
        if session["department_name"]=="Human Resources" or
session["department_name"]=="Administration":
            if request.method=="POST":
                termination_date=request.form["termination_date"]
                termination_date=datetime.strptime(termination_date, "%Y-%m-
%d").date()
                termination_reason=request.form["termination_reason"]
                employment_status=request.form["employment_status"]
                employee_tin_number=request.form["employee_tin_number"]
                stmt=(
                    update(Employee)
```

```
                    .where(Employee.employee_tin_number==employee_tin_number)
                    .values(termination_date=termination_date,
                            termination_reason=termination_reason,
                            employment_status=employment_status)
                )
                db.session.execute(stmt)
                db.session.commit()
                cache.clear()
            csrf_token = generate_csrf()
            return render_template("employee_termination.html",
                                   csrf_token=csrf_token )
        else:
            return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/terminated_employee_list")
@login_required
@cache.cached(timeout=600)
def terminated_employee_list():
    try:
        if session["department_name"]=="Human Resources" or
session["department_name"]=="Administration":
            return render_template("terminated_employee_list.html",
                                   db_department=db_department)
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/terminated_employee_list/employee/data")
@login_required
@cache.cached(timeout=600)
def terminated_employee_list_data():
    try:
        if session["department_name"]=="Human Resources" or
session["department_name"]=="Administration":
            result=db.session.query(Employee.firstname,Employee.lastname,

Employee.phonenumber,Employee.department_name,
                                    Employee.position,Employee.salary,

Employee.termination_date,Employee.employee_tin_number
                                    ).where(
                                        Employee.employment_status!="Active"
                                    ).all()
            lst=[]
            for i in result:
                lst.append({
                    "firstname":i[0],
                    "lastname":i[1],
                    "phonenumber":i[2],
                    "department_name":i[3],
                    "position":i[4],
                    "salary":i[5],
                    "termination_date":i[6],
                    "employee_tin_number":i[7]
                })
            return jsonify(lst)
        return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
```

```
        db.session.rollback()
        return render_template("404.html")


@app.route("/terminated_employee_list/employee/data/<employee_tin_number>")
@login_required
@cache.cached(timeout=600)
def terminated_employee_data(employee_tin_number):
    try:
        if session["department_name"]=="Human Resources" or
session["department_name"]=="Administration":

employee=db.session.query(Employee).where(Employee.employee_tin_number==employee
_tin_number).first()

emergency_contact=db.session.query(EmergencyContact).where(EmergencyContact.fyid
a_id==employee.emergency_contact_fyida_id).first()
            employee={
                "emergency_contact":{
                    "name":emergency_contact.firstname + " "
+emergency_contact.lastname,
                    "fyida_id":emergency_contact.fyida_id,
                    "email":emergency_contact.email,
                    "gender":emergency_contact.gender,
                    "phonenumber":emergency_contact.phonenumber,

"location":str(emergency_contact.location).replace("<Location
","").replace(">","")
                },
                "employee_tin_number":employee.employee_tin_number,
                "location":str(employee.location).replace("<Location
","").replace(">",""),
                "department_name":employee.department_name,
                "salary":employee.salary,
                "name":employee.firstname + " " +employee.lastname,
                "fyida_id":employee.fyida_id,
                "date_of_employement":employee.date_of_employement,
                "bank_account_number":employee.bank_account_number,
                "gender":employee.gender,
                "email":employee.email,
                "job_description":employee.job_description,
                "position":employee.position,
                "termination_date":employee.termination_date,
                "termination_reason":employee.termination_reason,
                "employment_status":employee.employment_status,
            }
            return render_template("my_account.html",
                                   employee=employee,
                                   termaination=True)
        return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/restate/<employee_tin_number>",methods=["GET","POST"])
@login_required
def restate(employee_tin_number):
    try:
        if session["department_name"]=="Human Resources" or
session["department_name"]=="Administration":
            if request.method=="POST":
                employment_status=request.form["employment_status"]
                stmt=(
```

```
                    update(
                        Employee
                    ).where(
                        Employee.employee_tin_number==employee_tin_number
                    ).values(
                        employment_status=employment_status
                    )
                )
                db.session.execute(stmt)
                db.session.commit()
                cache.clear()
                return redirect("/dashboard")
        return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/all/employee/data")
@login_required
@cache.cached(timeout=600)
def employee_data():
    try:
        if session["department_name"]=="Human Resources" or
session["department_name"]=="Administration":
            result=db.session.query(Employee.firstname,Employee.lastname,

Employee.phonenumber,Employee.department_name,
                                    Employee.position,Employee.salary,

Employee.bank_account_number,Employee.employee_tin_number
                                    ).where(
                                        Employee.employment_status=="Active"
                                    ).all()
            lst=[]
            for i in result:
                lst.append({
                    "firstname":i[0],
                    "lastname":i[1],
                    "phonenumber":i[2],
                    "department_name":i[3],
                    "position":i[4],
                    "salary":i[5],
                    "bank_account_number":i[6],
                    "employee_tin_number":i[7]
                })
            return jsonify(lst)
        return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/employee_info_for_hr/<employee_tin_number>")
@login_required
@cache.cached(timeout=600)
def employee_info_for_hr(employee_tin_number):
    try:
        if session["department_name"]=="Human Resources" or
session["department_name"]=="Administration":

employee=db.session.query(Employee).where(Employee.employee_tin_number==employee
_tin_number).first()
```

```
emergency_contact=db.session.query(EmergencyContact).where(EmergencyContact.fyid
a_id==employee.emergency_contact_fyida_id).first()
            employee={
                "emergency_contact":{
                    "name":emergency_contact.firstname + " "
+emergency_contact.lastname,
                    "fyida_id":emergency_contact.fyida_id,
                    "email":emergency_contact.email,
                    "gender":emergency_contact.gender,
                    "phonenumber":emergency_contact.phonenumber,

"location":str(emergency_contact.location).replace("<Location
","").replace(">","")
                },
                "employee_tin_number":employee.employee_tin_number,
                "location":str(employee.location).replace("<Location
","").replace(">",""),
                "department_name":employee.department_name,
                "salary":employee.salary,
                "name":employee.firstname + " " +employee.lastname,
                "fyida_id":employee.fyida_id,
                "date_of_employement":employee.date_of_employement,
                "bank_account_number":employee.bank_account_number,
                "gender":employee.gender,
                "email":employee.email,
                "job_description":employee.job_description,
                "position":employee.position
            }
            return render_template("my_account.html",
                                   employee=employee)
        return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/my_account")
@login_required
def account():
    try:

employee=db.session.query(Employee).where(Employee.employee_tin_number==session[
"employee_tin_number"]).first()

emergency_contact=db.session.query(EmergencyContact).where(EmergencyContact.fyid
a_id==employee.emergency_contact_fyida_id).first()
        employee={
            "emergency_contact":{
                "name":emergency_contact.firstname + " "
+emergency_contact.lastname,
                "fyida_id":emergency_contact.fyida_id,
                "email":emergency_contact.email,
                "gender":emergency_contact.gender,
                "phonenumber":emergency_contact.phonenumber,
                "location":str(emergency_contact.location).replace("<Location
","").replace(">","")
            },
            "employee_tin_number":employee.employee_tin_number,
            "location":str(employee.location).replace("<Location
","").replace(">",""),
            "department_name":employee.department_name,
            "salary":employee.salary,
            "name":employee.firstname + " " +employee.lastname,
            "fyida_id":employee.fyida_id,
```

```
            "date_of_employement":employee.date_of_employement,
            "bank_account_number":employee.bank_account_number,
            "gender":employee.gender,
            "email":employee.email,
            "job_description":employee.job_description,
            "position":employee.position
        }

        return render_template("my_account.html",
                               employee=employee)
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/employee_list")
@login_required
@cache.cached(timeout=600)
def employee_list():
    try:
        if session["department_name"]=="Human Resources" or
session["department_name"]=="Administration":
            return render_template("employee_list.html",
                                   db_department=db_department)
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/item_regsisteration",methods=["GET","POST"])
@login_required
def item_registeration():
    try:
        if session["department_name"]=="Store" or
session["department_name"]=="Administration":
            if request.method=="POST":
                item_name=request.form["item_name"]
                item_price=request.form["item_price"]
                unit=request.form["unit"]
                location_name=request.form["location"]
                item_category=request.form["item_category"]
                item_subcategory=request.form["item_subcategory"]
                item_quantity=request.form["item_quantity"]
                item_description=request.form["item_description"]
                item_name=request.form["item_name"]
                item_shelf_life=request.form["item_shelf_life"]
                currency=request.form["currency"]
                item_shelf_life = datetime.strptime(item_shelf_life, "%Y-%m-
%d").date()
                item=Item(
                    item_name=item_name,item_price=item_price,
                    currency_name=currency,item_quantity=item_quantity,
                    unit_name=unit,category_name=item_category,

location_name=location_name,subcategory_name=item_subcategory,

created_by_employee_tin_number=session["employee_tin_number"],
                    item_description=item_description,
                    item_shelf_life=item_shelf_life)

                db.session.add(item)
                db.session.commit()
                cache.clear()
                return redirect("/dashboard")
```

```
            csrf_token = generate_csrf()
            return render_template("item_registeration.html",
                                   csrf_token=csrf_token,
                                   db_location=db_location,
                                   db_unit=db_unit,
                                   db_currency=db_currency,
                                   db_category=db_category,
                                   db_subcategory=db_subcategory)
        else:
            return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/item_info/<item_id>")
@login_required
def item_info(item_id):
    try:
        item=db.session.query(Item).where(Item.item_id==item_id).first()
        return render_template("item_info.html",item=item[0])
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/item_listing")
@login_required
@cache.cached(timeout=600)
def item_listing():
    try:
        if session["department_name"]=="Store" or
session["department_name"]=="Administration":
            item_list=db.session.query(
                Item.item_name,Item.item_price,Item.item_quantity,
                Item.unit_name,Item.item_shelf_life,Item.item_id
            ).order_by(Item.item_shelf_life.asc()).all()
            return render_template("item_list.html",
                                   item_list=item_list)
        return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/checkout_info/<checkout_id>")
def checkout_info(checkout_id):
    try:

item=db.session.query(CheckOut).where(CheckOut.checkout_id==checkout_id).first()
        return render_template("checkout_info.html",item=item[0])
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/checkin_info/<checkin_id>")
def checkin_info(checkin_id):
    try:

item=db.session.query(CheckIn).where(CheckIn.checkin_id==checkin_id).first()
        return render_template("checkin_info.html",item=item[0])
    except Exception as e:
        logging.exception(str(e))
```

```
        db.session.rollback()
        return render_template("404.html")

@app.route("/checkout_list")
@login_required
@cache.cached(timeout=600)
def checkout_list():
    try:
        if session["department_name"]=="Store":
            checkout_list_name=db.session.query(
                CheckOut.item_name,CheckOut.item_status,CheckOut.item_quantity,
                CheckOut.item_siv,CheckOut.unit_name,CheckOut.checkout_id
                ).order_by(CheckOut.checkout_date.asc()).where(

CheckOut.employee_tin_number==session["employee_tin_number"]).all()
            return render_template("checkout_list.html",
                                   checkout_list_name=checkout_list_name)

        elif session["department_name"]=="Administration":
            checkout_list_name=db.session.query(
                CheckOut.item_name,CheckOut.item_status,CheckOut.item_quantity,
                CheckOut.item_siv,CheckOut.unit_name,CheckOut.checkout_id
                ).order_by(CheckOut.checkout_date.asc()).all()
            return render_template("checkout_list.html",
                                   checkout_list_name=checkout_list_name)

        return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/item_checkout",methods=["GET","POST"])
@login_required
def item_checkout():
    try:
        if session["department_name"]=="Store" or
session["department_name"]=="Administration":
            if request.method=="POST":
                item_name=request.form["item_name"]
                return_employee_id=request.form["return_employee_id"]
                checkout_date=request.form["checkout_date"]
                item_quantity=request.form["item_quantity"]
                item_siv=request.form["item_siv"]
                item_status=request.form["item_status"]
                department=request.form["department"]
                location_name=request.form["location"]
                item_description=request.form["item_description"]
                unit_name=request.form["unit"]
                checkout_date = datetime.strptime(checkout_date, "%Y-%m-
%d").date()


item=db.session.query(Item).filter(Item.item_name==item_name).first()
                budget=db.session.query(Budget).where(
                    Budget.item_name==item_name
                    and
                    Budget.department==session["department_name"]
                ).first()

                if item.item_quantity-float(item_quantity)<0:
                    return render_template("checkout.html",negative=True)

                if budget.item_quantity_deduct<=0:
```

```
                    return render_template("checkout.html",negative=True)

                stmt=(
                    update(
                        Item
                    ).where(Item.item_name==item_name)
                    .values(item_quantity=Item.item_quantity-
float(item_quantity))
                )

                db.session.execute(stmt)
                db.session.commit()
                cache.clear()
                checkout_item=CheckOut(

item_name=item_name,return_employee_id=return_employee_id,checkout_date=checkout
_date,

item_quantity=item_quantity,item_siv=item_siv,department=department,

location_name=location_name,item_description=item_description,item_status=item_s
tatus,

unit_name=unit_name,employee_tin_number=session["employee_tin_number"])
                db.session.add(checkout_item)
                db.session.commit()
                cache.clear()
            csrf_token = generate_csrf()
            return render_template("checkout.html",
                                   csrf_token=csrf_token,
                                   item_name_list=item_name_list,
                                   db_location=db_location,
                                   db_unit=db_unit,
                                   db_department=db_department)
        else:
            return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/checkin_list")
@login_required
@cache.cached(timeout=600)
def checkin_list():
    try:
        if session["department_name"]=="Store":
            checkin_list_name=db.session.query(
                CheckIn.item_name,CheckIn.item_price,CheckIn.item_quantity,
                CheckIn.item_grr,CheckIn.unit_name
                ).where(

CheckIn.employee_tin_number==session["employee_tin_number"]).all()
            return render_template("checkin_list.html",
                                   checkin_list_name=checkin_list_name)

        elif session["department_name"]=="Administration":
            checkin_list_name=db.session.query(
            CheckIn.item_name,CheckIn.item_price,CheckIn.item_quantity,
            CheckIn.item_grr,CheckIn.unit_name,CheckIn.checkin_id
            ).order_by(CheckIn.checkin_date.asc()).all()
            return render_template("checkin_list.html",
                                   checkin_list_name=checkin_list_name)
        return render_template("404.html")
```

```
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/item_checkin",methods=["GET","POST"])
@login_required
def item_checkin():
    try:
        if session["department_name"]=="Store" or
session["department_name"]=="Administration":
            item_name_list=db.session.query(Item.item_name).all()
            if request.method=="POST":
                item_name=request.form["item_name"]
                reciving_employee_id=request.form["reciving_employee_id"]
                vendor_name=request.form["vendor_name"]
                checkin_date=request.form["checkin_date"]
                checkin_date = datetime.strptime(checkin_date, "%Y-%m-
%d").date()
                item_price=request.form["item_price"]
                item_quantity=request.form["item_quantity"]
                item_grr=request.form["item_grr"]
                item_description=request.form["item_description"]
                unit=request.form["unit"]
                item_status=request.form["item_status"]
                currency=request.form["currency"]
                item_shelf_life=request.form["item_shelf_life"]
                item_shelf_life=datetime.strptime(item_shelf_life, "%Y-%m-
%d").date()

item=db.session.query(Item).where(Item.item_name==item_name).first()

                if item.item_quantity==0:
                    stmt=(
                        update(Item)
                        .where(Item.item_name==item_name)
                        .values(item_quantity=float(item_quantity),
                                item_price=float(item_quantity)
                        )
                    )

                    db.session.execute(stmt)
                    db.session.commit()
                    cache.clear()
                else:
                    stmt=(
                        update(Item)
                        .where(Item.item_name==item_name)
                        .values(item_quantity=Item.item_quantity+float(item_quan
tity),

item_price=(Item.item_price+float(item_quantity))/2
                        )
                    )

                    db.session.execute(stmt)
                    db.session.commit()
                    cache.clear()

                checkin_item=CheckIn(

item_name=item_name,reciving_employee_id=reciving_employee_id,
```

```
employee_tin_number=session["employee_tin_number"],item_price=item_price,
                        item_quantity=item_quantity,item_grr=item_grr,
                        item_description=item_description,unit_name=unit,
                        checkin_date=checkin_date,currency_name=currency,

item_shelf_life=item_shelf_life,item_status=item_status,vendor_name=vendor_name)

                db.session.add(checkin_item)
                db.session.commit()
                cache.clear()
                return render_template("checkin.html",
                                       item_name_list=item_name_list,
                                       db_currency=db_currency,
                                       db_unit=db_unit,
                                       db_vendor_name=db_vendor_name)
            csrf_token = generate_csrf()
            return render_template("checkin.html",
                                   csrf_token=csrf_token,
                                   item_name_list=item_name_list,
                                   db_currency=db_currency,
                                   db_unit=db_unit,
                                   db_vendor_name=db_vendor_name)
        else:
            return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/customer_list")
@login_required
@cache.cached(timeout=600)
def customer_list():
    try:
        if session["department_name"]=="Sales" or
session["department_name"]=="Administration":
            return render_template("customer_list.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/customer_list/list/<employee_tin_number>")
@login_required
@cache.cached(timeout=600)
def customer_list_employee_tin_number(employee_tin_number):
    try:
        if session["department_name"]=="Sales" or
session["department_name"]=="Administration":
            customer_list_name=db.session.query(Customer).where(

Customer.regsistered_employee_tin_number==employee_tin_number
                            ).all()

            lst=[]

            for i in customer_list_name:
                lst.append(
                    {
                        "customer_tin":i[0],
                        "customer_name":i[1],
                        "customer_email":i[2],
                        "customer_phonenumber":i[3],
                        "customer_location":i[4]
```

```
                    }
                )
            return jsonify(lst)
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/sales_list")
@login_required
@cache.cached(timeout=600)
def sales_list():
    try:
        if session["department_name"]=="Sales" or
session["department_name"]=="Administration":
            return render_template("sales_list.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/sales_list/list/<employee_tin_number>")
@login_required
@cache.cached(timeout=600)
def sales_list_employee_tin_number(employee_tin_number):
    try:
        if session["department_name"]=="Sales" or
session["department_name"]=="Administration":
            sales_list_name=db.session.query(Sales).where(
                                Sales.employee_tin_number==employee_tin_number
                            ).all()

            lst=[]

            for i in sales_list_name:
                lst.append(
                    {
                        "customer_tin":i[6],
                        "sales_date":i[1],
                        "item_name":i[2],
                        "item_quantity":i[3],
                        "total_price":i[4],
                        "unit_name":i[8],
                    }
                )
            return jsonify(lst)
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/sales_registeration",methods=["GET","POST"])
@login_required
def sales_registeration():
    try:
        if session["department_name"]=="Sales" or
session["department_name"]=="Administration":
            if request.method=="POST":
                item_name=request.form["item_name"]
                item_quantity=request.form["item_quantity"]
                item_price=request.form["item_price"]
                customer_tin=request.form["customer_tin"]
                unit_name=request.form["unit_name"]
                currency_name=request.form["currency_name"]
```

```
item=db.session.query(Item).filter(Item.item_name==item_name).first()

                if item.item_quantity-int(item_quantity)<0:
                    csrf_token = generate_csrf()
                    return render_template("sales_registeration.html",
                                           csrf_token=csrf_token,
                                           negative=True,
                                           db_currency=db_currency,
                                           db_unit=db_unit)

                stmt=(
                    update(
                        Item
                    ).where(Item.item_name==item_name)
                    .values(item_quantity=Item.item_quantity-
float(item_quantity))
                )

                db.session.execute(stmt)
                db.session.commit()
                cache.clear()
                sales=Sales(

item_name=item_name,item_quantity=item_quantity,item_price=item_price,

customer_tin=customer_tin,unit_name=unit_name,currency_name=currency_name,
                    employee_tin_number=session["employee_tin_number"]
                )

                db.session.add(sales)
                db.session.commit()
                cache.clear()
            csrf_token = generate_csrf()
            return render_template("sales_registeration.html",
                                   csrf_token=csrf_token,
                                   item_name_list=item_name_list,
                                   db_currency=db_currency,
                                   db_unit=db_unit)
        else:
            return render_template("404.html")

    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/purchase_order",methods=["GET","POST"])
@login_required
def purchase_order():
    try:
        if request.method=="POST":
            item_name=request.form["item_name"]
            ordered_quantity=request.form["ordered_quantity"]
            purchase_reason=request.form["purchase_reason"]
            order_status="Pending"
            purchase=PurchaseOrder(
                item_name=item_name,ordered_quantity=ordered_quantity,
                purchase_reason=purchase_reason,order_status=order_status,
                employee_tin_number=session["employee_tin_number"]
            )
            db.session.add(purchase)
            db.session.commit()
```

```
            cache.clear()
        csrf_token=generate_csrf()
        return render_template("purchase_order.html",
                               csrf_token=csrf_token,
                               item_name_list=item_name_list)

    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/customer_registeration",methods=["GET","POST"])
@login_required
def customer_registeration():
    try:
        if session["department_name"]=="Sales" or
session["department_name"]=="Administration":
            if request.method=="POST":
                customer_name=request.form["customer_name"]
                customer_email=request.form["customer_email"]
                customer_phonenumber=request.form["customer_phonenumber"]
                customer_tin=request.form["customer_tin"]
                customer_location=request.form["customer_location"]

                customer=Customer(

customer_tin=customer_tin,customer_phonenumber=customer_phonenumber,

customer_name=customer_name,customer_location=customer_location,

customer_email=customer_email,regsistered_employee_tin_number=session["employee_
tin_number"]
                )

                db.session.add(customer)
                db.session.commit()
                cache.clear()
            csrf_token=generate_csrf()
            return render_template("customer_registeration.html",
                                   csrf_token=csrf_token,
                                   db_location=db_location)
        else:
            return render_template("404.html")

    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/pending_listing")
@login_required
@cache.cached(timeout=600)
def pending_listing():
    try:
        if session["department_name"]=="Administration":
            pending_list_name=db.session.query(

PurchaseOrder.employee_tin_number,PurchaseOrder.item_name,

PurchaseOrder.order_status,PurchaseOrder.order_date,PurchaseOrder.purchase_order
_id
                               ).where(
                                   PurchaseOrder.order_status== "Pending"
                               ).order_by(PurchaseOrder.order_date.asc()).all()
```

```
            return render_template("pending_list_all.html",
                                   pending_list_name=pending_list_name)

        elif session["department_name"]=="Procurement":
            pending_list_name=db.session.query(

PurchaseOrder.employee_tin_number,PurchaseOrder.item_name,

PurchaseOrder.order_status,PurchaseOrder.order_date,PurchaseOrder.purchase_order
_id
                               ).where(

PurchaseOrder.employee_tin_number==session["employee_tin_number"]
                                  and PurchaseOrder.order_status== "Pending"
                               ).order_by(PurchaseOrder.order_date.asc()).all()
            return render_template("pending_list.html",
                                   pending_list_name=pending_list_name)
        return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/purchase_order_approval/
<purchase_order_id>",methods=["GET","POST"])
@login_required
def purchase_order_approval(purchase_order_id):
    try:

purchase_order_data=db.session.query(PurchaseOrder).where(PurchaseOrder.order_st
atus==purchase_order_id).first()
        if request.method=="POST" and
session["department_name"]=="Administration":
            order_status=request.form["order_status"]
            stmt=(
                update(
                    PurchaseOrder
                ).where(
                    PurchaseOrder.purchase_order_id==purchase_order_id
                ).values(
                    order_status=order_status
                )
            )
            db.session.execute(stmt)
            db.session.commit()
            cache.clear()
        return render_template("purchase_order_info.html",
                               purchase_order_data=purchase_order_data)
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/approved_listing")
@login_required
@cache.cached(timeout=600)
def approved_listing():
    try:
        if session["department_name"]=="Administration":
            approved_list_name=db.session.query(

PurchaseOrder.employee_tin_number,PurchaseOrder.item_name,

PurchaseOrder.order_status,PurchaseOrder.order_date,PurchaseOrder.purchase_order
```

```
_id
                            ).where(
                                PurchaseOrder.order_status== "Approved"
                            ).order_by(PurchaseOrder.order_date.asc()).all()
            return render_template("approved_list.html",
                                   approved_list_name=approved_list_name)
        elif session["department_name"]=="Procurement":
            approved_list_name=db.session.query(

PurchaseOrder.employee_tin_number,PurchaseOrder.item_name,

PurchaseOrder.order_status,PurchaseOrder.order_date,PurchaseOrder.purchase_order
_id
                            ).where(

PurchaseOrder.employee_tin_number==session["employee_tin_number"]
                                and PurchaseOrder.order_status== "Approved"
                            ).order_by(PurchaseOrder.order_date.asc()).all()
            return render_template("approved_list.html",
                                   approved_list_name=approved_list_name)
        return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/rejected_listing")
@login_required
@cache.cached(timeout=600)
def rejected_listing():
    try:
        if session["department_name"]=="Administration":
            rejected_list_name=db.session.query(

PurchaseOrder.employee_tin_number,PurchaseOrder.item_name,

PurchaseOrder.order_status,PurchaseOrder.order_date,PurchaseOrder.purchase_order
_id
                                    ).where(

PurchaseOrder.employee_tin_number==session["employee_tin_number"]
                                        and PurchaseOrder.order_status==
"Decline"
                                    ).order_by(PurchaseOrder.order_date.asc()).a
ll()
            return render_template("rejected_list.html",
                                   rejected_list_name=rejected_list_name)
        elif session["department_name"]=="Procurement":
            rejected_list_name=db.session.query(

PurchaseOrder.employee_tin_number,PurchaseOrder.item_name,

PurchaseOrder.order_status,PurchaseOrder.order_date,PurchaseOrder.purchase_order
_id
                            ).where(

PurchaseOrder.employee_tin_number==session["employee_tin_number"]
                                and PurchaseOrder.order_status== "Decline"
                            ).order_by(PurchaseOrder.order_date.asc()).all()
            return render_template("rejected_list.html",
                                   rejected_list_name=rejected_list_name)
        return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
```

```
        db.session.rollback()
        return render_template("404.html")

@app.route("/vendor_info/<vendor_tin>",methods=["GET","POST"])
@login_required
def vendor_info(vendor_tin):
    try:
        if session["department_name"]=="Procurement":
            vendor=db.session.query(Vendor).where(
                Vendor.vendor_tin==vendor_tin,

Vendor.regsistered_employee_tin_number==session["employee_tin_number"]
            ).first()

            if request.method=="POST":
                vendor_tin_new=request.form["vendor_tin"]
                vendor_email=request.form["vendor_email"]
                vendor_phonenumber=request.form["vendor_phonenumber"]
                location=request.form["location"]
                vendor_name=request.form["vendor_name"]

                stmt=(
                    update(
                        Vendor
                    ).where(
                        Vendor.vendor_tin==vendor_tin
                    ).values(
                        vendor_tin=vendor_tin_new,
                        vendor_email=vendor_email,
                        vendor_phonenumber=vendor_phonenumber,
                        location=location,
                        vendor_name=vendor_name
                    )
                )

                db.session.execute(stmt)
                db.session.commit()
                cache.clear()
                return render_template("vendor_info.html",
                                       vendor=vendor)
            return render_template("vendor_info.html",
                                   vendor=vendor)

        return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/utility_registeration",methods=["GET","POST"])
@login_required
def utility_registeration():
    try:
        if session["department_name"]=="Finance" or
session["department_name"]=="Administration":
            if request.method=="POST":
                utility_type=request.form["utility_type"]
                total_cost=request.form["total_cost"]
                utility_name=request.form["utility_name"]
                department_name=request.form["department"]
                currency_name=request.form["currency"]
                location_name=request.form["location"]

                utility=UtilityCost(
```

```
                    utility_type=utility_type,
                    utility_name=utility_name,
                    total_cost=total_cost,
                    department_name=department_name,
                    currency_name=currency_name,
                    location_name=location_name,

recorded_by_employee_tin_number=session["employee_tin_number"]
                )
                db.session.add(utility)
                db.session.commit()
                cache.clear()
            csrf_token=generate_csrf()
            return render_template("utility_registeration.html",
                                   csrf_token=csrf_token,
                                   db_utility=db_utility)
        return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/utility_listing")
@login_required
@cache.cached(timeout=600)
def utility_listing():
    try:
        if session["department_name"]=="Finance" or
session["department_name"]=="Administration":
            utility_list_name=db.session.query(
                UtilityCost.utility_cost_id,
                UtilityCost.utility_name,
                UtilityCost.utility_type,
                UtilityCost.total_cost,
                UtilityCost.location_name,
                UtilityCost.department_name,
                UtilityCost.currency_name
                ).order_by(UtilityCost.utility_name.asc()).all()
            return render_template("utility_list.html",
                                   utility_list_name=utility_list_name)
        return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/vendor_listing")
@login_required
@cache.cached(timeout=600)
def vendor_listing():
    try:
        if session["department_name"]=="Procurement":
            vendor=db.session.query(
                Vendor.vendor_tin,Vendor.vendor_name,
                Vendor.vendor_phonenumber,Vendor.vendor_email,Vendor.location
                ).where(Vendor.regsistered_employee_tin_number==session["employe
e_tin_number"]).all()
            return render_template("vendor_list.html",vendor_lst=vendor)
        elif session["department_name"]=="Administration":
            vendor=db.session.query(
                Vendor.vendor_tin,Vendor.vendor_name,
                Vendor.vendor_phonenumber,Vendor.vendor_email,Vendor.location
                ).all()
            return render_template("vendor_list.html",
```

```
                                    vendor_lst=vendor)
        return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/vendor_regsisteration",methods=["GET","POST"])
@login_required
def vendor_regsisteration():
    try:
        if session["department_name"]=="Procurement" or
session["department_name"]=="Administration":
            if request.method=="POST":
                vendor_name=request.form["vendor_name"]
                vendor_tin=request.form["vendor_tin"]
                vendor_phonenumber=request.form["vendor_phonenumber"]
                vendor_email=request.form["vendor_email"]
                location=request.form["location"]

                vendor=Vendor(
                    vendor_name=vendor_name,vendor_tin=vendor_tin,

vendor_phonenumber=vendor_phonenumber,vendor_email=vendor_email,

location=location,regsistered_employee_tin_number=session["employee_tin_number"]
                )

                db.session.add(vendor)
                db.session.commit()
                cache.clear()
            csrf_token=generate_csrf()
            return render_template("vendor_regsisteration.html",
                                   csrf_token=csrf_token,
                                   db_location=db_location)
        else:
            return render_template("404.html")

    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/budget_registeration",methods=["GET","POST"])
@login_required
def budget_registeration():

    try:
        if session["department_name"]=="Finance":
            csrf_token=generate_csrf()
            if request.method=="POST":
                item_name=request.form["item_name"]
                department=request.form["department"]
                item_quantity=request.form["item_quantity"]
                item_budget=request.form["item_budget"]
                currency=request.form["currency"]
                unit=request.form["unit"]
                date_from=datetime.strptime(request.form["date_from"], "%Y-%m-
%d").date()
                date_to=datetime.strptime(request.form["date_to"], "%Y-%m-
%d").date()

                budget=Budget(
                    item_name=item_name,
```

```
                    department_name=department,
                    item_quantity=item_quantity,
                    item_budget=item_budget,
                    unit_name=unit,
                    currency_name=currency,
                    date_from=date_from,
                    date_to=date_to,

recorded_by_employee_tin_number=session["employee_tin_number"]
                )
                db.session.add(budget)
                db.session.commit()
                cache.clear()
                return
render_template("budget_registeration.html",csrf_token=csrf_token)
            return
render_template("budget_registeration.html",csrf_token=csrf_token)
        else:
            return render_template("404.html")

    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/budget_list")
@login_required
@cache.cached(timeout=600)
def budget_list():
    try:
        if session["department_name"]=="Finance":
            budget_list_name=db.session.query(
                Budget.item_name,
                Budget.department_name,
                Budget.item_budget,
                Budget.item_budget_deduct,
                Budget.item_quantity,
                Budget.item_quantity_deduct,
                Budget.currency_name,
                Budget.unit_name,
                Budget.date_from,
                Budget.date_to
                ).order_by(Budget.recorded_at.desc()).all()

            return
render_template("budget_list.html",budget_list_name=budget_list_name)
        else:
            return render_template("404.html")

    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/finanical_data",methods=["GET","POST"])
@login_required
def finanical_data():
    try:
        if session["department_name"]=="Finance" or
session["department_name"]=="Administration":
            csrf_token=generate_csrf()
            if request.method=="POST":
                department=request.form["department"]
                if department=="All":
```

```
                    employee=db.session.query(
                        Employee.employee_tin_number,
                        Employee.firstname,
                        Employee.lastname,
                        Employee.salary,
                        Employee.department_name
                    ).where(Employee.employment_status=="Active").all()
                else:
                    employee=db.session.query(
                        Employee.employee_tin_number,
                        Employee.firstname,
                        Employee.lastname,
                        Employee.salary,
                        Employee.department_name
                    ).where(Employee.department_name==department and
Employee.employment_status=="Active").all()
                employee_dict=[]
                total_pension=0
                total_income_tax=0
                total_net_salary=0
                total_gross_salary=0
                for i in employee:
                    salary=i[3]
                    firstname=i[1]
                    fathername=i[2]
                    pension=i[3]*0.07
                    total_pension+=pension
                    income_tax=0
                    net_salary=0

                    if salary<=4000 and salary>=2001:
                        income_tax=(0.15*salary)-300
                        net_salary=salary-(income_tax+pension)
                        total_income_tax+=income_tax
                        total_net_salary+=net_salary
                        total_gross_salary+=salary
                    elif salary<=7000 and salary>=4001:
                        income_tax=(0.2*salary)-500
                        net_salary=salary-(income_tax+pension)
                        total_income_tax+=income_tax
                        total_net_salary+=net_salary
                        total_gross_salary+=salary
                    elif salary<=10000 and salary>=7001:
                        income_tax=(0.25*salary)-850
                        net_salary=salary-(income_tax+pension)
                        total_income_tax+=income_tax
                        total_net_salary+=net_salary
                        total_gross_salary+=salary
                    elif salary<=14000 and salary>=10001:
                        income_tax=(0.3*salary)-1350
                        net_salary=salary-(income_tax+pension)
                        total_income_tax+=income_tax
                        total_net_salary+=net_salary
                        total_gross_salary+=salary
                    elif salary>=14001:
                        income_tax=(0.35*salary)-2050
                        net_salary=salary-(income_tax+pension)
                        total_income_tax+=income_tax
                        total_net_salary+=net_salary
                        total_gross_salary+=salary

                    employee_dict.append({
                        "employee_tin_number":i[0],
                        "name":firstname+" "+fathername,
```

```
                        "gross_salary":salary,
                        "pension":"{:.2f}".format(pension),
                        "net_salary":"{:.2f}".format(net_salary),
                        "income_tax":"{:.2f}".format(income_tax),
                        "department":i[4]
                    })
                total_employee_info={
                    "total_pension":"{:.2f}".format(total_pension),
                    "total_income_tax":"{:.2f}".format(total_income_tax),
                    "total_net_salary":"{:.2f}".format(total_net_salary),
                    "total_gross_salary":"{:.2f}".format(total_gross_salary),
                    "number_of_employee":len(employee)
                }
                return render_template("finanical_data.html",
                                       employee_dict=employee_dict,
                                       db_department=db_department,
                                       csrf_token=csrf_token,
                                       total_employee_info=total_employee_info
                                       )
            return render_template("finanical_data.html",
                                       employee_dict=[],
                                       db_department=db_department,
                                       csrf_token=csrf_token,
                                       total_employee_info={}
                                       )
        else:
            return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/finance")
@login_required
def finance():
    try:
        if session["department_name"]=="Finance":
            return render_template("finance.html")
        return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/store")
@login_required
def store():
    try:
        if session["department_name"]=="Store":
            return render_template("store.html")
        return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/sales")
@login_required
def sales():
    try:
        if session["department_name"]=="Sales":
            return render_template("sales.html")
        return render_template("404.html")
    except Exception as e:
```

```
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/procurement")
@login_required
def procurement():
    try:
        if session["department_name"]=="Procurement":
            return render_template("procurement.html")
        return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/human_resources")
@login_required
def human_resources():
    try:
        if session["department_name"]=="Human Resources":
            return render_template("human_resources.html")
        return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/administration")
@login_required
def administration():
    try:
        if session["department_name"]=="Administration":
            return render_template("administration.html")
        return render_template("404.html")
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/login",methods=["GET","POST"])
@limiter.limit("5 per minute")
def login():
    try:
        if "logged_in" in session and session["logged_in"]==True:
            if session["department_name"]=="Finance":
                return redirect("/finance")
            elif session["department_name"]=="Store":
                return redirect("/store")
            elif session["department_name"]=="Sales":
                return redirect("/sales")
            elif session["department_name"]=="Procurement":
                return redirect("/procurement")
            elif session["department_name"]=="Human Resources":
                return redirect("/human_resources")
            elif session["department_name"]=="Administration":
                return redirect("/administration")
        elif request.method=="POST":
            employee_tin_number=request.form["employee_id"]
            password=request.form["password"]

employee=db.session.query(Employee).filter(Employee.employee_tin_number==employe
e_tin_number).first()
            if employee is None:
```

```
                return redirect("/login")
            is_vaild=bcrypt.checkpw(password.encode("utf-8"),employee.password)
            if is_vaild==True and employee.employment_status=="Active":
                login_user(employee)
                session["employee_tin_number"]=employee.employee_tin_number
                session["logged_in"]=True
                session["department_name"]=employee.department_name
                if session["department_name"]=="Finance":
                    return redirect("/finance")
                elif session["department_name"]=="Store":
                    return redirect("/store")
                elif session["department_name"]=="Sales":
                    return redirect("/sales")
                elif session["department_name"]=="Procurement":
                    return redirect("/procurement")
                elif session["department_name"]=="Human Resources":
                    return redirect("/human_resources")
                elif session["department_name"]=="Administration":
                    return redirect("/administration")
            return redirect("/login")
        csrf_token = generate_csrf()
        return render_template("login.html",csrf_token=csrf_token)
    except Exception as e:
        logging.exception(str(e))
        db.session.rollback()
        return render_template("404.html")

@app.route("/logout")
def logout():
    try:
        cache.clear()
        logout_user()
        session.clear()
        return redirect("/login")
    except Exception as e:
        logging.exception(str(e))
        return render_template("404.html")

@app.route("/")
def index():
    return redirect("/login")

if __name__=="__main__":

    app.run(debug=True)
```