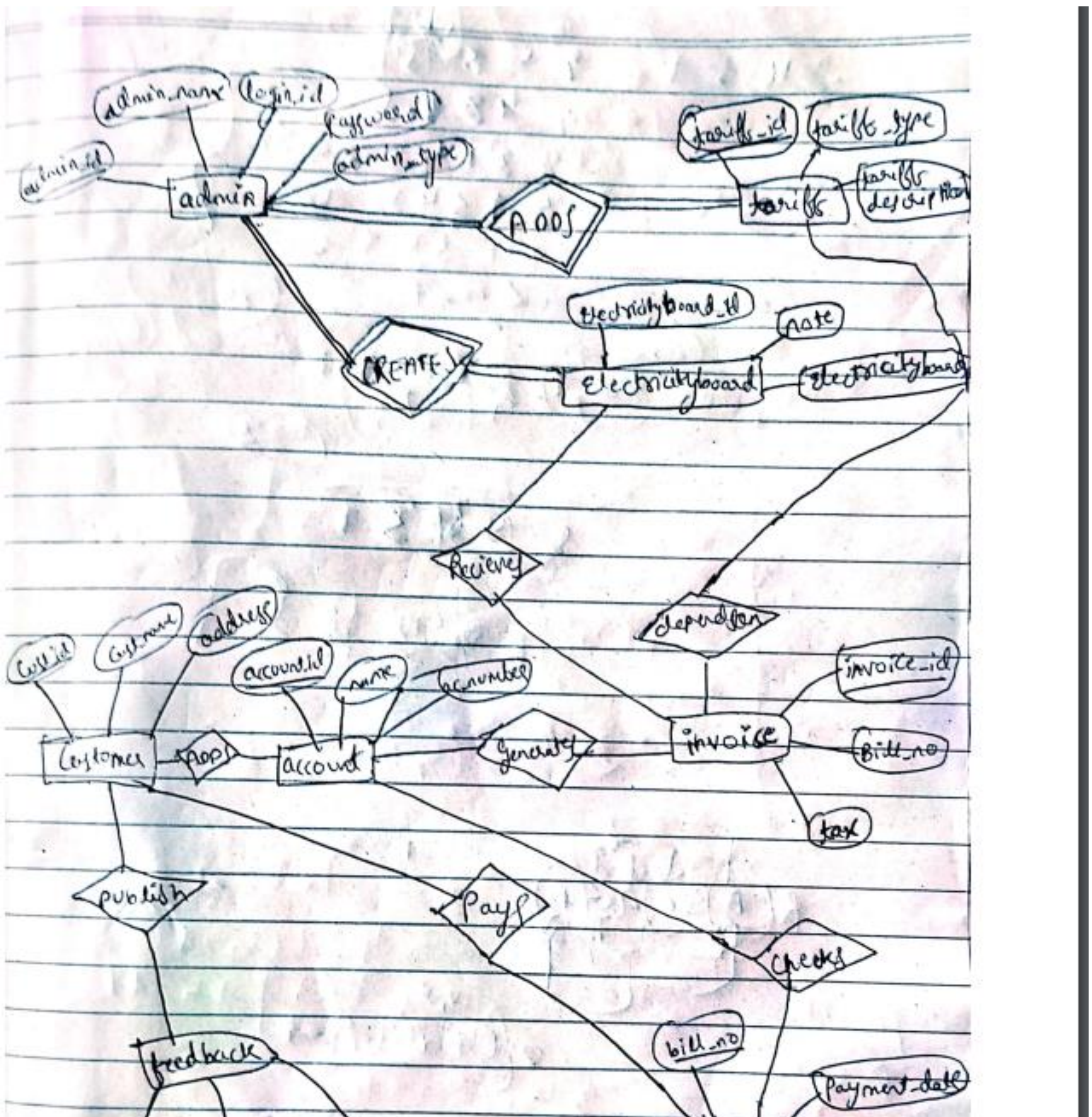
DBMS PROJECT(ELECTRICITY BILLING SYSTEM)

TEAM MEMBERS:

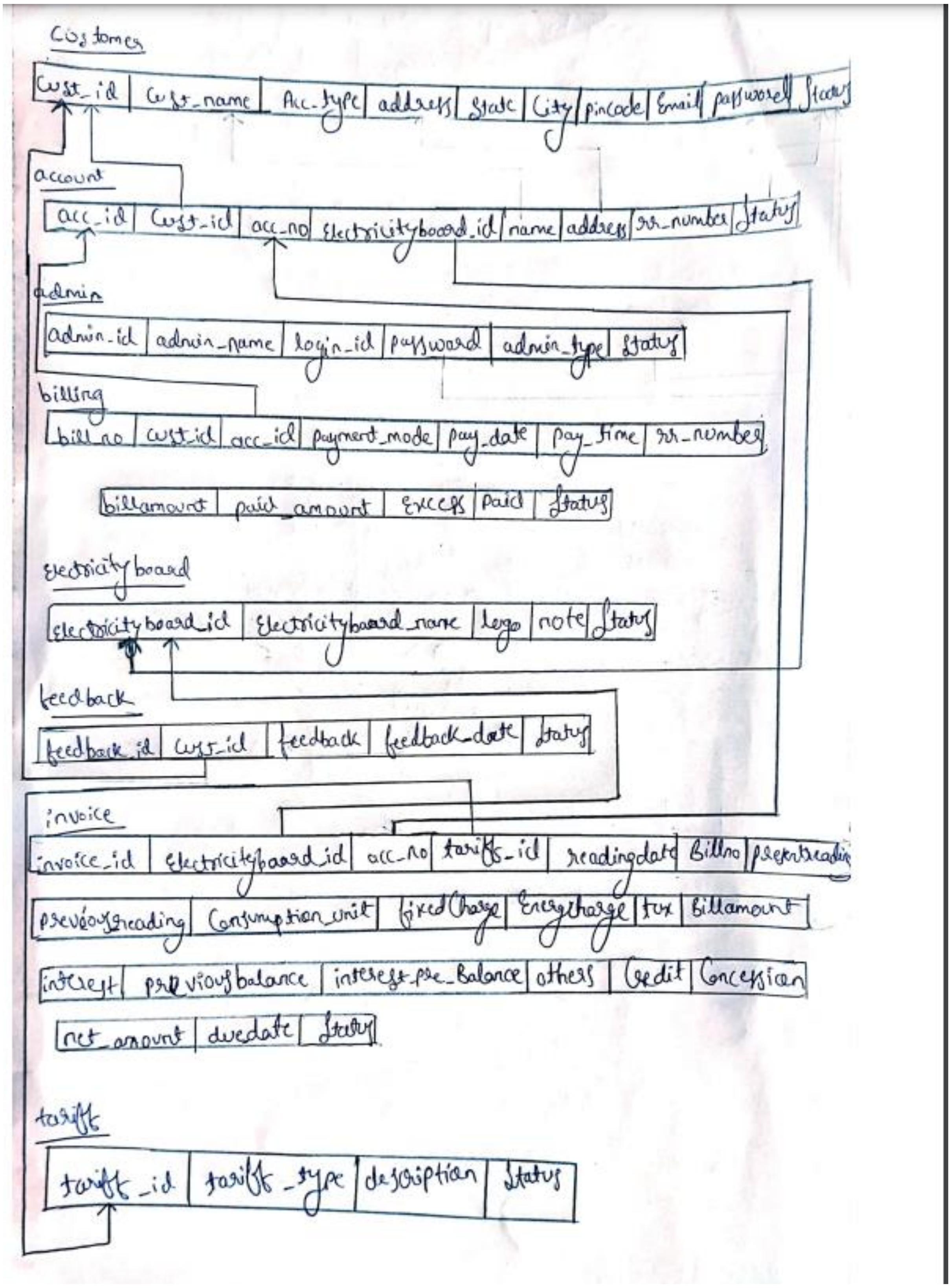
K SAI VIGNESH	SRN: PES1UG21CS260
KARTHIK .S. KADEWADI	SRN:PES1UG21CS271

ER DIAGRAM:

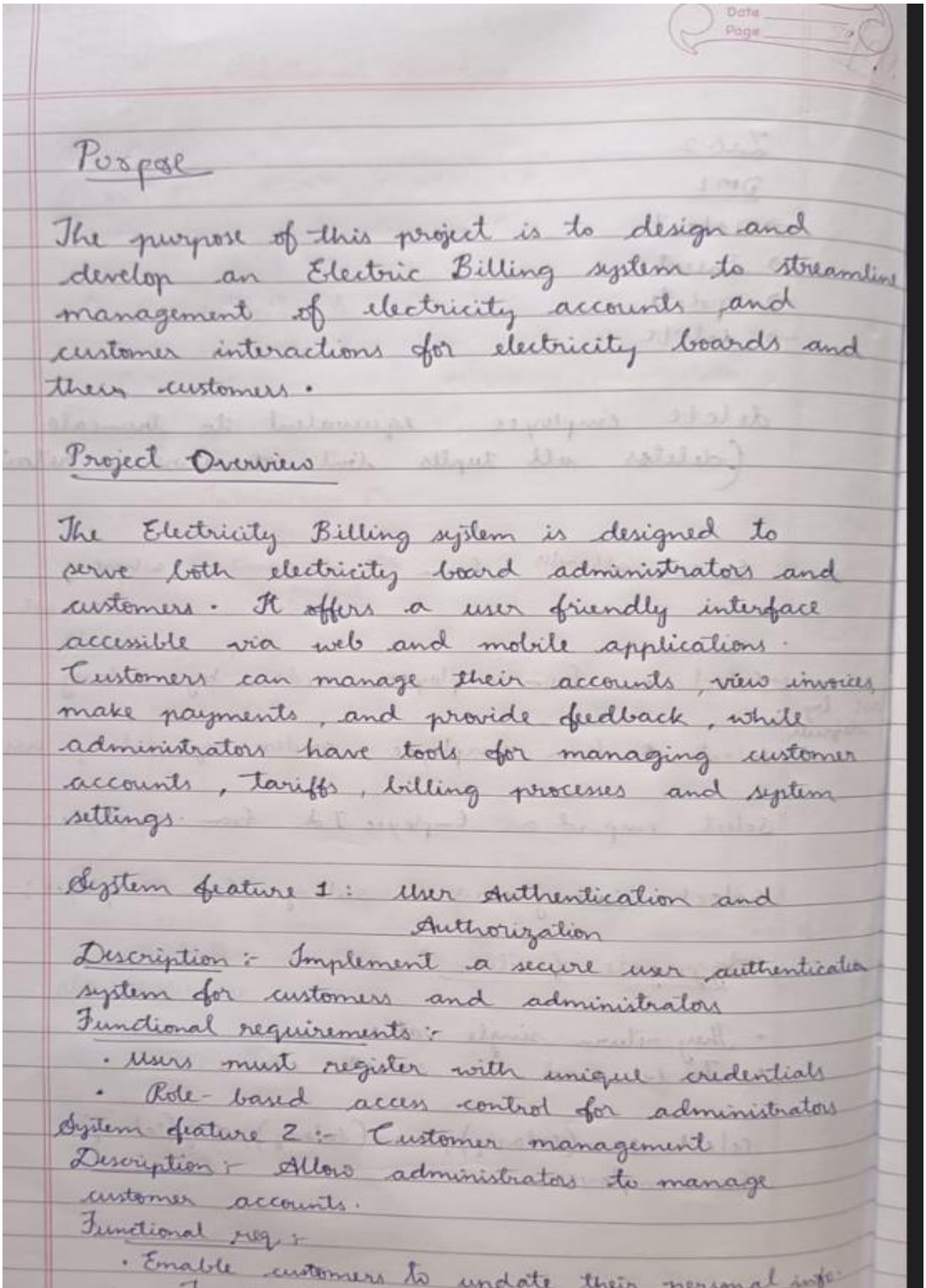


DUE TO SPACE CONSTRAINTS WE COULDN'T SHOW ALL THE ATTRIBUTES BECAUSE INVOICE TABLE ITSELF HAS SO MANY ATTRIBUTES

RELATIONAL SCHEMA:



PROJECT USER REQUIREMENT SPECIFICATION:



Purpose

The purpose of this project is to design and develop an Electric Billing system to streamline management of electricity accounts, and customer interactions for electricity boards and their customers.

Project Overview

The Electricity Billing system is designed to serve both electricity board administrators and customers. It offers a user friendly interface accessible via web and mobile applications. Customers can manage their accounts, view invoices, make payments, and provide feedback, while administrators have tools for managing customer accounts, tariffs, billing processes and system settings.

System feature 1 : User Authentication and Authorization

Description :- Implement a secure user authentication system for customers and administrators

Functional requirements :-

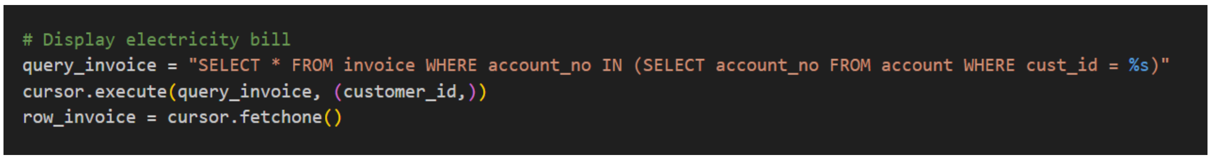
- Users must register with unique credentials
- Role-based access control for administrators

System feature 2 :- Customer management

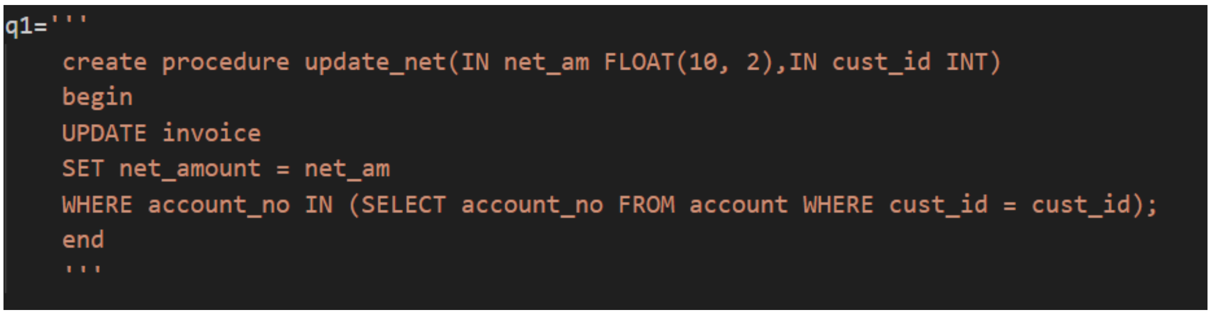
Description :- Allow administrators to manage customer accounts.

Functional req. :-

- Enable customers to update their personal info.
- Track customer acc status.

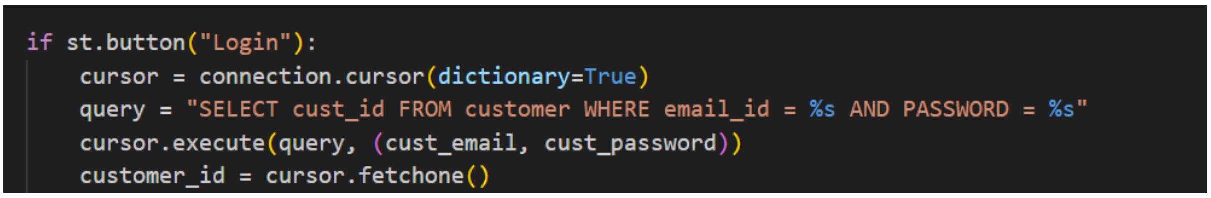


```
# Display electricity bill
query_invoice = "SELECT * FROM invoice WHERE account_no IN (SELECT account_no FROM account WHERE cust_id = %s)"
cursor.execute(query_invoice, (customer_id,))
row_invoice = cursor.fetchone()
```

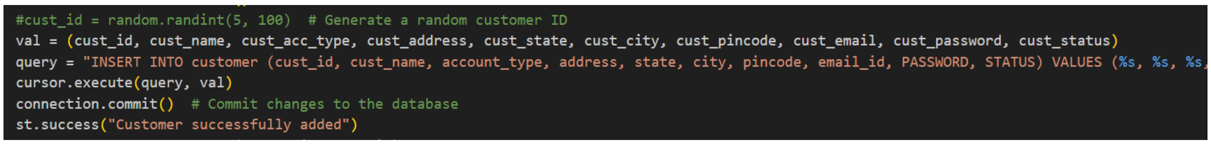


```
q1='''
    create procedure update_net(IN net_am FLOAT(10, 2),IN cust_id INT)
    begin
    UPDATE invoice
    SET net_amount = net_am
    WHERE account_no IN (SELECT account_no FROM account WHERE cust_id = cust_id);
    end
    '''
```

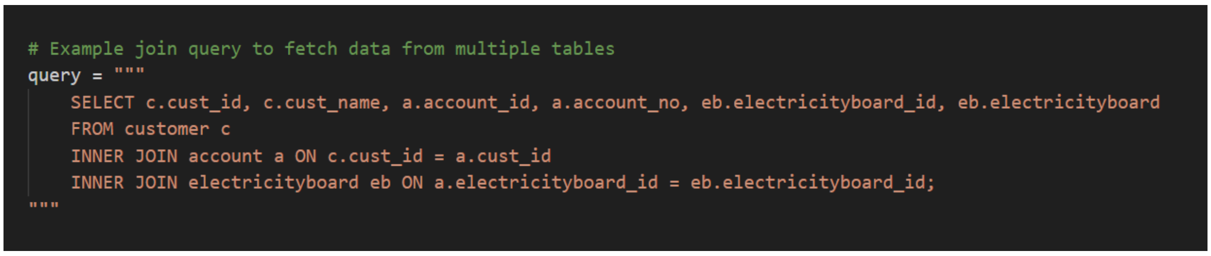
```
CREATE PROCEDURE CalculateNetAmount(in cu float(10,2),OUT
netAmountResult FLOAT(10, 2))
    BEGIN
        DECLARE presentReadingParam FLOAT(10, 2);
        DECLARE previousReadingParam FLOAT(10, 2);
        DECLARE consumptionUnitParam FLOAT(10, 2);
        DECLARE fixedChargeParam FLOAT(10, 2);
        DECLARE energyChargeParam FLOAT(10, 2);
        DECLARE taxParam FLOAT(10, 2);
        DECLARE interestParam FLOAT(10, 2);
        DECLARE previousBalanceParam FLOAT(10, 2);
        DECLARE interestPreBalanceParam FLOAT(10, 2);
        DECLARE othersParam FLOAT(10, 2);
        DECLARE creditParam FLOAT(10, 2);
        DECLARE concessionParam FLOAT(10, 2);


        SELECT
            present_reading,
            previous_reading,
            consumption_unit,
            fixed_charge,
            energy_charge,
            tax,
            interest,
            previous_balance,
            interest_pre_balance,
            others,
            credit,
            consession
        INTO
            presentReadingParam,
            previousReadingParam,
            consumptionUnitParam,
            fixedChargeParam,
            energyChargeParam,
            taxParam,
            interestParam,
            previousBalanceParam,
            interestPreBalanceParam,
            othersParam,
            creditParam,
            concessionParam
        FROM invoice
        WHERE account_no IN (SELECT account_no FROM account WHERE
cust_id = %s);

        IF consumptionUnitParam < 200 THEN
            SET energyChargeParam = 0;
        ELSEIF consumptionUnitParam >= 200 AND consumptionUnitParam
< 300 THEN
            SET energyChargeParam = 1.5 * energyChargeParam;
        ELSE
            SET energyChargeParam = 2 * energyChargeParam;
        END IF;

        SET netAmountResult = (
            (cu + energyChargeParam) +
            fixedChargeParam +
            taxParam +
            interestParam +
            previousBalanceParam +
            interestPreBalanceParam +
            othersParam -
            creditParam -
            concessionParam
        );

        UPDATE invoice
        SET net_amount = netAmountResult
        WHERE account_no IN (SELECT account_no FROM account WHERE
cust_id = %s);

    END'''
```



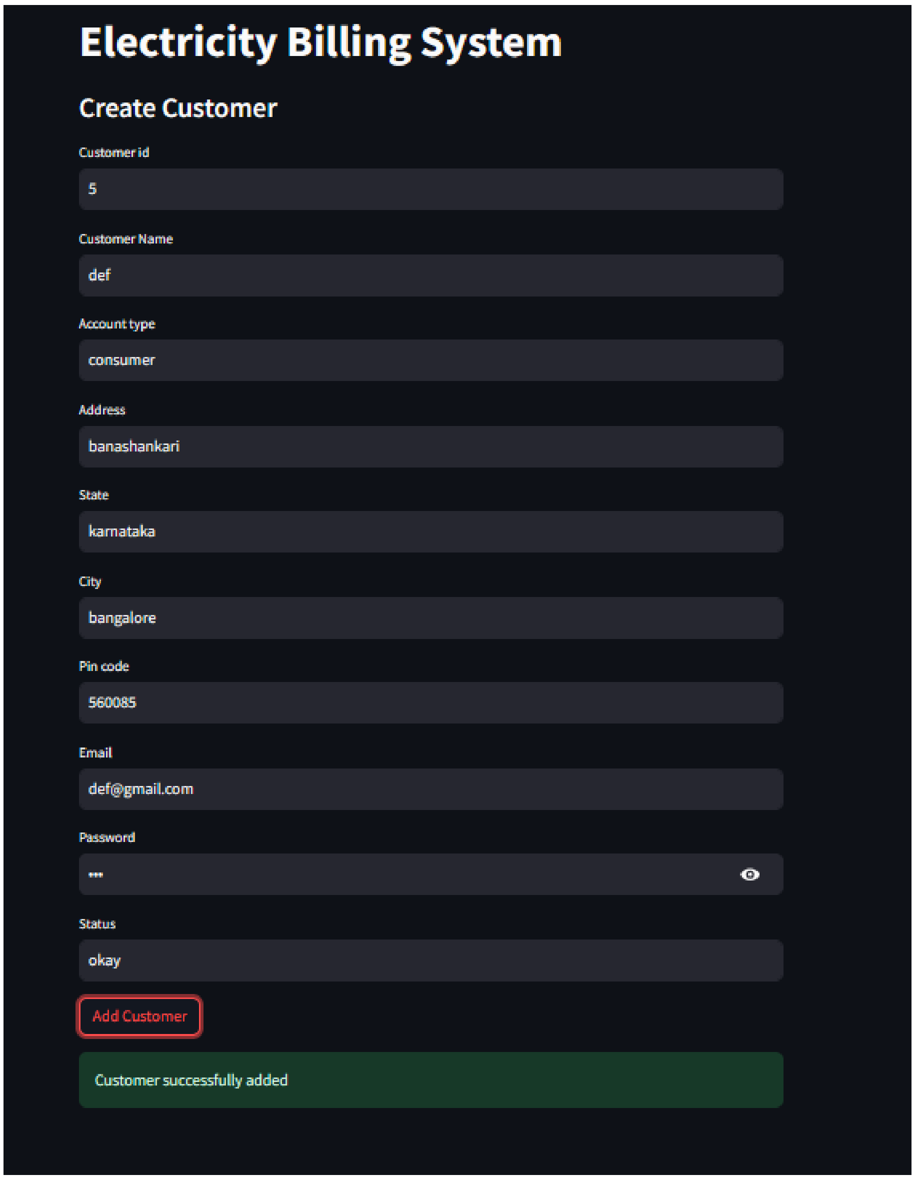
```
if st.button("Login"):
    cursor = connection.cursor(dictionary=True)
    query = "SELECT cust_id FROM customer WHERE email_id = %s AND PASSWORD = %s"
    cursor.execute(query, (cust_email, cust_password))
    customer_id = cursor.fetchone()
```



```
#cust_id = random.randint(5, 100)  # Generate a random customer ID
val = (cust_id, cust_name, cust_acc_type, cust_address, cust_state, cust_city, cust_pincode, cust_email, cust_password, cust_status)
query = "INSERT INTO customer (cust_id, cust_name, account_type, address, state, city, pincode, email_id, PASSWORD, STATUS) VALUES (%s, %s, %s,
cursor.execute(query, val)
connection.commit()  # Commit changes to the database
st.success("Customer successfully added")
```



```
# Example join query to fetch data from multiple tables
query = """
    SELECT c.cust_id, c.cust_name, a.account_id, a.account_no, eb.electricityboard_id, eb.electricityboard
    FROM customer c
    INNER JOIN account a ON c.cust_id = a.cust_id
    INNER JOIN electricityboard eb ON a.electricityboard_id = eb.electricityboard_id;
"""
```



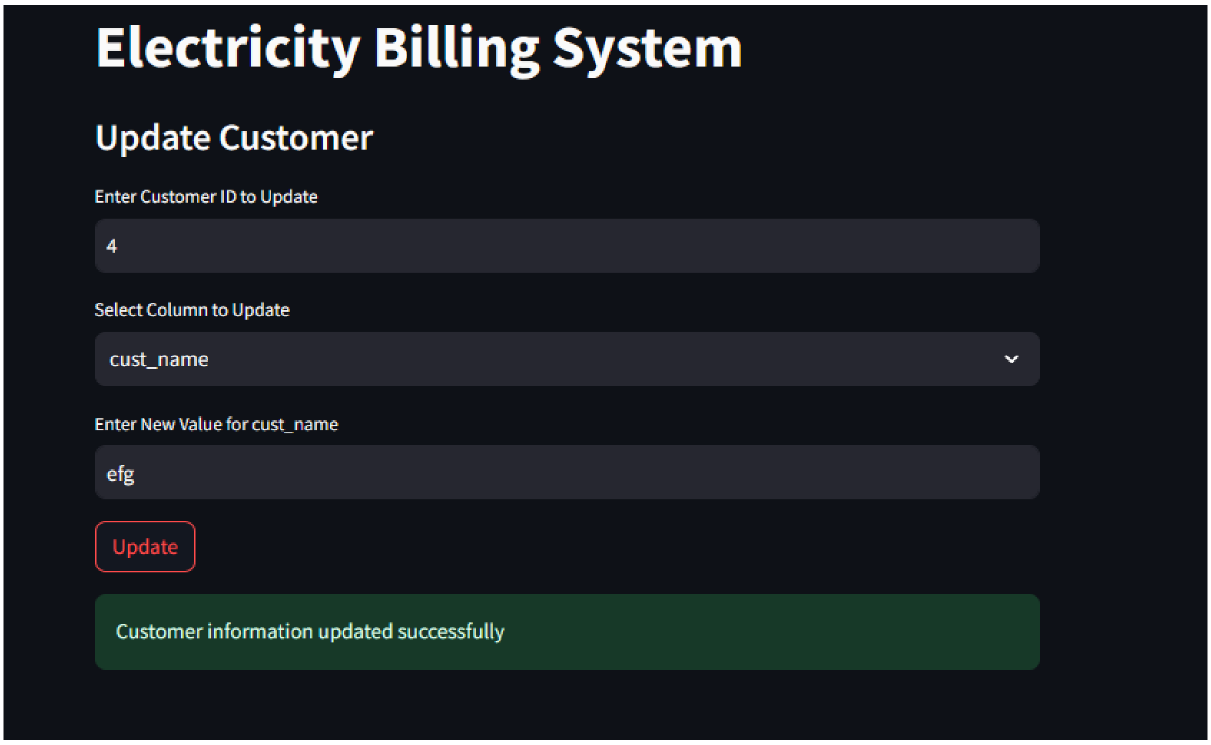
Electricity Billing System

Create Customer

Customer id	5
Customer Name	def
Account type	consumer
Address	banashankari
State	karnataka
City	bangalore
Pin code	560085
Email	def@gmail.com
Password	•••
Status	okay

Add Customer

Customer successfully added



Electricity Billing System

Update Customer

Enter Customer ID to Update

4

Select Column to Update

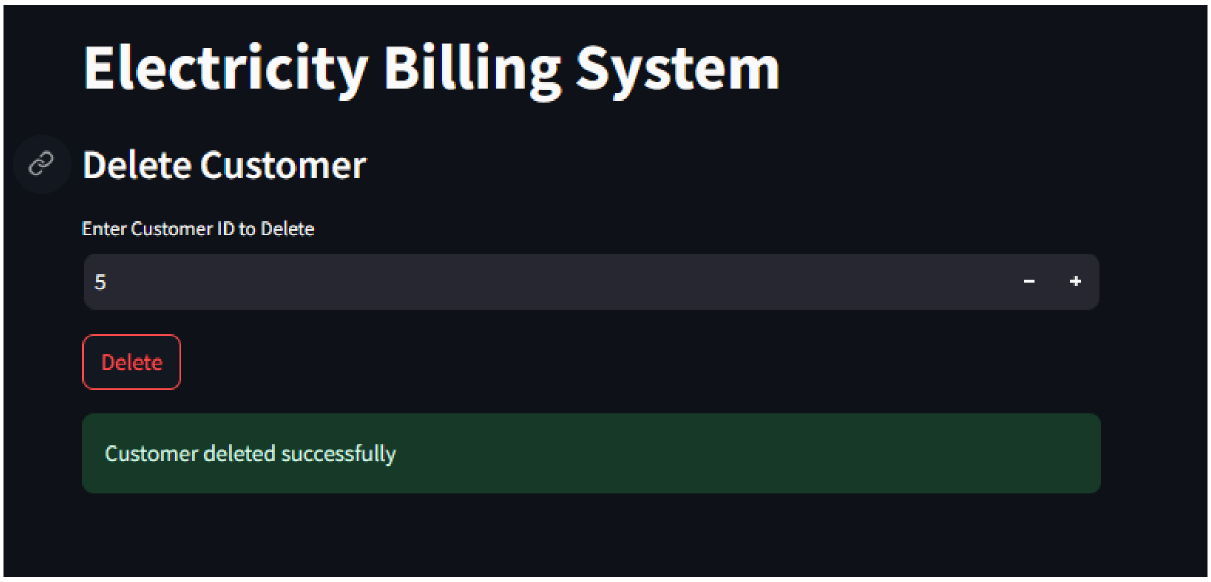
cust_name

Enter New Value for cust_name

efg

Update

Customer information updated successfully



Electricity Billing System

Delete Customer

Enter Customer ID to Delete

5

Delete

Customer deleted successfully