# Database System

## Inventory Management System

20K-0218 Muhammad Hatif Mujahid
20K-0208 Wahaj Javed Alam
20K-0316 Minhal Irfan

# ❖ Introduction

The method through which you keep track of your products across the whole supply chain, from purchase to manufacture to final sales, is known as an **Inventory Management System (or Inventory System)**. It controls how you go about managing your company's inventory.

Our system has two types of accessing modes:

1. **Admin**
2. **Retailer**

**1)Admin:**

- Monitors all the Inventories
- Allows a Retailer to Create account by Approving it
- Assigns Inventories to Retailers
- Accesses transactions of all the Inventories (History of Transactions)

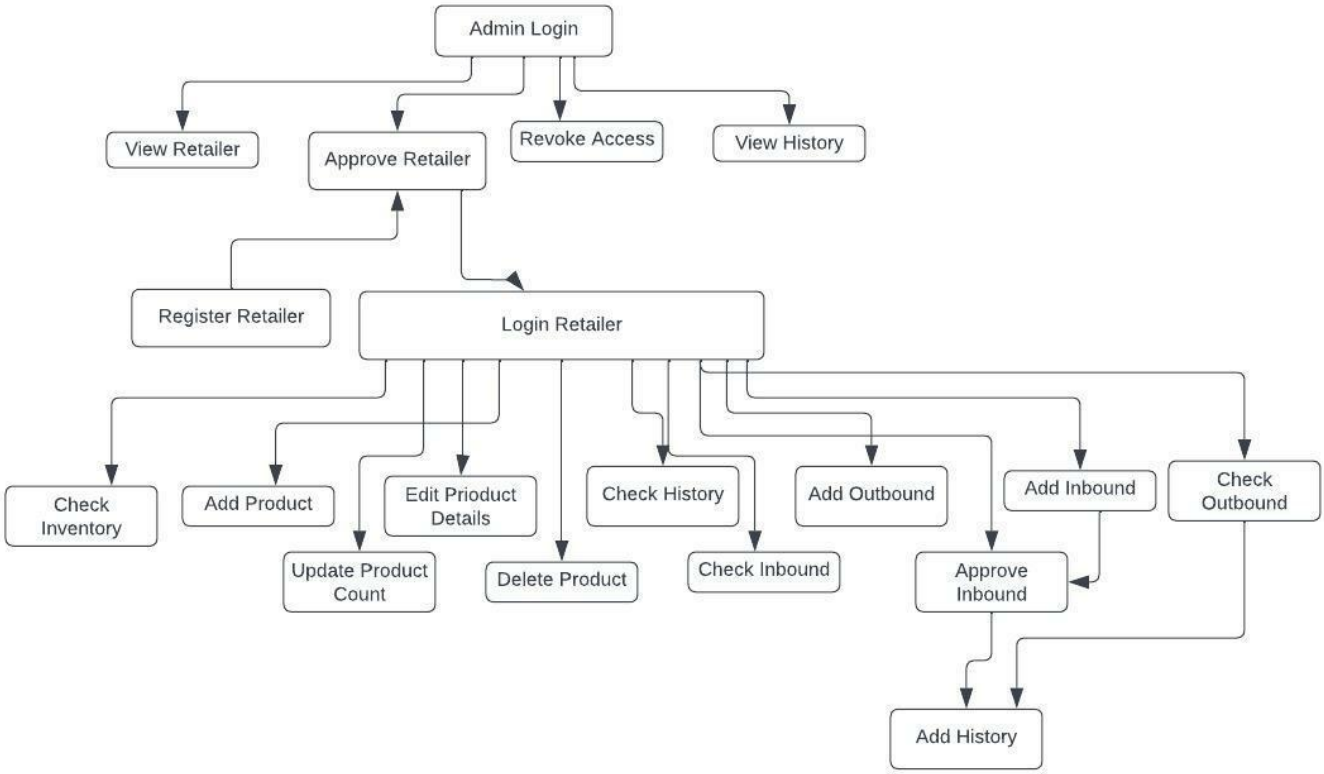
**2)Retailer:** Manages inventory assigned to it ,it:

- Can Create a new account(A new Retailer can Register)
- Accesses all the Inbound/Outbound transactions of the Inventory (View the history of Transactions into/out of his Inventory)
- Can add a new product
- Delete a product from inventory
- Update a Product's Details from the Inventory

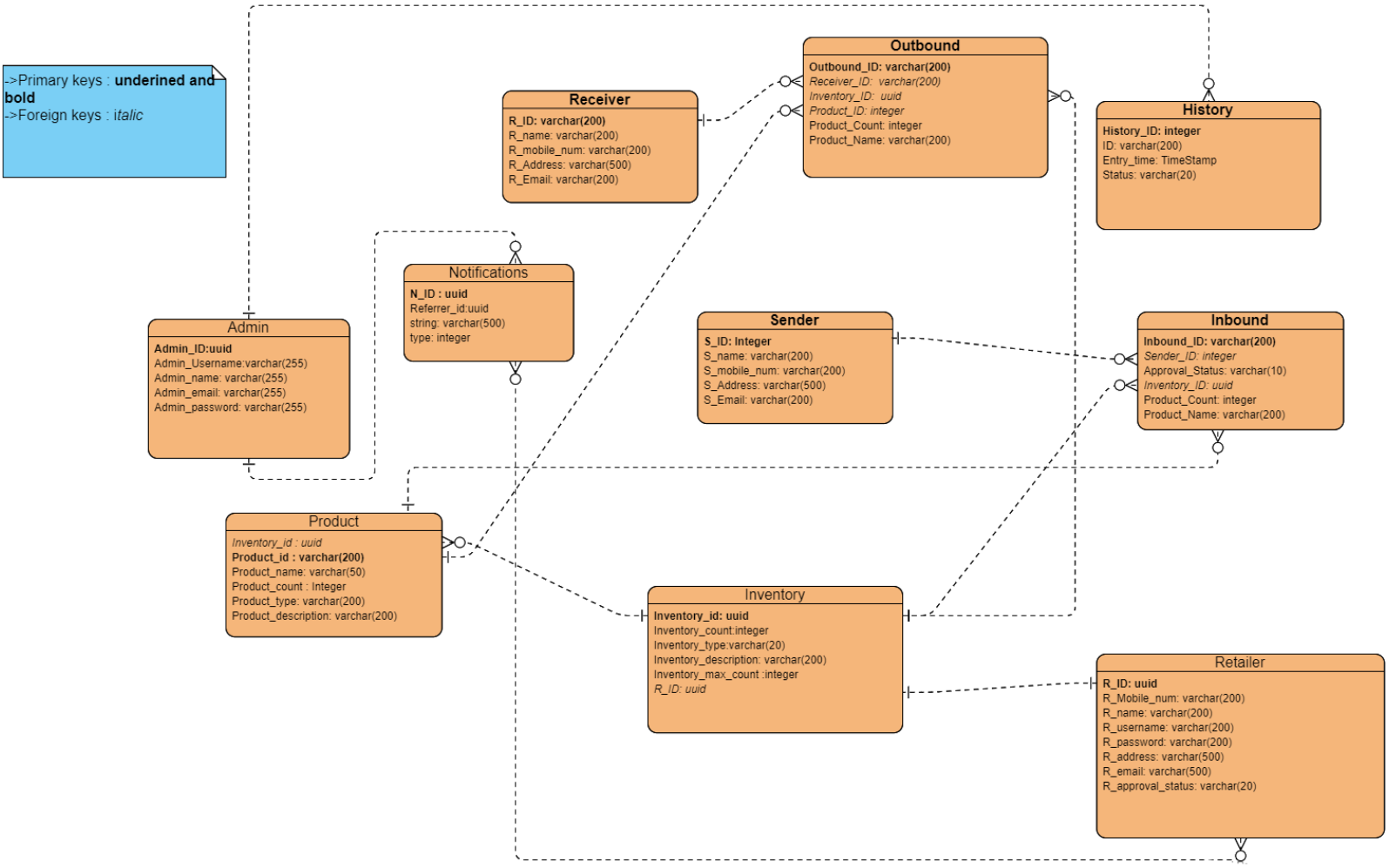
## ❖ Technology used

| | |
|---|---|
| Database | PostgreSQL |
| Backend | Node, Express JS |
| Front-End | React JS, MUI |

## ❖ System Design (not done)



# ❖ ER DIAGRAM


->Primary keys : **underined and bold**
->Foreign keys : *italic*
Outbound
Outbound_ID: varchar(200)
Receiver_ID: varchar(200)
Inventory_ID: uuid
Product_ID: integer
Product_Count: integer
Product_Name: varchar(200)
Receiver
R_ID: varchar(200)
R_name: varchar(200)
R_mobile_num: varchar(200)
R_Address: varchar(500)
R_Email: varchar(200)
History
History_ID: integer
ID: varchar(200)
Entry_time: TimeStamp
Status: varchar(20)
Notifications
N_ID : uuid
Referrer_id:uuid
string: varchar(500)
type: integer
Admin
Admin_ID:uuid
Admin_Username:varchar(255)
Admin_name: varchar(255)
Admin_email: varchar(255)
Admin_password: varchar(255)
Sender
S_ID: Integer
S_name: varchar(200)
S_mobile_num: varchar(200)
S_Address: varchar(500)
S_Email: varchar(200)
Inbound
Inbound_ID: varchar(200)
Sender_ID: integer
Approval_Status: varchar(10)
Inventory_ID: uuid
Product_Count: integer
Product_Name: varchar(200)
Product
Inventory_id : uuid
Product_id : varchar(200)
Product_name: varchar(50)
Product_count : Integer
Product_type: varchar(200)
Product_description: varchar(200)
Inventory
Inventory_id: uuid
Inventory_count:integer
Inventory_type:varchar(20)
Inventory_description: varchar(200)
Inventory_max_count :integer
R_ID: uuid
Retailer
R_ID: uuid
R_Mobile_num: varchar(200)
R_name: varchar(200)
R_username: varchar(200)
R_password: varchar(200)
R_address: varchar(500)
R_email: varchar(500)
R_approval_status: varchar(20)

# ❖ Normalization

**INITIAL TABLES( BEFORE NORMALIZATION)**

- **History**(History_ID,ID,Entry_time,status)
- **Admin**(Admin_ID,Admin_Name,Admin_UserName,Admin_Email,Admin_Password)
- **OutBound**(Outbound_ID,Inventory_ID,Product_ID,Product_Count,Receiver_ID,R_Name,R_Mobile_Num,R_Address,R_email)
- **InBound**(InBound_ID,Sender_ID,Approval_status,Inventory_ID,Product_Count,Product_Name,S_Name,S_Mobile_Num,S_Address,S_Email)
- **Inventory**(Inventory_ID,Inventory_count,Inventory_type,Inventory_description,Inventory_max_count,Product_ID,Product_name,Product_count,Product_description,Product_type,R_ID,R_mobile_num,R_username,R_name,R_password,R_address,R_email,R_approval_status)
- **Notifications**(N_ID,Referrer_ID,string,type)

**FINAL TABLES (AFTER NORMALIZATION)**

- **History**

| History_ID | ID | Entry_time | status |
|---|---|---|---|

- **Notifications**

| N_ID | Referrer_ID | String | type |
|---|---|---|---|

- **Admin**

| Admin_ID | Admin_name | Admin_Username | Admin_Email | Admin_password |
|---|---|---|---|---|

- **Inventory**

| Inventory_ID | Inventory_count | Inventory_type | Inventory_description | Inventory_max_count | Retailer_ID |
|---|---|---|---|---|---|

Foreign keys: *Retailer_ID* referencing *R_ID of Retailer* table

- **Product**

| Product_ID | Product_name | Product_count | Product_description | Product_type | Inventory_ID |
|---|---|---|---|---|---|

Foreign Keys: *Inventory_ID* referencing *Inventory_ID of Inventory* Table

- **Retailer**

| R_ID | R_mobile_num | R_username | R_name | R_password | R_address | R_email | R_approal_status |
|---|---|---|---|---|---|---|---|

- **Receiver**

| R_ID | R_name | R_mobile_num | R_address | R_email |
|---|---|---|---|---|

- **Outbound**

| Outbound_ID | Inventory_ID | Product_ID | Product_count | Receiver_ID |
|---|---|---|---|---|

Foreign Keys: 1) *Inventory_ID* referencing *Inventory_ID of Inventory* Table
2) *Product_ID* referencing *Product_ID of Product* Table
3) *Receiver_ID* referencing *R_ID of Receiver* Table

- **Sender**

| S_ID | S_name | S_mobile_num | S_address | S_email |
|---|---|---|---|---|

- **Inbound**

| Inbound_ID | Sender_ID | Approval_status | Inventory_ID | Product_count | Product_name |
|---|---|---|---|---|---|

Foreign Keys: 1) *Inventory_ID* referencing *Inventory_ID of Inventory* Table
2) *Sender_ID* referencing *R_ID of Sender* Table

# ❖ Triggers,Procedures,Functions,Views(the implemented ones)

## 1. Triggers

1.1. **CHECK_PASS** → ( Check password length <8)

1.2. **CHECK_NUM** → (Check phone number length ==11)

1.3. **RECIEVER_NEW** → ( autogenerates *Receiver_ID* of a new Receiver who signed up , calls function *R_ID()* to generate the new ID)

1.4. **PRODUCT_NEW** → ( autogenerates *Product_ID* of a new Product added, calls function *Product_New_ID()* to generate the new ID)

1.5. **INBOUND_NEW** → ( autogenerates *Inbound_ID* of a new Inbound Transaction In the Inventory, calls function *Inbound_New_ID()* to generate the new ID)

1.6. **OUTBOUND_NEW** → ( autogenerates *Outbound_ID* of a new Inbound Transaction In the Inventory, calls function *Outbound_New_ID()* to generate the new ID)

1.7. **NEW_INVENTORY** → (Update inventory_count when a new product added,calls the function *New_Inventory_Count*)

1.8. **UPDATE_INVENTORY** → (Update inventory_count when a Product's count is updated ,calls the function *Update_Inventory_Count*)

1.9. **DELETE_INVENTORY** → (Updates inventory_count when product deleted,calls function *Delete_Inventory_Count()*)

1.10. **ADD_INBOUND_HISTORY** → (Trigger for updating history from outbound)

1.11. **ADD_OUTBOUND_HISTORY** → (Trigger for updating history from inbound)

## 2. Views

2.1. **RETAILER_ACCESSES** → shows all the details of the Retailer and its assigned Inventory

## 3. Functions

3.1. **CHECK_PASSWORD()** → To check if the length of password entered is less than **8.**

3.2. **CHECK_PHONE()** → To check if the length of phone number entered is less than **11**

3.3. **R_ID ()** → To AutoGenerate Receiver_ID for every new receiver who signed up

3.4. **PRODUCT_NEW_ID()** → To AutoGenerate Product_ID for every new Product added.

3.5. **INBOUND_NEW_ID()** → To AutoGenerate Inbound_ID for every new Inbound Transaction.

3.6. **OUTBOUND_NEW_ID()** → To AutoGenerate Outbound_ID for every new Outbound Transaction.

3.7. **NEW_INVENTORY_COUNT()** → Updates Inventory Count if a new product is added into the inventory IF the Inventory still has capacity ie, if the total count still remains less than the MAX_COUNT for each inventory even after the updation of count ;else, returns "not Possible" and donot adds Product into the Inventory.

3.8. **UPDATE_INVENTORY_COUNT()** → Updates Inventory Count if a product count is updated into the inventory IF the Inventory still has capacity ie, if the total count still remains less than the MAX_COUNT for each inventory even after the updation of count ;else, returns "not Possible" and donot allows the count of the Product to be updated into the Inventory.

3.9. **DELETE_INVENTORY_COUNT()** → Updates Inventory count if a Product is deleted from the Inventory.

3.10. **ADD_INBOUND_HISTORY()** → Inserts A Product into Inventory only if the retailer allows the product to be added

# ❖ Connectivity Screenshots

1. Check the current state of the table

```
wahaj@DESKTOP-HCO43HH:~/db-project-inventory$ sudo -u postgres psql
psql (12.12 (Ubuntu 12.12-0ubuntu0.20.04.1))
Type "help" for help.

postgres=# \c ivms_db
You are now connected to database "ivms_db" as user "postgres".
ivms_db=# select * from retailer;
 r_id | r_mobile_num | inventory_id | r_name | r_username | r_password | r_address | r_email | r_approval_status
------+--------------+--------------+--------+------------+------------+-----------+---------+-------------------
(0 rows)

ivms_db=#
```

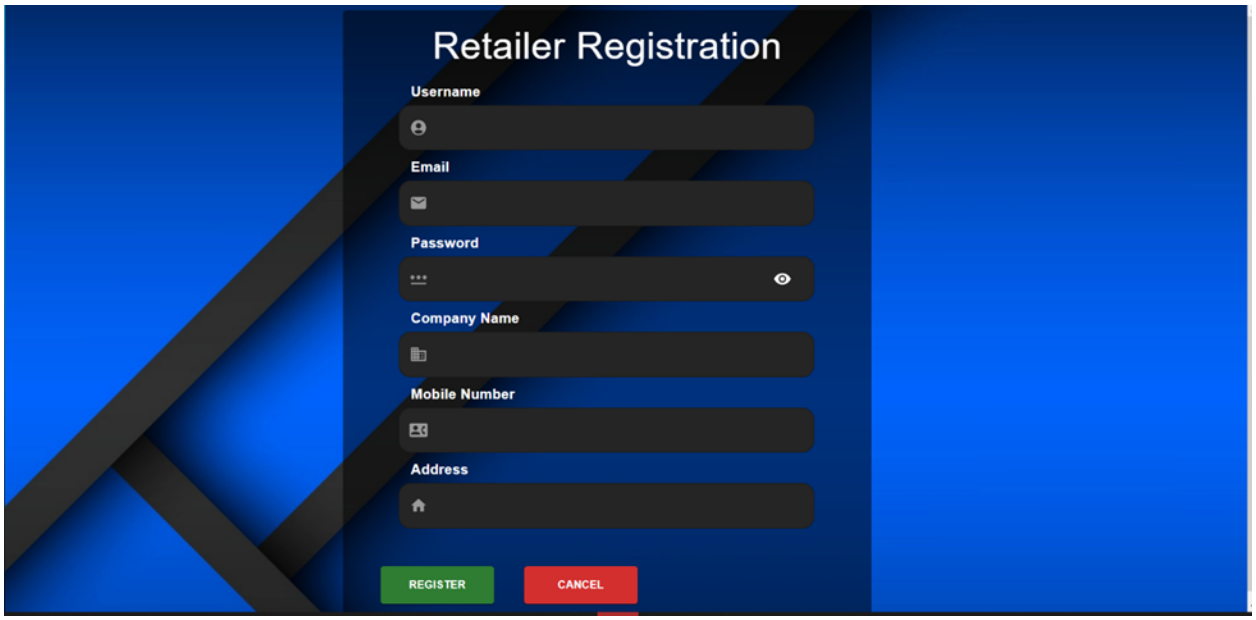
2. Register API code

```
router.post("/register/retailer", async (req, res) => {
    const {
        username,
        email,
        companyName,
        password,
        mobile,
        address,
    } = req.body;

    try {
        const user = await pool.query(
            "SELECT * FROM retailer WHERE r_email = $1",
            [email]
        );

        if (user.rows.length > 0) {
            return res.status(401).json("Company already exists!");
        }
        const salt = await bcrypt.genSalt(10);
        const bcryptPassword = await bcrypt.hash(password, salt);

        let newUser = await pool.query(
            "INSERT INTO retailer (r_name,r_username, r_password,r_address,r_mobile_num,r_email) VALUES ($1, $2, $3, $4,$5,$6) RETURNING r_id",
            [companyName, username, bcryptPassword, address, mobile, email]
        );
        let notif = await pool.query(
            "INSERT INTO NOTIFICATIONS(referrer_id,string,type) VALUES ($1,$2,$3)",
            [newUser.rows[0].r_id, "Approve Retailer", 1]
        );
        const jwtToken = jwtGenerator(newUser.rows[0].r_id);
        return res.json({ jwtToken });
    } catch (err) {
        console.error(err.message);
        res.status(500).send("Server error");
    }
});
```

3. Start server

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   COMMENTS

wahaj@DESKTOP-HCO43HH:~/db-project-inventory/server$ nodemon server
[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Server is starting on port 5000
```

4. Go to sign up page




5. Fill in Data

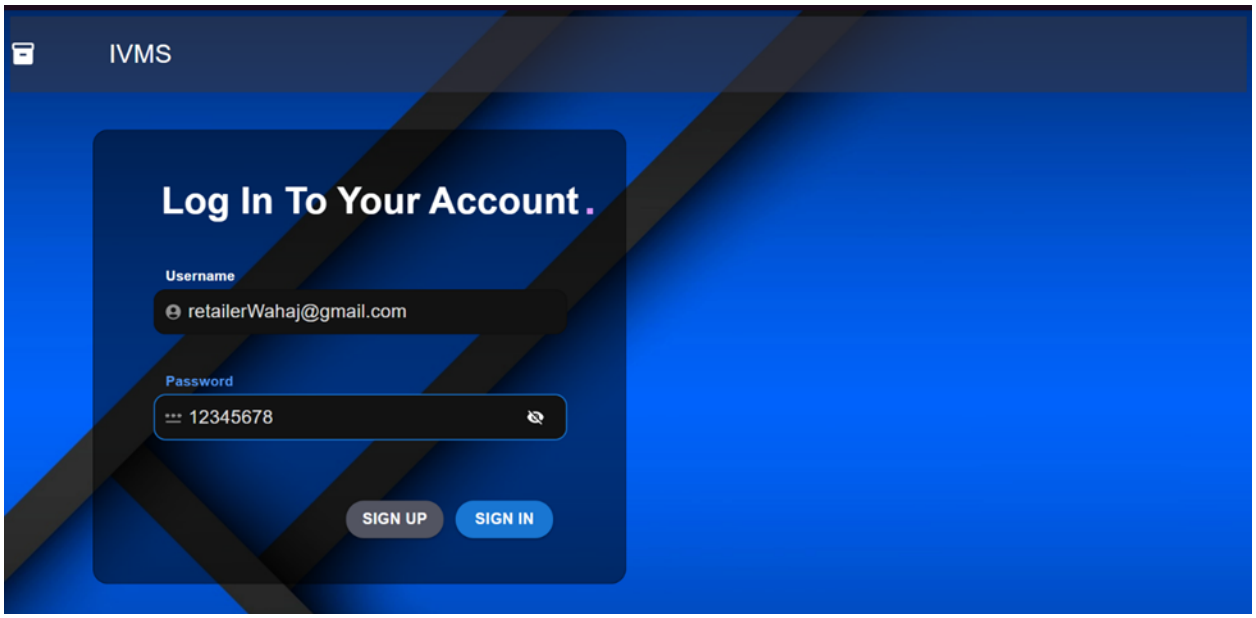

6. Clicking on register calls the API from frontend

```javascript
async function handleSubmit(e) {
    e.preventDefault();
    const inputs = {
        username: username,
        password: password,
        email: email,
        companyName: companyName,
        mobile: mobile,
        address: address,
    };
    try {
        const response = await fetch(
            "http://localhost:5000/authentication/register/retailer",
            {
                method: "POST",
                headers: { "Content-type": "application/json" },
                body: JSON.stringify(inputs),
            }
        );
        const parseRes = await response.json();
        localStorage.setItem("token", parseRes.jwtToken);
        localStorage.setItem("type", "retailer");
        const history = useNavigate();
        history("/");
        setAuth(true);
    } catch (err) {
        console.error(err);
    }
}
```

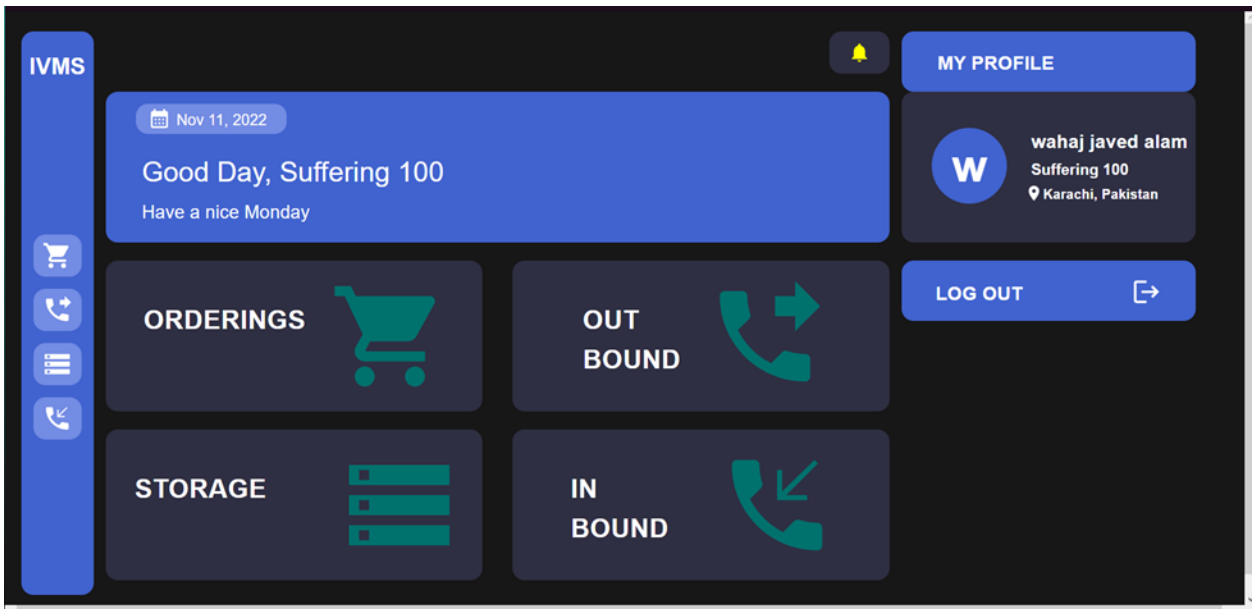
7. The new account is now recorded In the database

```
ivms_db=# \x on
Expanded display is on.
ivms_db=# select * from retailer;
-[ RECORD 1 ]-----+-----------------------------------------------------------
r_id              | 1d2eab56-7a29-4fe6-a909-5572f1f068ee
r_mobile_num      | 03101026205
inventory_id      |
r_name            | Suffering 100
r_username        | wahaj javed alam
r_password        | $2b$10$RR4DDlwnLUd5SEu70PM9DeizE2HBK9X7Gcsfc7Ee8xp/qeBdhkfja
r_address         | Karachi, Pakistan
r_email           | retailerWahaj@gmail.com
r_approval_status | FALSE

ivms_db=#
```

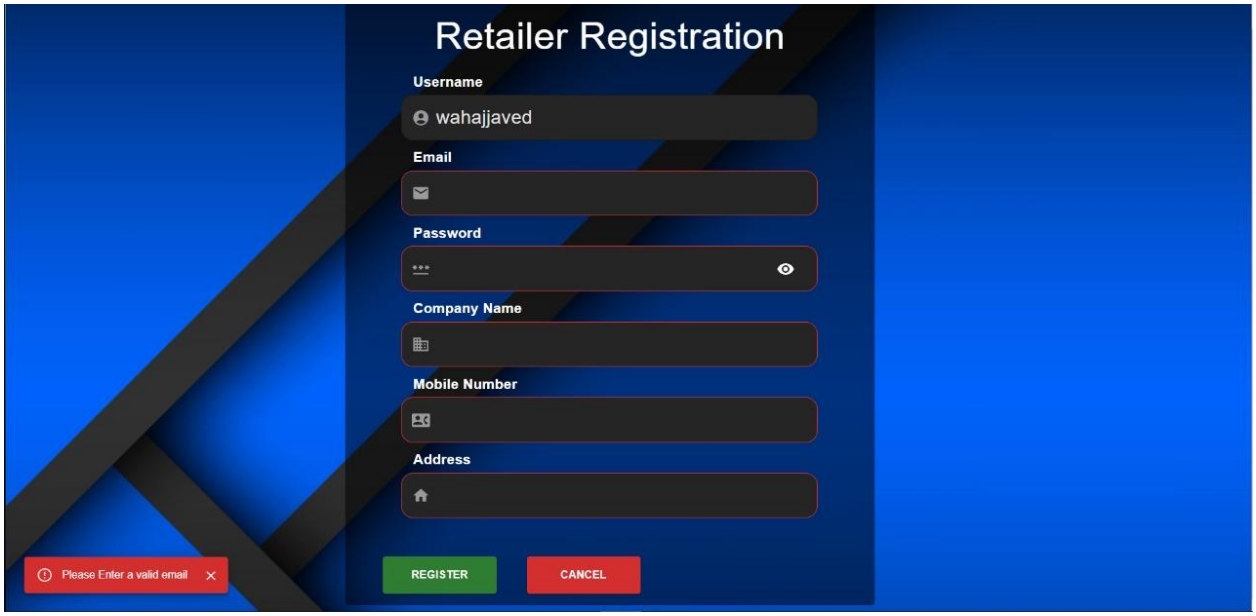
8. Input login Credentials

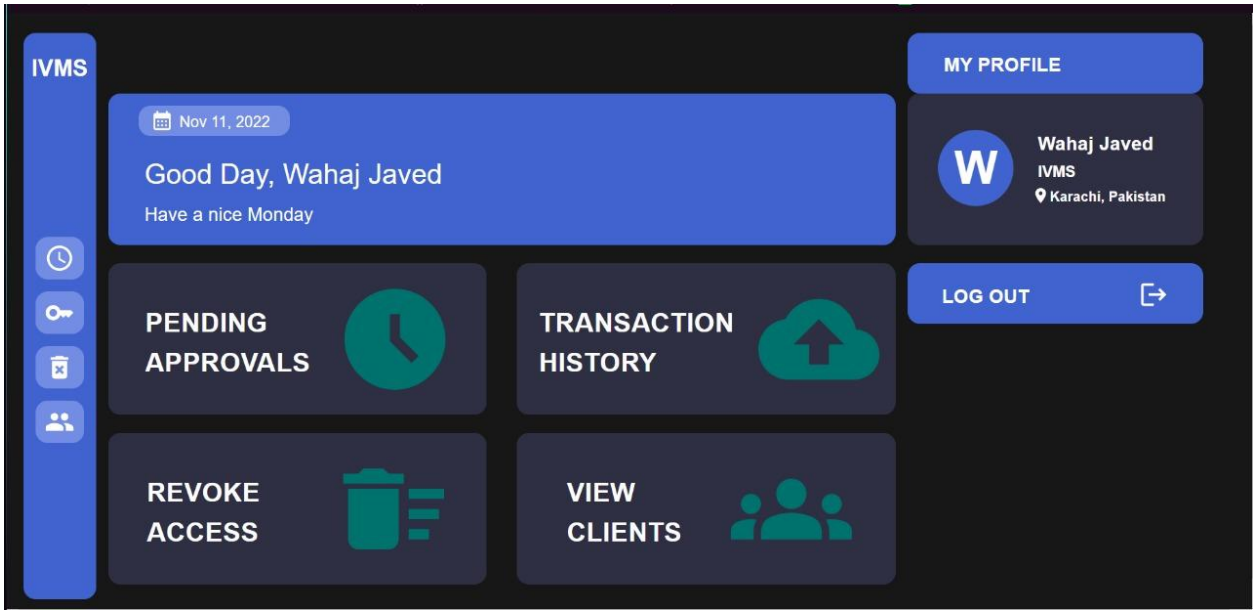


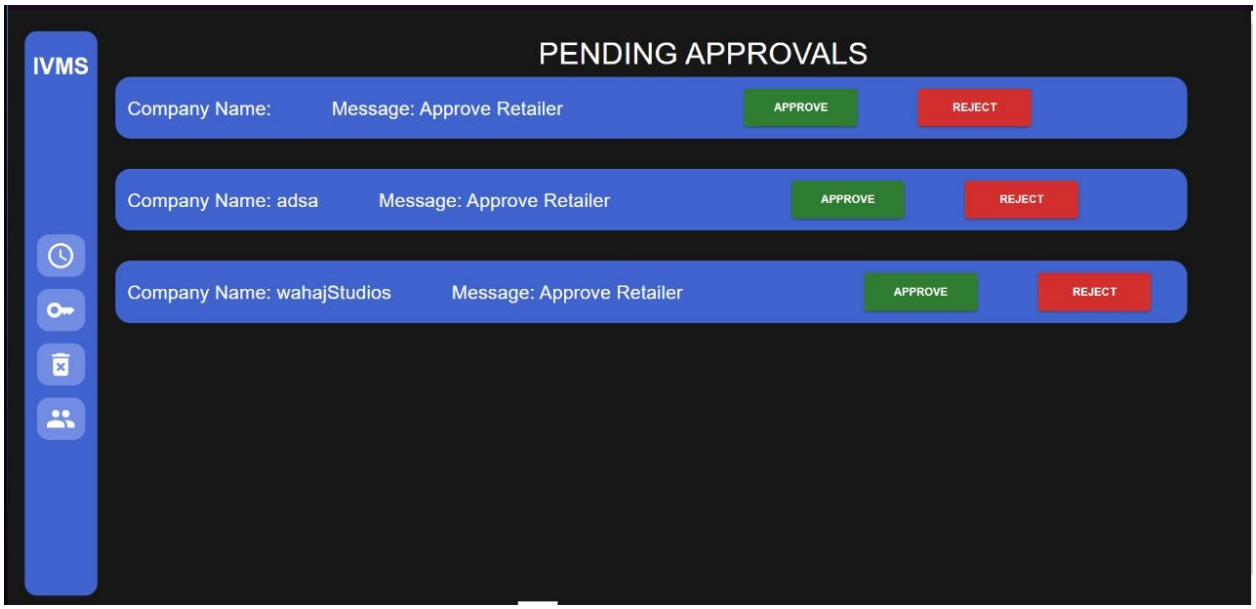

9. Dashboard loaded

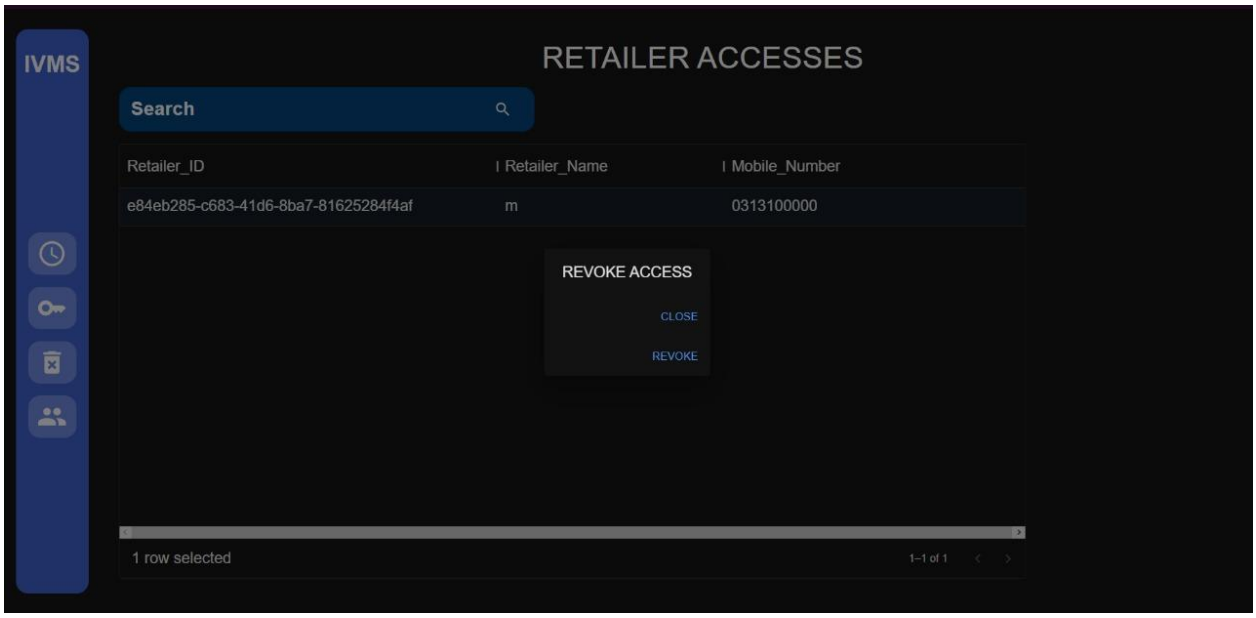


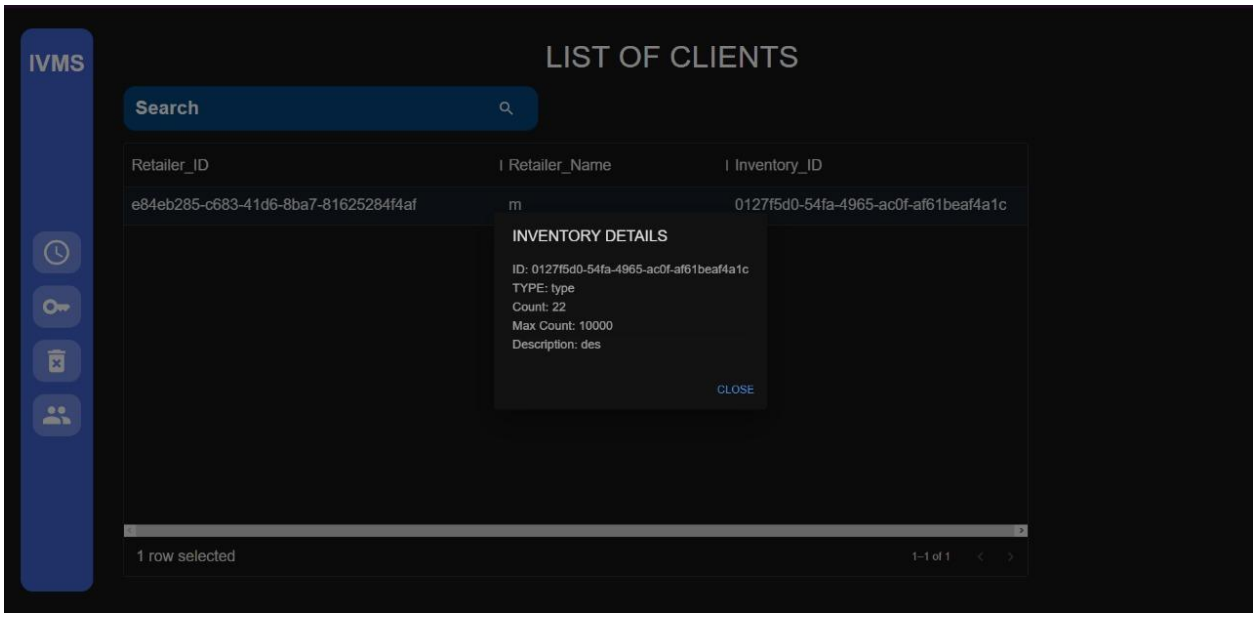
## ❖ Screenshots of project

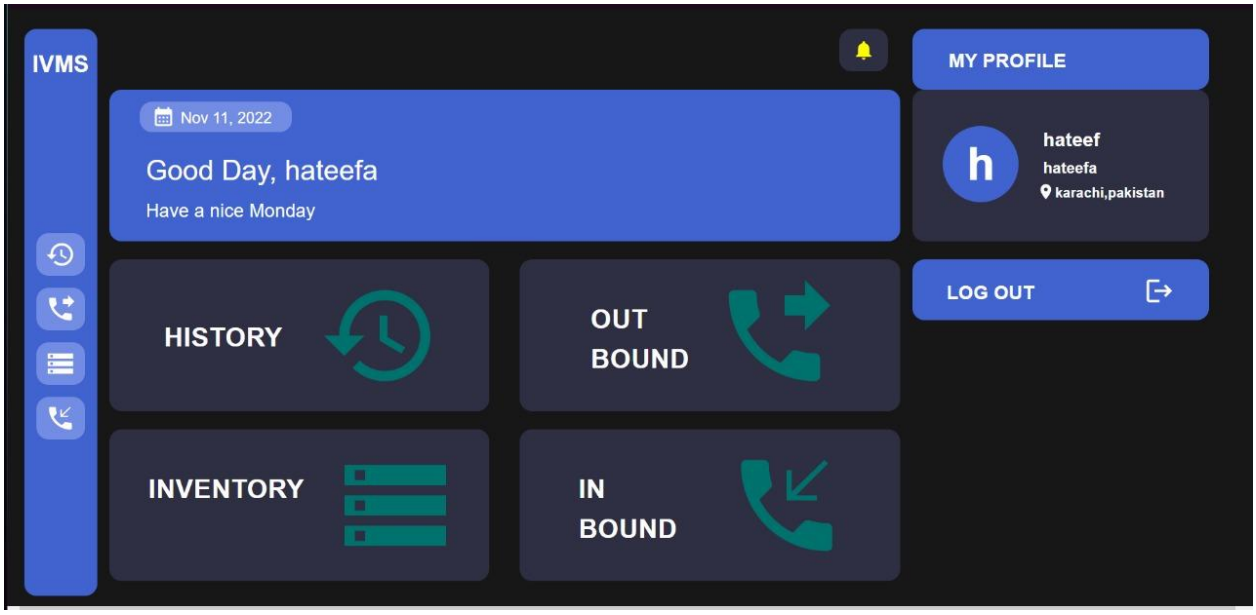


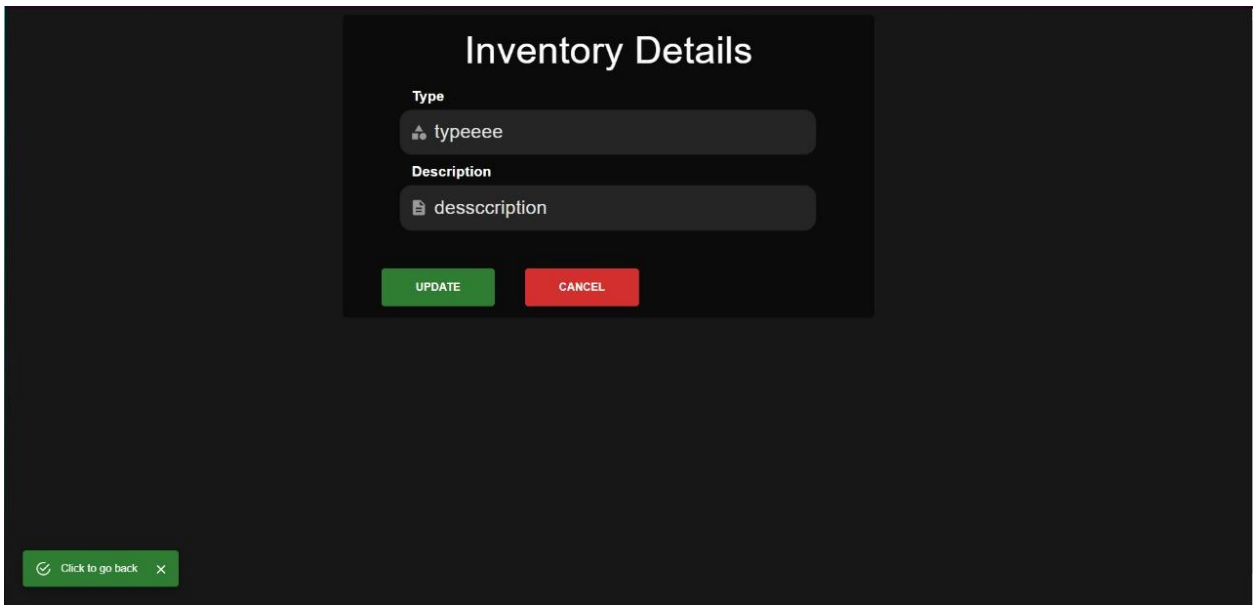



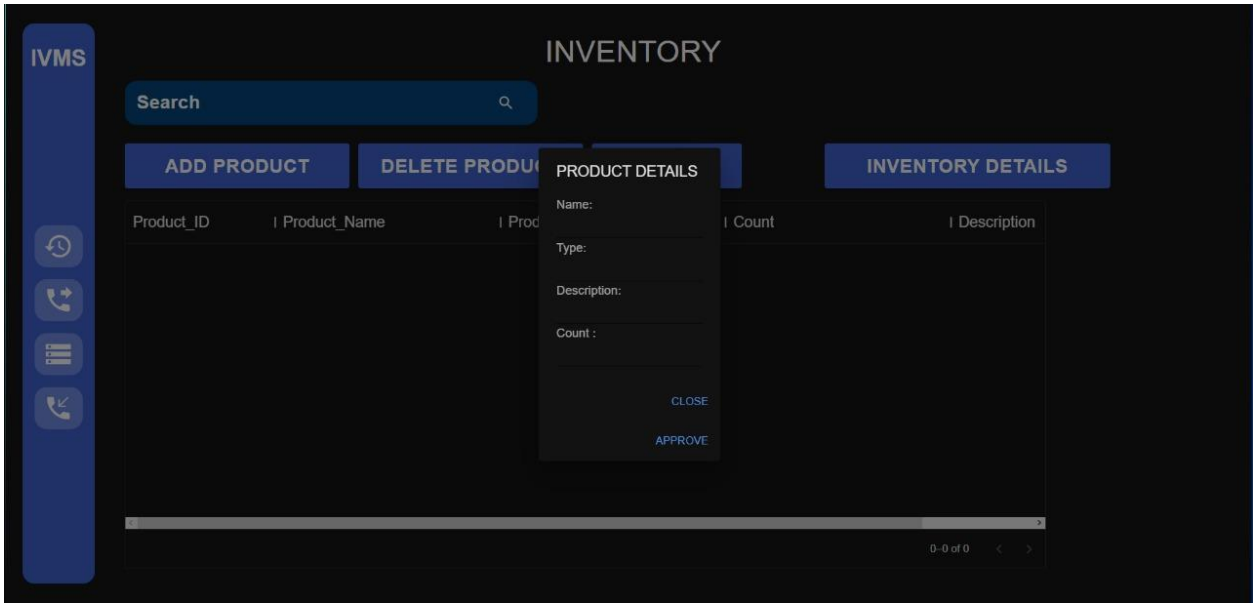

IVMS
PENDING APPROVALS
Company Name:
Message: Approve Retailer
APPROVE
REJECT
Company Name: adsa
Message: Approve Retailer
APPROVE
REJECT
Company Name: wahajStudios
Message: Approve Retailer
APPROVE
REJECT

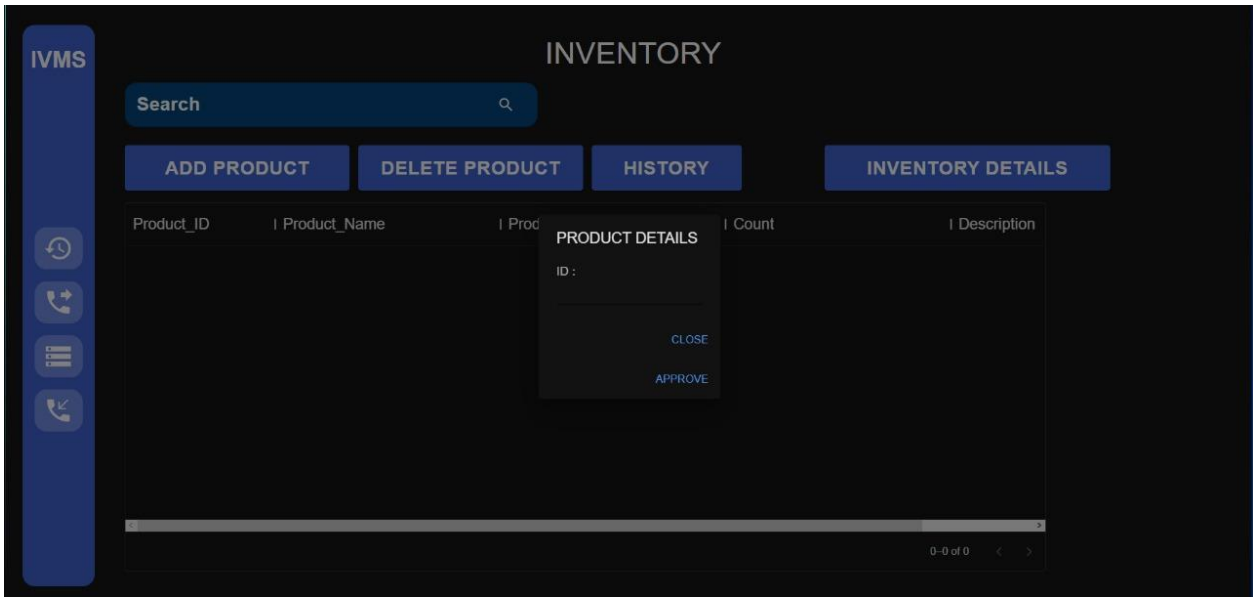

IVMS
RETAILER ACCESSES
Search
Retailer_ID
Retailer_Name
Mobile_Number
e84eb285-c683-41d6-8ba7-81625284f4af
m
0313100000
REVOKE ACCESS
CLOSE
REVOKE
1 row selected
1–1 of 1

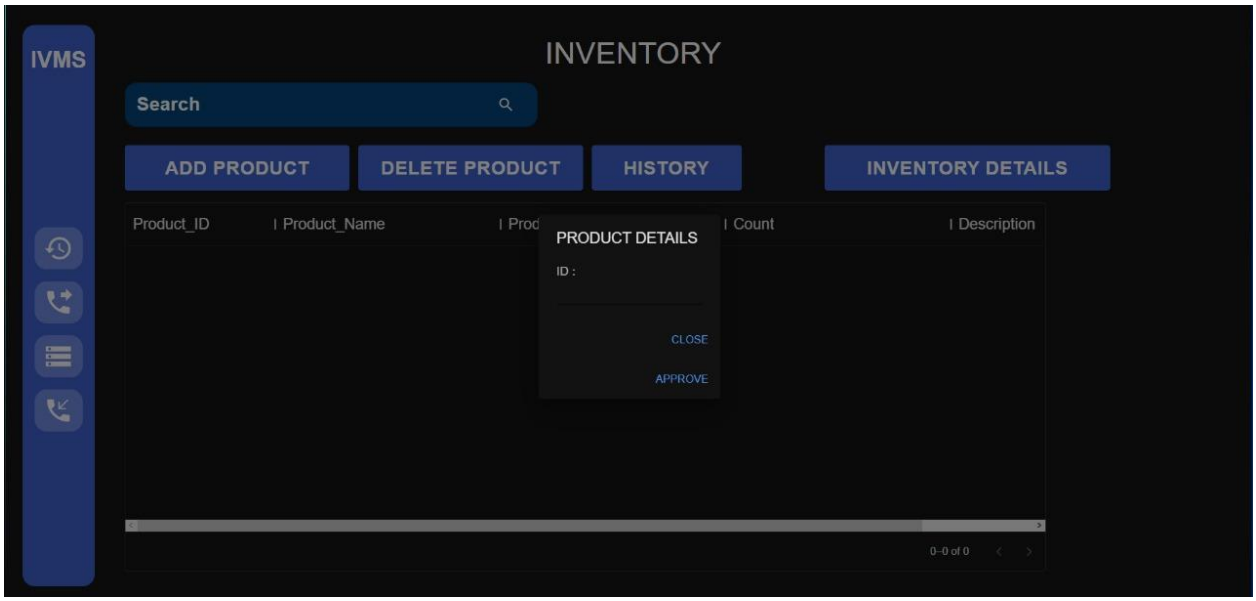

IVMS
LIST OF CLIENTS
Search
Retailer_ID
Retailer_Name
Inventory_ID
e84eb285-c683-41d6-8ba7-81625284f4af
m
0127f5d0-54fa-4965-ac0f-af61beaf4a1c
INVENTORY DETAILS
ID: 0127f5d0-54fa-4965-ac0f-af61beaf4a1c
TYPE: type
Count: 22
Max Count: 10000
Description: des
CLOSE
1 row selected
1–1 of 1

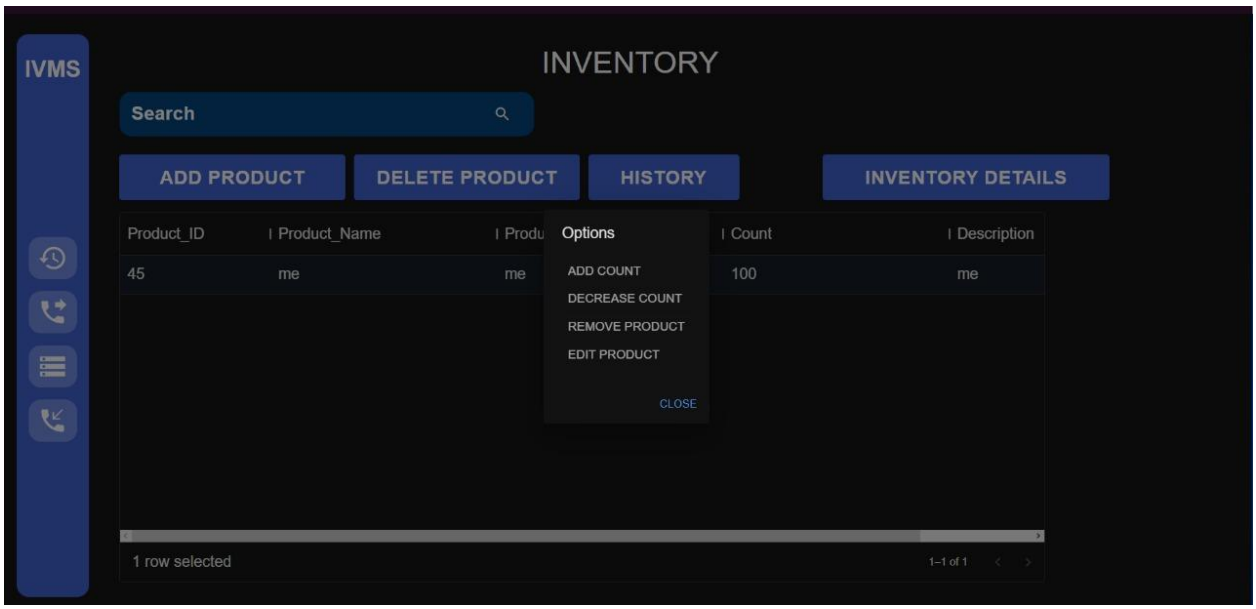

IVMS
MY PROFILE
Nov 11, 2022
Good Day, hateefa
Have a nice Monday
h
hateef
hateefa
karachi,pakistan
LOG OUT
HISTORY
OUT BOUND
INVENTORY
IN BOUND

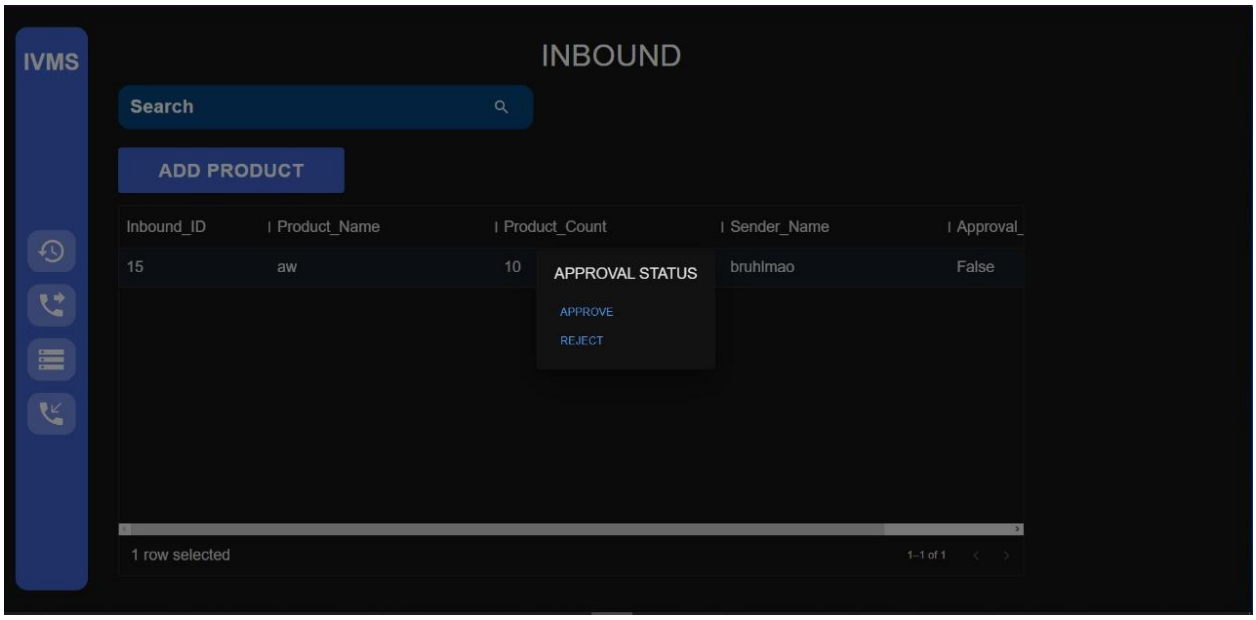

Inventory Details
Type
typeeee
Description
dessccription
UPDATE
CANCEL
Click to go back

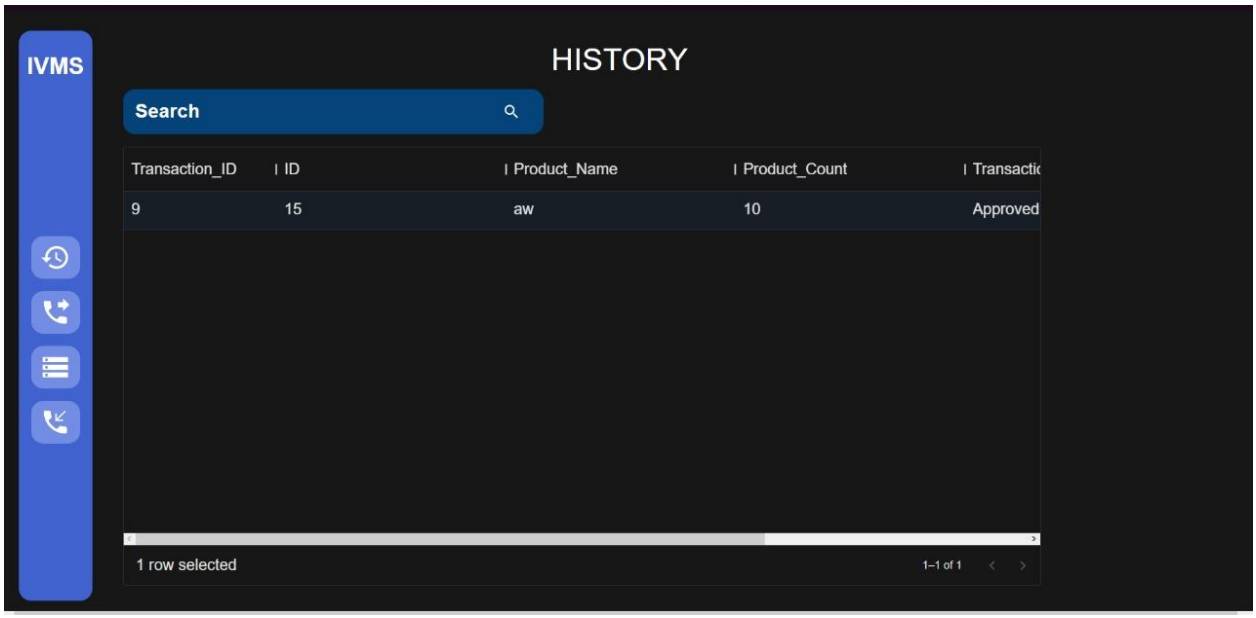

IVMS
INVENTORY
Search
ADD PRODUCT
DELETE PRODU
INVENTORY DETAILS
Product_ID
Product_Name
Count
Description
PRODUCT DETAILS
Name:
Type:
Description:
Count :
CLOSE
APPROVE
0-0 of 0

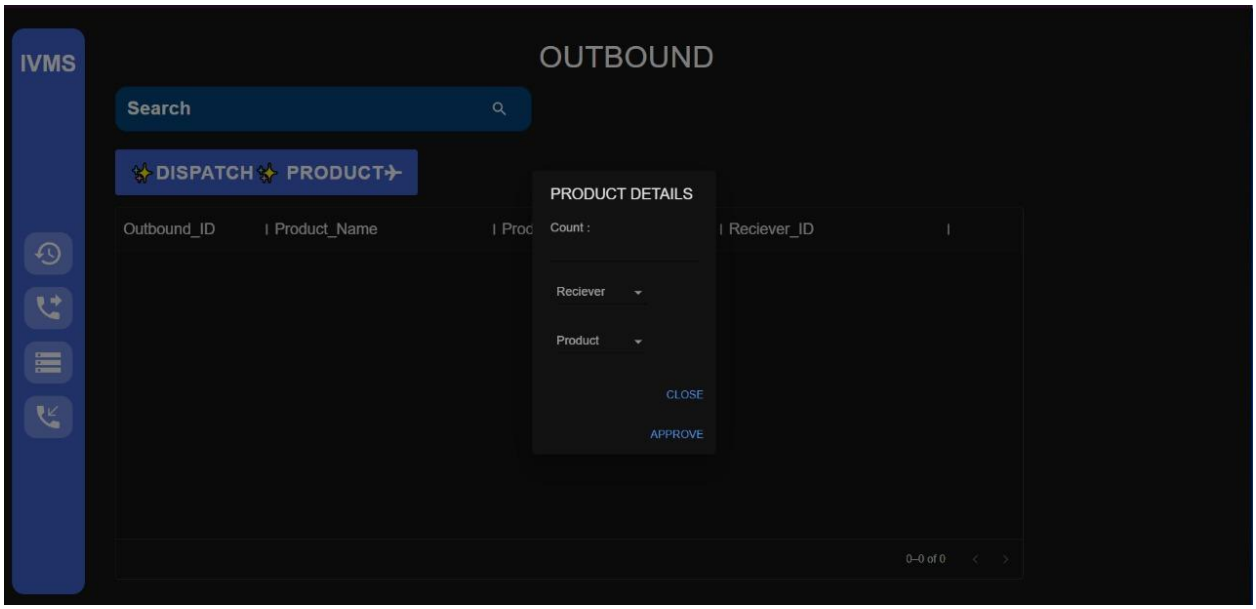

IVMS
INVENTORY
Search
ADD PRODUCT
DELETE PRODUCT
HISTORY
INVENTORY DETAILS
Product_ID
Product_Name
Prod
Count
Description
PRODUCT DETAILS
ID :
CLOSE
APPROVE
0–0 of 0

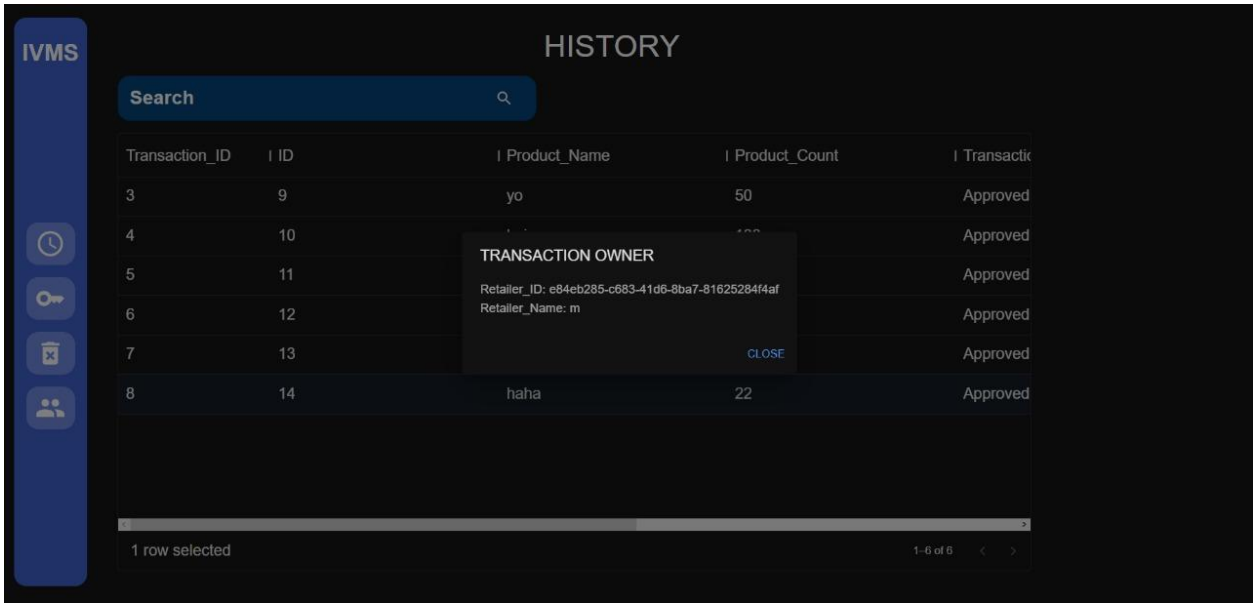

IVMS
INVENTORY
Search
ADD PRODUCT
DELETE PRODUCT
HISTORY
INVENTORY DETAILS
Product_ID
Product_Name
Prod
Count
Description
PRODUCT DETAILS
ID :
CLOSE
APPROVE
0–0 of 0


IVMS
INVENTORY
Search
ADD PRODUCT
DELETE PRODUCT
HISTORY
INVENTORY DETAILS
Product_ID
Product_Name
Produ
Count
Description
45
me
me
100
me
Options
ADD COUNT
DECREASE COUNT
REMOVE PRODUCT
EDIT PRODUCT
CLOSE
1 row selected
1–1 of 1


IVMS
INBOUND
Search
ADD PRODUCT
Inbound_ID
Product_Name
Product_Count
Sender_Name
Approval_
15
aw
10
bruhlmao
False
APPROVAL STATUS
APPROVE
REJECT
1 row selected
1–1 of 1


IVMS
HISTORY
Search
Transaction_ID
ID
Product_Name
Product_Count
Transactio
9
15
aw
10
Approved
1 row selected
1–1 of 1


IVMS
OUTBOUND
Search
DISPATCH
PRODUCT
Outbound_ID
Product_Name
Reciever_ID
PRODUCT DETAILS
Count :
Reciever
Product
CLOSE
APPROVE
0–0 of 0


IVMS
HISTORY
Search
Transaction_ID
ID
Product_Name
Product_Count
Transactio
3 9 yo 50 Approved
4 10 Approved
5 11 Approved
6 12 Approved
7 13 Approved
8 14 haha 22 Approved
TRANSACTION OWNER
Retailer_ID: e84eb285-c683-41d6-8ba7-81625284f4af
Retailer_Name: m
CLOSE
1 row selected
1–6 of 6