DBMS MINIPROJECT

Project Title: APMC Trader System

Group: G21
Name: Nischal H S
SRN: PES1UG21CS393

USER REQUIREMENT SPECIFICATION

Table of Contents

1. **Introduction**
   - Purpose of the project
   - Scope of the project
2. **Project Description**
   - Project overview
   - Major project functionalities
3. **System Features and Function Requirements**
   - User Authentication and Registration
   - Trader Profile Management
   - Employee Management
   - Transaction Recording
   - Inventory Management
   - Data Security

1. Introduction

Purpose of the Project

The purpose of the APMC Trader System is to provide a comprehensive software solution for Agricultural Produce Market Committee (APMC) traders to manage their trading operations efficiently. The system aims to streamline transactions, inventory management, and financial tracking, enhancing the trading experience for both traders and their stakeholders.

Scope of the Project

The scope of the APMC Trader System includes the following aspects:

- Managing trader profiles and employee details.
- Tracking transactions with suppliers and buyers.
- Recording item/commodity information and inventory.
- Handling payment methods and transaction status.
- Ensuring data security and user access control.

2. Project Description

Project Overview

The APMC Trader System is a web-based application designed to facilitate trading activities within APMC markets. It acts as a centralized platform for traders to conduct transactions, manage inventory, and monitor financial data. The system enables traders to efficiently handle various aspects of their business operations, including buying from suppliers and selling to buyers.

Major Project Functionalities

The major functionalities of the APMC Trader System include:

- User Management: Users can register and log in with appropriate roles (trader, clerk, etc.).
- Trader Profile: Traders can create and update their profiles, including contact information.
- Employee Management: Traders can manage employee details, including salary and roles.
- Transaction Management: Traders can record transactions, specifying items, quantities, prices, payment methods, and status.
- Inventory Tracking: The system allows traders to manage their inventory of items and commodities.
- Data Security: The system ensures data security through user access control and authentication mechanisms.

3. System Features and Function Requirements

System Feature 1: User Authentication and Registration

Functional Requirement: Users must be able to register for an account with role-based access (trader, clerk, etc.). Registered users should be able to log in securely.

System Feature 2: Trader Profile Management

Functional Requirement: Traders should be able to create and update their profiles, providing essential contact information.

System Feature 3: Employee Management

Functional Requirement: Traders must have the ability to manage employee details, including adding employees, specifying roles, and tracking salaries.

System Feature 4: Transaction Recording

Functional Requirement: Traders should be able to record transactions, including items, quantities, prices, payment methods, and transaction status.

System Feature 5: Inventory Management

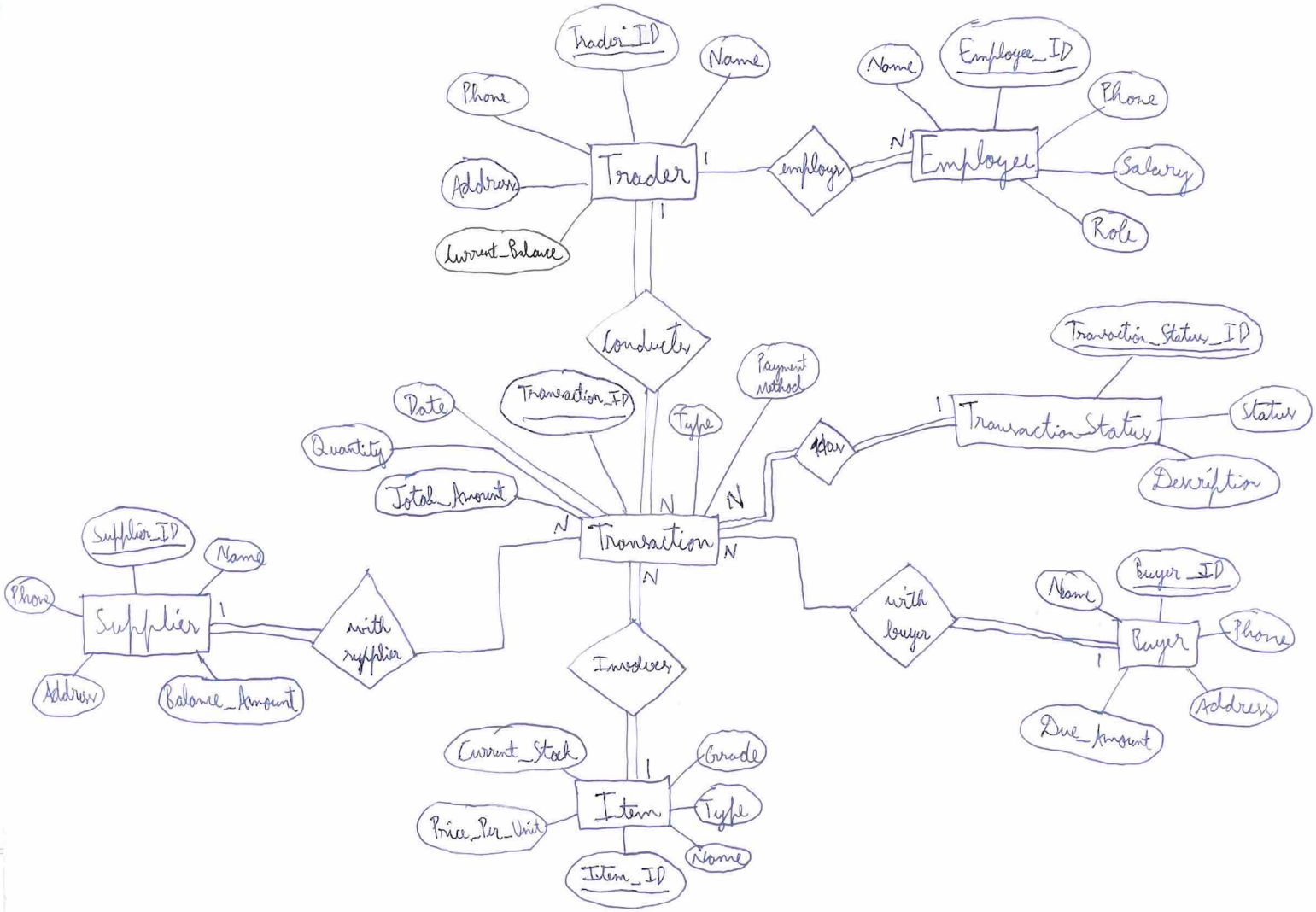
Functional Requirement: Traders need to manage their inventory of items and commodities, including quantity tracking.

System Feature 6: Data Security

Functional Requirement: Data security measures, including user access control and authentication, must be in place to safeguard sensitive information.

APMC Trader Management System - ER Diagram

Name: NISCHAL H.S.
SRN: PES1UG21CS393



APMC Trader Management System - Relational Schema

Name : Nishal H.S
SRN : PES1UG21CS39

Trader :

Trader_ID	Name	Address	Phone	Current_Balance

Employee :

Employee_ID	Name	Phone	Salary	Role	Trader_ID

Transaction :

Transaction_ID	Type	Date	Quantity	Total_Amount	Payment_Method

Trader_ID	Transaction_Status_ID	Item_ID

Transaction_Status :

Transaction_Status_ID	Status	Description

Supplier :

Supplier_ID	Name	Address	Phone	Balance_Amount

Buyer :

Buyer_ID	Name	Address	Phone	Due_Amount

Item :

Item_ID	Name	Type	Grade	Price_Per_Unit	Current_Stock

Seller_Transaction :

Transaction_ID	Supplier_ID

Buyer_Transaction :

Transaction_ID	Buyer_ID

Trigger

In this project trigger needs to be fired automatically when some cases of change to transaction status is made. The following code handles modifying transaction status(can be found on line 615 of app/utils.py):

```
    st.subheader("Modify Transaction Status")

        # Allow the user to select a transaction by Transaction ID
          selected_transaction_id = st.number_input("Select Transaction
ID", min_value=1, step=1)

        # Fetch the current status of the selected transaction
            current_status_query = f"SELECT Transaction_Status_ID FROM
Transaction WHERE Transaction_ID = {selected_transaction_id}"
                        current_status_id = fetch_data(connection,
current_status_query)

        if current_status_id:
            current_status_id = current_status_id[0][0]

            # Fetch the corresponding status string
                    current_status_string_query = f"SELECT status FROM
Transaction_Status WHERE Transaction_Status_ID = {current_status_id}"
                        current_status_string = fetch_data(connection,
current_status_string_query)[0][0]

                            st.info(f"Current Status of Transaction
{selected_transaction_id}: {current_status_string}")

               # Define the possible status modifications based on the
current status
            if current_status_string == "initialised":
                        new_status_options = ["cancelled", "fulfilled",
"completed"]
            elif current_status_string == "fulfilled":
                new_status_options = ["completed"]
            else:
                new_status_options = []

            if new_status_options:
                      new_status = st.selectbox("Select New Transaction
Status", new_status_options)
```

```
                if st.button("Modify Transaction Status"):
                            # Perform the database update for the new
transaction status
                    update_transaction_status_query = f"""
                        UPDATE Transaction
                                                SET  Transaction_Status_ID  =
{fetch_transaction_status_id(new_status)}
                                                    WHERE  Transaction_ID  =
{selected_transaction_id};
                    """
                                                execute_query(connection,
update_transaction_status_query)

                      st.success(f"Transaction {selected_transaction_id}
status modified to {new_status} successfully!")
            else:
                  st.warning("Cannot modify the transaction status based
on the current status.")
        else:
               st.warning("Invalid Transaction ID. Please enter a valid
Transaction ID.")
```

One example is when the transaction status is changed from fulfilled to completed, in that case the following trigger query is automatically fired:

```
CREATE TRIGGER trg_transaction_fulfilled_completed
AFTER UPDATE ON Transaction
FOR EACH ROW
BEGIN
     IF NEW.Transaction_Status_ID = (SELECT Transaction_Status_ID FROM
Transaction_Status WHERE status = 'completed') AND
        OLD.Transaction_Status_ID = (SELECT Transaction_Status_ID FROM
Transaction_Status WHERE status = 'fulfilled') THEN

        IF NEW.type = 'With Supplier' THEN
            UPDATE Supplier
            SET balance_amount = balance_amount - NEW.total_amount
                        WHERE  Supplier_ID  =  (SELECT  Seller_ID  FROM
Seller_Transaction WHERE Transaction_ID = NEW.Transaction_ID);

            UPDATE Trader
```

```
                SET current_holdings_amount = current_holdings_amount -
NEW.total_amount
            WHERE Trader_ID = NEW.Trader_ID;
        ELSE
            UPDATE Buyer
            SET balance_amount = balance_amount - NEW.total_amount
               WHERE Buyer_ID = (SELECT Buyer_ID FROM Buyer_Transaction
WHERE Transaction_ID = NEW.Transaction_ID);

            UPDATE Trader
                SET current_holdings_amount = current_holdings_amount +
NEW.total_amount
            WHERE Trader_ID = NEW.Trader_ID;
        END IF;
    END IF;
END;
```

Procedure

Procedure has been used to retrieve total salary of employees under the trader:

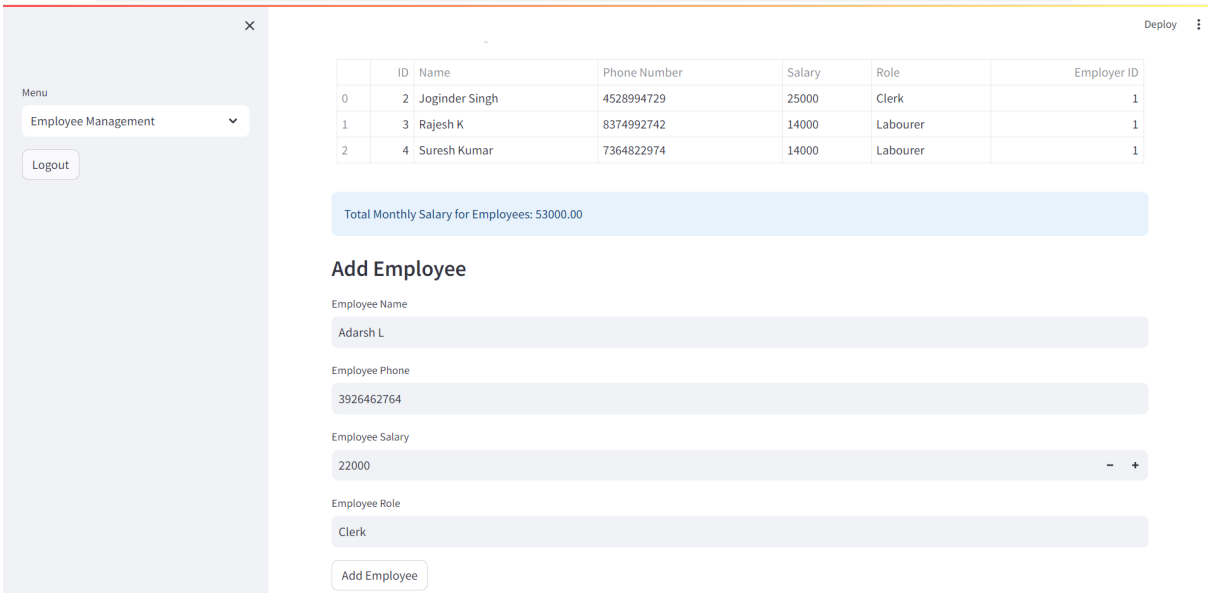
```
DELIMITER //

CREATE PROCEDURE get_employee_payment(trader_id INT)
BEGIN
    SELECT SUM(salary) AS total_payment
    FROM Employee
    WHERE Trader_ID = trader_id;
END;
//

DELIMITER ;
```

Queries – any 4 sample queries

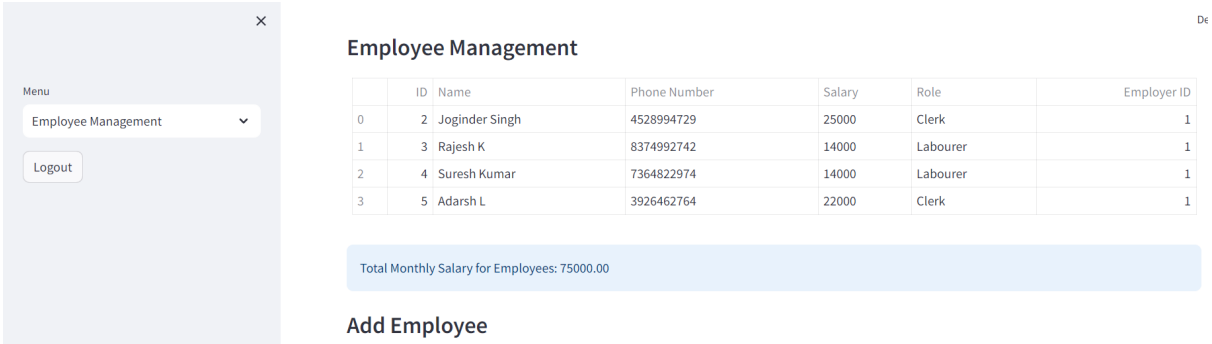
1. Add an employee
   Query:
   ```
   INSERT INTO Employee (name, phone, salary, role, Trader_ID)
                                    VALUES  ('{new_employee_name}',
   '{new_employee_phone}',                   {new_employee_salary},
   '{new_employee_role}', {trader_id});
   ```

Output:

Before Adding



After Adding



2. View Supplier Transactions

Query:

```
SELECT t.Transaction_ID, t.type, t.date, t.quantity, t.total_amount, t.payment_method,
                ts.status AS transaction_status, i.name AS item_name
            FROM Transaction t
             JOIN Transaction_Status ts ON t.Transaction_Status_ID = ts.Transaction_Status_ID
            JOIN Item i ON t.Item_ID = i.Item_ID
            WHERE t.Transaction_ID IN (
                    SELECT Transaction_ID FROM Seller_Transaction WHERE Seller_ID = {entity_id}
            )
```

Output:

Select Supplier

Shivaram Gowda (ID: 1)

Supplier Profile:

	Supplier_ID	Name	Address	Phone
0	1	Shivaram Gowda	Hoskote	2974382975

	Transaction_ID	type	date	quantity	total_amount	payment_method	transaction_status	item_name
0	2	With Supplier	2023-11-23	100	3200	UPI	completed	Carrot
1	3	With Supplier	2023-11-23	200	5000	Cash	completed	Carrot
2	4	With Supplier	2023-11-23	300	6999.99	Debit Card	fulfilled	Carrot
3	5	With Supplier	2023-11-23	200	4000	Cash	completed	Carrot

3. Edit trader profile

Query:

```
UPDATE Trader
            SET name = '{new_name}',
                address = '{new_address}',
                phone = '{new_phone}',
                password = '{new_password}'
            WHERE Trader_ID = {trader_id};
```

Output:

Before Update

	Detail	Value
0	Trader_ID	1
1	Name	Ramesh
2	Address	Kalasipalya Market
3	Phone	9999999999
4	Current Holdings Amount	987800.00
5	Password	********

Edit Trader Details

New Name

Ramesh A

New Address

Kalasipalya Market

New Phone

9999999999

New Password

••••••••

Update Details

After Update

Deploy

	Detail	Value
0	Trader_ID	1
1	Name	Ramesh A
2	Address	Kalasipalya Market
3	Phone	9999999999
4	Current Holdings Amount	987800.00
5	Password	********

Edit Trader Details

New Name

Ramesh A

New Address

Kalasipalya Market

New Phone

9999999999

New Password

••••••••

Update Details

4. Modify Item unit price:

Query:

```
UPDATE Item SET price_per_unit = {new_price} WHERE Item_ID =
{item_id};
```

Output:

Before modification

APMC Trader System

Inventory Management

All Items:

	Item_ID	Name	Type	Grade	Price_per_Unit	Current_Stock
0	1	Carrot	Ooty	Super	35	100
1	2	Carrot	Ooty	Good	24	500
2	3	Carrot	Ooty	Average	18	200

Update Item Price

Item ID to Update

1

Updated Price per Unit

38

Press Enter to apply

Update Item Price

After Modification

Inventory Management

All Items:

	Item_ID	Name	Type	Grade	Price_per_Unit	Current_Stock
0	1	Carrot	Ooty	Super	38	100
1	2	Carrot	Ooty	Good	24	500
2	3	Carrot	Ooty	Average	18	200

Update Item Price

Item ID to Update

1

Updated Price per Unit

38.00

Update Item Price

Item Price Updated Successfully!