Database Management System

Project

Farm Management System

Name: RAGHAVENDRA
SRN: PES1UG21CS470

Name: REEHAN SHAVEEZ
SRN: PES1UG21CS484

Abstract :

The Farm Management System is a comprehensive web-based platform designed to enhance agricultural productivity and streamline farm operations. With a focus on user-friendly features, the system provides farmers with tools to manage key aspects of their agricultural enterprise.

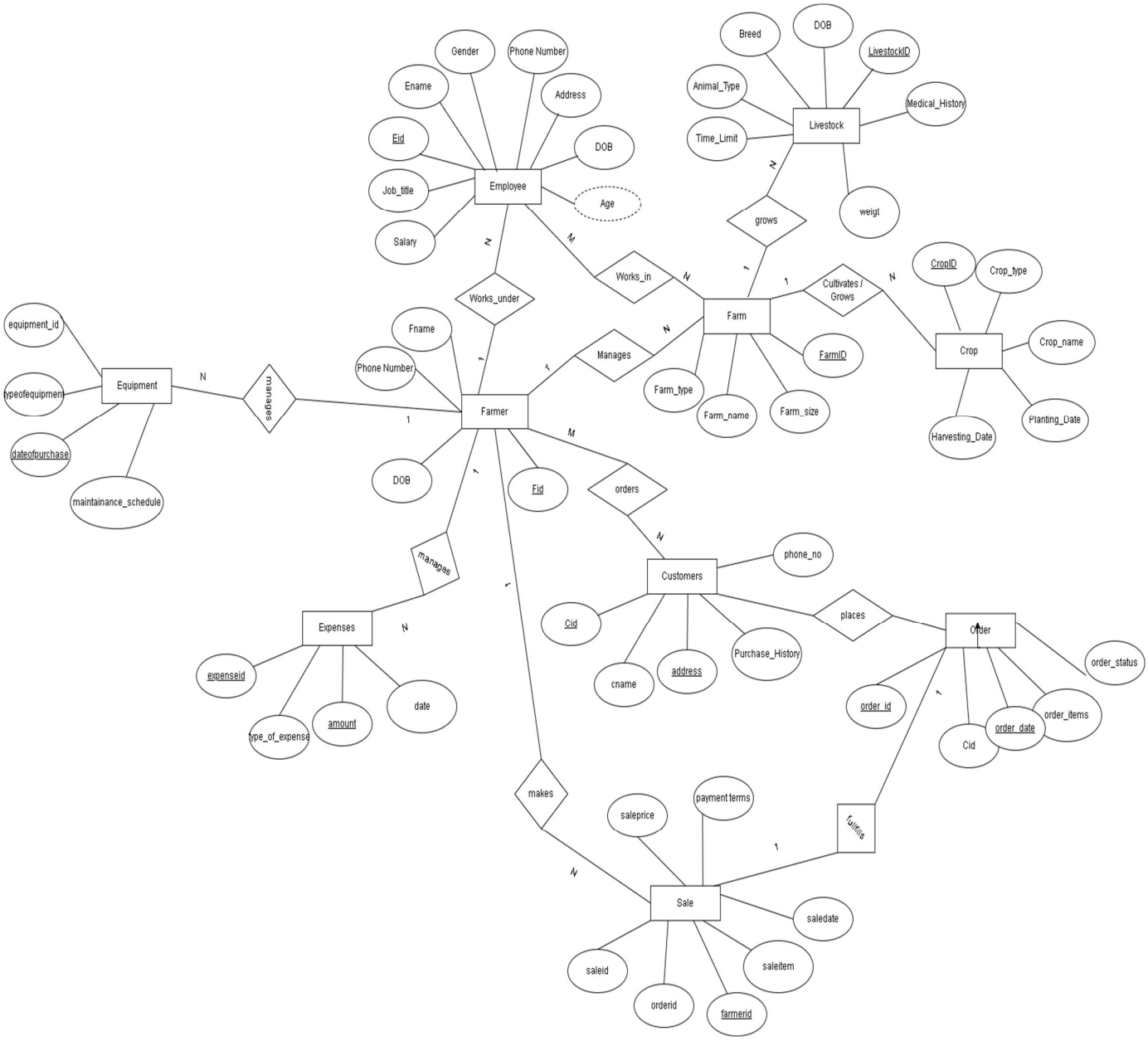
In the realm of crop management, the system allows farmers to track planting and harvesting dates, expected yields, and seasonal patterns. Livestock management features enable monitoring of animal health, breeding cycles, and inventory. Additionally, the platform facilitates efficient equipment management, keeping tabs on maintenance schedules, usage patterns, and procurement details.

The system also offers robust sales and order management capabilities, allowing farmers to track and analyze crop and livestock sales. Employee management features provide insights into workforce details, salaries, and departmental assignments.

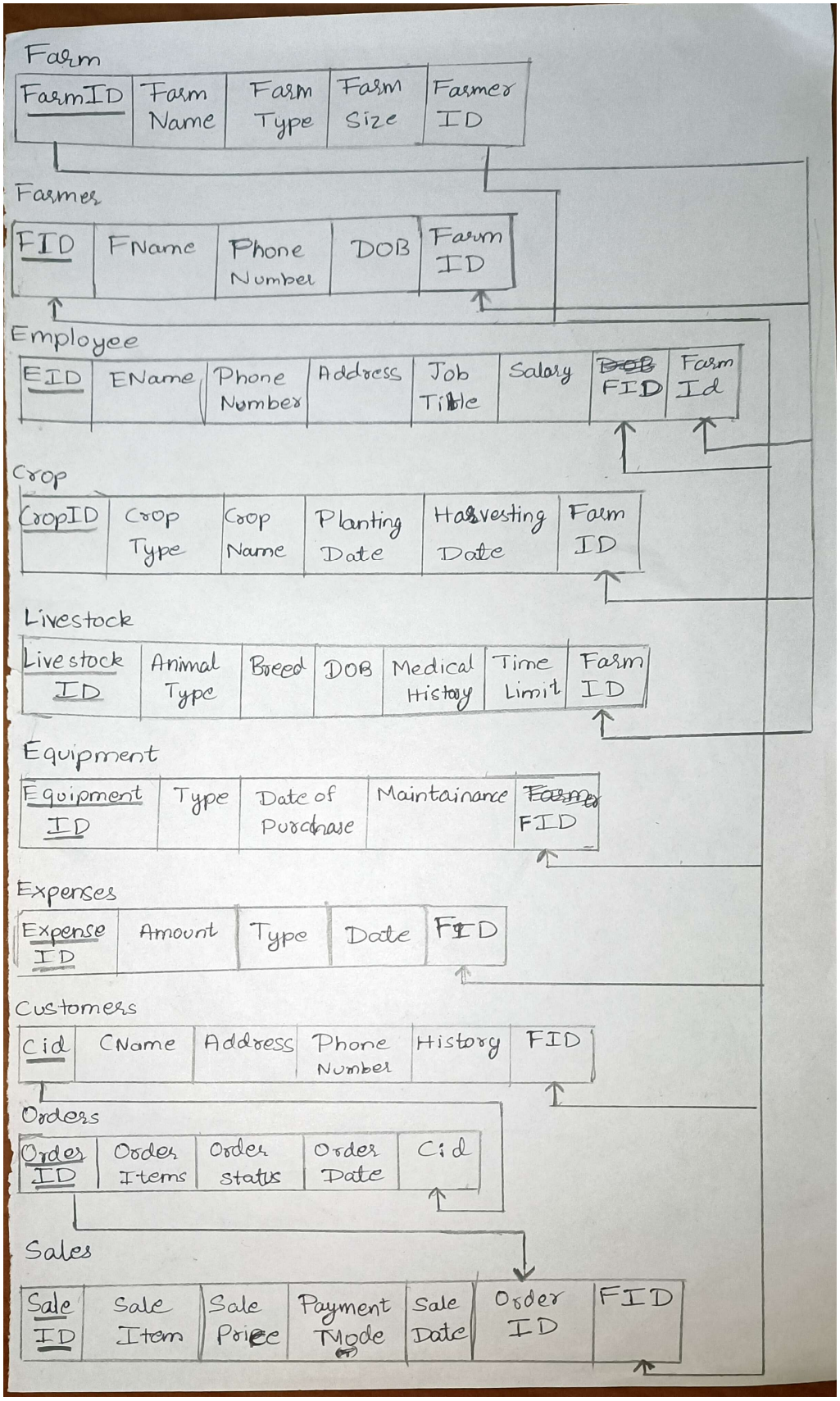
Through a intuitive user interface, farmers can easily add or remove crops, livestock, equipment, and employees. The platform ensures data security and accessibility, allowing farmers to make informed decisions based on real-time insights.

In summary, the Farm Management System serves as a centralized hub for farmers, fostering precision farming, data-driven decision-making, and overall operational efficiency in the agricultural domain.

ER Diagram:


Gender
Phone Number
Ename
Address
Eid
DOB
Job_title
Employee
Age
Salary
Breed
DOB
LivestockID
Animal_Type
Medical_History
Time_Limit
Livestock
weigt
grows
Works_in
Works_under
Farm
Cultivates / Grows
CropID
Crop_type
Crop_name
Crop
Planting_Date
Harvesting_Date
FarmID
Farm_type
Farm_name
Farm_size
equipment_id
typeofequipment
dateofpurchase
maintainance_schedule
Equipment
manages
Fname
Phone Number
Manages
Farmer
DOB
Fid
orders
manages
Expenses
expenseid
type_of_expense
amount
date
Customers
phone_no
Cid
cname
address
Purchase_History
places
Order
order_status
order_id
Cid
order_date
order_items
makes
saleprice
payment terms
fulfills
Sale
saledate
saleid
orderid
farmerid
saleitem

Relational schema:



DDL SQL Commands:

```
CREATE TABLE farmers (
  farmer_id INT NOT NULL AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL,
  phone_number VARCHAR(255) NOT NULL,
  dob DATE NOT NULL,
  PRIMARY KEY (farmer_id)
);

CREATE TABLE farms (
  farm_id INT NOT NULL AUTO_INCREMENT,
  farmer_id INT NOT NULL,
  name VARCHAR(255) NOT NULL,
  location VARCHAR(255) NOT NULL,
  size DECIMAL(10,2) NOT NULL,
  type_of_farming ENUM('conventional', 'organic', 'sustainable') NOT
NULL,

  PRIMARY KEY (farm_id),
  FOREIGN KEY (farmer_id) REFERENCES farmers(farmer_id)
);



CREATE TABLE crops (
  crop_id INT NOT NULL AUTO_INCREMENT,
  farm_id INT NOT NULL,
  variety VARCHAR(255) NOT NULL,
  planting_date DATE NOT NULL,
  harvesting_date DATE NOT NULL,
  expected_yield DECIMAL(10,2) NOT NULL,

  PRIMARY KEY (crop_id),
  FOREIGN KEY (farm_id) REFERENCES farms(farm_id)
);
```

```
CREATE TABLE livestock (
  livestock_id INT NOT NULL AUTO_INCREMENT,
  farm_id INT NOT NULL,
  breed VARCHAR(255) NOT NULL,
  age INT NOT NULL,
  weight DECIMAL(10,2) NOT NULL,
  health_status ENUM('healthy', 'sick', 'injured') NOT NULL,

  PRIMARY KEY (livestock_id),
  FOREIGN KEY (farm_id) REFERENCES farms(farm_id)
);

CREATE TABLE customers (
  customer_id INT NOT NULL AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL,
  address VARCHAR(255) NOT NULL,
  phone_number VARCHAR(255) NOT NULL,
  email_address VARCHAR(255) NOT NULL,

  PRIMARY KEY (customer_id)
);

CREATE TABLE orders (
  order_id INT NOT NULL AUTO_INCREMENT,
  farmer_id INT NOT NULL,
  customer_id INT NOT NULL,
  order_date DATE NOT NULL,
  item_cropid INT,
  item_livestockid INT,
  quantity_crop INT,
  quantity_livestock INT,

  PRIMARY KEY (order_id),
  FOREIGN KEY (farmer_id) REFERENCES farmers(farmer_id),
  FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
  FOREIGN KEY (item_cropid) REFERENCES crops(crop_id),
  FOREIGN KEY (item_livestockid) REFERENCES livestock(livestock_id)
);
```

```
CREATE TABLE crop_sales (
  sale_id INT NOT NULL AUTO_INCREMENT,
  sale_date DATE NOT NULL,
  customer_id INT NOT NULL,
  sale_amount DECIMAL(10,2) NOT NULL,
  sale_status ENUM('open', 'closed', 'canceled') NOT NULL,
  crop_id INT NOT NULL,
  quantity INT NOT NULL,
  unit_price DECIMAL(10,2) NOT NULL,
  PRIMARY KEY (sale_id),
  FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
  FOREIGN KEY (crop_id) REFERENCES crops(crop_id)
);

CREATE TABLE livestock_sales (
  sale_id INT NOT NULL AUTO_INCREMENT,
  sale_date DATE NOT NULL,
  customer_id INT NOT NULL,
  sale_amount DECIMAL(10,2) NOT NULL,
  sale_status ENUM('open', 'closed', 'canceled') NOT NULL,
  livestock_id INT NOT NULL,
  quantity INT NOT NULL,
  unit_price DECIMAL(10,2) NOT NULL,
  PRIMARY KEY (sale_id),
  FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
  FOREIGN KEY (livestock_id) REFERENCES livestock(livestock_id)
);


CREATE TABLE expenses (
  expense_id INT NOT NULL AUTO_INCREMENT,
  farmer_id INT NOT NULL,
  expense_date DATE NOT NULL,
  expense_type ENUM('feed', 'supplies', 'equipment', 'labor','others') NOT
NULL,
  expense_amount DECIMAL(10,2) NOT NULL,

  PRIMARY KEY (expense_id),
  FOREIGN KEY (farmer_id) REFERENCES farmers(farmer_id)
```

```
);

CREATE TABLE equipment (
  equipment_id INT NOT NULL AUTO_INCREMENT,
  farmer_id INT NOT NULL,
  equipment_type ENUM('tractor', 'combine', 'harvester','others') NOT
NULL,
  equipment_brand VARCHAR(255) NOT NULL,
  equipment_model VARCHAR(255) NOT NULL,
  equipment_year INT NOT NULL,

  PRIMARY KEY (equipment_id),
  FOREIGN KEY (farmer_id) REFERENCES farmers(farmer_id)
);
CREATE TABLE employees (
  employee_id INT NOT NULL AUTO_INCREMENT,
  farmer_id INT NOT NULL,
  name VARCHAR(255) NOT NULL,
  address VARCHAR(255) NOT NULL,
  phone_number VARCHAR(255) NOT NULL,
  hire_date DATE NOT NULL,
  position ENUM('manager', 'supervisor', 'worker') NOT NULL,
  salary DECIMAL(10,2) NOT NULL,
  department ENUM('farming', 'sales', 'administration') NOT NULL,
  expected_date_of_salary DATE NOT NULL,
  PRIMARY KEY (employee_id),
  FOREIGN KEY (farmer_id) REFERENCES farmers(farmer_id)
);
```

Tables is in 3NF:

The provided tables, namely State, Infrastructure, HDI, TouristAttraction, AdministrativeOffice, and StateGovernmentOfficial, are designed in a way that adheres to the Third Normal Form (3NF) of database normalization.

First Normal Form (1NF):

All tables have atomic values in each column, unique column names, and the order of data storage is insignificant.

Second Normal Form (2NF):

Each table is in 1NF, and all non-key attributes are fully functionally dependent on their respective primary keys.

Third Normal Form (3NF):

Each table is in 3NF, and there are no transitive dependencies. All non-key attributes are directly dependent on their respective primary keys.

It satisfies the principles of normalization up to the Third Normal Form, which promotes data integrity and reduces redundancy in the database design.

Loading SQL Data:

```
-- For farmers table
LOAD DATA INFILE 'D:/farmers.txt' INTO TABLE farmers
FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n';

-- For farms table
LOAD DATA INFILE 'D:/farms.txt' INTO TABLE farms
FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n';

-- For crops table
LOAD DATA INFILE 'D:/crops.txt' INTO TABLE crops
FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n';

-- For livestock table
LOAD DATA INFILE 'D:/livestock.txt' INTO TABLE livestock
FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n';

-- For customers table
LOAD DATA INFILE 'D:/customers.txt' INTO TABLE customers
FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n';

-- For orders table
LOAD DATA INFILE 'D:/orders.txt' INTO TABLE orders
FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n';

-- For crop_sales table
LOAD DATA INFILE 'D:/crop_sales.txt' INTO TABLE crop_sales
FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n';

-- For livestock_sales table
LOAD DATA INFILE 'D:/livestock_sales.txt' INTO TABLE livestock_sales
FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n';

-- For expenses table
LOAD DATA INFILE 'D:/expenses.txt' INTO TABLE expenses
FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n';

-- For equipment table
```

LOAD DATA INFILE 'D:/equipment.txt' INTO TABLE equipment
FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n';

-- For employees table
LOAD DATA INFILE 'D:/employees.txt' INTO TABLE employees
FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n';

CRUD Operations:

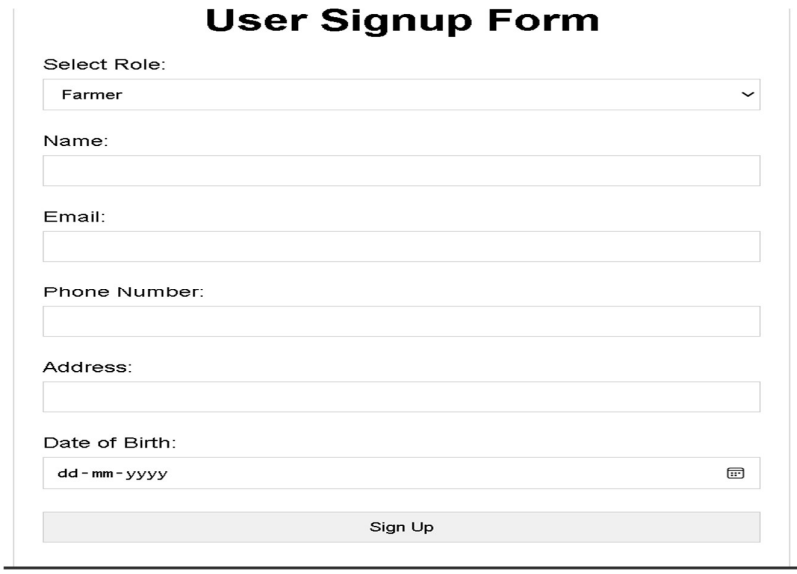
```python
@app.route('/signup',methods=['GET','POST'])
def signup():
    if request.method == 'POST':
        role = request.form['role']
        name = request.form['name']
        user_data = validate_user_credentials(role,name)
        if not user_data:
            db = pymysql.connect(**db_config)
            cursor = db.cursor()

            phone_number = request.form['phone_number']
            if role == "farmers":
                dob = request.form['dob']
                insert_query = "INSERT INTO farmers (name, phone_number, dob) VALUES (%s, %s, %s)"
                cursor.execute(insert_query, (name, phone_number, dob))
                db.commit()

            elif role == "customers":
                address = request.form['address']
                email_address = request.form['email']
                insert_query = "INSERT INTO customers (name, address, phone_number,email_address) VALUES (%s, %s, %s,%s)"
                cursor.execute(insert_query, (name, address, phone_number,email_address))
                db.commit()

            cursor.close()
            db.close()

        return redirect('/login')
    return render_template('signup.html')
```

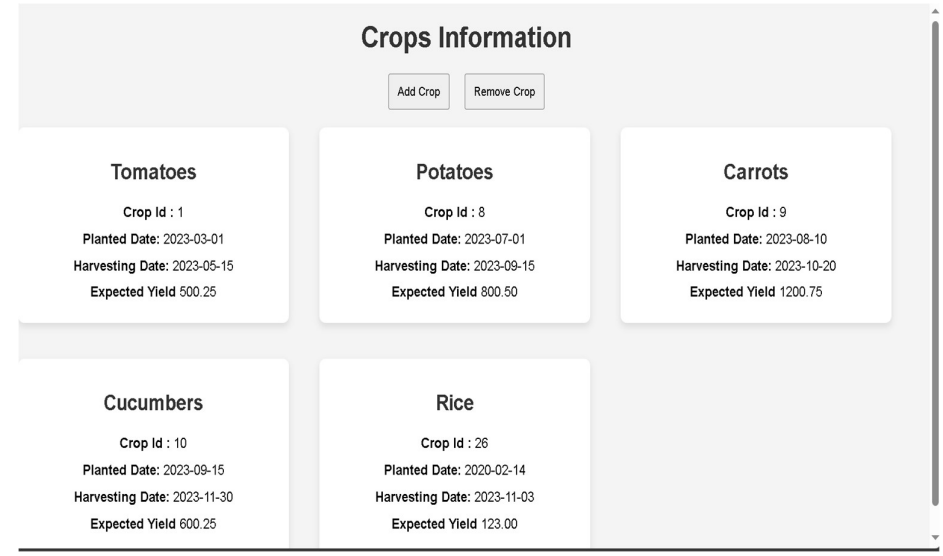
```python
#newly added code here
@app.route('/crops')
def crops():

    user_data = session.get('user_data')
    print(user_data)
    farmer_id = int(user_data[0])

    db = pymysql.connect(**db_config)
    cursor = db.cursor()
    # Fetch data from the database for Livestock
    cursor.execute('CALL GetFarmerCrops(%s)',(farmer_id))
    crop_data = cursor.fetchall()

    cursor.close()
    db.close()

    return render_template('crop.html', crop_data = crop_data)


@app.route('/livestock')
def livestock():
    user_data = session.get('user_data')
    farmer_id = int(user_data[0])

    db = pymysql.connect(**db_config)
    cursor = db.cursor()
    # Fetch data from the database for Livestock
    cursor.execute('CALL GetLivestockByFarmerId(%s)',(farmer_id))
    livestock_data = cursor.fetchall()
    cursor.close()
    db.close()
```

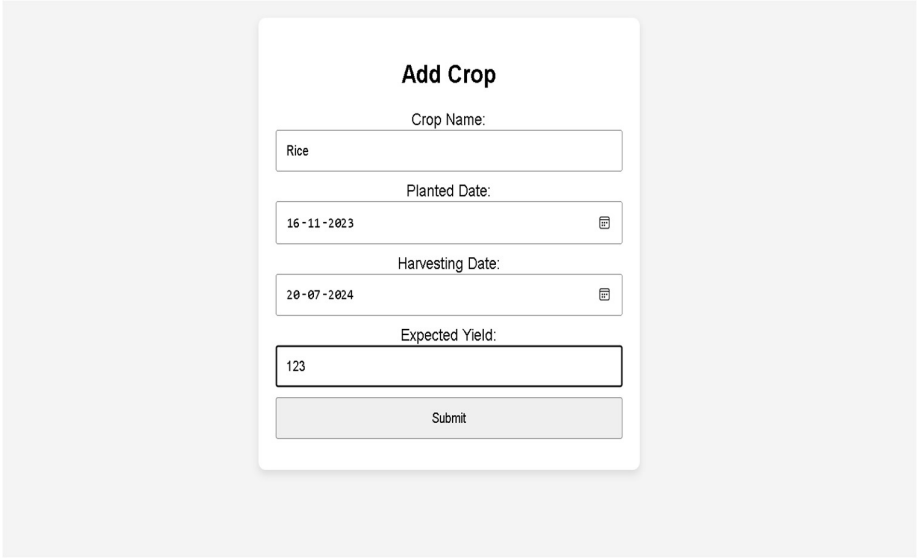
```python
@app.route('/expenses')
def expenses():
    user_data = session.get('user_data')
    farmer_id = int(user_data[0])
    db = pymysql.connect(**db_config)
    cursor = db.cursor()
    # Fetch data from the database for Orders
    cursor.execute('SELECT * from expenses where farmer_id = (%s)',(farmer_id))
    expenses_data = cursor.fetchall()
    cursor.close()
    db.close()

    return render_template('expenses.html', expenses_data=expenses_data)

@app.route('/addCrop',methods=['GET','POST'])
def addCrop():
    if request.method == 'POST':
        user_data = session.get('user_data')
        farm_id = int(user_data[0])
        variety = request.form['cropName']
        planting_date = request.form['plantedDate']
        harvesting_date = request.form['harvestingDate']
        expected_yield = request.form['expected_yield']
        db = pymysql.connect(**db_config)
        cursor = db.cursor()
        cursor.execute('INSERT INTO crops(farm_id, variety, planting_date, harvesting_date, expected_yield) VALUES (%s,%s,%s,%s,%s)',(
        db.commit()
```

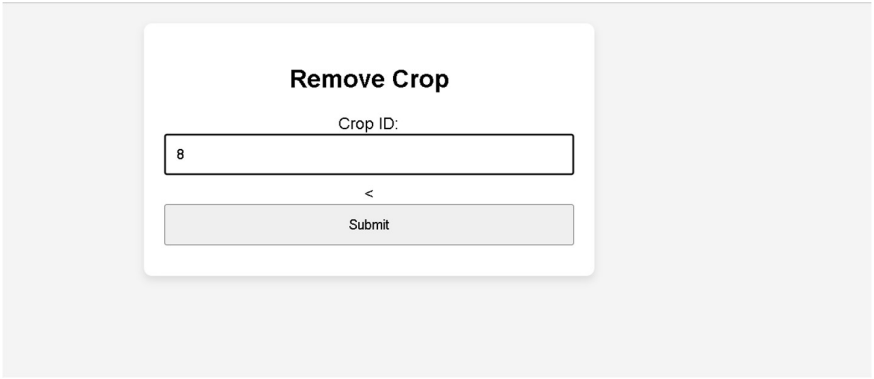
```python
@app.route('/removeCrop',methods=['GET','POST'])
def removeCrop():
    if request.method == 'POST':
        user_data = session.get('user_data')
        farm_id = int(user_data[0])
        id = int(request.form['cropID'])
        print(id)
        db = pymysql.connect(**db_config)
        cursor = db.cursor()
        cursor.execute('DELETE FROM crops WHERE crop_id = %s and farm_id = %s', (id,farm_id))
        db.commit()
        cursor.close()
        db.close()
        return redirect('crops')
    return render_template('removecrop.html')
```

Output of CRUD:









Procedures, Functions and Views:

```
DELIMITER //
CREATE PROCEDURE AddCrop(IN farmId INT, IN variety VARCHAR(255), IN plantingDate
DATE, IN harvestingDate DATE, IN expectedYield DECIMAL(10,2))
BEGIN
   INSERT INTO crops (farm_id, variety, planting_date, harvesting_date, expected_yield)
   VALUES (farmId, variety, plantingDate, harvestingDate, expectedYield);
END //
DELIMITER ;
DELIMITER //

CREATE PROCEDURE RemoveCropById(IN p_crop_id INT)
BEGIN
   DELETE FROM crops WHERE crop_id = p_crop_id;
END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE AddLivestock(
   IN p_farm_id INT,
   IN p_breed VARCHAR(255),
   IN p_age INT,
   IN p_weight DECIMAL(10,2),
   IN p_health_status ENUM('healthy', 'sick', 'injured')
)
BEGIN
   INSERT INTO livestock (farm_id, breed, age, weight, health_status)
   VALUES (p_farm_id, p_breed, p_age, p_weight, p_health_status);
END //

DELIMITER ;
DELIMITER //

CREATE PROCEDURE RemoveLivestockById(IN p_livestock_id INT)
BEGIN
   DELETE FROM livestock WHERE livestock_id = p_livestock_id;
END //

DELIMITER ;
DELIMITER //
```

```
CREATE PROCEDURE AddEmployee(
    IN p_farmer_id INT,
    IN p_name VARCHAR(255),
    IN p_address VARCHAR(255),
    IN p_phone_number VARCHAR(255),
    IN p_hire_date DATE,
    IN p_position ENUM('manager', 'supervisor', 'worker'),
    IN p_salary DECIMAL(10,2),
    IN p_department ENUM('farming', 'sales', 'administration'),
    IN p_expected_date_of_salary DATE
)
BEGIN
    INSERT INTO employees (farmer_id, name, address, phone_number, hire_date, position,
salary, department, expected_date_of_salary)
    VALUES (p_farmer_id, p_name, p_address, p_phone_number, p_hire_date, p_position,
p_salary, p_department, p_expected_date_of_salary);
END //

DELIMITER ;
DELIMITER //

CREATE PROCEDURE RemoveEmployeeById(IN p_employee_id INT)
BEGIN
    DELETE FROM employees WHERE employee_id = p_employee_id;
END //

DELIMITER ;


DELIMITER //
CREATE PROCEDURE GetFarmerCrops(IN farmerId INT)
BEGIN
    SELECT * FROM crops
    WHERE farm_id IN (SELECT farm_id FROM farms WHERE farmer_id = farmerId);
END //
DELIMITER ;

DELIMITER //
CREATE FUNCTION CalculateTotalSales(customerId INT) RETURNS DECIMAL(10,2)
READS SQL DATA
BEGIN
    DECLARE totalSales DECIMAL(10,2);
    SELECT SUM(sale_amount) INTO totalSales
    FROM crop_sales
    WHERE customer_id = customerId;

    RETURN totalSales;
END //
DELIMITER ;

DELIMITER //

CREATE PROCEDURE GetLivestockByFarmerId(IN p_farmerId INT)
BEGIN
    SELECT *
    FROM livestock
    WHERE farm_id IN (SELECT farm_id FROM farms WHERE farmer_id = p_farmerId);
END //
```

```
DELIMITER ;
DELIMITER //

CREATE PROCEDURE GetEquipmentByFarmerId(IN p_farmerId INT)
BEGIN
   SELECT *
   FROM equipment
   WHERE farmer_id = p_farmerId;
END //

DELIMITER ;
DELIMITER //

CREATE PROCEDURE GetOrdersByFarmerId(IN p_farmerId INT)
BEGIN
   SELECT *
   FROM orders
   WHERE farmer_id = p_farmerId;
END //

DELIMITER ;
DELIMITER //

CREATE PROCEDURE GetSalesByFarmerId(IN p_farmerId INT)
BEGIN
   SELECT *
   FROM crop_sales
   WHERE crop_id IN (SELECT crop_id FROM crops WHERE farm_id IN (SELECT farm_id
FROM farms WHERE farmer_id = p_farmerId))
   UNION
   SELECT *
   FROM livestock_sales
   WHERE livestock_id IN (SELECT livestock_id FROM livestock WHERE farm_id IN
(SELECT farm_id FROM farms WHERE farmer_id = p_farmerId));
END //

DELIMITER ;
DELIMITER //

CREATE PROCEDURE GetEmployeesByFarmerId(IN p_farmerId INT)
BEGIN
   SELECT *
   FROM employees
   WHERE farmer_id = p_farmerId;
END //

DELIMITER ;
DELIMITER //
CREATE PROCEDURE UpdateLivestockHealth(
   IN livestockID INT,
   IN healthStatus ENUM('healthy', 'sick', 'injured')
)
BEGIN
   UPDATE livestock SET health_status = healthStatus WHERE livestock_id = livestockID;
END //
DELIMITER ;

DELIMITER //
CREATE FUNCTION CalculateAverageYield(farmID INT)
RETURNS DECIMAL(10,2)
```

```
DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE avgYield DECIMAL(10,2);
    SELECT AVG(expected_yield) INTO avgYield FROM crops WHERE farm_id = farmID;
    RETURN avgYield;
END //
DELIMITER ;
```

One of the use case :

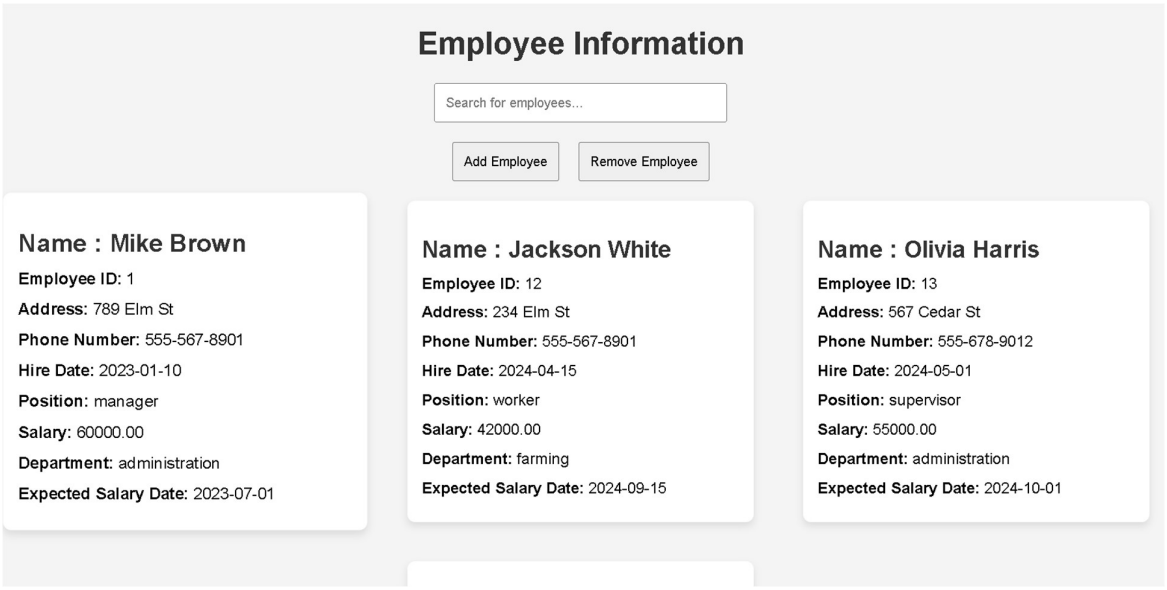
```python
@app.route('/sales')
def sales():
    user_data = session.get('user_data')
    print(user_data)
    farmer_id = int(user_data[0])
    db = pymysql.connect(**db_config)
    cursor = db.cursor()
    # Fetch data from the database for Orders
    cursor.execute('CALL GetSalesByFarmerId(%s)',(farmer_id))
    crop_sales_data = cursor.fetchall()
    cursor.close()
    db.close()

    return render_template('sales.html', crop_sales_data=crop_sales_data)
@app.route('/employees')
def employees():
    user_data = session.get('user_data')
    farmer_id = int(user_data[0])
    db = pymysql.connect(**db_config)
    cursor = db.cursor()
    # Fetch data from the database for Orders
    cursor.execute('CALL GetEmployeesByFarmerId(%s)',(farmer_id))
    employee_data = cursor.fetchall()
    cursor.close()
    db.close()
```

Crop Sales

Sale ID	Sale Date	Customer ID	Sale Amount	Sale Status	Crop ID	Quantity	Unit Price
1	2023-06-10	1	1500.50	closed	1	80	18.75
1	2023-06-20	1	900.75	closed	1	2	300.25

Livestock Sales

OUTPUT :




```
mysql> DELIMITER ;
mysql> CALL GetSalesByFarmerId(1);
+---------+------------+-------------+-------------+-------------+---------+----------+------------+
| sale_id | sale_date  | customer_id | sale_amount | sale_status | crop_id | quantity | unit_price |
+---------+------------+-------------+-------------+-------------+---------+----------+------------+
|       1 | 2023-06-10 |           1 |     1500.50 | closed      |       1 |       80 |      18.75 |
|       1 | 2023-06-20 |           1 |      900.75 | closed      |       1 |        2 |     300.25 |
+---------+------------+-------------+-------------+-------------+---------+----------+------------+
2 rows in set (0.06 sec)

Query OK, 0 rows affected (0.07 sec)

mysql> CALL GetSalesByFarmerId(2);
+---------+------------+-------------+-------------+-------------+---------+----------+------------+
| sale_id | sale_date  | customer_id | sale_amount | sale_status | crop_id | quantity | unit_price |
+---------+------------+-------------+-------------+-------------+---------+----------+------------+
|       2 | 2023-06-15 |           2 |     1200.25 | closed      |       2 |       30 |      22.50 |
|       2 | 2023-06-25 |           2 |     1200.50 | closed      |       2 |        1 |     600.50 |
+---------+------------+-------------+-------------+-------------+---------+----------+------------+
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.03 sec)
```

```
mysql> SHOW PROCEDURE STATUS WHERE Db = 'fms';
+------+------------------------+-----------+----------------+---------------------+---------------------+---------------+---------+----------------------+----------------------+-----------------------+
| Db   | Name                   | Type      | Definer        | Modified            | Created             | Security_type | Comment | character_set_client | collation_connection | Database Collation    |
+------+------------------------+-----------+----------------+---------------------+---------------------+---------------+---------+----------------------+----------------------+-----------------------+
| fms  | AddEmployee            | PROCEDURE | root@localhost | 2023-11-24 13:23:00 | 2023-11-24 13:23:00 | DEFINER       |         | cp850                | cp850_general_ci     | utf8mb4_0900_ai_ci    |
| fms  | AddLivestock           | PROCEDURE | root@localhost | 2023-11-24 13:22:05 | 2023-11-24 13:22:05 | DEFINER       |         | cp850                | cp850_general_ci     | utf8mb4_0900_ai_ci    |
| fms  | GetEmployeesByFarmerId | PROCEDURE | root@localhost | 2023-11-24 00:16:07 | 2023-11-24 00:16:07 | DEFINER       |         | cp850                | cp850_general_ci     | utf8mb4_0900_ai_ci    |
| fms  | GetEquipmentByFarmerId | PROCEDURE | root@localhost | 2023-11-24 00:15:24 | 2023-11-24 00:15:24 | DEFINER       |         | cp850                | cp850_general_ci     | utf8mb4_0900_ai_ci    |
| fms  | GetFarmerCrops         | PROCEDURE | root@localhost | 2023-11-24 00:08:34 | 2023-11-24 00:08:34 | DEFINER       |         | cp850                | cp850_general_ci     | utf8mb4_0900_ai_ci    |
| fms  | GetLivestockByFarmerId | PROCEDURE | root@localhost | 2023-11-24 00:14:14 | 2023-11-24 00:14:14 | DEFINER       |         | cp850                | cp850_general_ci     | utf8mb4_0900_ai_ci    |
| fms  | GetOrdersByFarmerId    | PROCEDURE | root@localhost | 2023-11-24 00:15:36 | 2023-11-24 00:15:36 | DEFINER       |         | cp850                | cp850_general_ci     | utf8mb4_0900_ai_ci    |
| fms  | GetSalesByFarmerId     | PROCEDURE | root@localhost | 2023-11-24 00:15:55 | 2023-11-24 00:15:55 | DEFINER       |         | cp850                | cp850_general_ci     | utf8mb4_0900_ai_ci    |
| fms  | RemoveCropById         | PROCEDURE | root@localhost | 2023-11-24 13:23:42 | 2023-11-24 13:23:42 | DEFINER       |         | cp850                | cp850_general_ci     | utf8mb4_0900_ai_ci    |
| fms  | RemoveEmployeeById     | PROCEDURE | root@localhost | 2023-11-24 13:23:10 | 2023-11-24 13:23:10 | DEFINER       |         | cp850                | cp850_general_ci     | utf8mb4_0900_ai_ci    |
| fms  | RemoveLivestockById    | PROCEDURE | root@localhost | 2023-11-24 13:22:48 | 2023-11-24 13:22:48 | DEFINER       |         | cp850                | cp850_general_ci     | utf8mb4_0900_ai_ci    |
| fms  | UpdateLivestockHealth  | PROCEDURE | root@localhost | 2023-11-26 17:04:42 | 2023-11-26 17:04:42 | DEFINER       |         | cp850                | cp850_general_ci     | utf8mb4_0900_ai_ci    |
+------+------------------------+-----------+----------------+---------------------+---------------------+---------------+---------+----------------------+----------------------+-----------------------+
12 rows in set (0.29 sec)
```

Function used to calculate the average yield of the farmer for a particular farm

```
DELIMITER //
CREATE FUNCTION CalculateAverageYield(farmID INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE avgYield DECIMAL(10,2);
    SELECT AVG(expected_yield) INTO avgYield FROM crops WHERE farm_id = farmID;
    RETURN avgYield;
END //
DELIMITER ;
```

```
mysql>
mysql> SELECT CalculateAverageYield(1) AS AVG_YIELD;
+-----------+
| AVG_YIELD |
+-----------+
|    557.96 |
+-----------+
1 row in set (0.07 sec)
```

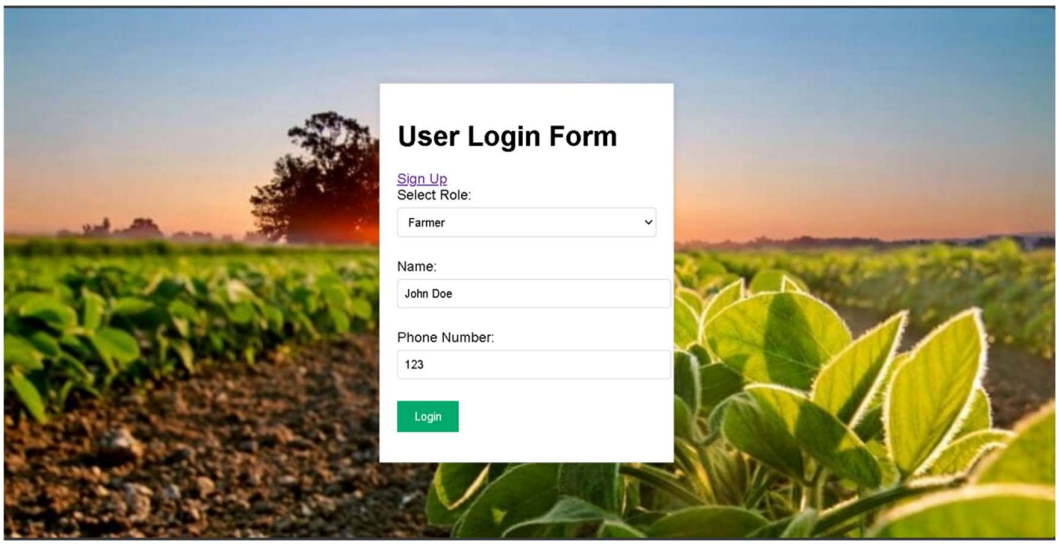
Trigger used to invoke if the equipment year is not current :

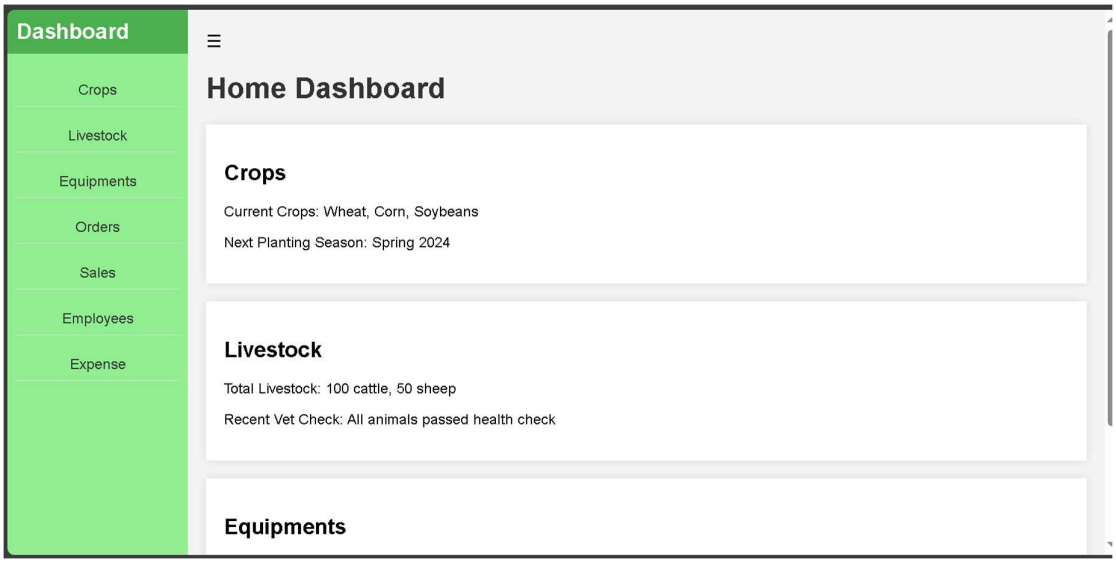
```
DELIMITER //
```

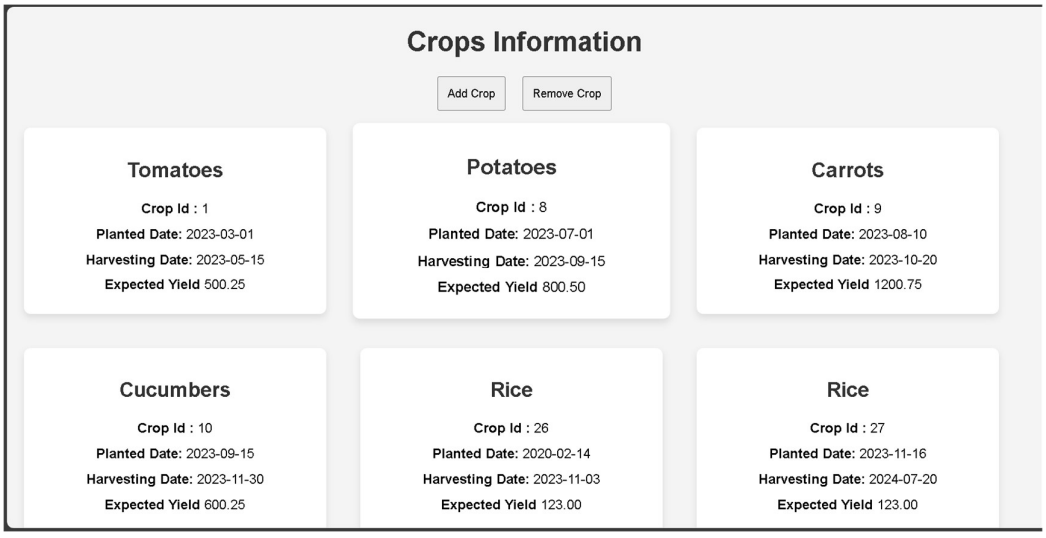
```
CREATE TRIGGER check_equipment_year
BEFORE INSERT ON equipment
FOR EACH ROW
BEGIN
   IF NEW.equipment_year > YEAR(CURDATE()) THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Invalid equipment year. Year cannot be in the future.';
   END IF;
END //
DELIMITER ;
```

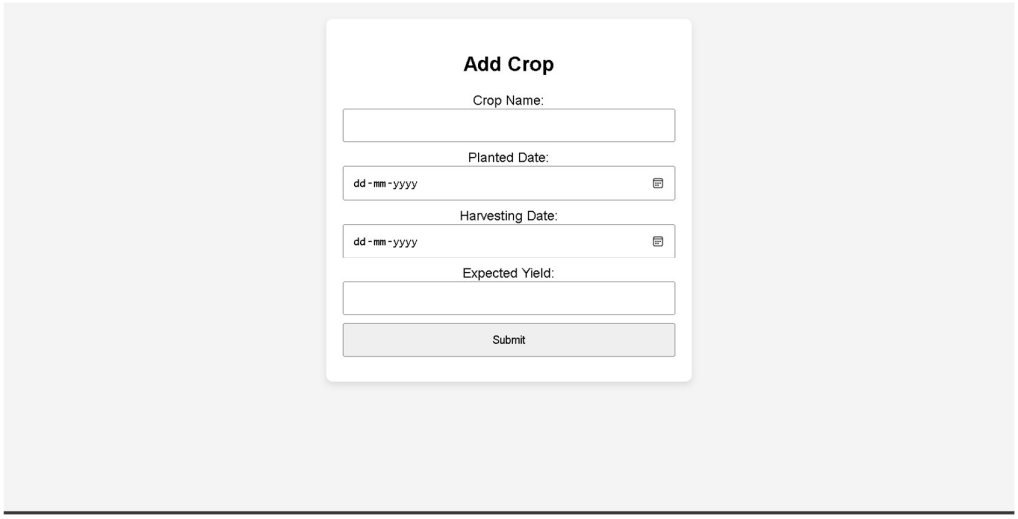
```
mysql>
mysql> -- Inserting equipment with a valid year (e.g., 2022)
mysql> INSERT INTO equipment (farmer_id, equipment_type, equipment_brand, equipment_model, equipment_year)
    -> VALUES (1, 'tractor', 'BrandX', 'Model123', 2022);
Query OK, 1 row affected (0.02 sec)

mysql>
mysql> -- Inserting equipment with a future year (e.g., 2030)
mysql> INSERT INTO equipment (farmer_id, equipment_type, equipment_brand, equipment_model, equipment_year)
    -> VALUES (1, 'combine', 'BrandY', 'Model456', 2030);
ERROR 1644 (45000): Invalid equipment year. Year cannot be in the future.
```

OUTPUT OF THE WEBSITE FARM MANAGEMENT SYSTEM :









Equipment Information

Add Equipment | Remove Equipment

Equipment tractor
Equipment ID: 1
Equipment Brand: John Deere
Equipment Model: X750
Equipment Year: 2020

Equipment harvester
Equipment ID: 6
Equipment Brand: John Deere
Equipment Model: S780
Equipment Year: 2023

Equipment harvester
Equipment ID: 14
Equipment Brand: John Deere
Equipment Model: S790
Equipment Year: 2023

Equipment tractor
Equipment ID: 15
Equipment Brand: Case IH
Equipment Model: Magnum 340
Equipment Year: 2022

Equipment tractor
Equipment ID: 18
Equipment Brand: BrandX
Equipment Model: Model123
Equipment Year: 2022

Livestock Information

Add livestock | Remove livestock

Holstein
Age: 2
Weight: 450.75
Health Status: healthy

Cow
Age: 3
Weight: 600.00
Health Status: healthy

Sheep
Age: 1
Weight: 37.50
Health Status: healthy

Pig
Age: 2
Weight: 95.25
Health Status: healthy

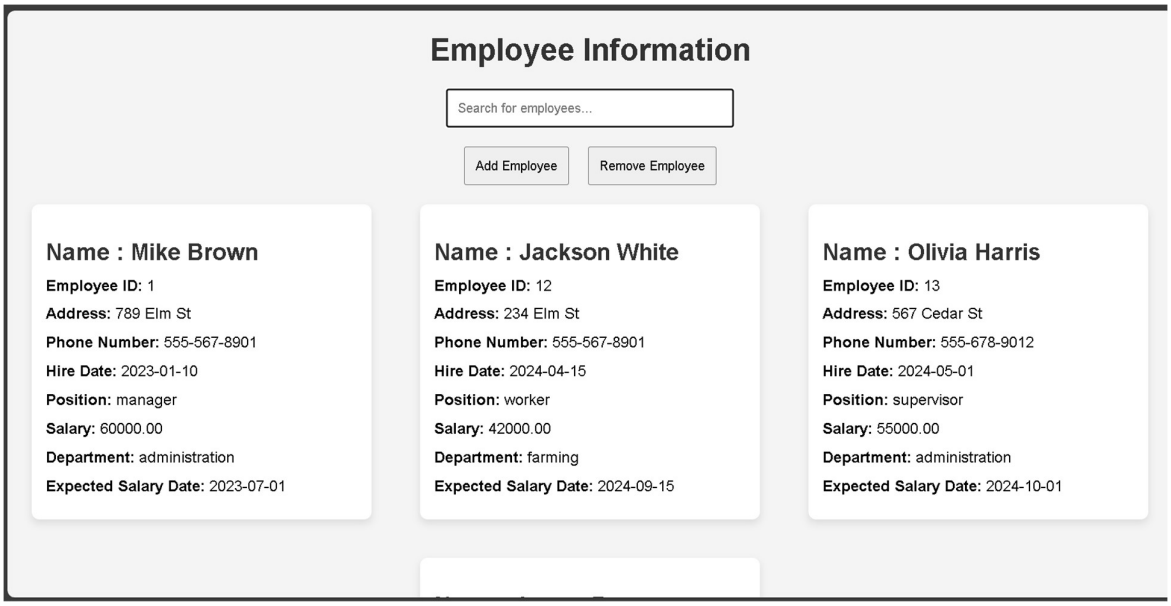
Expenses Information

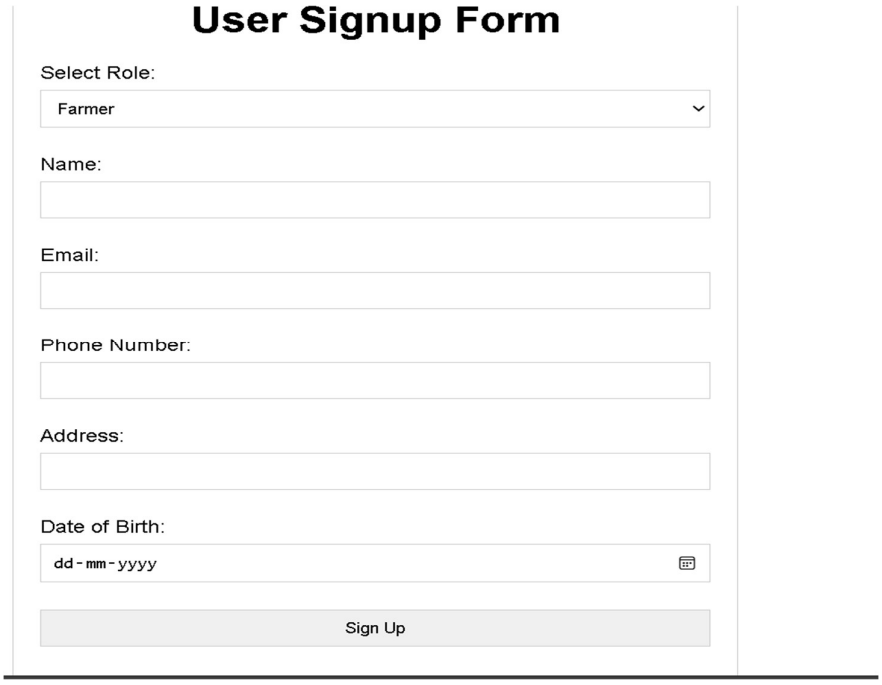
Expense ID: 1
Expense Date: 2023-06-05
Expense Type: feed
Expense Amount: 500.00

Expense ID: 6
Expense Date: 2023-11-20
Expense Type: feed
Expense Amount: 800.00

Orders

Order ID	Customer ID	Order Date	Item Crop ID	Item Livestock ID	Quantity Crop	Quantity Livestock
1	1	2023-06-01	1	None	100	None
21	4	2023-11-15	None	19	None	4





GITHUB Repository:

https://github.com/Raghavendra369/DBMS-Project