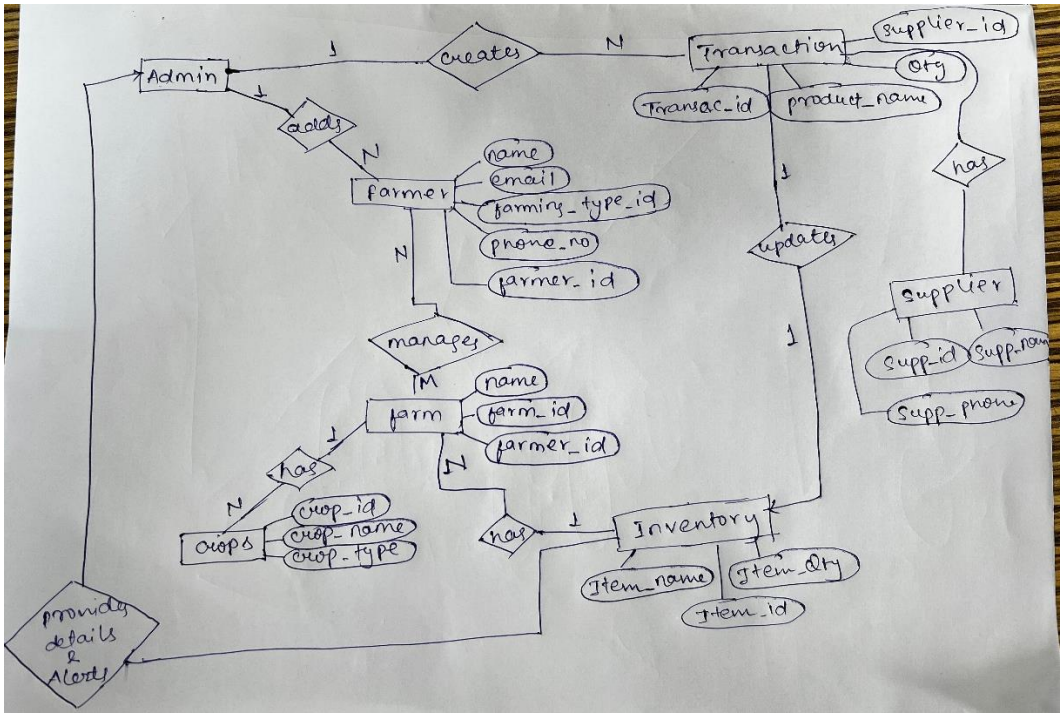
Farm Management System

Team:

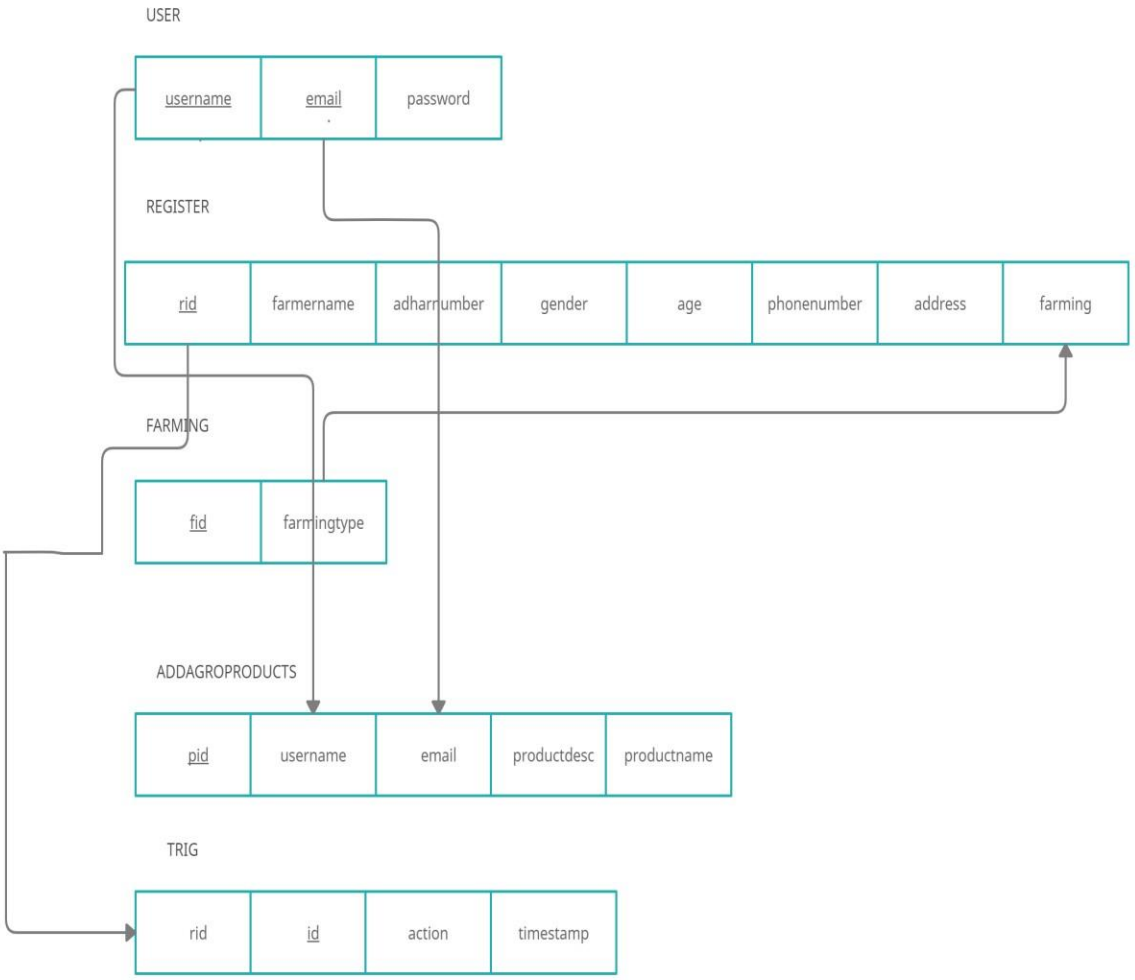
Mohammed Nihal Mantri (PES1UG22CS356)

Safdar Hakeem Ahmed (PES1UG23CS822)

ER Diagram:



Relationship Schema:



Queries:

Multi-Table Nested and Aggregate Query:

Lists the farmingtype while accessing every username and their associated products in the addagroproducts table. Aggregates the total price of products added by each user and displays the total.

```
SELECT farming.farmingtype, addagroproducts.username,
SUM(addagroproducts.price) AS total_price
FROM farming
JOIN register ON farming.fid = register.farming
JOIN addagroproducts ON register.username = addagroproducts.username
GROUP BY farming.farmingtype, addagroproducts.username;
```

Multi-Table Join and Aggregate Query:

Returns the farming type with the maximum number of registered farmers associated with it.

```
SELECT farming.farmingtype, COUNT(register.rid) AS farmer_count
FROM farming
JOIN register ON farming.fid = register.farming
GROUP BY farming.farmingtype
ORDER BY farmer_count DESC
LIMIT 1;
```

Multi-Table Aggregate and Ranking Query:

Lists the farming types in descending order based on the number of farmers registered under each type.

```
SELECT farming.farmingtype, COUNT(register.rid) AS farmer_count
FROM farming
JOIN register ON farming.fid = register.farming
GROUP BY farming.farmingtype
ORDER BY farmer_count
DESC;
```

Procedure Call:

Calls the procedure to reset the price of a specific product in the addagroproducts table.

```
CALL reset_product_price('silk');
```

Update Queries:

Updates the price of a specific product in the addagroproducts table.

```
UPDATE addagroproducts
SET price = 600
WHERE productname = 'GIRIJA CAULIFLOWER';
```

Updates the details (address, age, and phone number) of a specific farmer in the register table.

```
UPDATE register
SET address = 'New Address', age = 35, phonenumber = '9876543210'
WHERE farmername = 'arkpro';
```

Updates the timestamp in the trig table after a product's details are updated.

```
UPDATE trig
SET timestamp = NOW()
WHERE action = 'FARMER UPDATED';
```

Insert Queries:

Inserts a new product into the addagroproducts table.

```
INSERT INTO addagroproducts (username, email, pid, productname,
productdesc, price)
VALUES ('newuser', 'newuser@example.com', 4, 'Wheat', 'High-quality wheat
for baking', 300);
```

Adds a new farming type into the farming table.

```
INSERT INTO farming (fid, farmingtype)
VALUES (4, 'Organic Farming');
```

Registers a new farmer into the register table and ensures triggers track the addition.

```
INSERT INTO register (rid, farmername, adharnumber, age, gender,
phonenumber, address, farming)
VALUES (10, 'newfarmer', '123456789012', 40, 'Male', '9876543210', 'New
Address', 'Organic Farming');
```

Triggers:

Increments the trig table's timestamp and logs an action when a farmer is deleted.

```
CREATE TRIGGER `deletion` BEFORE DELETE ON `register`
FOR EACH ROW
INSERT INTO trig VALUES (NULL, OLD.rid, 'FARMER DELETED', NOW());
```

Verifies data integrity by ensuring no duplicate farming types are added to the farming table.

```
CREATE TRIGGER prevent_duplicate_farming BEFORE INSERT ON `farming`
FOR EACH ROW
BEGIN
  IF EXISTS (SELECT 1 FROM farming WHERE farmingtype = NEW.farmingtype)
THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Duplicate farming type not allowed';
  END IF;
END;
```

Checks and logs updates to the register table for monitoring farmer information changes.

```
CREATE TRIGGER `updation` AFTER UPDATE ON `register`
FOR EACH ROW
INSERT INTO trig VALUES (NULL, NEW.rid, 'FARMER UPDATED', NOW());
```

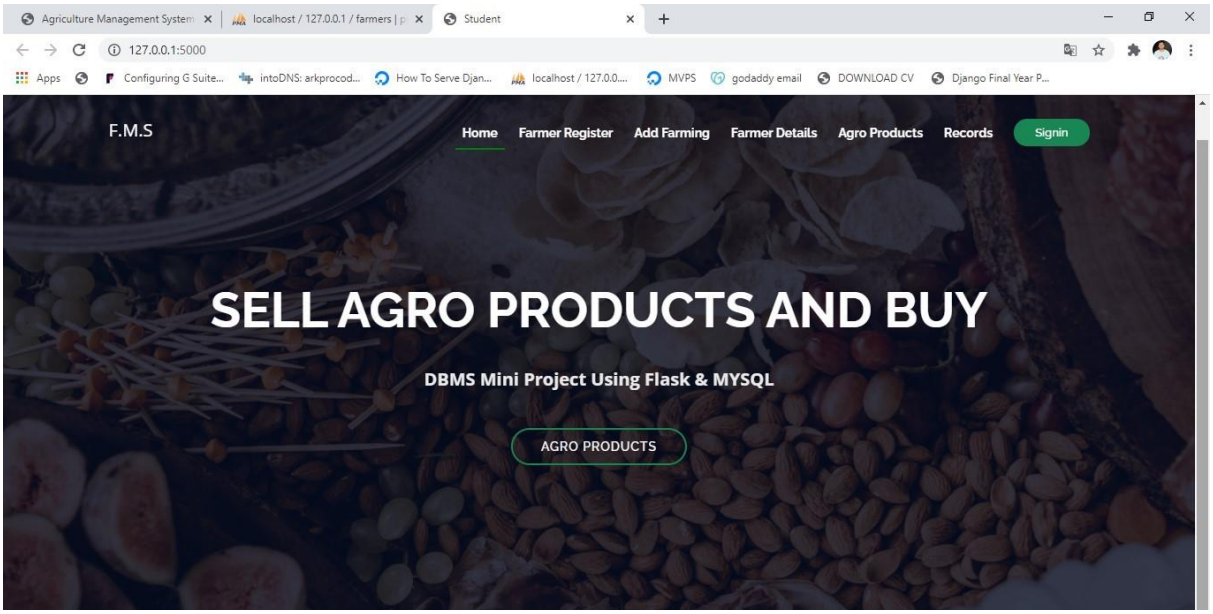
Procedures:

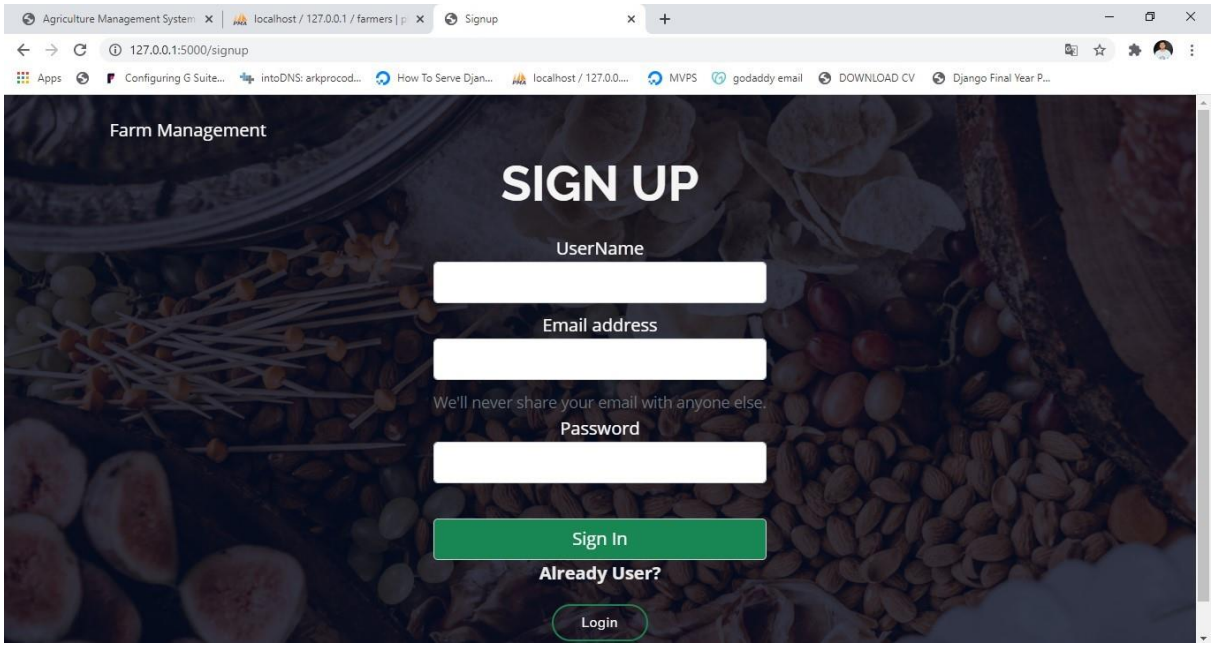
Resets the price of all products for a specific farmer in the addagroproducts table.

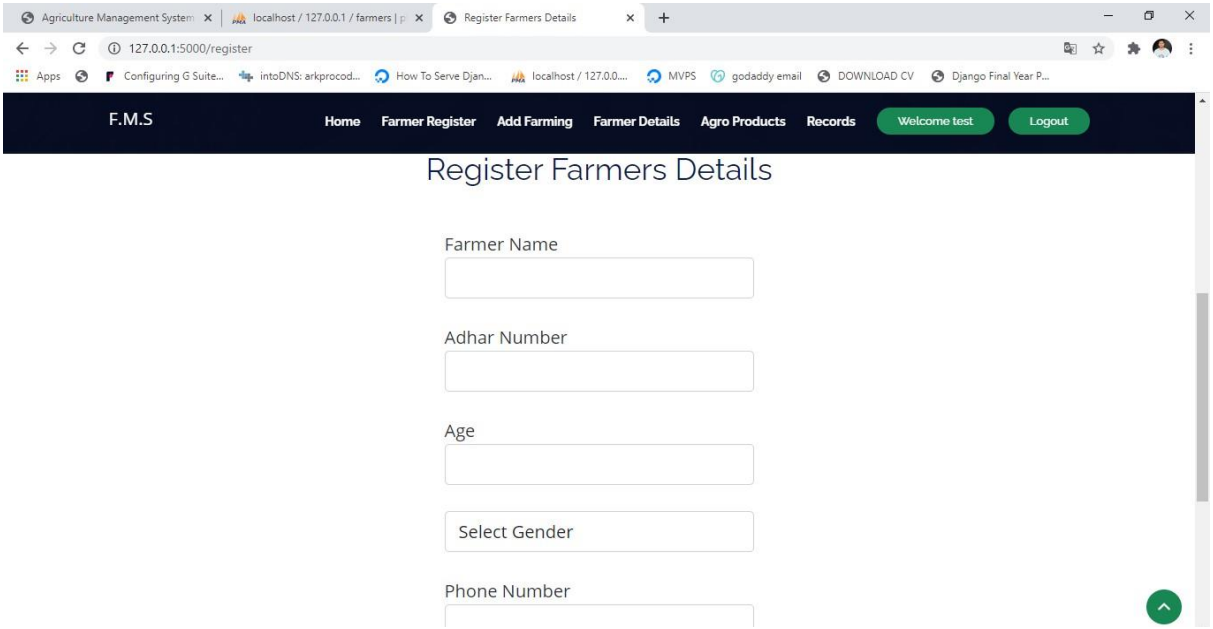
```
DELIMITER $$
CREATE PROCEDURE reset_product_price(IN productName VARCHAR(100))
BEGIN
  UPDATE addagroproducts
  SET price = 0
  WHERE productname = productName;
END$$
DELIMITER ;
```

USER INTERFACE

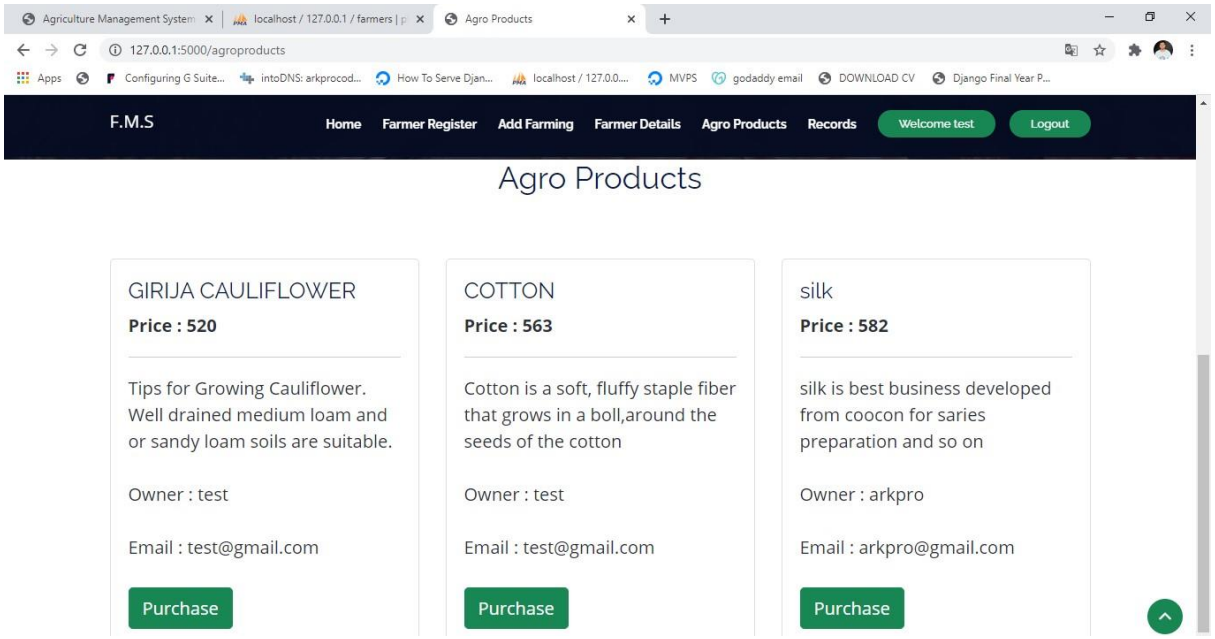
SIGN IN PAGE:

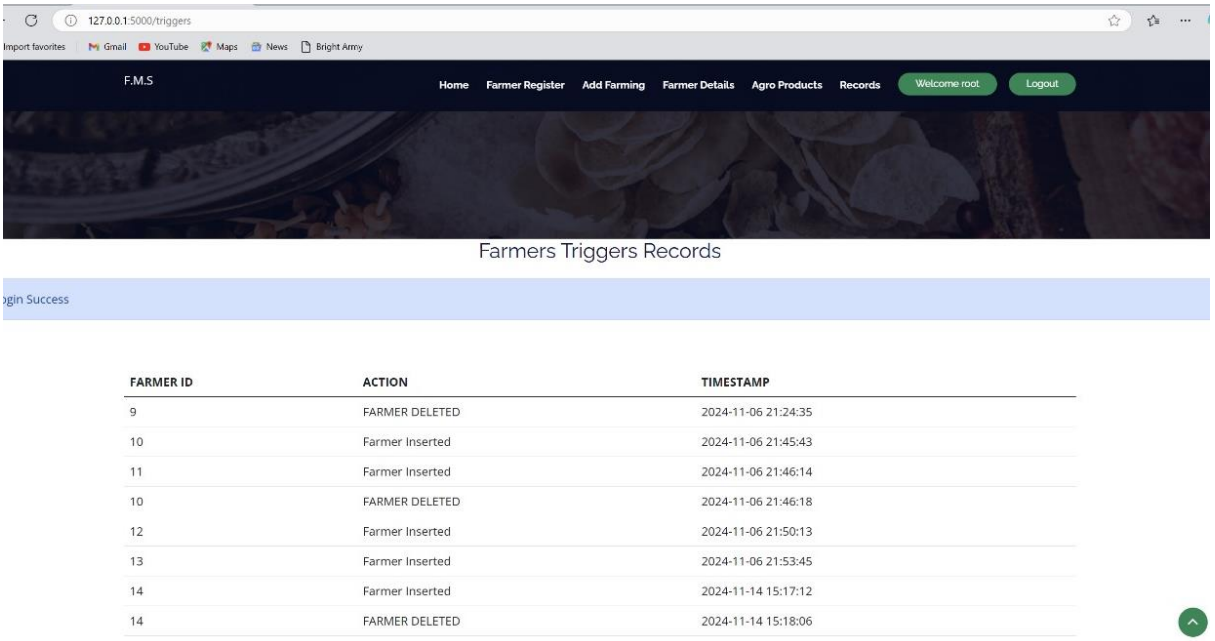




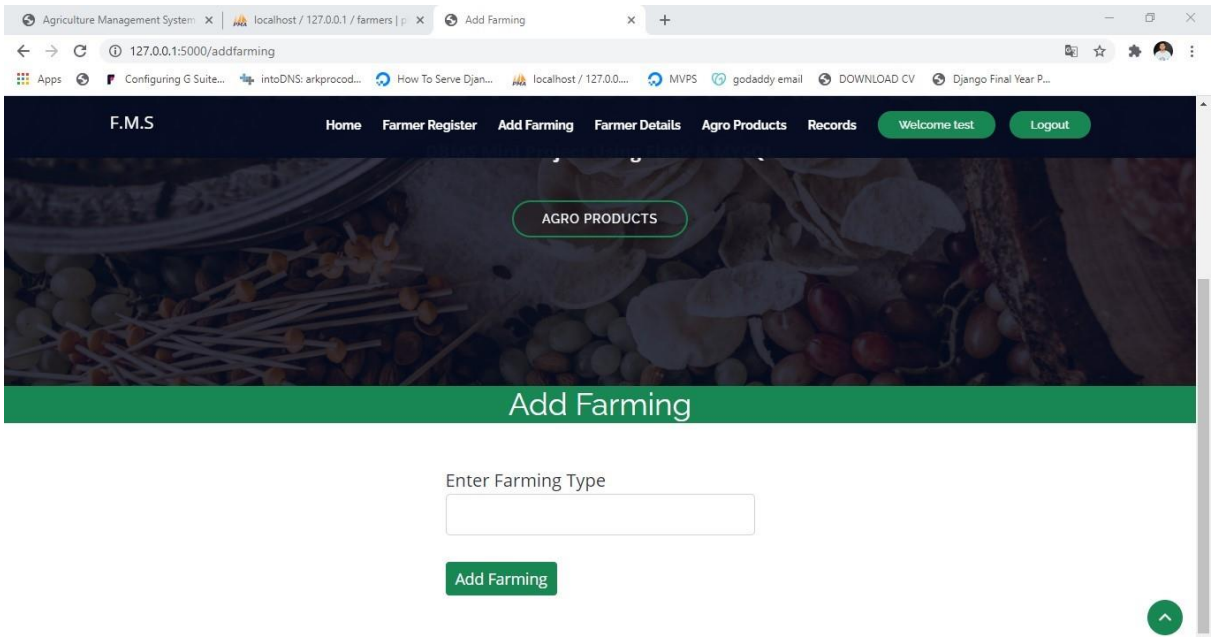


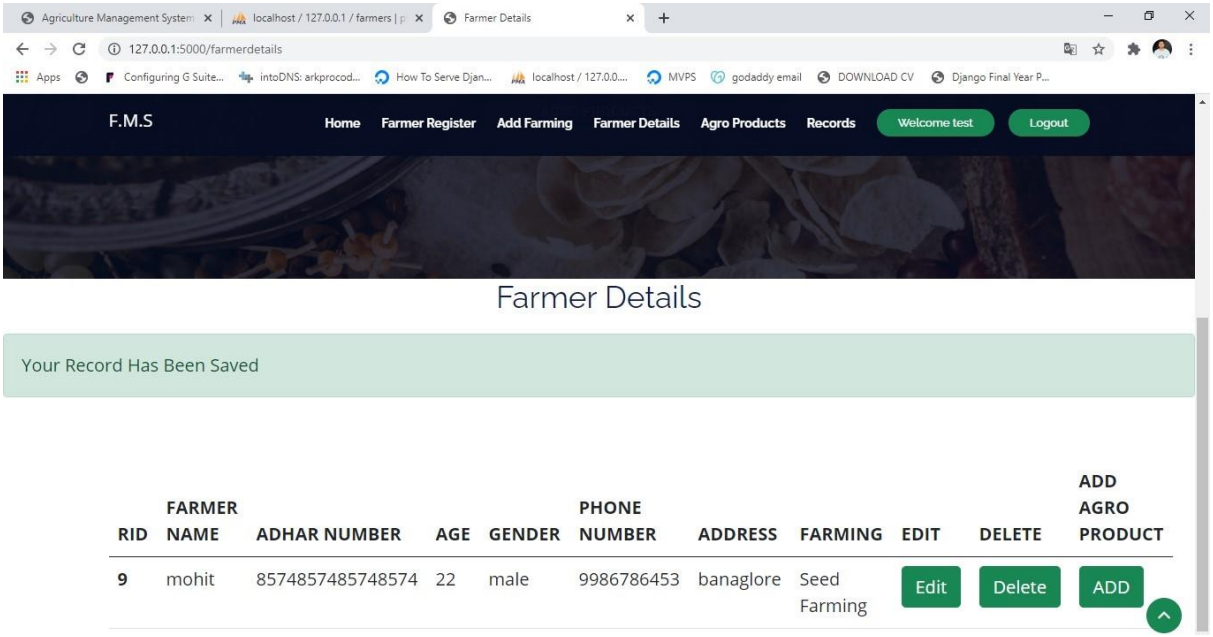
REGISTERATION PAGE & PRODUCTS:



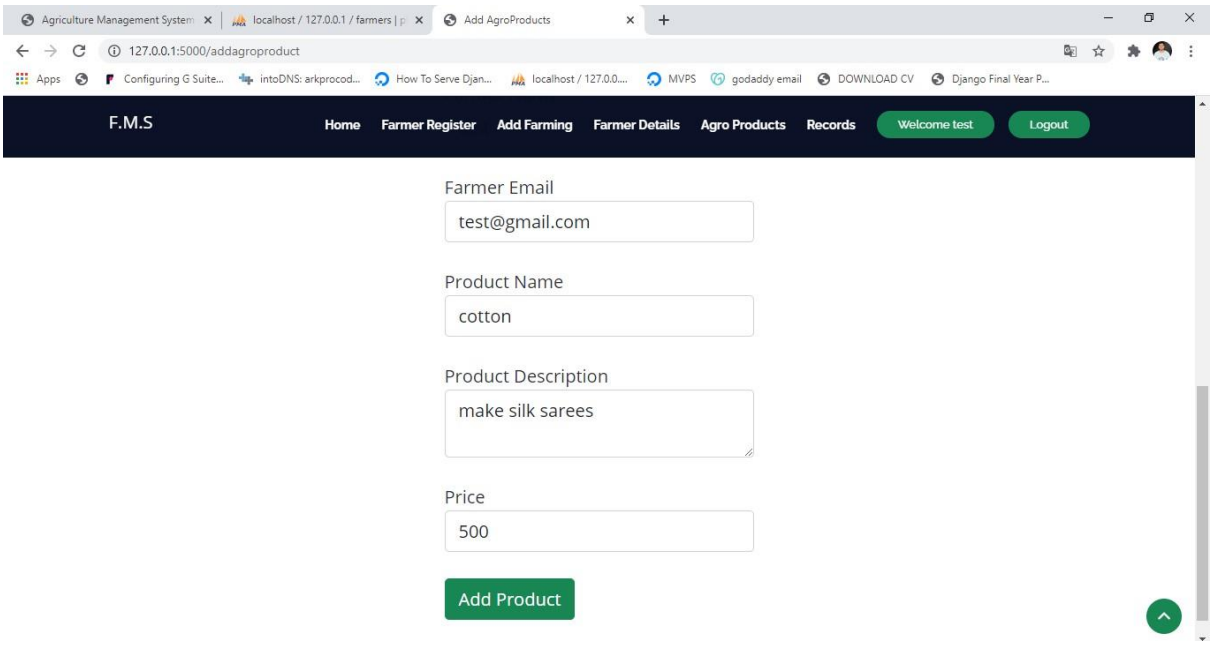


TRIGGERS RECORDS





ADDING AGRO PRODUCTS



CONCLUSION

FARM MANAGEMENT SYSTEM successfully implemented based on online selling which helps us in administrating the agroproducts user for managing the tasks performed in farmers. The project successfully used various functionalities of Xampp and python flask and also create the fully functional database management system for online portals.

Using MySQL as the database is highly beneficial as it is free to download, popular and can be easily customized. The data stored in the MySQL database can easily be retrieved and manipulated according to the requirements with basic knowledge of SQL.

With the theoretical inclination of our syllabus it becomes very essential to take the atmost advantage of any opportunity of gaining practical experience that comes along. The building blocks of this Major Project “Farm Management System” was one of these opportunities. It gave us the requisite practical knowledge to supplement the already taught theoretical concepts thus making us more competent as a computer engineer. The project from a personal point of view also helped us in understanding the following aspects of project development:

• The planning that goes into implementing a project.

• The importance of proper planning and an organized methodology.

• The key element of team spirit and co-ordination in a successful project.

FUTURE ENHANCEMENT

• Enhanced database storage facility

• Enhanced user friendly GUI

• more advanced results systems

- online payments