ABSTRACT

The purpose of farm management system is to automate the existing manual system by the help of computerized equipment's and full-fledged computer software .Fulfilling their requirement, so that their valuable information can be stored for a longer period with easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with.

Farm management system, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help organization in better utilization of resources. The organization can maintain computerized records without redundant entries. That means that one need not be distracted by information that is not relevant, while being able to reach the information.

The aim is to automate the existing manual system by the help of computerized equipment's and full-fledged computer software .Fulfilling their requirement, so that their valuable information can be stored for a longer period with easy accessing and manipulation of the same. Basically the project describes how to manage goods performance and better service for clients.

TABLE OF CONTENTS

TABLE OF FIGURES

Chapter 1

INTRODUCTION

1.1 Overview of Database Management System

The farmers can sell their productions online and the buyer can purchase various agricultural products online. Buyer can send purchase request to check the quality of the product. After collecting all the farm produce from the farmers, it should be sold to the customers. This project covers these entries and the data collections. There are 2 types of users: Customer & Farmers. The login id and password must be required to login the system. The article and agro products section helps farmers to share their products and increase profitability.

Functions of database management system:

- Provides Recovery services
- Provides utility
- Provides data Independence
- Provides a clear and logical view of the process that manipulates data.

Advantages of DBMS:

- Segregation of application program
- Minimal data duplicity
- Reduced development time and maintenance need
- Easy retrieval of data

1.2 Problem Statement

The main aim of developing “Farm Management System Project” application is to help farmers by providing all kinds agriculture related information in the site.” Farm Management System Project” is web application which helps farmers to share best-practice farming processes. It helps farmers to improve their productivity and profitability. It enables farmers to sell their product online and farmers can purchase tools and seeds directly from seller. Farmers can view their profile and they can register, edit and delete data.

The farmers can sell their productions online and the buyer can purchase various agriculture products online. Buyer can send purchase request to check the quality of the Agro product through mails.

1.3 Objectives

- The main objective of the project is to design and develop a user friendly-system
- Easy to use and an efficient computerized system.
- To develop an accurate and flexible system, it will eliminate data redundancy.
- To study the functioning of Farm management System.
- To make a software fast in processing, with good user interface.
- To make software with good user interface so that user can change it and it should be used for a long time without error and maintenance.
- To provide synchronized and centralized farmer and seller database.
- Computerization can be helpful as a means of saving time and money.
- To provide better Graphical User Interface (GUI).
- Less chances of information leakage.
- Provides Security to the data by using login and password method.
- To provide immediate storage and retrieval of data and information.
- Improving arrangements for farmers co-ordination.
- Reducing loss.

1.4 Dataset Description

- The entity User is created with user_id , Email and password as the unique key.
- The entity Registration is created with the attributes registration_id, farmer_name, aadhar_no, gender, age, phone_number and farming. The registration_no is the Primary key and farming as foreign key with reference to farming_type.
- The Entity Farming is created with the attributes farming_id , farming_type and farming_id is the primary key.
- The Entity Agro_product is created with the attributes product_id, user_id, email, product_description, product_name where user_id and email are foreign key with references to User attributes.
- The Entity Trig is created with the attributes registration_id, id, action, timestamp where registration_id is the foreign key referring entity Registration.

Chapter 2

SYSTEM REQUIREMENTS

2.1 Software Requirement

Frontend- HTML, CSS, Java Script, Bootstrap Backend-Python flask (Python 3.7), SQLAlchemy.

- Operating System: Windows 10
- Google Chrome/Internet Explorer
- XAMPP (Version-3.7)
- Python main editor (user interface): PyCharm Community
- workspace editor: Sublime text 3/ Visual studio code

2.2 Hardware Requirement

- Computer with a 1.1 GHz or faster processor
- Minimum 2GB of RAM or more
- 2.5 GB of available hard-disk space
- 5400 RPM hard drive
- 1366 × 768 or higher-resolution display
- DVD-ROM drive

Chapter 3

SYSTEM DESIGN

.

3.1 ER Diagram

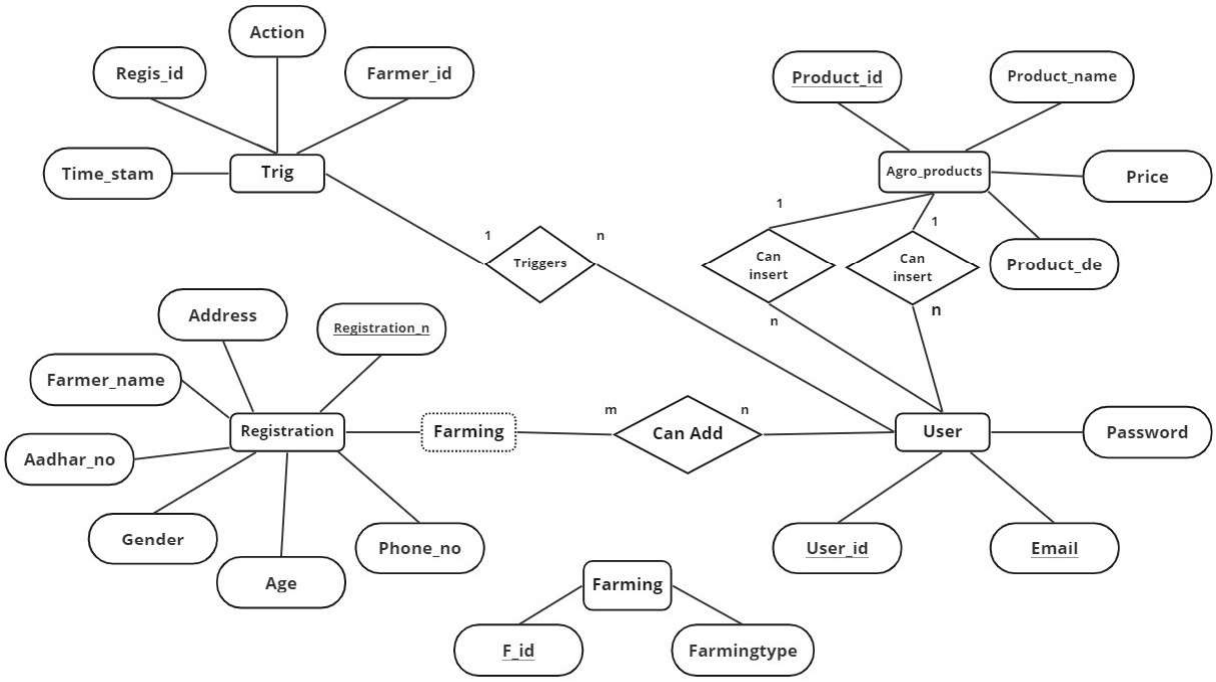


Fig 3.1 - ER diagram for farm management system

An entity relationship diagram(ERD) is a data modeling technique that graphically illustrates an information system's entities and the relationships between those entities. An Entity Relationship Diagram contains different symbols and connectors that visualize two important information. The major entities within the system scope and the interrelationships among these entities

3.2 ER to Relation Mapping

Step 1: Mapping the regular entities – Our ER diagram corresponds to five entities User, Registration, Farming, Agroproducts, Trig. But this does not include the foreign and relational attribute which are Farming type in entity Farming, User_id and Email in entity Agroproducts and Registartion_id in entity Trigger.

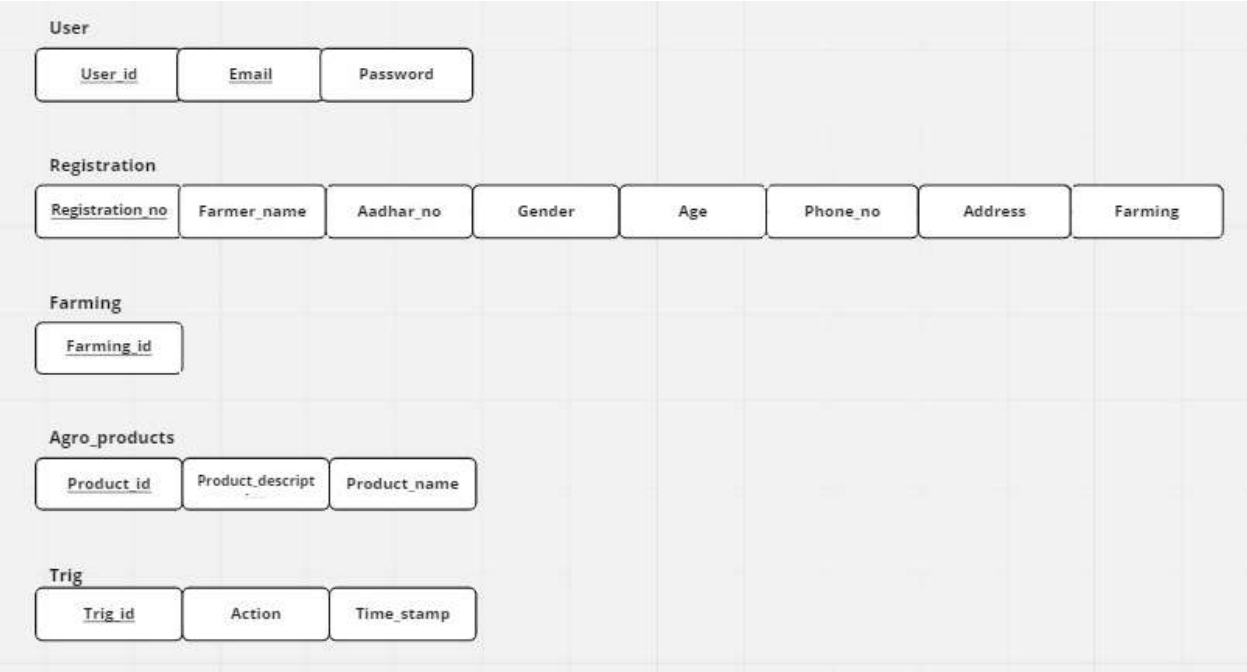


Fig 3.2.1 – Regular entity mapping

Step 2: **Mapping of weak entity types –** We do not have weak entities in our ER.

Step 3: **Mapping of 1 : 1 cardinality relationship –** We do not have this type of relationship.

Step 4: **Mapping of 1 : n cardinality relationship –** We have three 1:n type of relationship .

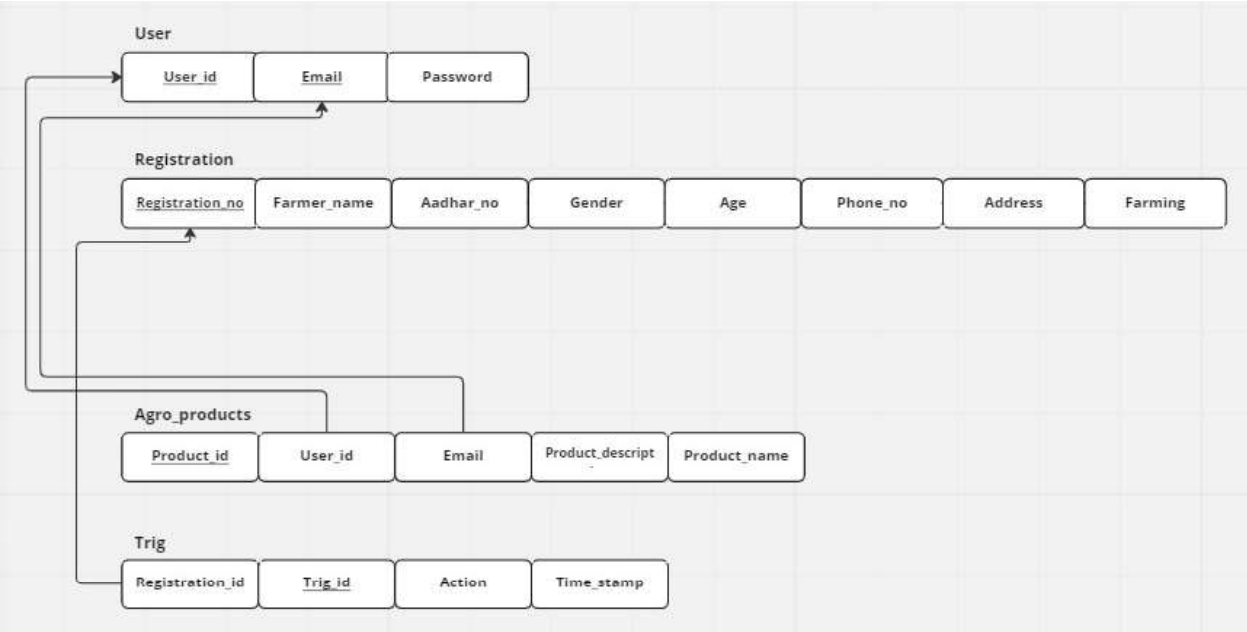


Fig 3.2.2 - 1: n cardinality relationship

Step 5: **Mapping of m : n cardinality relationship –** We have one n:m type of relationship. Here as the user n number of users can add m number of farming type the entity user and registration are indirectly related to each other with the help of farming entity.

Fig 3.2.3 – m: n cardinality relationship

3.3 Schema Diagram

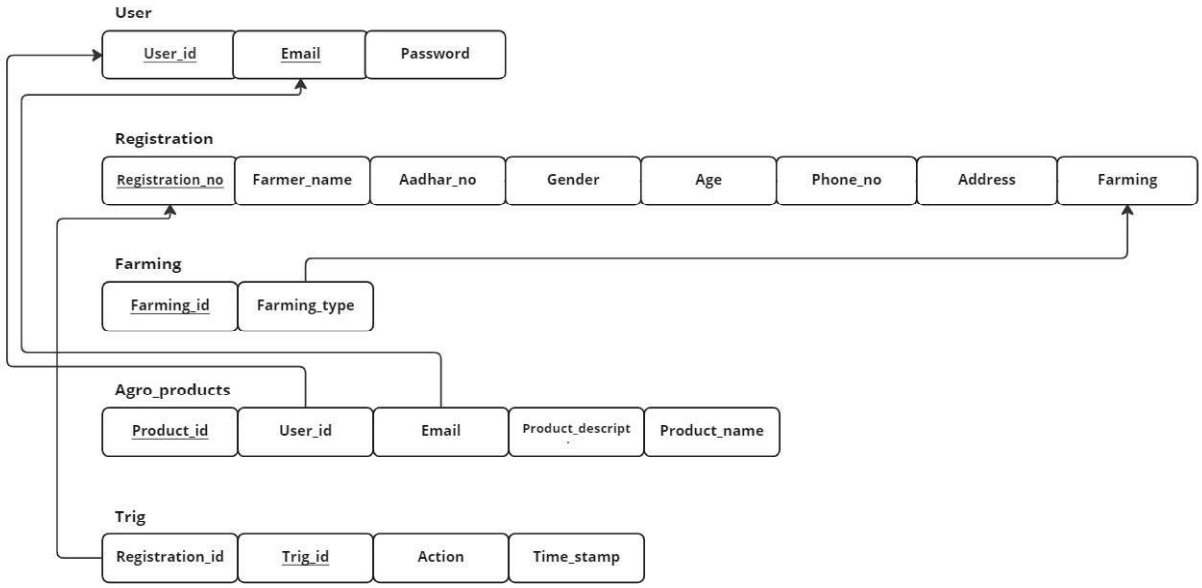


Fig 3.3 – Schema diagram for farm management system

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and the relations among them are associated. It formulates all the constraints that are to be applied on data

3.4 Overview of GUI

GUI is a program interface that takes advantage of the computer's graphics capabilities to make the program easier to use Well designed graphical user interfaces can free the user from learning complex command languages On the other hand, many users find that they work more effectively with a command driven interface, especially if they already know the command language

- **Hyper Text Markup Language**
  HTML is the standard markup language for creating web pages and web applications With JavaScript it forms a triad of cornerstone technologies for the World Wide Web Web browsers receive HTML documents from a web server or from local storage and render them into multimedia web pages HTML describes the structure of a web page semantically and originally includedcues for the appearance of the document

- **Cascading Style Sheet**
  CSS is a style sheet language used for describing the presentation of a document written in a markup language like HTML CSS is a cornerstone technology of the World Wide Web alongside HTML and JavaScript CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts This separation can improve content accessibility provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate css file, and reduce complexity and repetition in the structural content

3.5 Normalization

- **1NF**: A relation is said to be in 1 NF form when it contains atomic values, or in simple does not contain multi valued attributes As our schema in Fig doesn't have multi valued attributes, the relational model can be said to be in 1 NF form

- **2NF**: In the second normal form, all non key attributes are fully functional dependent on the primary key and should be in 1 NF form .When observed in Fig

User Table – Primary key: User_id and Email – all other non-key attribute fully dependent on it as in Fig 3.5.1

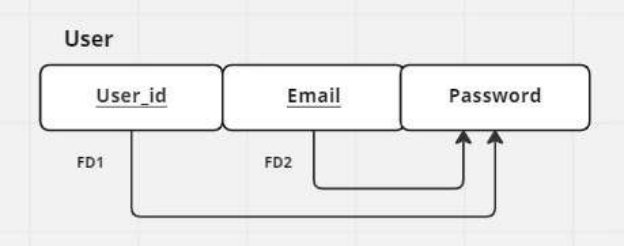


Fig 3.5.1 – FDs in User table

Registration Table – Primary key: Registration_no–all other non-key attribute fully dependent on it as in Fig 3.5.2

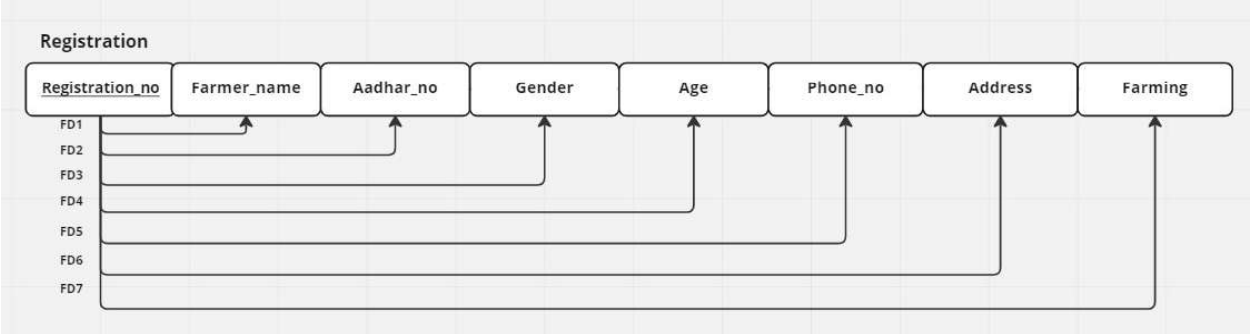


Fig 3.5.2 – FDs in Registration table

Farming Table – Primary key: Farming_id – all other non-key attribute fully dependent on it as in Fig 3.5.3

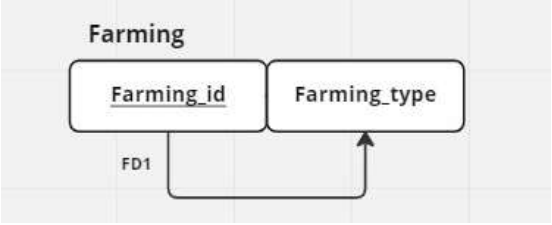


Fig 3.5.3 – FDs in Farming table

Agro_Products Table – Primary key: Product_id – all other non-key attribute fully dependent on it as in Fig 3.5.4

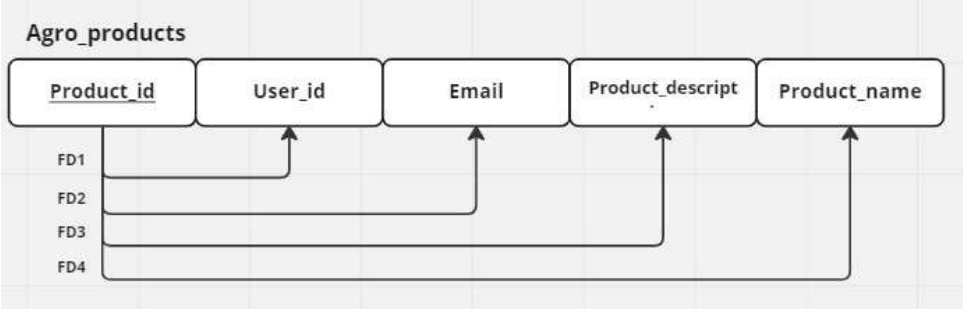


Fig 3.5.4 – FDs in Agro products table

Trig Table – Primary key: Trig_id – all other non-key attribute fully dependent on it as in Fig 3.5.5

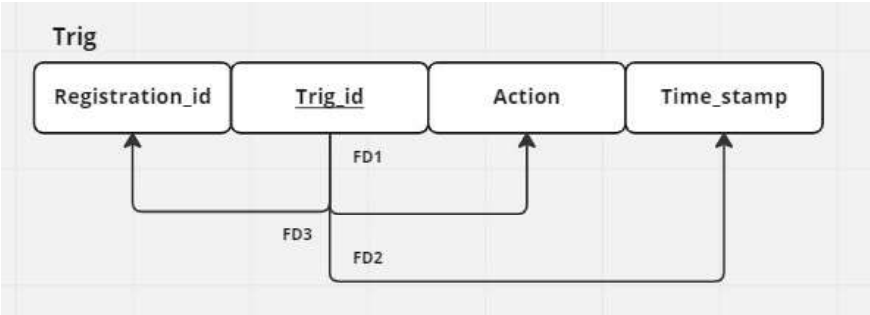


Fig 3.5.5 – FDs in Trig table

Therefore, the relational schema is in 2 NF form as all relations have full dependencies.

- **3 NF**: A relation will be in 3 NF if it is in 2 NF and not contain any transitive partial dependency 3 NF is used to reduce the data duplication. It is also used to achieve the data integrity. If there is no transitive dependency for non prime attributes, then the relation must be in third normal form. Transitive functional dependency. A dependency X --> Y in a relation is said to be a transitive dependency if there exists an attribute Z such that X -->Z and Z -->Y. As there are no transitive dependencies in our relational schema, the relational model is said to be in 3 NF form.

Chapter 4

IMPLEMENTATION

4.1 Table Creation

- CREATE TABLE addagroproducts (
  username varchar(50) NOT NULL,
  email varchar(50) NOT NULL,
  pid int(11) NOT NULL,
  productname varchar(100) NOT NULL,
  productdesc text NOT NULL,
  price int(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

- CREATE TABLE farming (
  fid int(11) NOT NULL,
  farmingtype varchar(200) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

- CREATE TABLE register (
  rid int(11) NOT NULL,
  farmername varchar(50) NOT NULL,
  adharnumber varchar(20) NOT NULL,
  age int(100) NOT NULL,
  gender varchar(50) NOT NULL,
  phonenumber varchar(12) NOT NULL,
  address varchar(50) NOT NULL,
  farming varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

- CREATE TABLE test (
  id int(11) NOT NULL,
  name varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

- CREATE TABLE trig (
  id int(11) NOT NULL,
  fid varchar(50) NOT NULL,
  action varchar(200) NOT NULL,
  timestamp datetime NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

- CREATE TABLE user (
  id int(11) NOT NULL,
  username varchar(50) NOT NULL,
  email varchar(50) NOT NULL,
  password varchar(500) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

- ALTER TABLE addagroproducts ADD PRIMARY KEY (pid);

- ALTER TABLE farming ADD PRIMARY KEY (fid);

- ALTER TABLE register ADD PRIMARY KEY (rid);

- ALTER TABLE test ADD PRIMARY KEY (id);

- ALTER TABLE trig ADD PRIMARY KEY (id);

- ALTER TABLE user ADD PRIMARY KEY (id);

- ALTER TABLE addagroproducts
  MODIFY pid int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;

- ALTER TABLE farming
  MODIFY fid int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;

- ALTER TABLE register
  MODIFY rid int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=9;

- ALTER TABLE test
  MODIFY id int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

- ALTER TABLE trig
  MODIFY id int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;

- ALTER TABLE user
  MODIFY id int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;
  COMMIT;

4.2 Description of Table

- **User Table**

```
mysql> describe user;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| id       | int          | NO   | PRI | NULL    | auto_increment |
| username | varchar(50)  | NO   |     | NULL    |                |
| email    | varchar(50)  | NO   |     | NULL    |                |
| password | varchar(500) | NO   |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
4 rows in set (0.01 sec)
```

Fig 4.2.1 – Description of User table

- **Registration Table**

```
mysql> describe register;
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| rid         | int         | NO   | PRI | NULL    | auto_increment |
| farmername  | varchar(50) | NO   |     | NULL    |                |
| adharnumber | varchar(20) | NO   |     | NULL    |                |
| age         | int         | NO   |     | NULL    |                |
| gender      | varchar(50) | NO   |     | NULL    |                |
| phonenumber | varchar(12) | NO   |     | NULL    |                |
| address     | varchar(50) | NO   |     | NULL    |                |
| farming     | varchar(50) | NO   |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
8 rows in set (0.01 sec)
```

Fig 4.2.2 – Description of Registration table

- **Farming Table**

```
mysql> describe farming;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| fid         | int          | NO   | PRI | NULL    | auto_increment |
| farmingtype | varchar(200) | NO   |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
2 rows in set (0.46 sec)
```

Fig 4.2.3 – Description of Farming table

- **AgroProducts Table**

```
mysql> describe addagroproducts;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| username    | varchar(50)  | NO   |     | NULL    |                |
| email       | varchar(50)  | NO   |     | NULL    |                |
| pid         | int          | NO   | PRI | NULL    | auto_increment |
| productname | varchar(100) | NO   |     | NULL    |                |
| productdesc | text         | NO   |     | NULL    |                |
| price       | int          | NO   |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
6 rows in set (0.01 sec)
```

Fig 4.2.4 – Description of Agro products table

- **Triggers Table**

```
mysql> describe trig;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| id        | int          | NO   | PRI | NULL    | auto_increment |
| fid       | varchar(50)  | NO   |     | NULL    |                |
| action    | varchar(200) | NO   |     | NULL    |                |
| timestamp | datetime     | NO   |     | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
4 rows in set (0.01 sec)
```

Fig 4.2.5 – Description of Triggers table

4.3 Populated Tables

- **User Table**

```
mysql> select * from user;
+----+----------+-----------------+------------------------------------------------------------------------------------------------------------------------+
| id | username | email           | password                                                                                                               |
+----+----------+-----------------+------------------------------------------------------------------------------------------------------------------------+
|  6 | Sam      | Sam@gmail.com   | gAAAAABjxHEZazcgUVqvAJflWB-hrhvI9wBTvl9w6NPcUzh9ypkameYxvltfimioGhNIrKw7ii8o9ctS7YHrguZzi4rDArdPhw==                   |
|  7 | prateek  | prateek@gmail.com | gAAAAABjxP_t4BB0zyx0sEvs5PT7FFgV8G0Cps5580-5b13kG_jL4wJIGPA2VJFrUITdOAK0hangwqi2Zc6g8zgZAHtevTpOpg==                 |
|  8 | poornima | poornima@gmail.com | gAAAAABjxTWuPrvK7zAbuoZwbtuWzaLyp_iOIdvxhAUs8vG2g3PgdRroGGgpHzcYpfnWcIpcL6wlMyFGECuczujGj7-4TI9xbw==                 |
|  9 | Khush    | khush@gmail.com | pbkdf2:sha256:260000$p2E5wwkUvCx0q88O$df04736f3557638bb97a0cc6fb342daac6c07b961f970ebfce81057b48ef914d                   |
+----+----------+-----------------+------------------------------------------------------------------------------------------------------------------------+
```

Fig 4.3.1 – Values in User table

- **Registration Table**

```
mysql> select * from register;
+-----+------------+--------------+-----+--------+-------------+--------------+----------+
| rid | farmername | adharnumber  | age | gender | phonenumber | address      | farming  |
+-----+------------+--------------+-----+--------+-------------+--------------+----------+
|  13 | Abhi       | 879867898765 |  68 | male   | 9857876456  | Bangalore    | silk     |
|  15 | khush      | 987887908789 |  40 | female | 7898987667  | chennai      | Honeybee |
|  16 | sita       | 596284623789 |  49 | female | 9999987667  | Andrapradesh | Flowers  |
+-----+------------+--------------+-----+--------+-------------+--------------+----------+
3 rows in set (0.00 sec)
```

Fig 4.3.2 – Values in Registration table

- **Farming Table**

```
mysql> select * from farming;
+-----+--------------+
| fid | farmingtype  |
+-----+--------------+
|   1 | Seed Farming |
|   2 | coccon       |
|   3 | silk         |
|   4 | Diary        |
+-----+--------------+
4 rows in set (0.00 sec)
```

Fig 4.3.3 – Values in Farming table

- **AgroProducts Table**

```
mysql> select * from addagroproducts;
+----------+------------------+-----+-----------------+-----------------------------------------------------------------------------------------------------------------+-------+
| username | email            | pid | productname     | productdesc                                                                                                     | price |
+----------+------------------+-----+-----------------+-----------------------------------------------------------------------------------------------------------------+-------+
| Mohammed | mohammed@gmail.com |   1 | GIRIJA CAULIFLOWER |  Tips for Growing Cauliflower. Well drained medium loam and or sandy loam soils are suitable. |   520 |
| Manish   | ard@gmail.com    |   2 | COTTON          | Cotton is a soft, fluffy staple fiber that grows in a boll,around the seeds of the cotton       |   563 |
| Jatin    | jatin@gmail.com  |   3 | silk            | silk is best business developed from coocon for saries preparation and so on                    |   582 |
| Sam      | Sam@gmail.com    |   5 | Milk            | fresh milk every morning                                                                        |    50 |
+----------+------------------+-----+-----------------+-----------------------------------------------------------------------------------------------------------------+-------+
4 rows in set (0.00 sec)
```

Fig 4.3.4 – Values in Agro products table

- **Triggers Table**

```
mysql> select * from trig;
+----+-----+-------------------------+---------------------+
| id | fid | action                  | timestamp           |
+----+-----+-------------------------+---------------------+
|  6 |   6 | User Inserted: Sam      | 2023-01-16 03:03:13 |
|  7 |   9 | Farmer Inserted: Sam    | 2023-01-16 03:04:20 |
|  8 |   4 | Sam inserted Silk       | 2023-01-16 03:05:01 |
|  9 |  10 | Farmer Inserted: Edith  | 2023-01-16 03:06:30 |
| 10 |   5 | Sam inserted Milk       | 2023-01-16 03:06:58 |
| 11 |   7 | User Inserted: prateek  | 2023-01-16 13:12:37 |
| 12 |   9 | Farmer Deleted: Sam     | 2023-01-16 15:48:37 |
| 13 |  10 | Farmer Deleted: Edith   | 2023-01-16 15:48:41 |
| 14 |  11 | Farmer Inserted: Sam    | 2023-01-16 15:49:50 |
| 15 |  12 | Farmer Inserted: Sam    | 2023-01-16 15:52:28 |
| 16 |   8 | User Inserted: poornima | 2023-01-16 17:01:58 |
| 17 |   4 | Sam deleted Silk        | 2023-01-19 03:09:10 |
| 18 |  11 | Farmer Updated: Sam     | 2023-01-19 13:22:38 |
| 19 |  12 | Farmer Updated: Sam     | 2023-01-19 13:22:38 |
| 20 |  11 | Farmer Updated: Sam     | 2023-01-19 13:29:52 |
| 21 |  12 | Farmer Updated: Sam     | 2023-01-19 13:29:52 |
| 22 |  11 | Farmer Deleted: Sam     | 2023-01-19 13:30:15 |
| 23 |  12 | Farmer Deleted: Sam     | 2023-01-19 13:30:20 |
| 24 |  13 | Farmer Inserted: Abhi   | 2023-01-19 14:10:14 |
| 25 |  14 | Farmer Inserted: Kiran  | 2023-01-19 14:21:52 |
| 26 |  13 | Farmer Updated: Abhi    | 2023-01-19 14:22:07 |
| 27 |  14 | Farmer Updated: Abhi    | 2023-01-19 14:22:07 |
| 28 |   9 | User Inserted: Khush    | 2023-01-19 15:48:20 |
| 29 |  13 | Farmer Updated: Abhi    | 2023-01-19 15:49:32 |
| 30 |  14 | Farmer Updated: Abhi    | 2023-01-19 15:49:32 |
| 31 |  13 | Farmer Updated: Abhi    | 2023-01-20 20:40:19 |
| 32 |  14 | Farmer Updated: Abhi    | 2023-01-20 20:40:19 |
| 33 |  13 | Farmer Updated: Abhi    | 2023-01-20 21:09:15 |
| 34 |  14 | Farmer Updated: Abhi    | 2023-01-20 21:09:15 |
| 35 |  13 | Farmer Updated: Abhi    | 2023-01-21 00:10:13 |
| 36 |  14 | Farmer Updated: Abhi    | 2023-01-21 00:10:13 |
| 37 |  14 | Farmer Deleted: Abhi    | 2023-01-21 03:04:21 |
| 38 |  15 | Farmer Inserted: khush  | 2023-01-21 03:06:30 |
| 39 |  16 | Farmer Inserted: sita   | 2023-01-21 03:08:08 |
+----+-----+-------------------------+---------------------+
34 rows in set (0.00 sec)
```

Fig 4.3.5 – Values in Triggers table

4.4 SQL Triggers and Stored Procedure

Trigger for deletion of farmer details

- DELIMITER $$
  CREATE TRIGGER deletionreg BEFORE DELETE ON register FOR EACH ROW
  BEGIN INSERT INTO trig(fid, action, timestamp) VALUES(OLD.rid,CONCAT
  ('Farmer Deleted: ', OLD.farmername),NOW());
  END$$
  DELIMITER ;

Trigger for insertion of farmer details

- DELIMITER $$
  CREATE TRIGGER insertionreg AFTER INSERT ON register FOR EACH ROW
  BEGIN INSERT INTO trig(fid, action, timestamp) VALUES(NEW.rid,CONCAT
  ('Farmer Inserted: ', NEW.farmername),NOW());
  END$$
  DELIMITER ;

Trigger for updation or farmer details

- DELIMITER $$
  CREATE TRIGGER updationreg AFTER UPDATE ON register FOR EACH ROW
  BEGIN INSERT INTO trig(fid, action, timestamp) VALUES(NEW.rid,CONCAT
  ('Farmer Updated: ', NEW.farmername),NOW());
  END$$
  DELIMITER ;

Trigger for deletion of user

- DELIMITER $$
  CREATE TRIGGER deletionusr BEFORE DELETE ON user FOR EACH ROW
  BEGIN INSERT INTO trig(fid, action, timestamp) VALUES(OLD.id,CONCAT
  ('User Deleted: ', OLD.username),NOW());
  END$$
  DELIMITER ;

Trigger for registration of user

- DELIMITER $$
  CREATE TRIGGER insertionusr AFTER INSERT ON user FOR EACH ROW
  BEGIN INSERT INTO trig(fid, action, timestamp) VALUES(NEW.id,CONCAT
  ('User Inserted: ', NEW.username),NOW());
  END$$
  DELIMITER ;

Trigger for updation of user details

- DELIMITER $$
  CREATE TRIGGER updationusr AFTER UPDATE ON user FOR EACH ROW
  BEGIN INSERT INTO trig(fid, action, timestamp) VALUES(NEW.id,CONCAT
  ('User Updated: ', NEW.username),NOW());
  END$$
  DELIMITER ;

Trigger for deletion of agroproducts

- DELIMITER $$
  CREATE TRIGGER deletionagro BEFORE DELETE ON addagroproducts FOR EACH ROW BEGIN INSERT INTOtrig(fid,action,timestamp VALUES (OLD.pid,CONCAT (OLD.username, " deleted ", OLD.productname),NOW());
  END$$
  DELIMITER ;

Trigger for insertion of agroproducts

- DELIMITER $$
  CREATE TRIGGER insertionagro AFTER INSERT ON addagroproducts FOR EACH ROW BEGIN INSERT INTO trig(fid, action, timestamp) VALUES (NEW.pid,CONCAT (NEW.username, " inserted ", NEW.productname),NOW());
  END$$
  DELIMITER ;

Trigger for updation of agroproducts

- DELIMITER $$
  CREATE TRIGGER updationagro AFTER UPDATE ON addagroproducts FOR EACH ROW BEGIN INSERT INTO trig(fid, action, timestamp) VALUES (NEW.pid,CONCAT (NEW.username, " updated ", NEW.productname),NOW());
  END$$
  DELIMITER ;

- **Stored Procedure:** Here is a stored procedure for validation of mobile number. As determined validate takes input mobile number and verify it is 10 digit or not and whether the first digit is starting from 7 to 9 and rest must be from 0 to 9.

```
1  CREATE DEFINER=`root`@`localhost` PROCEDURE `validate`()
2  BEGIN
3  IF not(select phone regexp '^[6-9]([0-9]{9})$') then
4      signal sqlstate '45000' SET message_text='Enter valid 12 digit aadhaar number';
5  END IF;
6  END
```

Fig 4.4 – Stored procedure screen shot

4.5 Database Connectivity

Database connection is a facility in computer science that allows client software to talk to database server software, whether on the same machine or not. A connection is required to send commands and receive answers, usually in the form of a result set. Flask is a micro web frame work written in python. Micro framework is normally a framework with little to no dependencies on external libraries.

There are three steps to make Python Flask database interaction
1. Create an app.py
2. Setting up SQLAlchemy and Configuration with the Database
3. Perform databasequery

- **Creating an app.py**

```
from flask import Flask, render_template, request, session, redirect, url_for, flash
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin
from flask_login import login_user, logout_user, login_manager, LoginManager
from flask_security import current_user, login_required
from flask_mail import Mail, Message
from cryptography.fernet import Fernet
import re

phone = re.compile(r'^([6-9])([0-9]{9})$')
aadhar = re.compile(r'^\d{12}')

local_server = True
app = Flask(__name__)
app.secret_key = b'INH4OKQ_O1Vv2Zl_uwYdutE8CVhlbEfb2LhYPkmV9K0='
```

Fig 4.5.1 – Creating an app.py

- **Setting up SQLAlchemy and Configuration with the Database**

```
app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql://root:12345@localhost/farmers'
db = SQLAlchemy(app)
```

Fig 4.5.2 – Setting up SQLAlchemy and config with database

4.6 Modules

The below flowchart explains how the system runs in the real world. The system can easily be implemented under various situations Reusability is possible as and when required in this application. There is feasibility in all the modules which makes task of the user easier.

The User can login with his credentials if he is already an existing user or else can register by giving required details and then can register himself as a farmer to sell products by registering their products separately and can also include an extra farming type. The customer can click on agro products to see which products are available and can order by contacting the seller or farmer.

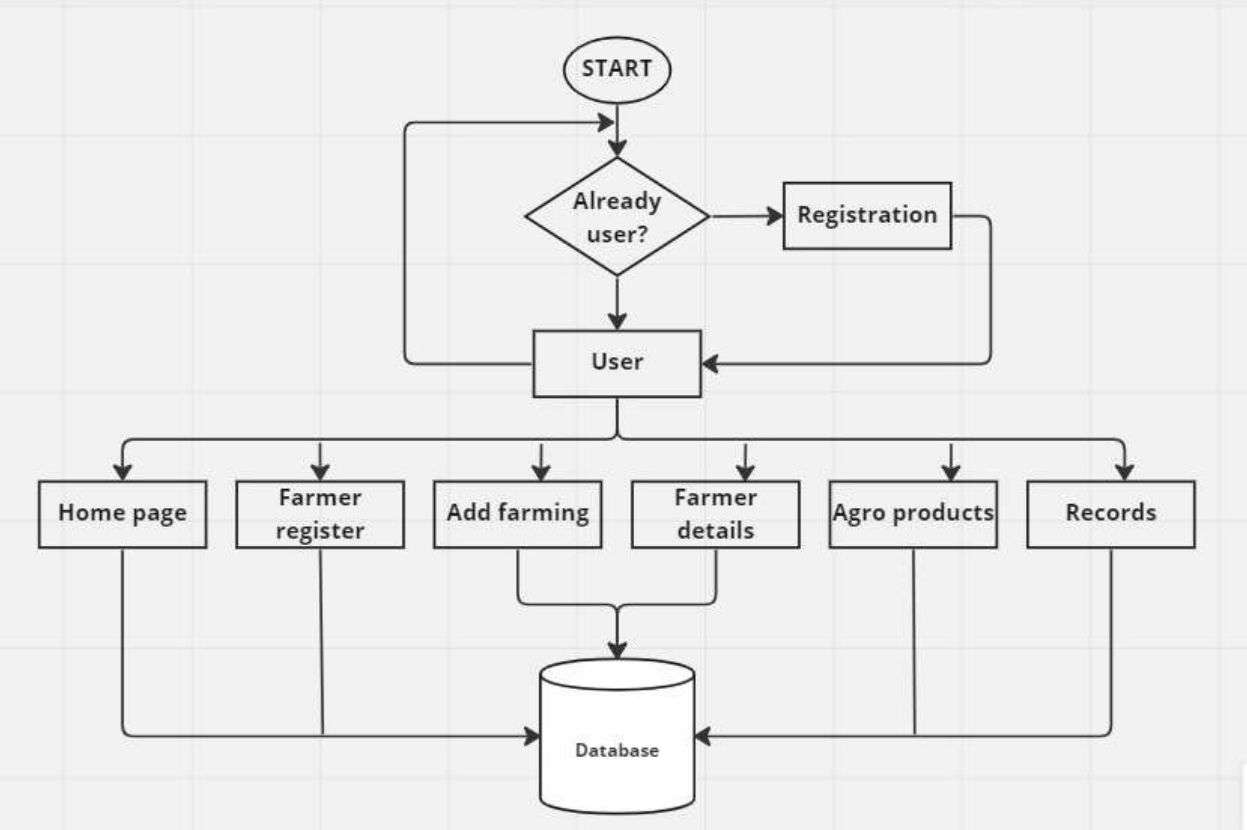
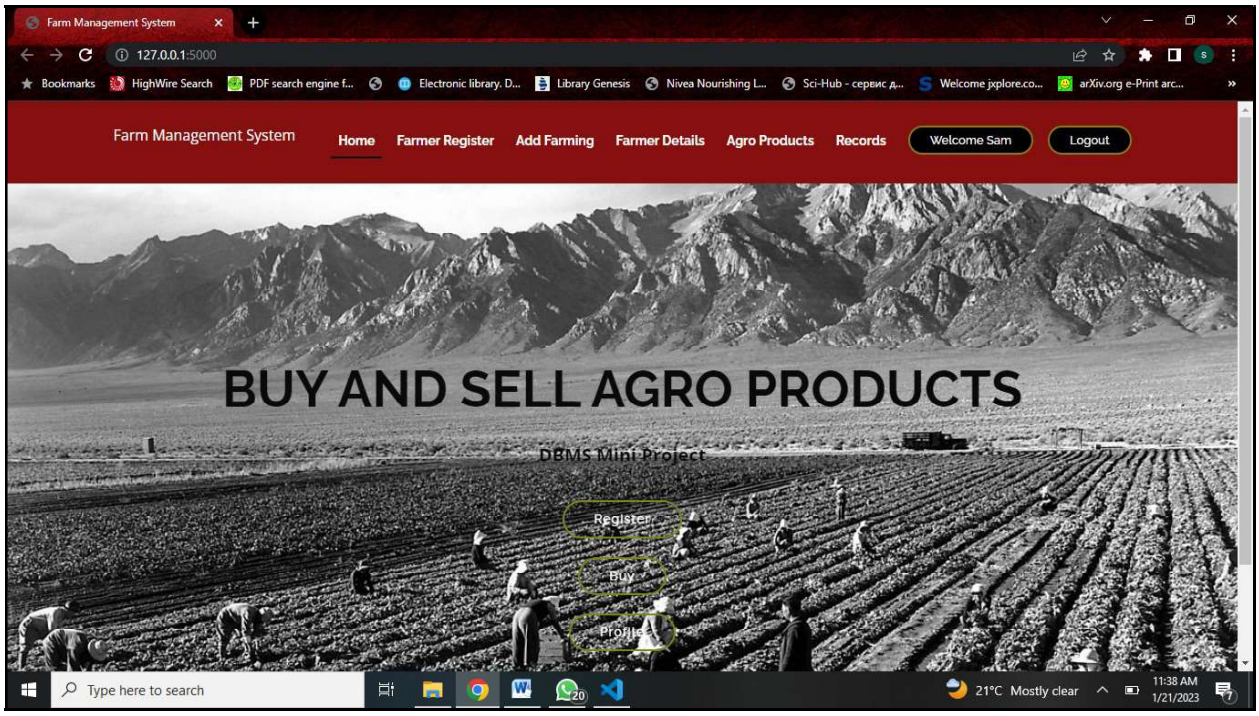


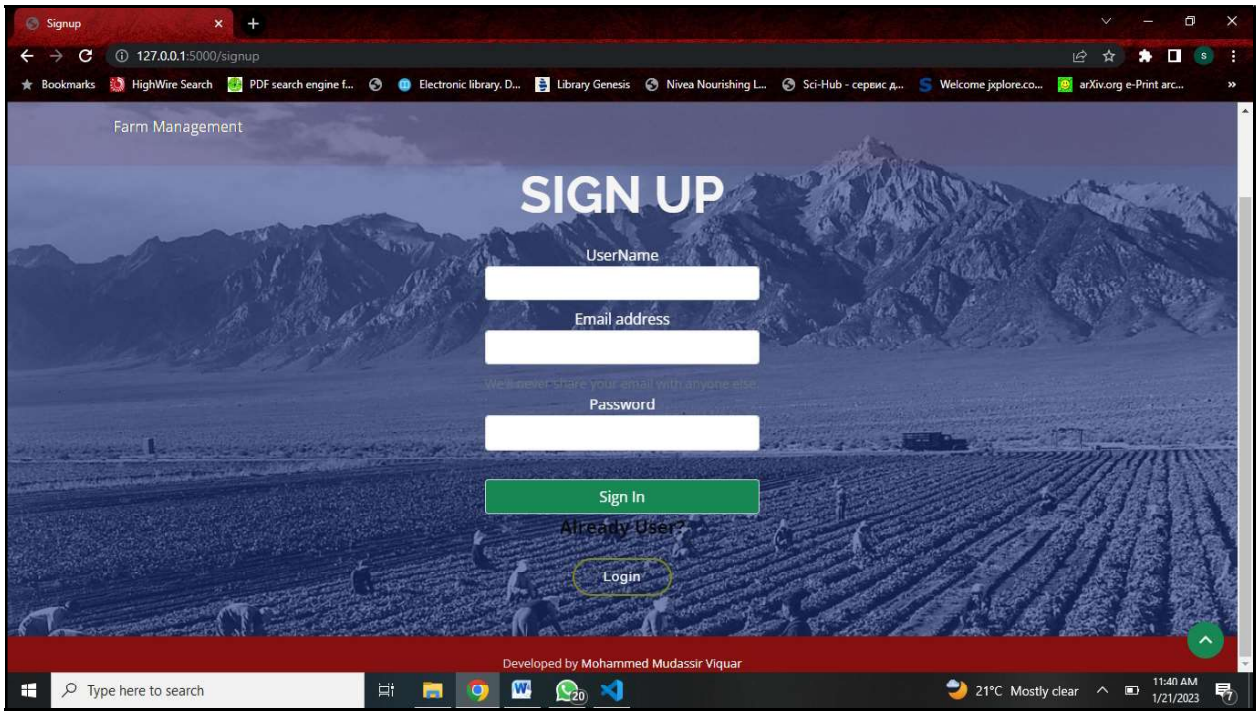
Fig 4.6 – Modules for Farm management system and their dataflow

Chapter 5

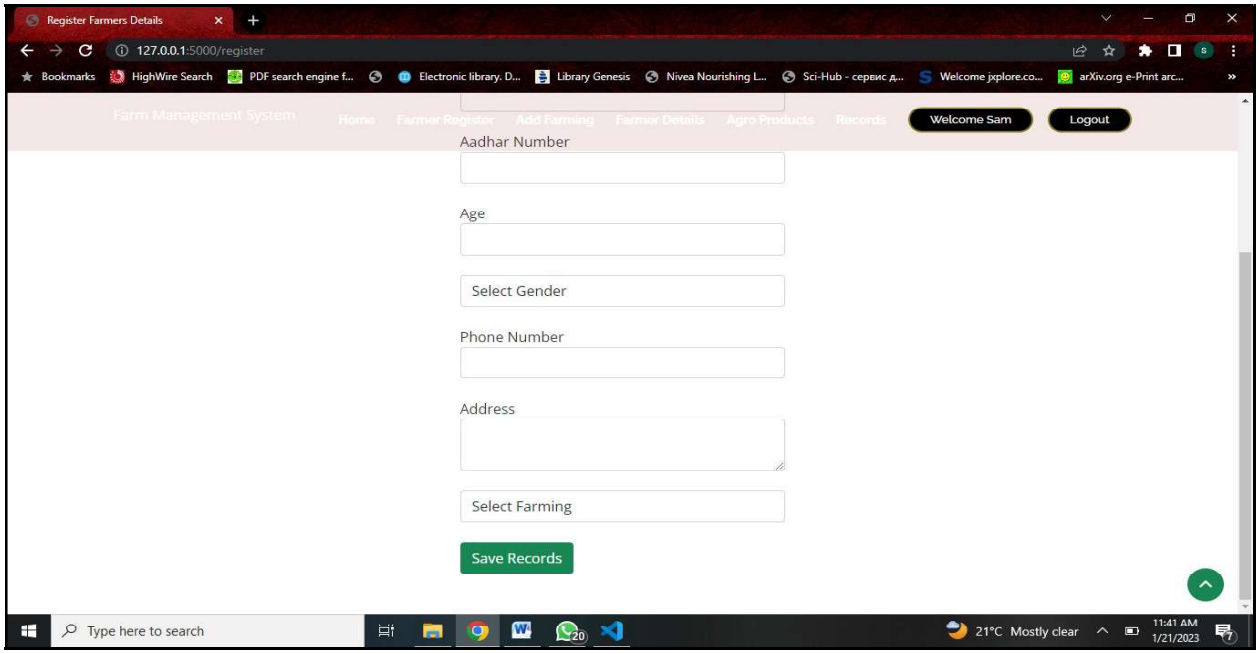
RESULTS



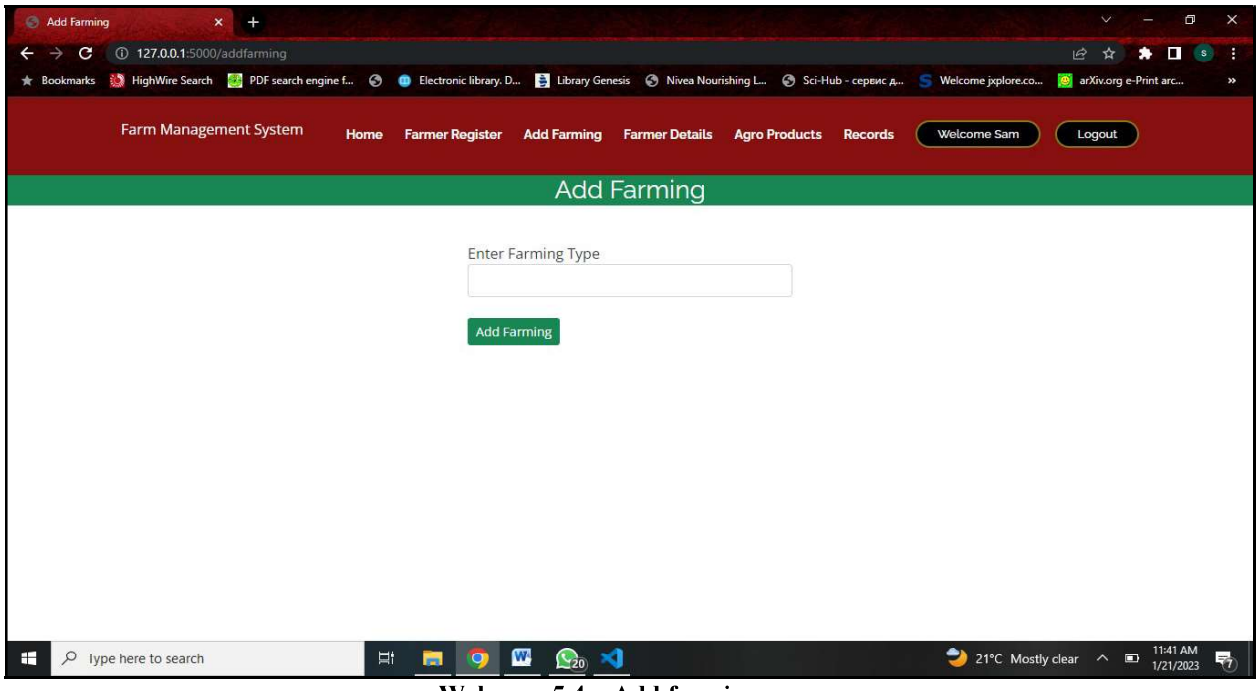
Webpage 5.1 – Home page



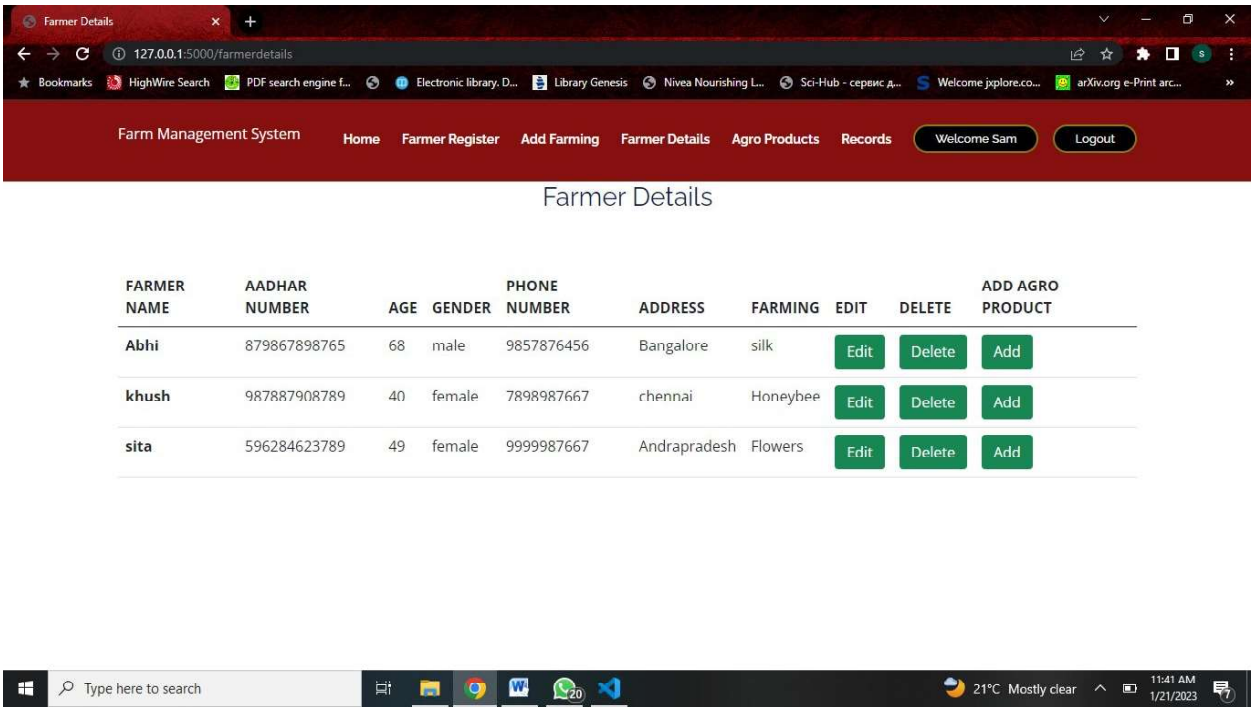
Webpage 5.2 – Signup page



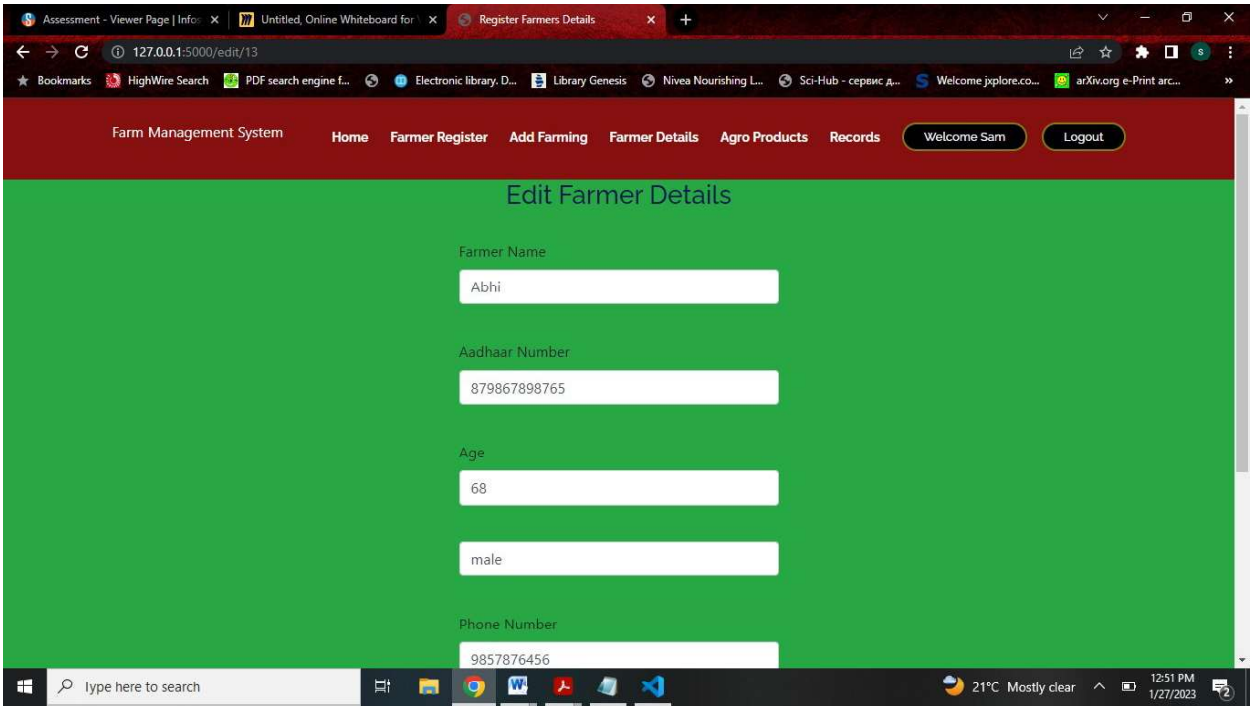
Webpage 5.3 – Farmers registration



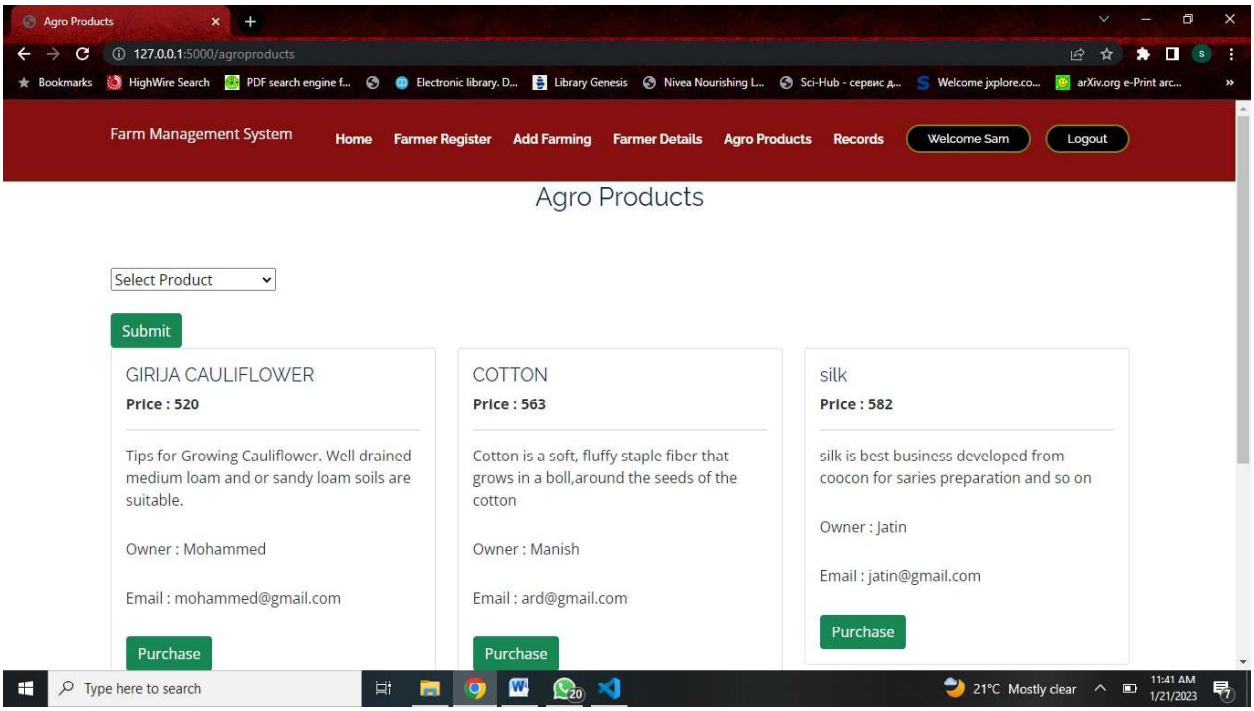
Webpage 5.4 – Add farming page



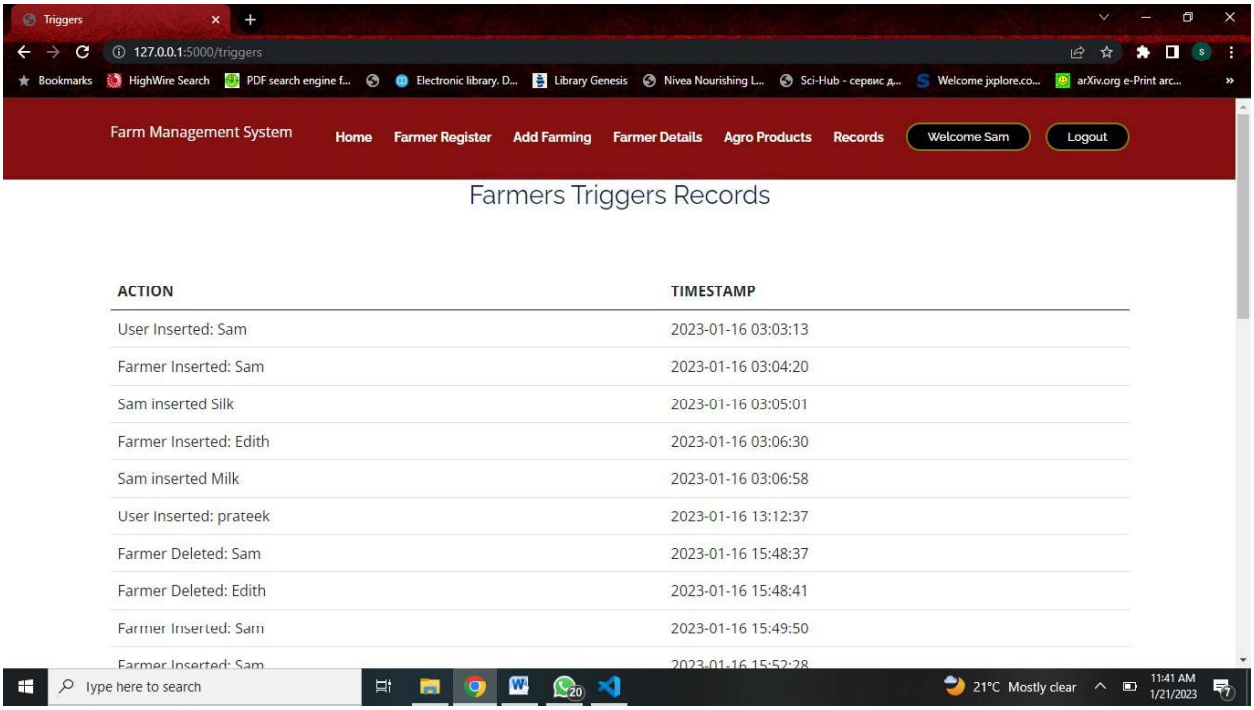
Webpage 5.5 – Farmer details page



Webpage 5.6 – Edit farmer details page



Webpage 5.7 – Agro products page



Webpage 5.8 – Triggers record page

Chapter 6

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