

Farmer Management System

A PROJECT REPORT

Submitted by

| | |
|----------------------|-----------------------|
| SHAMSHAD ALAM | (220101120029) |
| RAJ KAPOOR | (220101120042) |
| SUNIL KUMAR | (220101120041) |
| SUMANT KUMAR | (220101120045) |
| WAHAB KHAN | (220101120040) |

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING



Centurion
UNIVERSITY

*Shaping Lives...
Empowering Communities...*

SCHOOL OF ENGINEERING AND TECHNOLOGY

PARALAKHEMUNDI CAMPUS

CENTURION UNIVERSITY OF TECHNOLOGY AND MANAGEMENT

ODISHA

JANUARY 2024

SPECIMEN CERTIFICATE

**DEPARTMENT OF COMPUTER SCIENCE &
ENGINEERING SCHOOL OF ENGINEERING AND TECHNOLOGY
PARALAKHEMUNDI CAMPUS**

BONAFIDE CERTIFICATE

Certified that this project report “**Farmer Management System**” is the bonafide work of “**RAJ KAPOOR**” who carried out the project work under my supervision. This is to further certify to the best of my knowledge, that this project has not been carried out earlier in this institute and the university.

SIGNATURE

(Prof. Rama Chandra Padhy)

**Assistant Professor Of
School of Engineering and Technology**

*Certified that the above mentioned project has been duly carried out as per
the norms of the college and statutes of the university.*

SIGNATURE

DEPARTMENT SEAL

HEAD OF THE DEPARTMENT / DEAN OF THE SCHOOL

Professor of School of Engineering and Technology

DECLARATION

I hereby declare that the project entitled “**FARMER MANAGEMENT SYSTEM**” submitted for the “**Database Creation And Maintenance project**” of **4th** semester B. Tech in Computer Science Engineering, is my original work and the project has not formed the basis for the award of any Degree / Diploma or any other similar titles in any other University / Institute.

Name of the Student:

Signature of the Student:

Registration No:

Place:

Date:

ACKNOWLEDGEMENTS

I wish to express my profound and sincere gratitude to **Prof.Rama Chandra Padhy**, School Of Engineering CUTM, parlakhemundi Campus, who guided me into the intricacies of this project nonchalantly with matchless magnanimity. I thank **Prof. Debendra Maharana**, Head of the Dept. of Department of Computer Science and Engineering, SoET, Parlakhemundi Campus and **Dr. Prafulla Kumar Panda** Dean School of Engineering and Technology, Parlakhemundi Campus for extending their support during Course of this investigation.

Name of the Student:

Signature of the Student:

Registration No:

Place:

Date:

TABLE OF CONTENTS

| CHAPTER NO. | TITLE | PAGE NO. |
|-------------|--|-----------|
| | CERTIFICATE | i |
| | DECLARATION | ii |
| | ACKNOWLEDGEMENT | iii |
| | LIST OF TABLE | iv |
| | LIST OF FIGURES | v |
| | ABSTRACT | vi |
| 1. | INTRODUCTION..... | 01 |
| 1.1 | Objectives | |
| 2. | STUDY OF EXISTING SYSTEM..... | 02 |
| 2.1 | Case study | |
| 2.2 | Proposed system | |
| 3. | METHODOLOGY..... | 09 |
| 3.1 | Software requirement specification | |
| 3.1.1 | Software requirements | |
| 3.1.2 | hardware requirements | |
| 3.2 | Conceptual design | |
| 3.2.1 | ER diagram | |
| 3.2.2 | Schema diagram | |
| 3.3 | Implementation | |
| 3.3.1 | Backend | |
| 3.3.1.1 | Database | |
| 3.3.1.2 | SQL | |
| 3.3.1.3 | Source code | |
| 3.3.2 | Frontend | |
| 3.3.3 | Trigger | |
| 3.3.4 | Stored procedure | |
| 4. | USER INTERFACES..... | 24 |
| 4.1 | Web page screen shots | |
| 4.2 | Database screen shots | |
| 5. | CONCLUSION AND FUTURE ENHANCEMENT | 30 |
| 6. | REFERENCE..... | 31 |

LIST OF FIGURES

| | |
|--------------------------------------|----|
| Fig. 3.2.1.1 E-R DIAGRAM | 3 |
| Fig. 3.2.2.1 SCHEMA DIAGRAM | 4 |
| Fig. 4.1.1. Home Page | 19 |
| Fig. 4.1.2. Sign up Page | 19 |
| Fig. 4.1.3. Register Farmer Details | 20 |
| Fig. 4.1.4. Agro Products | 20 |
| Fig. 4.1.5. Farmers triggers Records | 21 |
| Fig. 4.1.6. Add Farming | 21 |
| Fig. 4.1.7. Farmers Details | 22 |
| Fig. 4.1.8. ADD AGRO PRODUCTS | 22 |
| Fig. 4.2.1. Register data in XAMP | 23 |
| Fig. 4.2.2 Trig data in XAMP | 23 |
| Fig. 4.2.3 Add agro product IN XAMP | 24 |
| Fig. 4.2.4 User & password XAMP | 24 |

ABSTRACT

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 products 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 agricultural products online. Buyer can send purchase request to check the quality of the Agro product through mails

1.1 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.

2.1 CASE STUDY

Source Trace is collaborating with Small Farmers Agri-business consortium (SFACH) and Karnataka Horticulture Department, deploying its digital solutions to support the horticulture farmers of India. Karnataka Agriculture Department is committed to providing a responsive and effective mechanism for the welfare of farmers and farm-based communities and recognizes the need to harness the growing power of Information Technologies for the betterment of life of the farmers and management of Farmer Producer Organizations (FPOs) in Haryana. To deploy its digital solution, Source Trace is in the process of creating 100,000 farmer profiles. The system was developed using technologies such as, HTML, CSS ,JS and MySQL. PYTHON- FLASK, HTML and CSS are used to build the user interface and database was built using MySQL. The system is free of errors and very efficient and less time consuming due to the care taken to develop it. All the phases of software development cycle are employed and it is worthwhile to state that the system is very robust. Provision is made for future development in the system.

2.2 PROPOSED 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.

3.1 SOFTWARE REQUIREMENTS SPECIFICATION

3.1.1 SOFTWARE REQUIREMENTS:

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

3.1.2 HARDWARE REQUIREMENTS:

- 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

3.2 CONCEPTUAL DESIGN:

3.2.1 E-R DIAGRAM:

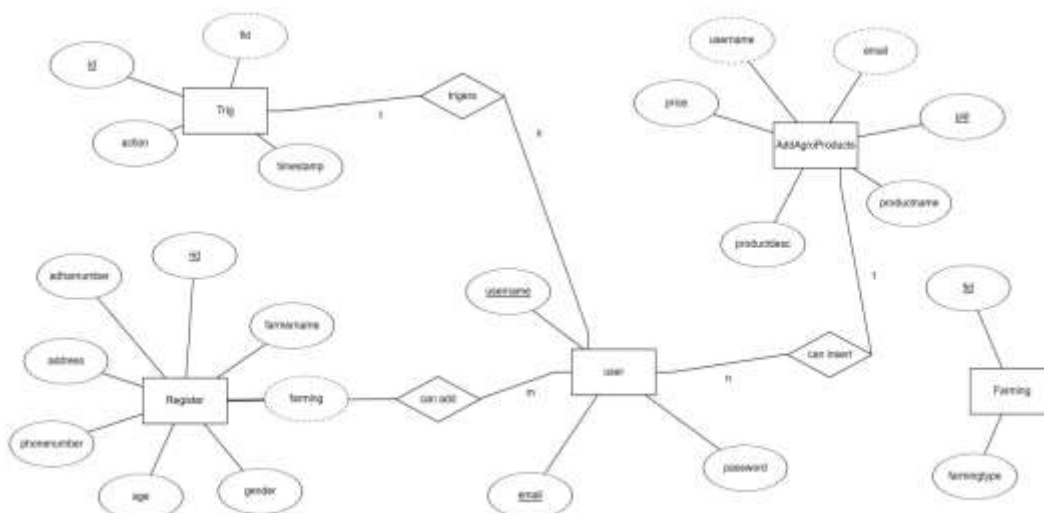


Fig.3.2.1.1

3.2.2 SCHEMA DIAGRAM:

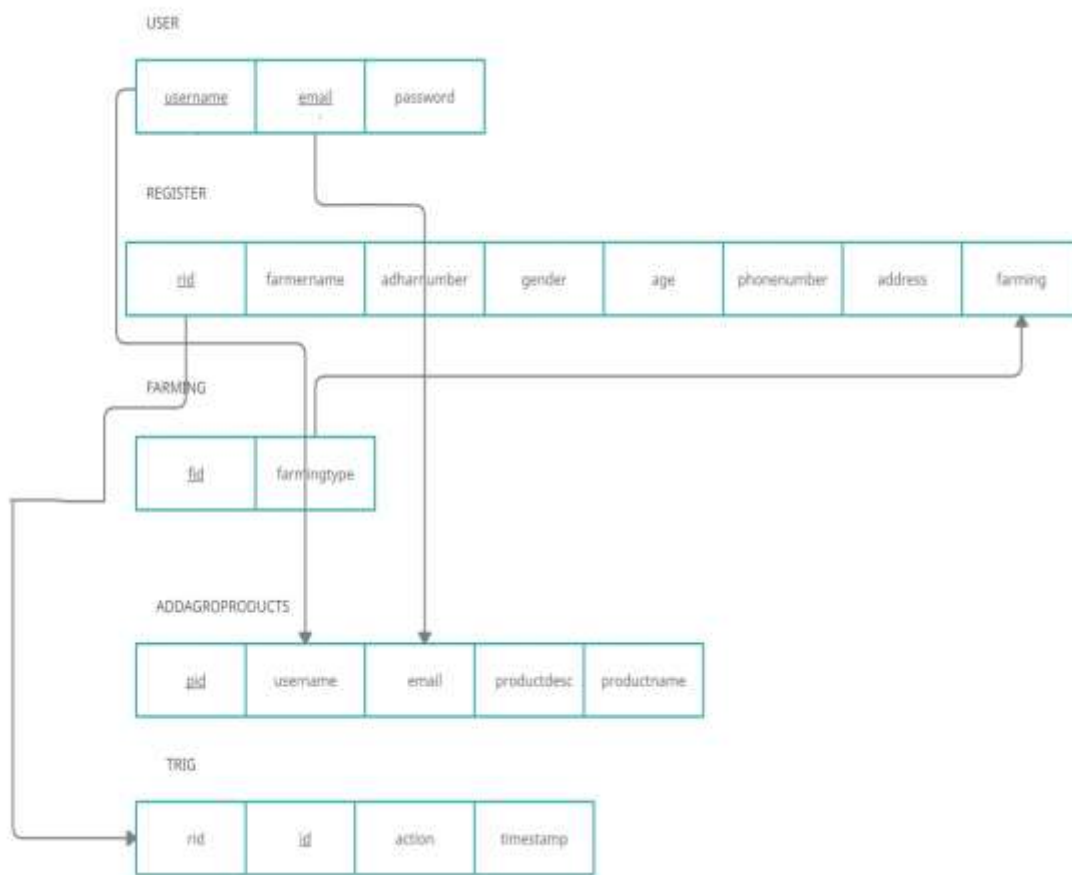


Fig.3.2.2.1

3.3 IMPLEMENTATION:

An "implementation" of Python should be taken to mean a program or environment which provides support for the execution of programs written in the Python language, as represented by the [CPython](#) reference implementation.

There have been and are several distinct software packages providing of what we all recognize as Python, although some of those are more like distributions or variants of some existing implementation than a completely new implementation of the language.

3.3.1 Backend(MySQL)

3.3.1.1 Database

A Database Management System (DBMS) is computer software designed for the purpose of managing databases, a large set of structured data, and run operations on the data requested by numerous users. Typical examples of DBMSs include Oracle, DB2, Microsoft Access, Microsoft SQL Server, Firebird, PostgreSQL, MySQL, SQLite, FileMaker and Sybase Adaptive Server Enterprise. DBMSs are typically used by Database administrators in the creation of Database systems. Typical examples of DBMS use include accounting, human resources and customer support systems. Originally found only in large companies with the computer hardware needed to support large data sets, DBMSs have more recently emerged as a fairly standard part of any company back office.

A DBMS is a complex set of software programs that controls the organization, storage, management, and retrieval of data in a database. A DBMS includes:

- A modeling language to define the schema of each database hosted in the DBMS, according to the DBMS data model.
- The dominant model in use today is the ad hoc one embedded in SQL, despite the objections of purists who believe this model is a corruption of the relational model, since it violates several of its fundamental principles for the sake of practicality and performance. Many DBMSs also support the Open Database Connectivity API that supports a standard way for programmers to access the DBMS.

Data structures (fields, records, files and objects) optimized to deal with very large amounts of data stored on a permanent data storage device (which implies relatively slow access compared to volatile main memory). A database query language and report writer to allow users to interactively interrogate the database, analyze its data and update it according to the users privileges on data.

- Data security prevents unauthorized users from viewing or updating the database. Using passwords, users are allowed access to the entire database or subsets of it called sub schemas. For example, an employee database can contain all the data about an individual employee, but one group of users may be authorized to view only payroll data, while others are allowed access to only work history and student data.
- If the DBMS provides a way to interactively enter and update the database, as well as interrogate it, this capability allows for managing personal databases. However, it may not leave an audit trail of actions or provide the kinds of controls necessary in a multi-user organization. These controls are only available when a set of application programs are customized for each data entry and updating function.

- ✓ A transaction mechanism, that ideally would guarantee the ACID properties, in order to ensure data integrity, despite concurrent user accesses (concurrency control), and faults (fault tolerance).
 - It also maintains the integrity of the data in the database.
 - The DBMS can maintain the integrity of the database by not allowing more than one user to update the same record at the same time. The DBMS can help prevent duplicate records via unique index constraints; for example, no two customers with the same customer numbers (key fields) can be entered into the database. See ACID properties for more information (Redundancy avoidance).

When a DBMS is used, information systems can be changed much more easily as the organization's information requirements change. Organizations may use one kind of DBMS for daily transaction processing and then move the detail onto another computer that uses another DBMS better suited for random inquiries and analysis. Overall systems design decisions are performed by data administrators and systems analysts. Detailed database design is performed by database administrators.

3.3.1.2 SQL

Structured Query Language (SQL) is the language used to manipulate relational databases.

SQL is tied very closely with the relational model.

- In the relational model, data is stored in structures called relations or tables.
SQL statements are issued for the purpose of:
- Data definition: Defining tables and structures in the database (DDL used to create, alter and drop schema objects such as tables and indexes) .

3.3.1.3 Source code :

```
from flask import Flask,render_template,request,session,redirect,url_for,flash
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin
from werkzeug.security import generate_password_hash,check_password_hash
from flask_login import login_user,logout_user,login_manager,LoginManager
from flask_login import login_required,current_user

# MY db connection local_server=
True app = Flask(__name__)
```

```

app.secret_key='harshithbhaskar'
# this is for getting unique user access login_manager=LoginManager(app)
login_manager.login_view='login'
@login_manager.user_loader def
load_user(user_id):
    return User.query.get(int(user_id))
#
app.config['SQLALCHEMY_DATABASE_URL']='mysql://username:password@localhost/databas_table_name'
app.config['SQLALCHEMY_DATABASE_URI']='mysql://root:@localhost/farmers' db=SQLAlchemy(app)
# here we will create db models that is tables class
Test(db.Model):
    id=db.Column(db.Integer,primary_key=True)    name=db.Column(db.String(100))

class Farming(db.Model):
    fid=db.Column(db.Integer,primary_key=True)    farmingtype=db.Column(db.String(100))
class Addagroproducts(db.Model):    username=db.Column(db.String(50))
email=db.Column(db.String(50))
pid=db.Column(db.Integer,primary_key=True)
productname=db.Column(db.String(100))
productdesc=db.Column(db.String(300))    price=db.Column(db.Integer)

class Trig(db.Model):
    id=db.Column(db.Integer,primary_key=True)
fid=db.Column(db.String(100))    action=db.Column(db.String(100))
timestamp=db.Column(db.String(100))

class User(UserMixin,db.Model):    id=db.Column(db.Integer,primary_key=True)
username=db.Column(db.String(50))
email=db.Column(db.String(50),unique=True)
password=db.Column(db.String(1000))

class Register(db.Model):
    rid=db.Column(db.Integer,primary_key=True)
farmername=db.Column(db.String(50))
adharnumber=db.Column(db.String(50))
age=db.Column(db.Integer)    gender=db.Column(db.String(50))
phonenumber=db.Column(db.String(50))
address=db.Column(db.String(50))

```

```

farming=db.Column(db.String(50))

@app.route('/')

def index():
    return render_template('index.html')

@app.route('/farmerdetails')
@login_required def
farmerdetails():
    query=db.engine.execute(f"SELECT * FROM `register`")    return
render_template('farmerdetails.html',query=query)

@app.route('/agroproducts') def
agroproducts():
    query=db.engine.execute(f"SELECT * FROM `addagroproducts`")    return
render_template('agroproducts.html',query=query)

@app.route('/addagroproduct',methods=['POST','GET'])
@login_required def
addagroproduct():    if
request.method=="POST":
    username=request.form.get('username')    email=request.form.get('email')
    productname=request.form.get('productname')
productdesc=request.form.get('productdesc')    price=request.form.get('price')

products=Addagroproducts(username=username,email=email,productname=productname,productdesc=prod
uctdesc,price=price)    db.session.add(products)    db.session.commit()    flash("Product Added","info")
return redirect('/agroproducts')

    return render_template('addagroproducts.html')

@app.route('/triggers')
@login_required def
triggers():
    query=db.engine.execute(f"SELECT * FROM `trig`")    return
render_template('triggers.html',query=query)

@app.route('/addfarming',methods=['POST','GET'])
@login_required def addfarming():
if request.method=="POST":
    farmingtype=request.form.get('farming')
    query=Farming.query.filter_by(farmingtype=farmingtype).first()    if
query:
    flash("Farming Type Already Exist","warning")
return redirect('/addfarming')
dep=Farming(farmingtype=farmingtype)

```

```

        db.session.add(dep)        db.session.commit()
flash("Farming Addes","success")    return
render_template('farming.html')

@app.route("/delete/<string:rid>",methods=['POST','GET'])
@login_required def
delete(rid):
    db.engine.execute(f'DELETE FROM `register` WHERE `register`.`rid`={rid}')
flash("Slot Deleted Successful","danger")    return redirect('/farmerdetails')
@app.route("/edit/<string:rid>",methods=['POST','GET'])
@login_required def
edit(rid):
    farming=db.engine.execute("SELECT * FROM `farming`")
posts=Register.query.filter_by(rid=rid).first()    if
request.method=="POST":
    farmername=request.form.get('farmername')
adharnumber=request.form.get('adharnumber')
age=request.form.get('age')        gender=request.form.get('gender')
phonenumner=request.form.get('phonenumner')
address=request.form.get('address')
    farmingtype=request.form.get('farmingtype')        query=db.engine.execute(f'UPDATE
`register` SET
farmername`='{farmername}','adharnumber`='{adharnumber}','age`='{age}','gender`='{gender}','phonenu
mber`='{phonenumner}','address`='{address}','farming`='{farmingtype}')        flash("Slot is
Updates","success")        return redirect('/farmerdetails')
    return render_template('edit.html',posts=posts,farming=farming)

@app.route('/signup',methods=['POST','GET']) def
signup():    if request.method == "POST":
username=request.form.get('username')
email=request.form.get('email')
password=request.form.get('password')
print(username,email,password)
    user=User.query.filter_by(email=email).first()        if
user:
    flash("Email Already Exist","warning")
return render_template('/signup.html')
encpassword=generate_password_hash(password)
    new_user=db.engine.execute(f'INSERT INTO `user` (`username`,`email`,`password`) VALUES

```



```

('{username}','{email}','{encpassword}'))

# this is method 2 to save data in db
# newuser=User(username=username,email=email,password=encpassword)
# db.session.add(newuser)      #
db.session.commit()

flash("Signup Succes Please Login","success")      return
render_template('login.html')

return render_template('signup.html')

@app.route('/login',methods=['POST','GET']) def
login():  if request.method == "POST":
email=request.form.get('email')
password=request.form.get('password')
user=User.query.filter_by(email=email).first()
    if user and check_password_hash(user.password,password):
        login_user(user)      flash("Login
Success","primary")      return
redirect(url_for('index'))      else:
        flash("invalid credentials","danger")
return render_template('login.html')

return render_template('login.html')

@app.route('/logout')
@login_required def
logout():  logout_user()
    flash("Logout SuccessFul","warning")      return
redirect(url_for('login'))
@app.route('/register',methods=['POST','GET'])
@login_required def
register():

    farming=db.engine.execute("SELECT * FROM `farming`")      if
request.method=="POST":
        farmername=request.form.get('farmername')
adharnumber=request.form.get('adharnumber')
age=request.form.get('age')      gender=request.form.get('gender')
phonenumner=request.form.get('phonenumner')

```

```

address=request.form.get('address')
farmingtype=request.form.get('farmingtype')
query=db.engine.execute(f'INSERT INTO `register`
(`farmername`,`adharnumber`,`age`,`gender`,`phonenumner`,`address`,`farming`) VALUES
('{farmername}','{adharnumber}','{age}','{gender}','{phonenumner}','{address}','{farmingtype}')')")
flash("Your Record Has Been Saved","success")    return redirect('/farmerdetails')
return render_template('farmer.html',farming=farming)

```

```

@app.route('/test') def
test():    try:
    Test.query.all()
    return 'My database is Connected'    except:
    return 'My db is not Connected'
app.run(debug=True)

```

3.3.2 FRONT END CODE

```

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta content="width=device-width, initial-scale=1.0" name="viewport">

<title>{% block title %}
{% endblock title %}</title>
<meta content="" name="description">
<meta content="" name="keywords">
{% block style %}
{% endblock style %}
<link
href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,700,700i|Raleway:300,400,50
0,700,800" rel="stylesheet">

<!-- Vendor CSS Files -->
<link href="static/assets/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">
<link href="static/assets/vendor/venobox/venobox.css" rel="stylesheet">
<link href="static/assets/vendor/font-awesome/css/font-awesome.min.css" rel="stylesheet">

```

```
<link href="static/assets/vendor/owl.carousel/assets/owl.carousel.min.css" rel="stylesheet"> <link
href="static/assets/vendor/aos/aos.css" rel="stylesheet">
```

```
<!-- Template Main CSS File -->
```

```
<link href="static/assets/css/style.css" rel="stylesheet">
```

```
</head>
```

```
<body>
```

```
<!-- ===== Header ===== -->
```

```
<header id="header">
```

```
<div class="container">
```

```
<div id="logo" class="pull-left">
```

```
<a href="/" class="scrollto">F.M.S</a>
```

```
</div>
```

```
<nav id="nav-menu-container">
```

```
<ul class="nav-menu">
```

```
<li class="{% block home %}
```

```
{% endblock home %}"><a href="/">Home</a></li>
```

```
<li><a href="/register">Farmer Register</a></li>
```

```
<li><a href="/addfarming">Add Farming</a></li>
```

```
<li><a href="/farmerdetails">Farmer Details</a></li>
```

```
<li><a href="/agroproducts">Agro Products</a></li>
```

```
<li><a href="/triggers">Records</a></li>
```

```
{% if current_user.is_authenticated %}
```

```
<li class="buy-tickets"><a href="">Welcome {{current_user.username}}</a></li>
```

```
<li class="buy-tickets"><a href="/logout">Logout</a></li>
```

```
{% else %}
```

```
<li class="buy-tickets"><a href="/signup">Signin</a></li>
```

```
{% endif %}
```

```
</ul>
```

```
</nav><!-- #nav-menu-container -->
```

```
</div>
```

```
</header><!-- End Header -->
```

```
<!-- ===== Intro Section ===== -->
```

```
<section id="intro">
```

```
<div class="intro-container" data-aos="zoom-in" data-aos-delay="100">
```

```
<h1 class="mb-4 pb-0">SELL AGRO PRODUCTS AND BUY </span> </h1>
```

```
<p class="mb-4 pb-0">DBMS Mini Project Using Flask & MYSQL</p>
```

```
<a href="/agroproducts" class="about-btn scrollto">AGRO PRODUCTS</a>
```

```
</div>
```

```
</section><!-- End Intro Section -->
```

```
<main id="main">
```

```
{% block body %}
```

```
{% with messages=get_flashed_messages(with_categories=true) %}
```

```
{% if messages %}
```

```
{% for category, message in messages %}
```

```
<div class="alert alert-{{category}} alert-dismissible fade show" role="alert">
```

```
  {{message}}
```

```
</div>
```

```
{% endfor %}
```

```
{% endif %}
```

```
{% endwith %}
```

```
{% endblock body %}
```

```
<a href="#" class="back-to-top"><i class="fa fa-angle-up"></i></a>
```

```
<!-- Vendor JS Files -->
```

```
<script src="static/assets/vendor/jquery/jquery.min.js"></script>
```

```
<script src="static/assets/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
```

```
<script src="static/assets/vendor/jquery.easing/jquery.easing.min.js"></script>
```

```
<script src="static/assets/vendor/php-email-form/validate.js"></script>
```

```
<script src="static/assets/vendor/venobox/venobox.min.js"></script>
```

```
<script src="static/assets/vendor/owl.carousel/owl.carousel.min.js"></script>
```

```
<script src="static/assets/vendor/superfish/superfish.min.js"></script>
<script src="static/assets/vendor/hoverIntent/hoverIntent.js"></script>
<script src="static/assets/vendor/aos/aos.js"></script>
```

```
<!-- Template Main JS File -->
<script src="static/assets/js/main.js"></script>
```

```
</body>
```

```
</html> <!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">
  <title>{% block title %}
  {% endblock title %}</title>
  <meta content="" name="description">
  <meta content="" name="keywords">
```

```
{% block style %}
{% endblock style %}
```

```
<link
href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,700,700i|Raleway:300,400,50
0,700,800" rel="stylesheet">
```

```
<!-- Vendor CSS Files -->
<link href="static/assets/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">
<link href="static/assets/vendor/venobox/venobox.css" rel="stylesheet">
<link href="static/assets/vendor/font-awesome/css/font-awesome.min.css" rel="stylesheet">
<link href="static/assets/vendor/owl.carousel/assets/owl.carousel.min.css" rel="stylesheet"> <link
href="static/assets/vendor/aos/aos.css" rel="stylesheet">
```

```
<!-- Template Main CSS File -->
<link href="static/assets/css/style.css" rel="stylesheet">
```

```
</head>
```

```

<body>

<!-- ===== Header ===== -->
<header id="header">
  <div class="container">
    <div id="logo" class="pull-left">

      <a href="/" class="scrollto">F.M.S</a>

    </div>

    <nav id="nav-menu-container">
      <ul class="nav-menu">
        <li class="{% block home %}"
          {% endblock home %}"><a href="/">Home</a></li>

        <li><a href="/register">Farmer Register</a></li>
        <li><a href="/addfarming">Add Farming</a></li>
        <li><a href="/farmerdetails">Farmer Details</a></li>
        <li><a href="/agroproducts">Agro Products</a></li>
        <li><a href="/triggers">Records</a></li>

        {% if current_user.is_authenticated %}
          <li class="buy-tickets"><a href="">Welcome {{current_user.username}}</a></li>
          <li class="buy-tickets"><a href="/logout">Logout</a></li>
        {% else %}
          <li class="buy-tickets"><a href="/signup">Signin</a></li>

        {% endif %}
      </ul>
    </nav><!-- #nav-menu-container -->
  </div>
</header><!-- End Header -->

<!-- ===== Intro Section ===== -->
<section id="intro">
  <div class="intro-container" data-aos="zoom-in" data-aos-delay="100">
    <h1 class="mb-4 pb-0">SELL AGRO PRODUCTS AND BUY </span> </h1>
    <p class="mb-4 pb-0">DBMS Mini Project Using Flask & MYSQL</p>
  </div>
</section>

```

```

    <a href="/agroproducts" class="about-btn scrollTo">AGRO PRODUCTS</a>
</div>
</section><!-- End Intro Section -->
<main id="main">

{% block body %}

{% with messages=get_flashed_messages(with_categories=true) %}
{% if messages %}
{% for category, message in messages %}

<div class="alert alert-{{category}} alert-dismissible fade show" role="alert">
    {{message}}

</div>

{% endfor %}
{% endif %}
{% endwith %}
{% endblock body %}

<a href="#" class="back-to-top"><i class="fa fa-angle-up"></i></a>

<!-- Vendor JS Files -->
<script src="static/assets/vendor/jquery/jquery.min.js"></script>
<script src="static/assets/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
<script src="static/assets/vendor/jquery.easing/jquery.easing.min.js"></script>
<script src="static/assets/vendor/php-email-form/validate.js"></script>
<script src="static/assets/vendor/venobox/venobox.min.js"></script>
<script src="static/assets/vendor/owl.carousel/owl.carousel.min.js"></script>
<script src="static/assets/vendor/superfish/superfish.min.js"></script>
<script src="static/assets/vendor/hoverIntent/hoverIntent.js"></script>
<script src="static/assets/vendor/aos/aos.js"></script>
<!-- Template Main JS File -->
<script src="static/assets/js/main.js"></script>
</body>

</html>

```

3.3.3 Trigger

It is the special kind of stored procedure that automatically executes when an event occurs in the database.

Triggers used :

1: Trigger name: on insert

Table: register

Time: after

Event: insert

INSERT INTO trig VALUES(null,NEW.rid,'Farmer Inserted',NOW())

2: Trigger name: on delete

Table: register

Time: after

Event: delete

Definition:INSERT INTO trig VALUES(null,OLD.rid,'FARMER DELETED',NOW())

3: Trigger name: on update

Table: register

Time: after

Event: update

Definition: INSERT INTO trig VALUES(null,NEW.rid,'FARMER UPDATED',NOW())

3.3.4 Stored procedure

Routine name: proc

Type: procedure

Definition: Select * from register;

4.1 WEB PAGE SCREEN SHOTS

HOME PAGE: This is the home page of farmer management system, there is different options like Farmer Register, Add Farming, Farmer Datils, Agro Products, records and Sign in and Sign up options are available in the home page.



Fig. 4.1.1. Home Page

Sign Up: This interface indicates that here Seller create there own id with valid email and password

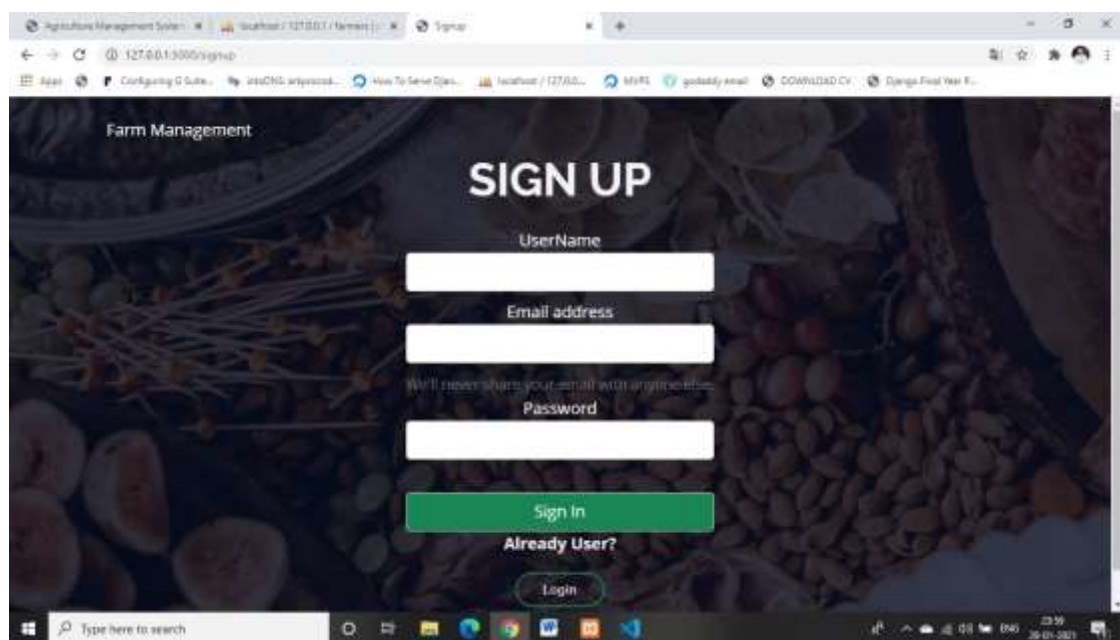
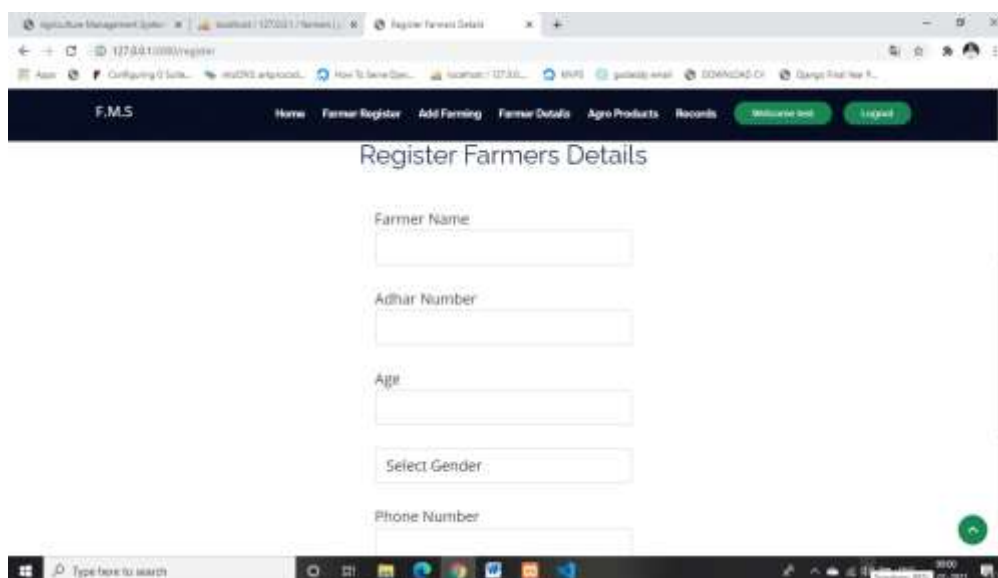


Fig.4.1.2. Sign up Page

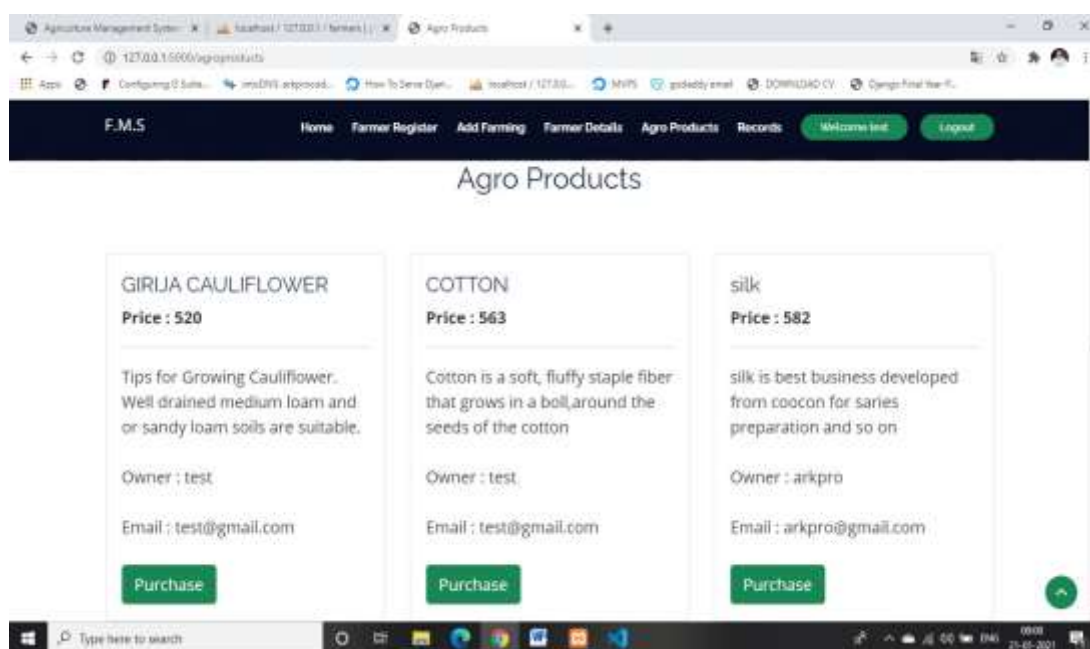
Register Farmer Details : This is the interface where Farmer add their personal details such as Father Name, Addhar Number, Age, Gender, Contact Number, e.t.c .



The screenshot shows a web browser window with the URL 'localhost:127.0.0.1/register'. The page has a dark blue header with the text 'F.M.S' and navigation links: 'Home', 'Farmer Register', 'Add Farming', 'Farmer Details', 'Agro Products', and 'Records'. There are also 'Welcome test' and 'Logout' buttons. The main content area is titled 'Register Farmers Details' and contains a form with the following fields: 'Farmer Name', 'Adhar Number', 'Age', 'Select Gender', and 'Phone Number'. A green circular button with a white arrow is located at the bottom right of the form.

Fig. 4.1.3. Register Farmer Details

Agro Products: In this page it shows list of products whose added by farmer and customer also buy the products using mail .

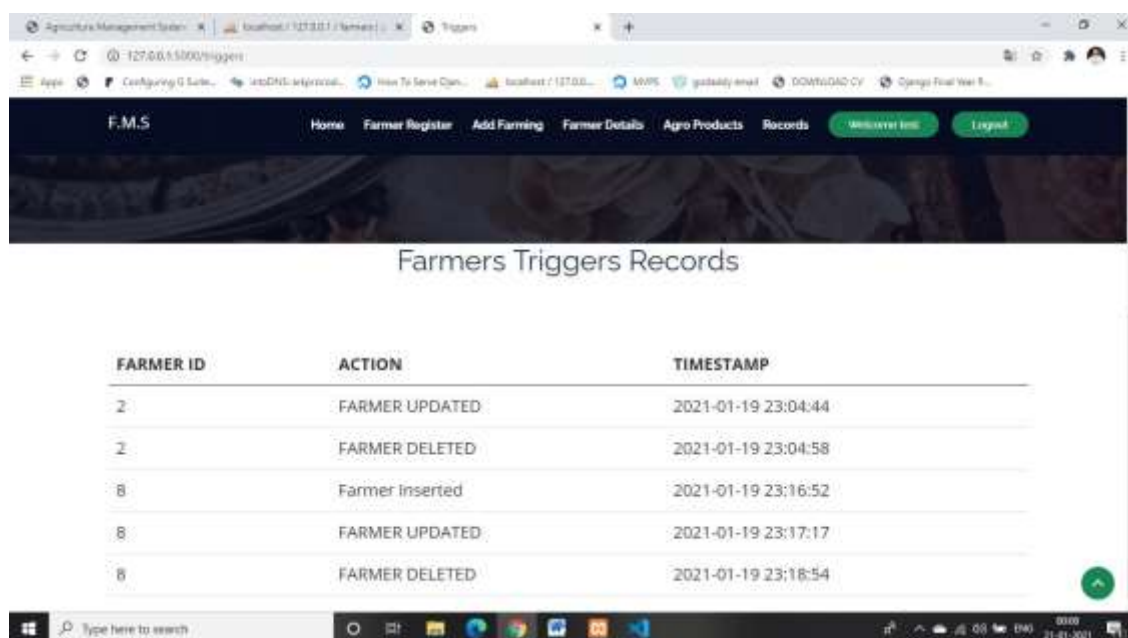


The screenshot shows a web browser window with the URL 'localhost:127.0.0.1/agroproducts'. The page has a dark blue header with the text 'F.M.S' and navigation links: 'Home', 'Farmer Register', 'Add Farming', 'Farmer Details', 'Agro Products', and 'Records'. There are also 'Welcome test' and 'Logout' buttons. The main content area is titled 'Agro Products' and displays three product cards. Each card has a title, price, description, owner information, email, and a 'Purchase' button.

| Product Name | Price | Description | Owner | Email |
|--------------------|-------|--|--------|------------------|
| GIRIJA CAULIFLOWER | 520 | Tips for Growing Cauliflower. Well drained medium loam and or sandy loam soils are suitable. | test | test@gmail.com |
| COTTON | 563 | Cotton is a soft, fluffy staple fiber that grows in a boll, around the seeds of the cotton | test | test@gmail.com |
| silk | 582 | silk is best business developed from cocoon for series preparation and so on | arkpro | arkpro@gmail.com |

Fig. 4.1.4. Agro Products

Farmers triggers Records: In this interface it shows the all types of updated details of farmers.

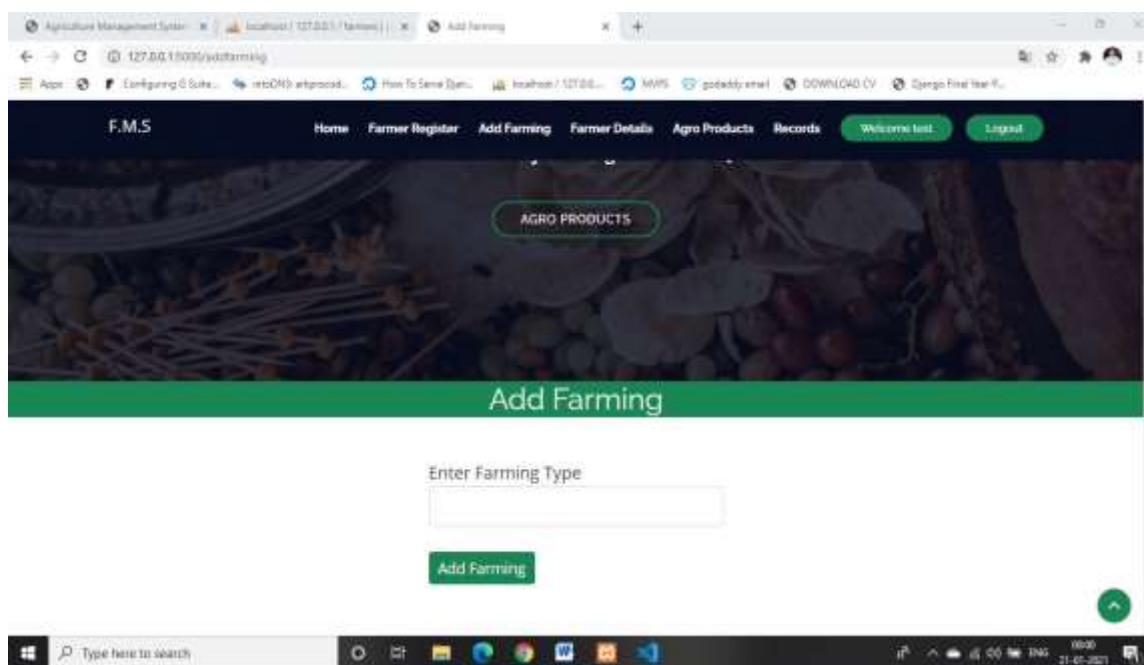


The screenshot shows a web browser window with the URL `localhost:127.0.0.1/farmers/`. The page title is "Farmers Triggers Records". The navigation bar includes "Home", "Farmer Register", "Add Farming", "Farmer Details", "Agro Products", and "Records". The main content area displays a table with the following data:

| FARMER ID | ACTION | TIMESTAMP |
|-----------|-----------------|---------------------|
| 2 | FARMER UPDATED | 2021-01-19 23:04:44 |
| 2 | FARMER DELETED | 2021-01-19 23:04:58 |
| 8 | Farmer Inserted | 2021-01-19 23:16:52 |
| 8 | FARMER UPDATED | 2021-01-19 23:17:17 |
| 8 | FARMER DELETED | 2021-01-19 23:18:54 |

Fig. 4.1.5. Farmers triggers Records

Add farming: This is the interface where farmers can add the crop types for sell purpose.



The screenshot shows a web browser window with the URL `localhost:127.0.0.1/farmers/addfarming/`. The page title is "Add Farming". The navigation bar includes "Home", "Farmer Register", "Add Farming", "Farmer Details", "Agro Products", and "Records". The main content area displays a form with the following elements:

- A green button labeled "AGRO PRODUCTS" at the top.
- A green bar with the text "Add Farming" below the button.
- A text input field labeled "Enter Farming Type".
- A green button labeled "Add Farming" below the input field.

Fig. 4.1.6. Add Farming

Farmers Details: In this interface it shows the farmer details and that farmer can Edit, Delete and Add their detail.

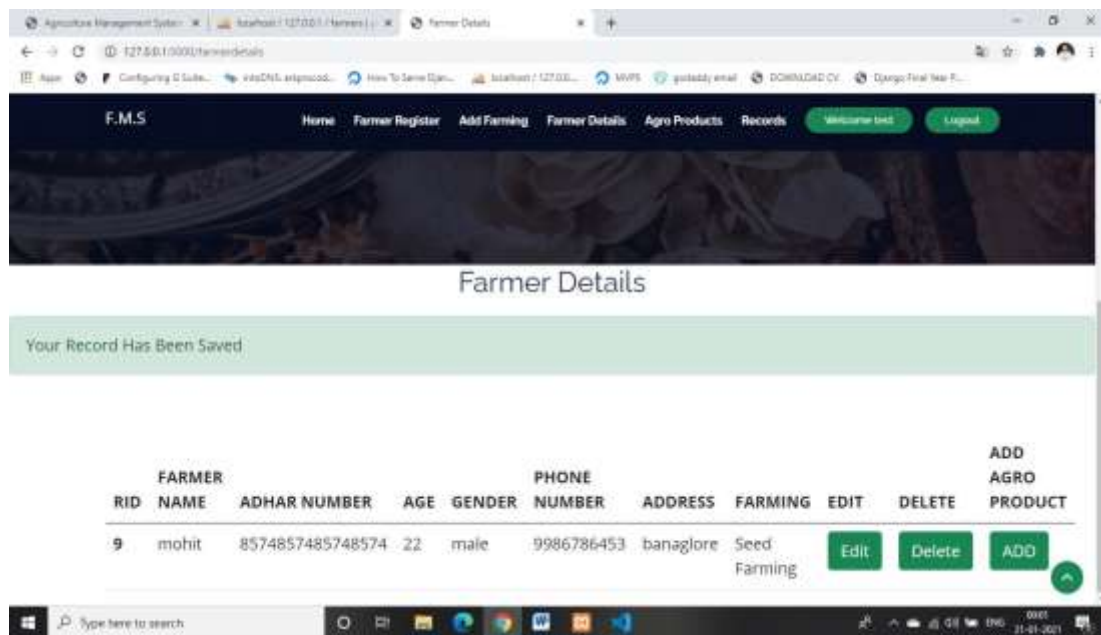


Fig. 4.1.7. Farmers Details

ADD AGRO PRODUCTS : This is the interface where farmer add the product and selling contact details and also add the product price.

The screenshot shows the 'Add Agro Products' page. It features a form with the following fields: 'Farmer Email' (containing 'test@gmail.com'), 'Product Name' (containing 'cotton'), 'Product Description' (containing 'make silk sarees'), and 'Price' (containing '500'). A green 'Add Product' button is located at the bottom of the form. The navigation bar at the top is identical to the previous screenshot, with the 'Agro Products' link highlighted.

Fig. 4.1.8. ADD AGRO PRODUCTS

4.2 DATABASE SCREEN SHOTS

REGISTER DATABASE: This is local xamp server interface where all registered data store in the database.

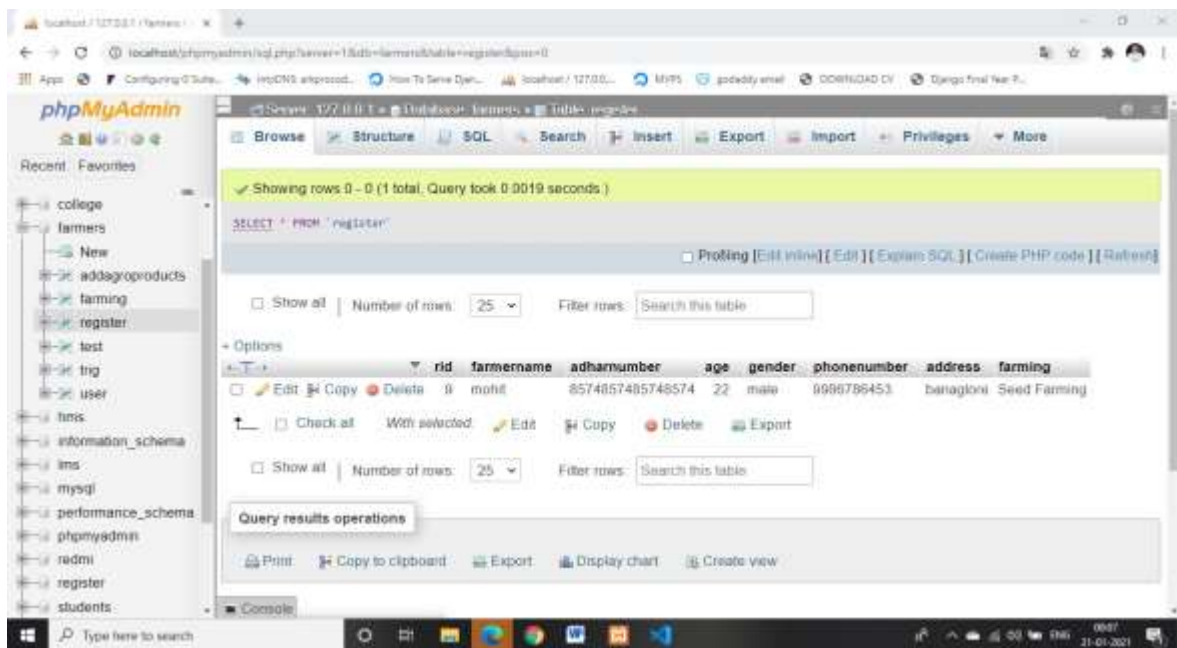


Fig. 4.2.1. Register data in XAMP

Trig database: In this interface trig type of data store in XAMP server, such as farmer details update, delete, insert, add.

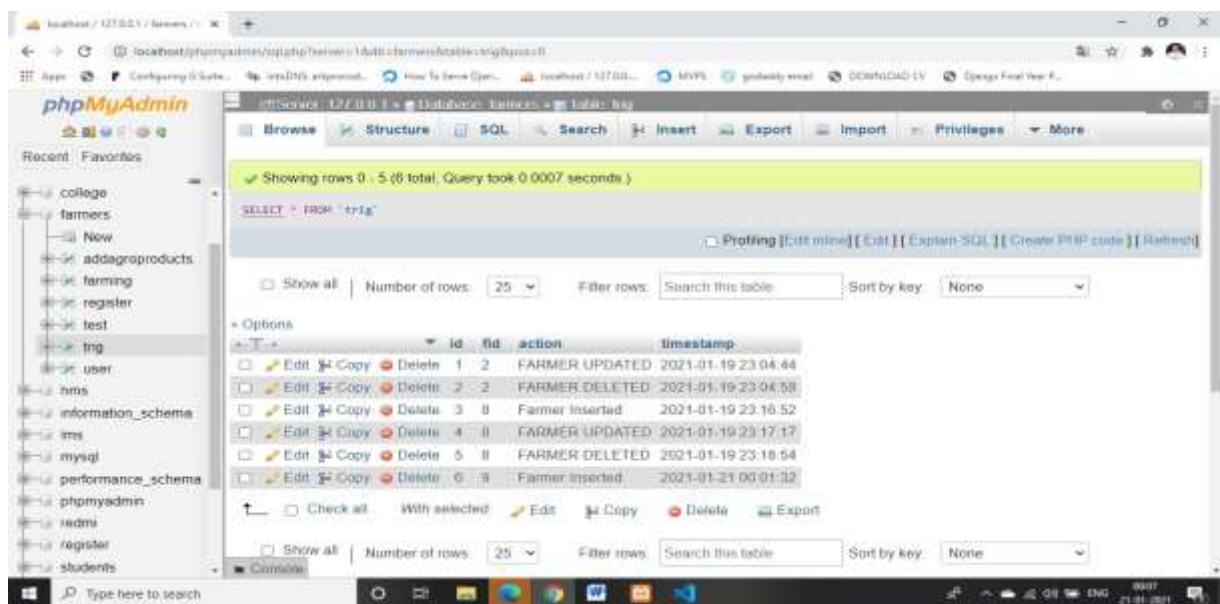


Fig. 4.2.2 Trig data in XAMP

Add agro product: This interface add agro product data store in XAMP server . show in the figure below.

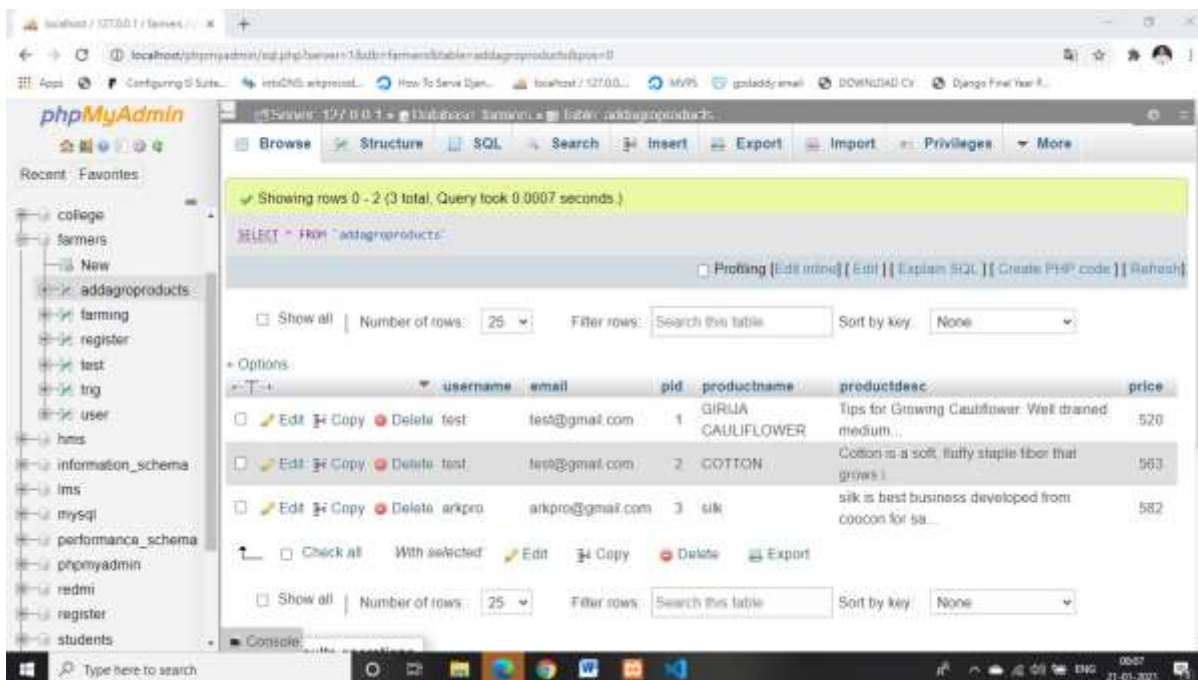


Fig. 4.2.3 Add agro product IN XAMP

User & password: This interface of the xamp server database where, store the data of farmer username , email, password in the database.

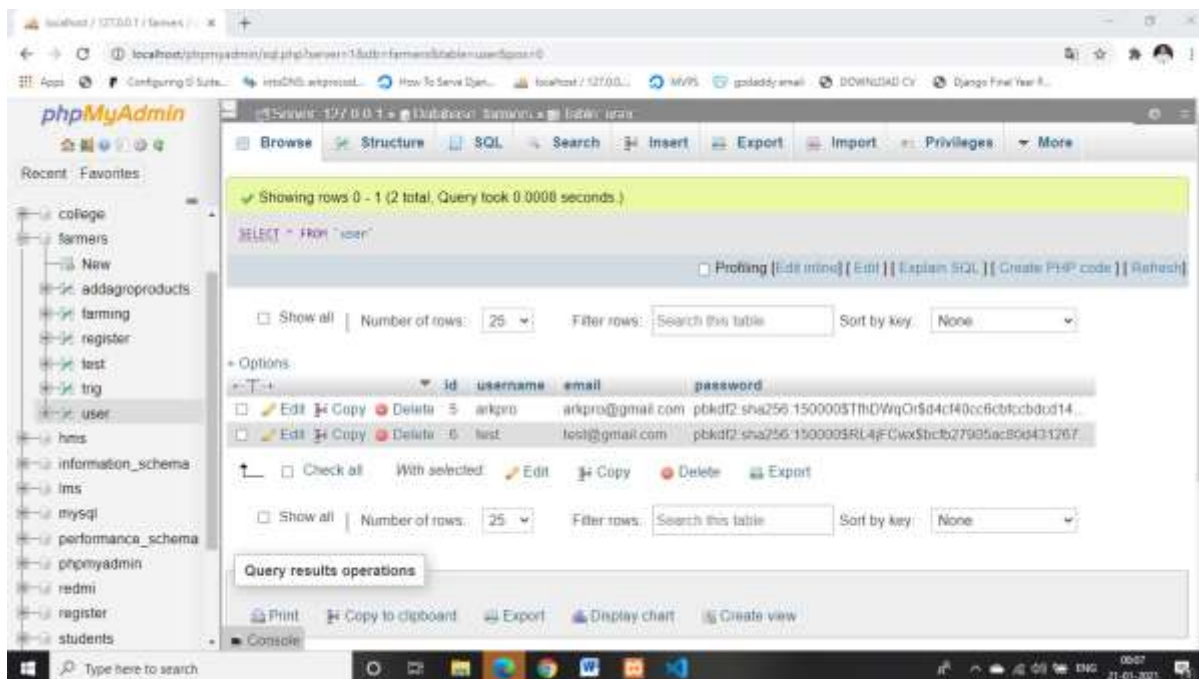


Fig. 4.2.4 User & password XAMP

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 utmost 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

- **More advanced results systems :** Now that we provide a simple interface where we can use the mail system to buy the products, we will do it in an it in an online mode where customers can buy directly from the website by using their current address.
- **Online payments:** In the future, we will provide an online payment system through an UPI ID, scanner, credit, or debit card. After adding these features, customers can pay directly on the webpage.

REFERENCE

- <https://www.youtube.com>
- <https://www.google.com>
- <http://www.getbootstrap.co>

ASSESSMENT

Internal:

| SL NO | RUBRICS | FULL MARK | MARKS OBTAINED | REMARKS |
|----------|---|--------------|-------------------|---------|
| 1 | Understanding the relevance, scope and dimension of the project | 10 | | |
| 2 | Methodology | 10 | | |
| 3 | Quality of Analysis and Results | 10 | | |
| 4 | Interpretations and Conclusions | 10 | | |
| 5 | Report | 10 | | |
| | Total | 50 | | |

Date:

Signature of the Faculty

COURSE OUTCOME (COs)

ATTAINMENT

Expected Course Outcomes (COs):

(Refer to COs Statement in the Syllabus)

1. Introduction to the Android platform for Mobile Application Development.
2. Understand Native Android Application, Android SDK features, Android Virtual Device (AVD), SDK manager, The Android Application Lifecycle.
3. Introduction Android Database, Introduction SQLite, and Content value working with SQLite Databases.

Course Outcome Attained:

How would you rate your learning of the subject based on the specified COs?

| | | | | | | | | | |
|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|-------------------------------------|
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

LOW

HIGH

Learning Gap (if any):

NO GAP.

Books / Manuals Referred:

1. Head First Android Development
 2. http://yuliana.lecturer.pens.ac.id/Android/Buku/professional_android_4_application_development.pdf
-
-



**Centurion
UNIVERSITY**

*Shaping Lives...
Empowering Communities...*

CENTURION UNIVERSITY OF TECHNOLOGY AND MANAGEMENT, ODISHA

CAMPUSES:

Paralakhemundi Campus

Village Alluri Nagar
P.O. – R Sitapur, Via- Uppalada
Paralakhemundi, Dist.- Gajapati
Odisha, India. PIN- 761211

Bhubaneswar Campus

Ramchandrapur
P.O. – Jatni, Bhubaneswar
Dist.- Khurda, Odisha,
India, PIN- 752050

Balangir Campus

Behind BSNL Office
IDCO land, Rajib Nagar
Dist.- Balangir, Odisha
India, PIN-767001

Rayagada Campus

IDCO Industrial Area
Pitamahal, Rayagada
Dist.-Rayagada, Odisha
India, PIN-765001

Balasore Campus

Gopalpur,
P.O.-Balasore
Dist.-Balasore, Odisha
India, PIN-756044

Chatrapur Campus

Ramchandrapur,
Kaliabali Chhak,
P.O-Chatrapur, Dist.-Ganjam
Odisha, India, PIN-761020