Farm Management System

A COURSE PROJECT REPORT

By

Anusha C (RA2111003010461)

Khushi Kumari (RA2111003010483)

Anuj Singh (RA2111003010459)

Under the guidance of **Dr. Baranidharan B**

In partial fulfillment for the Course

18CSC303J-Database Management System

In

School of Computing

FACULTY OF ENGINEERING AND TECHNOLOGY SRM INSTITUTE OFSCIENCE AND TECHNOLOGY

Kattankulathur, Chengalpattu District

APRIL 2024.

Acknowledgement

We would like to express our gratitude to our Professor, Dr. Baranidharan B who gave us the golden opportunity to do this wonderful project on the topic **"FARM MANAGEMENT SYSTEM"** which also helped us in doinga lot of research and we came to know about so many new things we are really thankful to him.

We are also thankful to all the other faculty, teaching and non-teaching staff members of our department for their kind co-operation and help.

Lastly, we would also like to thank our friends who helped us a lot in finishing this project within the limited time. We are making this project not only for marks but to also increase our knowledge.

ANUSHA C (RA211100301046)

KHUSHI KUMARI(RA2111003010483)

ANUJ SING (RA2111003010459)

Index

CHAPTER-1

1. INTRODUCTION

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 farmer's co-ordination.
- Reducing loss.

1.2 LIMITATIONS:

- Small size of farm business: Due to fragmentation and subdivision of holding the average size of operational holdings is very small
- Less labor per unit areas is required to farm large areas, especially since expensive alterations to land (like terracing) are completely absent.
- Mechanization can be used more effectively over large, flat areas

CHAPTER-2

STUDY OF EXISTING SYSTEM

2.1 CASE STUDY STUDY

SourceTrace 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

CHAPTER-3

DATABASE DESIGN

3.1.1 SOFTWARE REQUIREMENTS SPECIFICATION

3.1.2 SOFTWARE REQUIREMENTS:

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

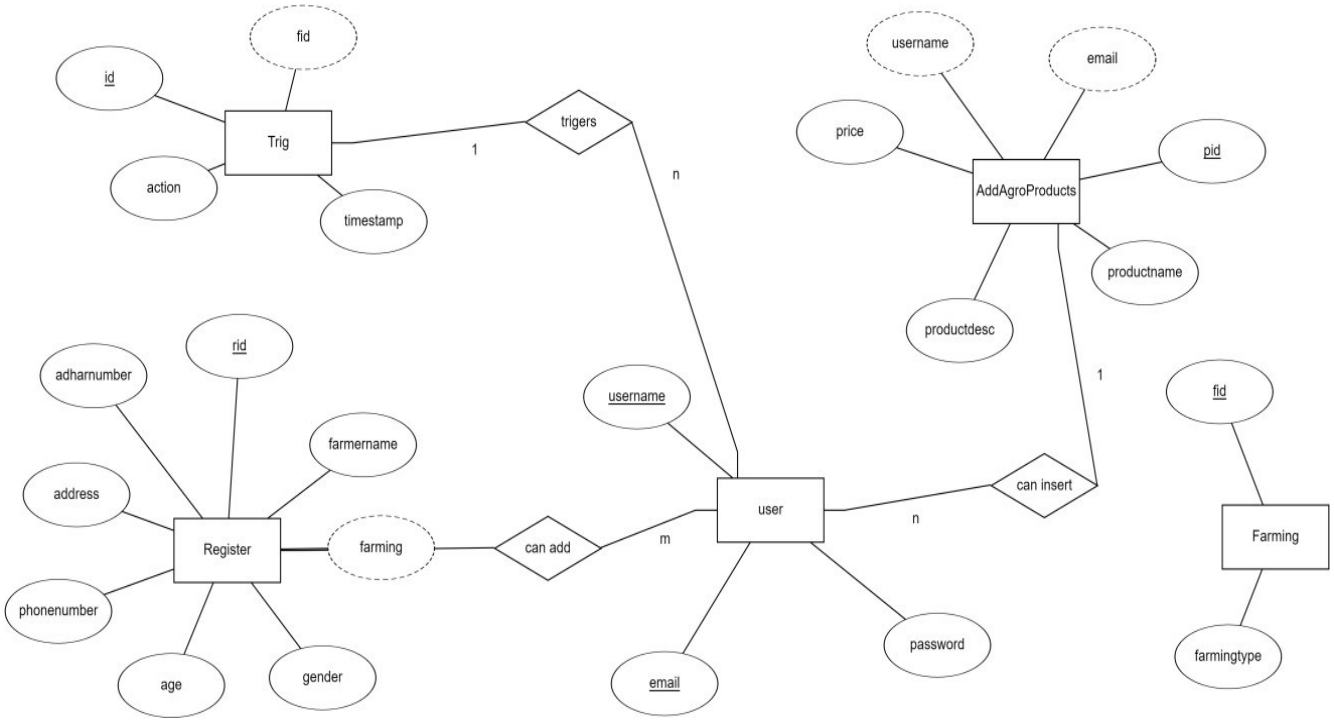
- Operating System: Windows 10
- Google Chrome/Internet Explorer
- XAMPP (Version-3.7)
- Python main editor (user interface): PyCharm Community
- Workspace editor: Sublime text 3

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

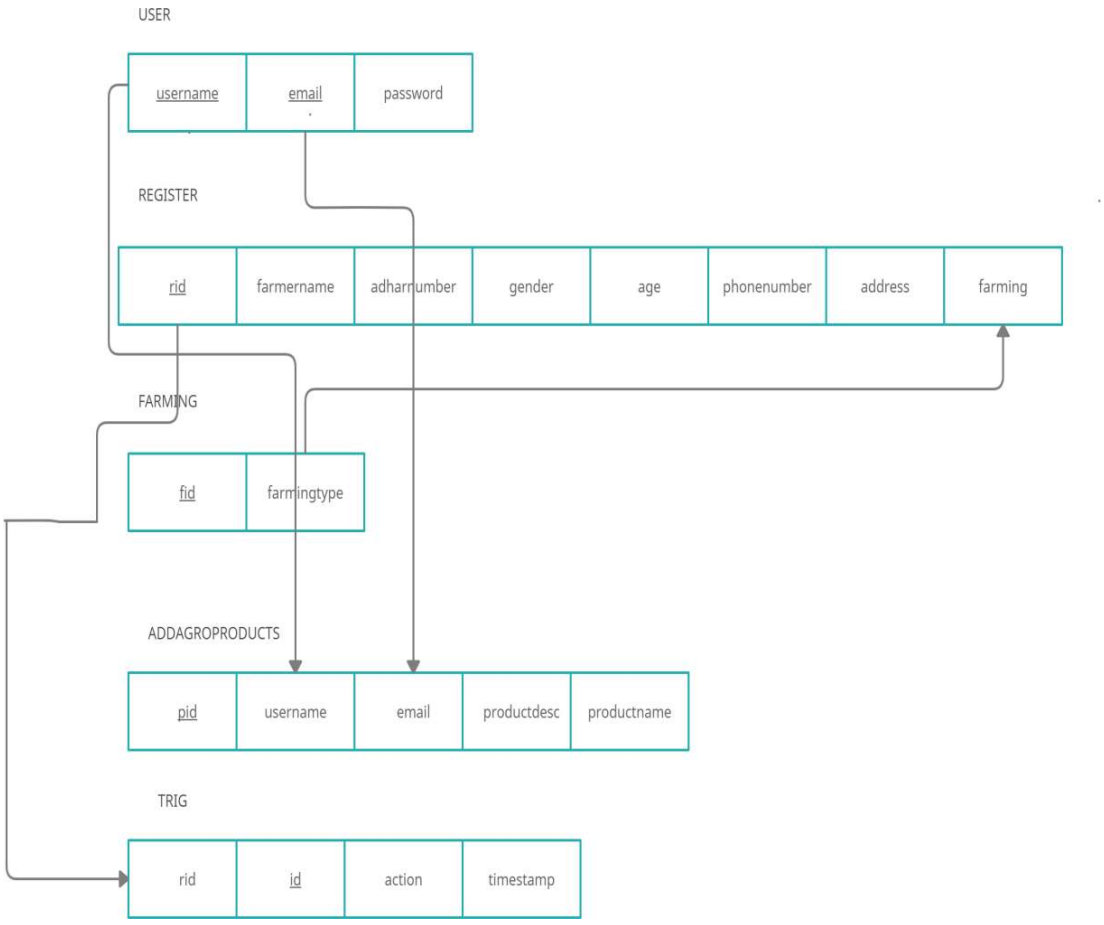
3.2 CONCEPTUAL DESIGN:

3.2.1 E-R DIAGRAM:



.

3.2.2 SCHEMA DIAGRAM:



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.

BackEnd (MySQL)

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. to the 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.

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)

4.2 : Stored Procedure

Routine name: proc
Type: procedure
Definition: Select * from register;

4.3: Triggers

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())

BACKEND PYHTON WITH MYSQL 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@localh
ost/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))


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')
 phonenumber=request.form.get('phonenumber')


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`='{phonenumber}',`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')
 phonenumber=request.form.get('phonenumber')
 address=request.form.get('address')
 farmingtype=request.form.get('farmingtype')


query=db.engine.execute(f"INSERT INTO `register`
(`farmername`,`adharnumber`,`age`,`gender`,`phonenumber`,`address`,`farming`)
VALUES
('{farmername}','{adharnumber}','{age}','{gender}','{phonenumber}','{address}','{farming
type}')")
 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)
```

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|Ra
leway: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|Ra
leway: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>
```

Farmer.html

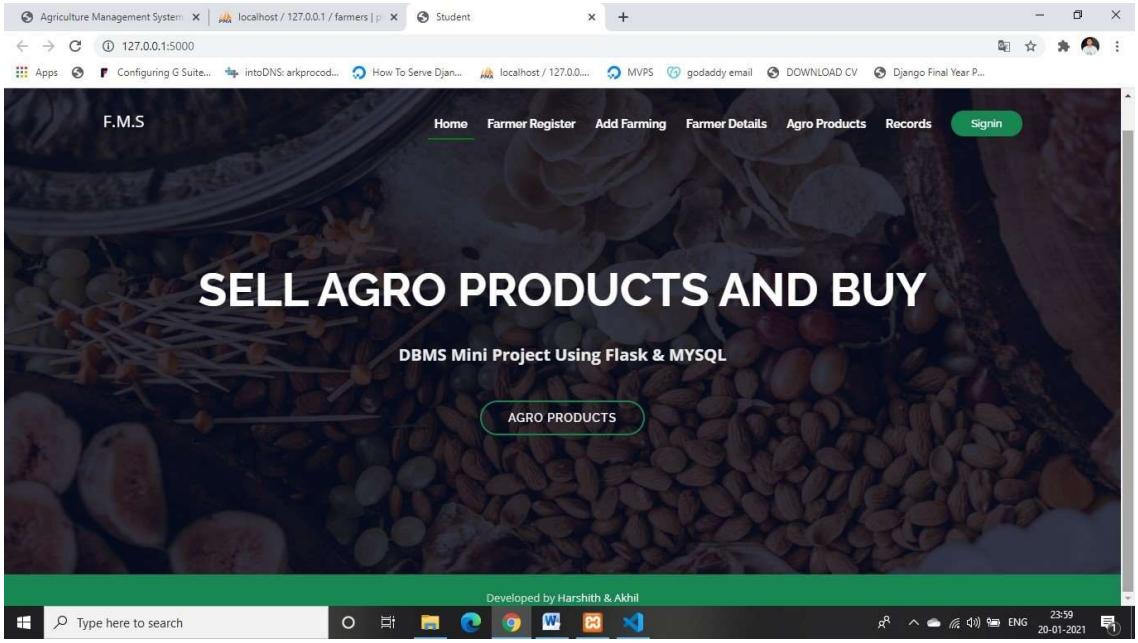
```
{% extends 'base.html' %}
{% block title %} Add Farming
{% endblock title %}


{% block body %}
<h3 class="text-center bg-success text-white"><span>Add Farming</span> </h3>


{% 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 %}
```

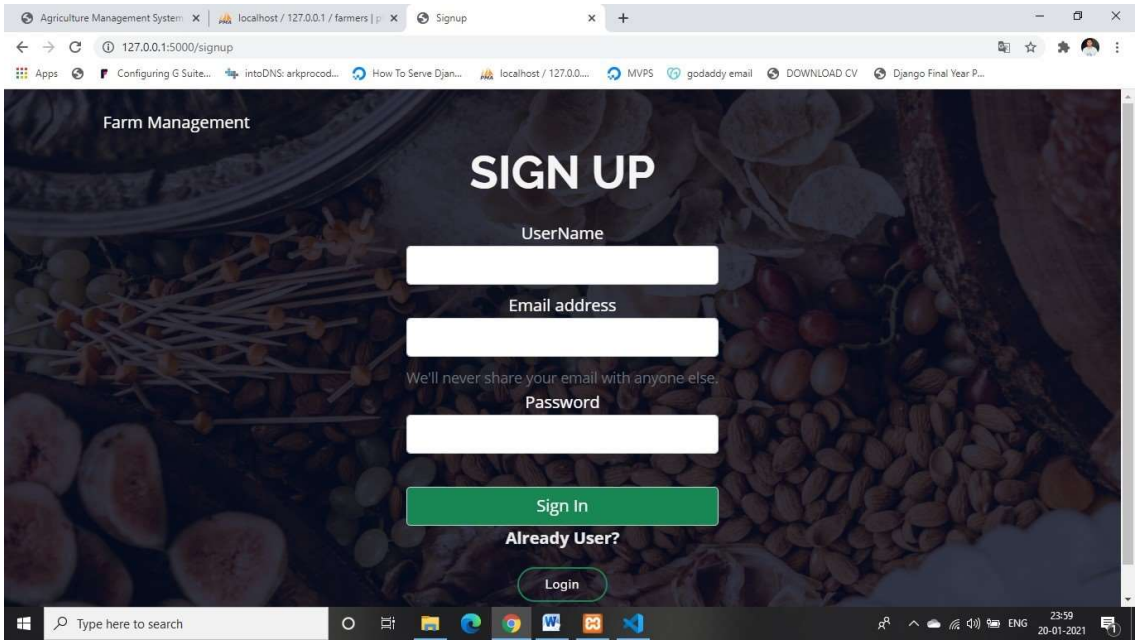
```
{% endwith %}
<br>
<div class="container">


<div class="row">


<div class="col-md-4"></div>
<div class="col-md-4">



<form action="/addfarming" method="post">



<div class="form-group">
<label for="dept">Enter Farming Type</label>
<input type="text" class="form-control" name="farming" id="farming">
</div>
<br>


<button type="submit" class="btn btn-success btn-sm btn-block">Add Farming</button>
</form>
<br>
<br>


</div>


<div class="col-md-4"></div>


</div></div>



{% endblock body %}
```

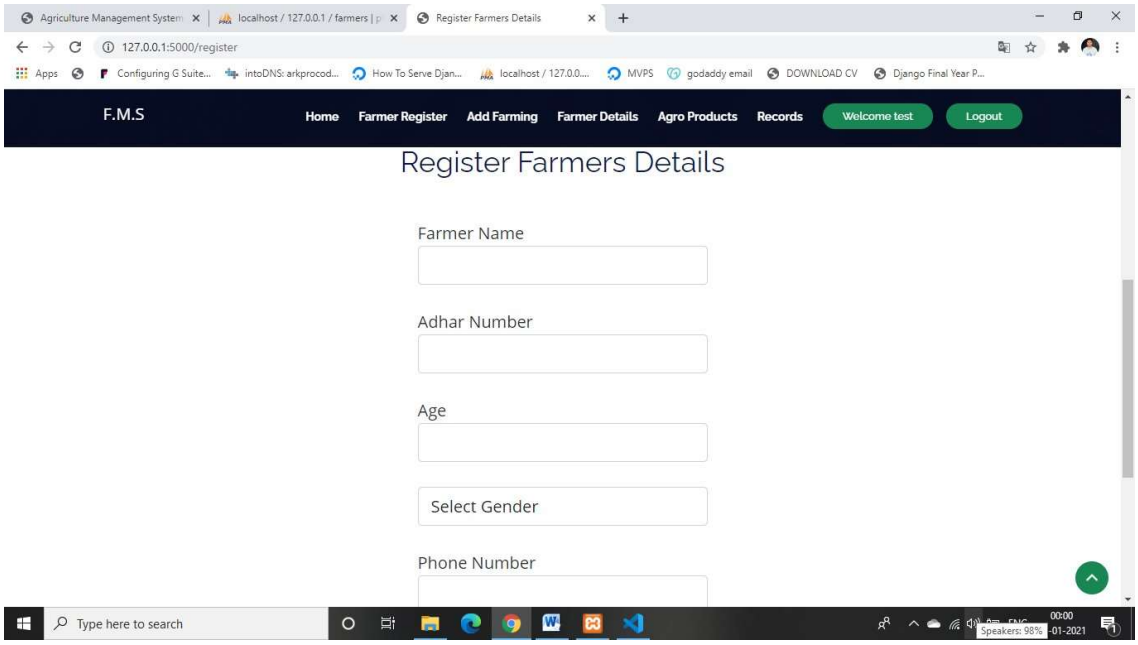
USER INTERFACE

4.1 SCREEN SHOTS

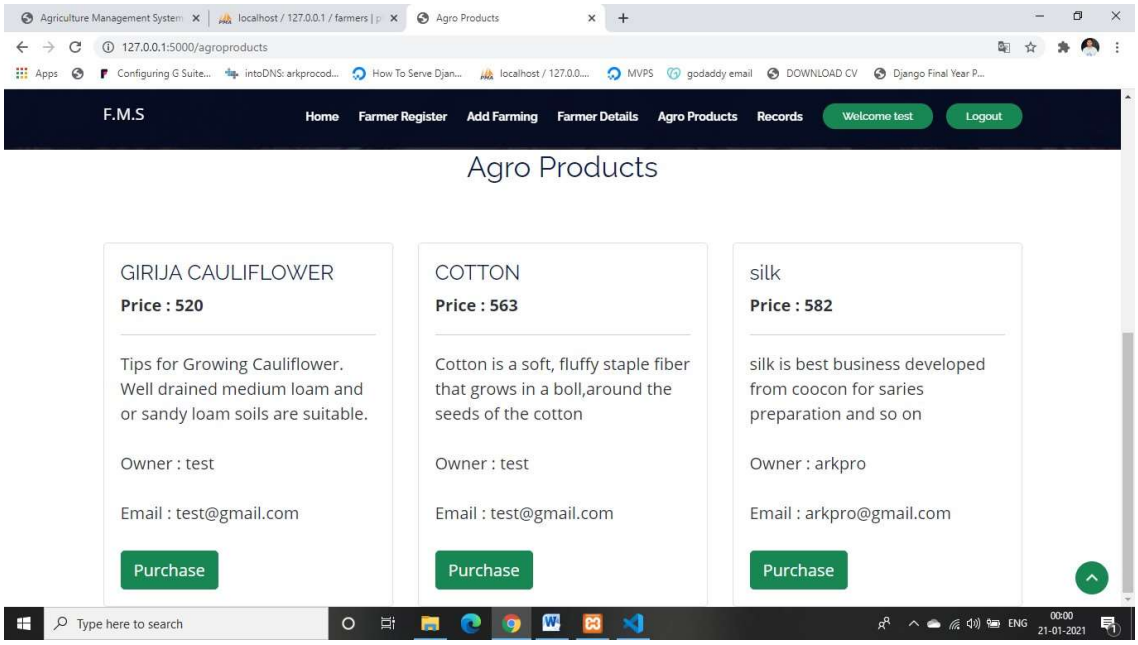
SIGN IN PAGE:

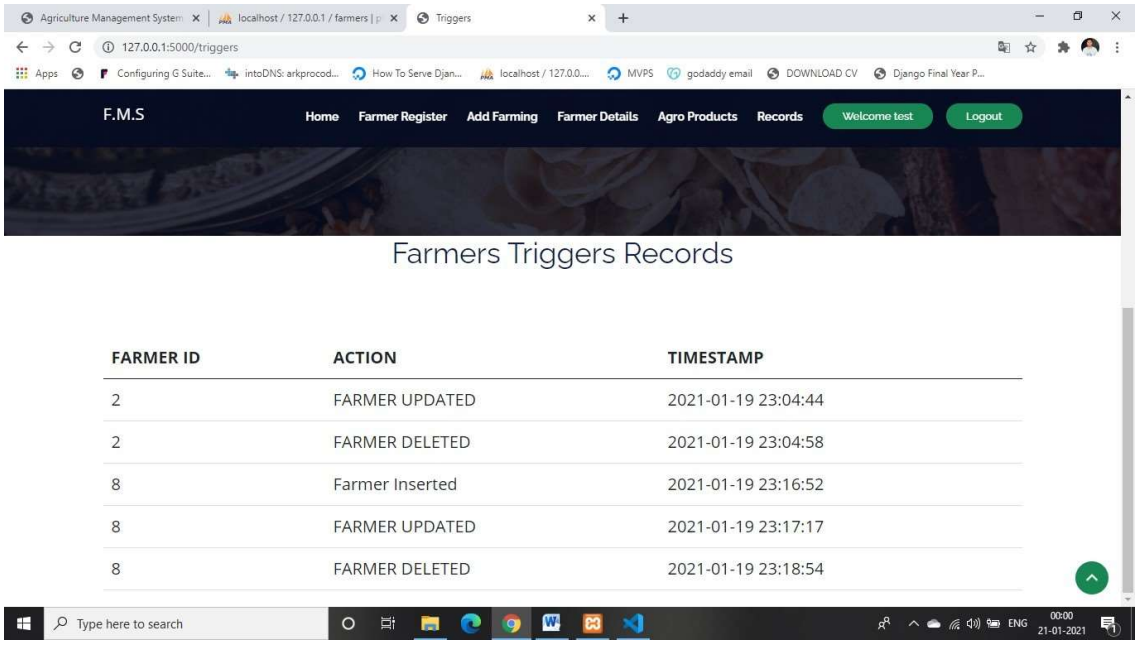




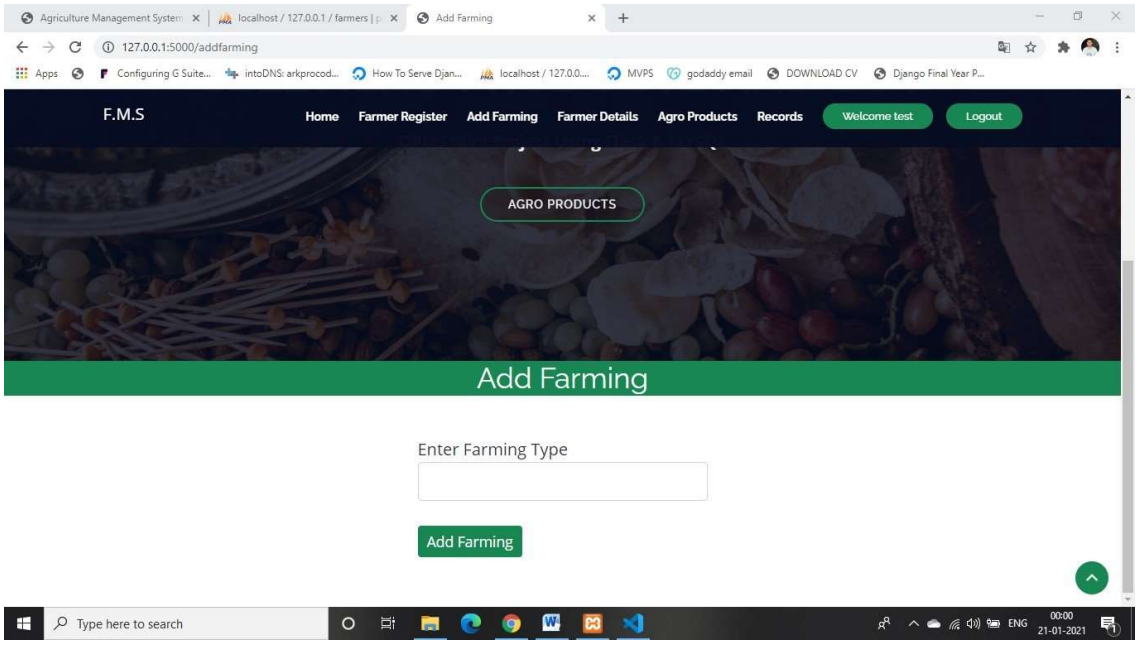


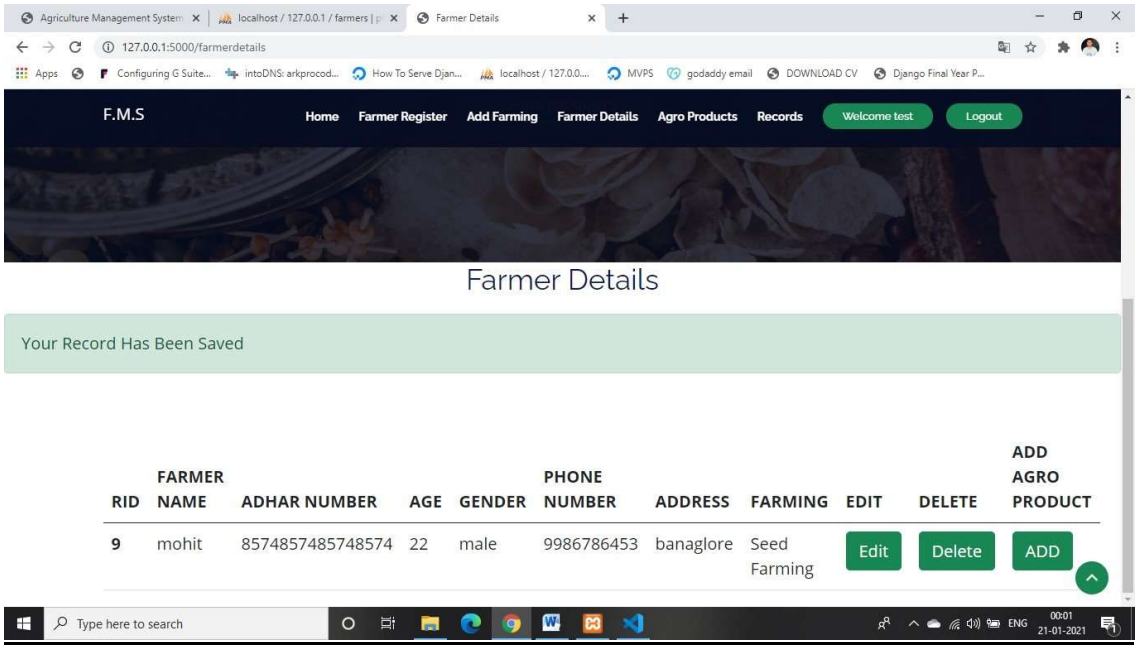
REGISTERATION PAGE & PRODUCTS:



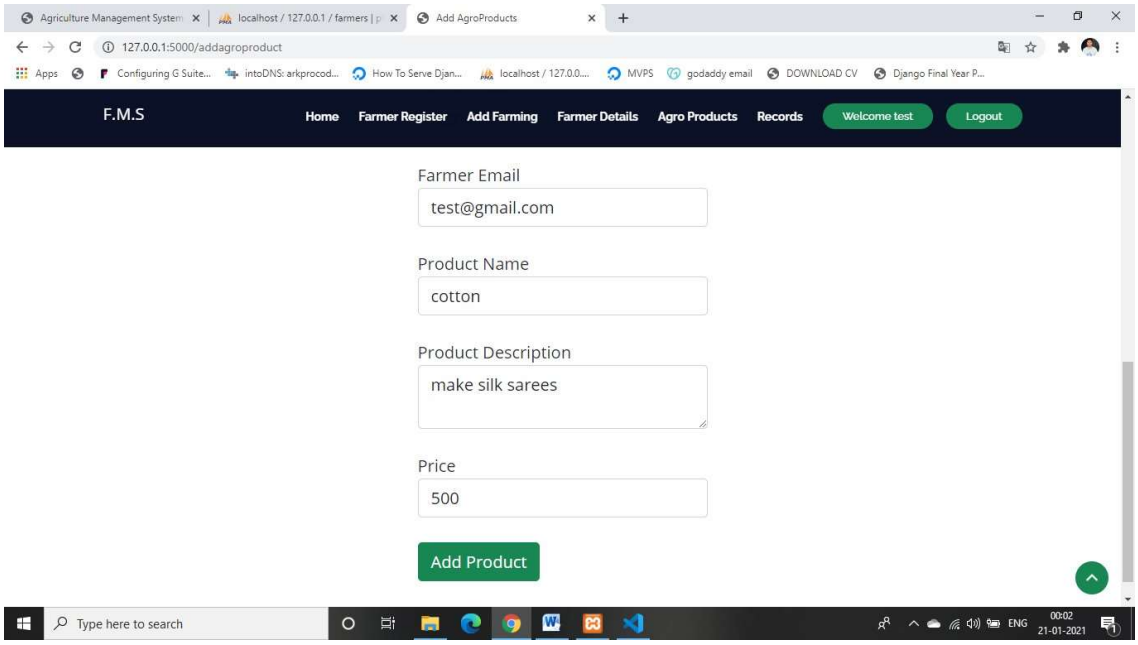


TRIGGERS RECORDS

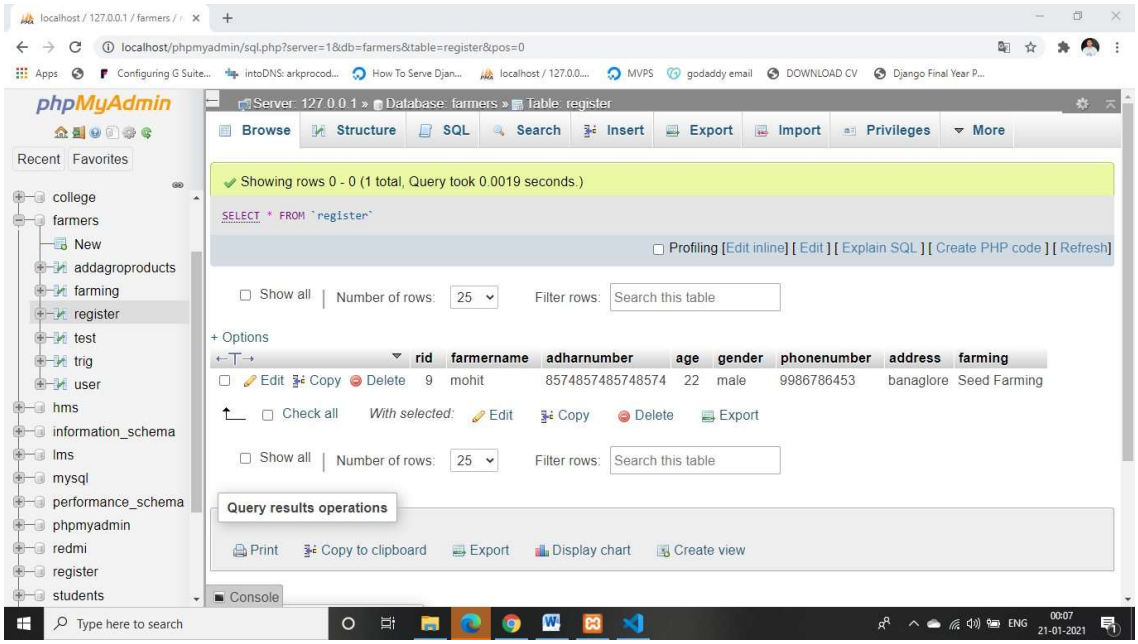


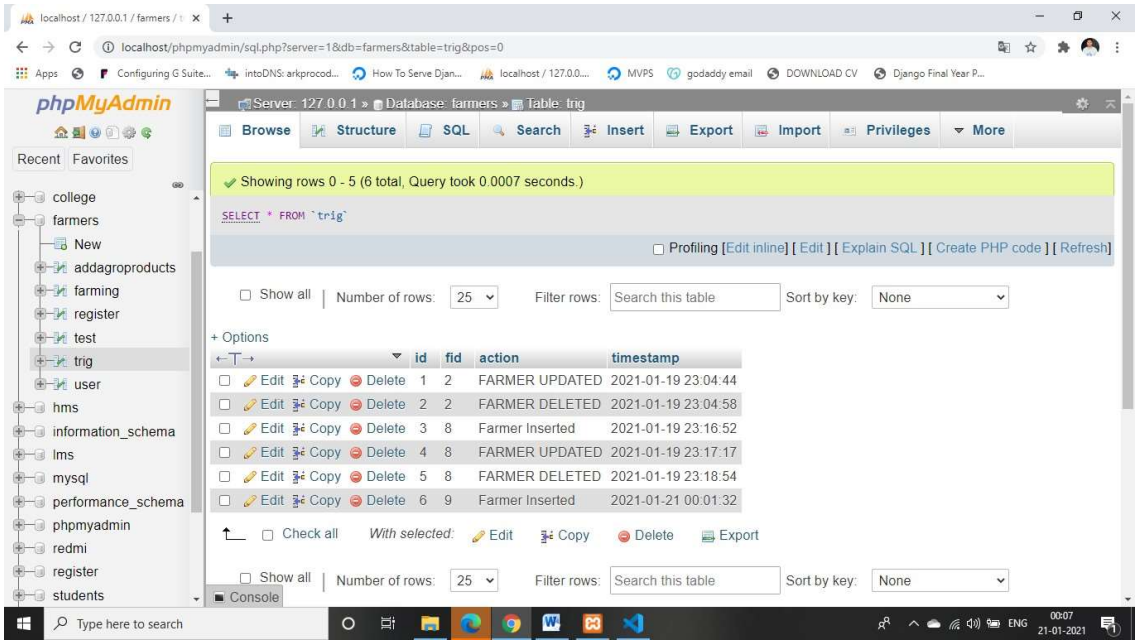


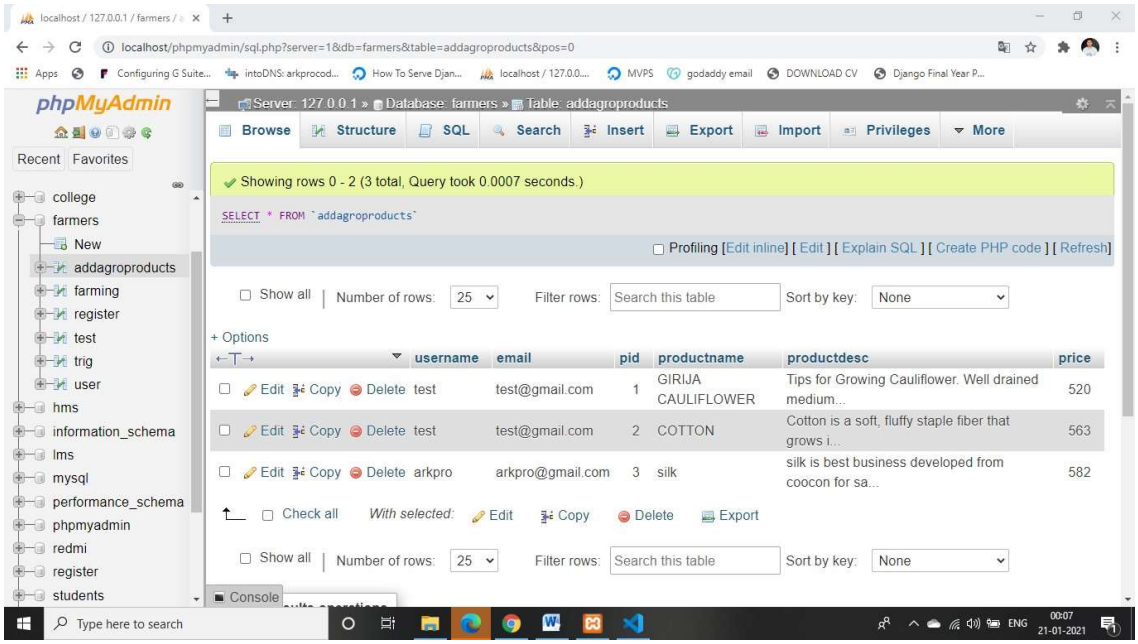
ADDING AGRO PRODUCTS

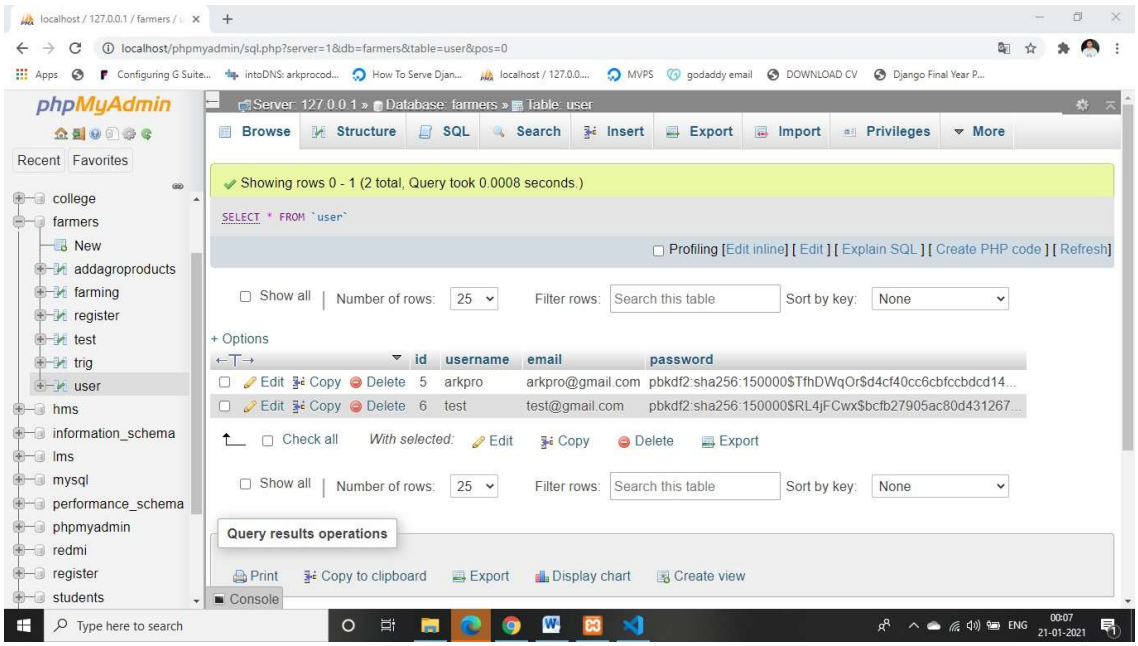


DATABASE:


localhost/phpmyadmin/sql.php?server=1&db=farmers&table=register&pos=0
phpMyAdmin
Server: 127.0.0.1 » Database: farmers » Table: register
Browse Structure SQL Search Insert Export Import Privileges More
Showing rows 0 - 0 (1 total, Query took 0.0019 seconds.)
SELECT * FROM `register`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
+ Options
rid farmername adharnumber age gender phonenumber address farming
Edit Copy Delete 9 mohit 8574857485748574 22 male 9986786453 banaglore Seed Farming
Check all With selected: Edit Copy Delete Export
Query results operations
Print Copy to clipboard Export Display chart Create view
college
farmers
New
addagroproducts
farming
register
test
trig
user
hms
information_schema
lms
mysql
performance_schema
phpmyadmin
redmi
register
students
Console
Type here to search
ENG 00:07 21-01-2021


localhost/phpmyadmin/sql.php?server=1&db=farmers&table=trig&pos=0
phpMyAdmin
Server: 127.0.0.1 » Database: farmers » Table: trig
Browse Structure SQL Search Insert Export Import Privileges More
Showing rows 0 - 5 (6 total, Query took 0.0007 seconds.)
SELECT * FROM `trig`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
+ Options
id fid action timestamp
Edit Copy Delete 1 2 FARMER UPDATED 2021-01-19 23:04:44
Edit Copy Delete 2 2 FARMER DELETED 2021-01-19 23:04:58
Edit Copy Delete 3 8 Farmer Inserted 2021-01-19 23:16:52
Edit Copy Delete 4 8 FARMER UPDATED 2021-01-19 23:17:17
Edit Copy Delete 5 8 FARMER DELETED 2021-01-19 23:18:54
Edit Copy Delete 6 9 Farmer Inserted 2021-01-21 00:01:32
Check all With selected: Edit Copy Delete Export
college
farmers
New
addagroproducts
farming
register
test
trig
user
hms
information_schema
lms
mysql
performance_schema
phpmyadmin
redmi
register
students
Console
Type here to search
ENG 00:07 21-01-2021


localhost / 127.0.0.1 / farmers /
localhost/phpmyadmin/sql.php?server=1&db=farmers&table=addagroproducts&pos=0
phpMyAdmin
Server: 127.0.0.1 » Database: farmers » Table: addagroproducts
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
More
Showing rows 0 - 2 (3 total, Query took 0.0007 seconds.)
SELECT * FROM `addagroproducts`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
+ Options
username email pid productname productdesc price
Edit Copy Delete test test@gmail.com 1 GIRIJA CAULIFLOWER Tips for Growing Cauliflower. Well drained medium... 520
Edit Copy Delete test test@gmail.com 2 COTTON Cotton is a soft, fluffy staple fiber that grows i... 563
Edit Copy Delete arkpro arkpro@gmail.com 3 silk silk is best business developed from coocon for sa... 582
Check all With selected: Edit Copy Delete Export
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Console
Recent Favorites
college
farmers
New
addagroproducts
farming
register
test
trig
user
hms
information_schema
lms
mysql
performance_schema
phpmyadmin
redmi
register
students
Type here to search
ENG
00:07
21-01-2021


localhost / 127.0.0.1 / farmers /
localhost/phpmyadmin/sql.php?server=1&db=farmers&table=user&pos=0
phpMyAdmin
Server: 127.0.0.1 » Database: farmers » Table: user
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
More
Showing rows 0 - 1 (2 total, Query took 0.0008 seconds.)
SELECT * FROM `user`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
+ Options
id username email password
Edit Copy Delete 5 arkpro arkpro@gmail.com pbkdf2:sha256:150000$TfhDWqOr$d4cf40cc6cbfccbdcd14...
Edit Copy Delete 6 test test@gmail.com pbkdf2:sha256:150000$RL4jFCwx$bcfb27905ac80d431267...
Check all With selected: Edit Copy Delete Export
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Query results operations
Print Copy to clipboard Export Display chart Create view
Console
Recent Favorites
college
farmers
New
addagroproducts
farming
register
test
trig
user
hms
information_schema
lms
mysql
performance_schema
phpmyadmin
redmi
register
students
Type here to search
ENG
00:07
21-01-2021

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 beeasily 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 personalpoint 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

REFERENCES

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

BIBLIOGRAPHY

It has been a matter of immense pleasure, honor and challenge to have this opportunity to take up this project and complete it successfully.

We have obtained information from various resources to design and implement our project. We have acquired most of the knowledge from the Internet.

The following are some of the resources:

- www.w3schools.com
- www.tutorialspoint.com
- Google and Youtube Tutorials.