

**Visvesvaraya Technological University
Belagavi-590 018, Karnataka**



A Mini Project Report on

“CROP MANAGEMENT SYSTEM”

Mini Project Report submitted in partial fulfillment of the requirement for the
DBMS Laboratory with mini project [18CSL58]

Bachelor of Engineering in

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Submitted by

Shrisha Udupa 1JT20AI041

Rangaswamy D 1JT20AI033



**Department of Artificial Intelligence and Machine Learning
Jyothy Institute of Technology Tataguni, Bengaluru-560082**

Jyothy Institute of Technology**Tataguni, Bengaluru-560082****Department of Artificial Intelligence and Machine Learning****CERTIFICATE**

Certified that the mini project work entitled “**Crop Management System**” carried out by **Shrisha Udupa [1JT20AI041]** and **Rangaswamy D [1JT20AI033]** bona-fide students of Jyothy Institute of Technology, in partial fulfilment for the award of **Bachelor of Engineering Artificial Intelligence and Machine Learning** in department of the **Visvesvaraya Technological University, Belagavi** during the year **2022-2023**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Mrs . Soumya

Assistant Professor

Dept. of AIML JIT

Bengaluru

DR . Madhu BR

Professor and HOD

Dept. of AIML

JIT, Bengaluru

External Examiner

1.

2.

Signature with Date:

ACKNOWLEDGEMENT

Firstly, we are very grateful to this esteemed institution “**Jyothy Institute of Technology**” for providing us an opportunity to complete our project.

We express our sincere thanks to our **Principal Dr . Gopalakrishna K** for providing us with adequate facilities to undertake this project.

We would like to thank **Dr . Madhu B R, Professor and Head of AIML** and Engineering Department for providing his valuable support.

We would like to thank our guides **Mrs. Soumya K N, Assistant Professor** for their keen interest and guidance in preparing this work.

Finally, we would thank all our friends who have helped us directly or indirectly in this project.

Shrisha Udupa [1JT20AI041]

Rangaswamy D [1JT20AI033]

ABSTRACT

For this application, we used the back end as XAMP to store the data which is used in the application, and for the user interface we have used HTML, CSS and the language used is Python

We have designed a database system named, 'Crop Management System' database system to make it easy for Users to manage crop transactions

The first activity while using the database is to add the User details to the system along with their personal details.

Later the user can store the crop required using crop id. Users can purchase any crop as well The Crop can be edited deleted and put for sale

The main aim of developing "Crop Management System Project" application is to help farmers by providing all kinds of agriculture related information in the site. "Crop Management System Project" is web application which helps Users to share Crop data.

It helps farmers to improve their productivity and profitability. It enables farmers to sell their products online and farmers can purchase seeds and crops 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 product through mails

TABLE OF CONTENTS

1. INTRODUCTION.....	6-8
1.1. INTRODUCTION TO DBMS.....	7
1.2. INTRODUCTION TO SQL.....	7
1.3. INTRODUCTION TO CROP MANAGEMENT SYSTEM.....	8
1.4. SCOPE AND IMPORTANCE OF WORK.....	8
2. DESIGN.....	9-12
2.1. THEORY OF ER DIAGRAM.....	10
2.2. ENTITIES.....	10
2.3. RELATIONSHIPS.....	11
2.4. ATTRIBUTES.....	11
2.5. ER DIAGRAM.....	13
2.6. SCHEMA DIAGRAM.....	14
2.7. LIST OF TABLES.....	14
3. IMPLEMENTATION.....	15-33
3.1. TABLE CREATION.....	16
3.2. TABLE INSERTION.....	21
3.3. BACKEND PYTHON WITH SQL CODE.....	22
3.4. TABLE DESCRIPTION.....	30
4. RESULTS AND SNAPSHOTS.....	34-39
5. CONCLUSION.....	40-42
5.1. FEATURES	41
5.2. REFERENCES	41
5.3. SOFTWARES USED	42
5.4. HARDWARE USED	42
5.5. FUTURE ENHANCEMENTS.....	42

CHAPTER 1: INTRODUCTION

1.INTRODUCTION

1.1 INTRODUCTION TO DBMS:

A database management system refers to technology for creating and managing databases. DBMS is a software tool to organize (create, retrieve, update and manage) data in a database. The main aim of DBMS is to supply a way to store up and retrieve database

information that is both convenient and efficient.

Advantages of databases:

- To develop software applications in less time.
Data independence and efficient use of data.
For uniform data administration.
For data integrity and security.
To use user-friendly declarative query language.

Components of DBMS

- Users: Users may be of any kind such as DB administrator, System developer, or database users.
- Database application: Database application may be Departmental, Personal, organization's and / or Internal.
- DBMS: Software that allows users to create and manipulate database access,
- Database: Collection of logical data as a single unit.

1.2 INTRODUCTION TO SQL:

SQL is an abbreviation of structured query language, is a language to request data from a database, to add, update, remove data within a database, or to manipulate the metadata of the database.

SQL is a declarative language in which the expected result or operation is given without the specific details about how to accomplish the task. The steps required to execute SQL statement are handled transparently by the SQL database. Sometimes SQL is characterised as non-procedural because procedural language generally require the details of the operations to be specified, such as opening and closing tables, loading and searching indexes, or flushing buffers and writing data to file system. Therefore, SQL is considered to be designed at a higher conceptual level of operation than procedural languages because the lower level logical and physical operation aren't specified and are determined by the SQL engine or server process that executes it.

SQL is used for database architecture and management. Thus, it is a vital tool used by any individual who seeks to pursue a career as a database administrator. For those unfamiliar with programming languages and website architecture, the work of SQL will often go unnoticed. Still, those who have seen behind the curtain will know it as one of the fundamental building blocks of modern database architecture

1.3 INTRODUCTION TO CROP MANAGEMENT SYSTEM:

The main objective of crop management system project is to create an application for the user either farmer or customer to modernize the transaction process. And to have the information to be stored digitally. So that the operator can see the details of his crop and also purchase from others . this enhances the transaction capacities and are farmer friendly in nature Since it is very easy to use anyone can use it with or without the knowledge of the modern computers

Crop management system can be used to store the crops that the farmer currently has which causes a systematic organization of the current crops

If a farmer tend to sell the crop he can give his name , description , price, and Email Id to order directly from the database

1.4 SCOPE AND IMPORTANCE OF WORK:

The scope of the project is clear to give a simple and attractive application to simplify the work as well as to reduce the efforts while doing it offline or we can say by doing with old methods.

In this project we are able to store the information of the crops, the user. and crops he has purchased and the payment details. This will mainly help the user to easily view the details of the crops. The main role is played by the farmer who can decide to sell or just store the crop . This application is surely be very useful and comfortable for the farmers and also to the customers.

CHAPTER 2: DESIGN

2 -THEORY OF ER DIAGRAM

An entity relationship diagram shows the relationships of entity sets stored in a database. It mainly describes the structure of a database with the help of a diagram, which is so called the entity relationship diagram. An ER model is the design r blueprint of the database that can later be implemented as a database. The main components of ER model are as said above entity set and relationship set.

ER diagrams are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities instead of relationships between entities themselves. ER diagrams also are often used in conjunction with data flow diagrams (DFDs), which map out the flow of information for processes or systems

2.1 ENTITIES:

An entity is an object that exists. It doesn't have to do anything, it just has to exist. In database administration, an entity can be a single thing, person, place, or an object. Data can be stored about such entities. A design tool that allows database administration to view the relationships between several entities is basically called as an ER diagram.

An entity is referred to as an object or thing that exists in the real world. For example, customer, car, pen, etc.

Entities are stored in the database, and they should be distinguishable, i.e., they should be easily identifiable from the group. For example, a group of pens that are from the same company cannot be identified, so they are only objects, but pens with different colours become unique and will be called an entity like a red pen, green pen, blue pen, black pen, etc

n a group of pens, we can easily identify any pen because of its different colours, so a pen of different colours is an entity.

For extracting data from the database, each data must be unique in its own way so that it becomes easier to differentiate between them. Distinct and unique data is known as an entity.

There are two kinds of entities, which are as follows:

1. Tangible Entity:

It is an entity in DBMS, which is a physical object that we can touch or see. In simple words, an entity that has a physical existence in the real world is called a tangible entity.

For example, in a database, a table represents a tangible entity because it contains a physical object that we can see and touch in the real world. It includes colleges, bank lockers, mobiles, cars, watches, pens, paintings, etc.

2. Intangible Entity:

It is an entity in DBMS, which is a non-physical object that we cannot see or touch. In simple words, an entity that does not have any physical existence in the real world is known as an intangible entity.

For example, a bank account logically exists, but we cannot see or touch it

2.2 RELATIONSHIPS:

A relational database collects different types of data sets that use tables, records, and columns. It is used to create a well defined relationship between database tables so that relational database can be easily stored. For example say we need to have a connection between the two entities such as staff and customer we can connect them using the relationship say staff serves customer where serves is the relation that exists between them.

One to One Relationship (1:1): It is used to create a relationship between two tables in which a single row of the first table can only be related to one and only one records of a second table. Similarly, the row of a second table can also be related to anyone row of the first table.

One to Many Relationship: It is used to create a relationship between two tables. Any single rows of the first table can be related to one or more rows of the second tables, but the rows of second tables can only relate to the only row in the first table. It is also known as a **many to one** relationship.

Many to Many Relationship: It is **many to many** relationships that create a relationship between two tables. Each record of the first table can relate to any records (or no records) in the second table. Similarly, each record of the second table can also relate to more than one record of the first table. It is also represented an **N:N** relationship.

2.3 ATTRIBUTES:

In general, an attribute is a characteristic. In a database management system, an attribute refers to a database component, such as a table. It also may refer to a database field. Attributes describe the instances in the column of a database.

Simple Attributes

Simple attributes are those that cannot be further divided into sub-attributes.

Composite Attributes

Composite attributes are made up of two or more simple attributes. For example, a person's address may be a composite attribute that is made up of the person's street address, city, state, and zip code. Composite attributes can be used to create more complex data models and can be helpful when trying to represent data in a concise way.

Single Valued Attributes

Single-valued attributes can only have one value. For example, a person's Social Security Number is a single-valued attribute. Social Security Numbers are used to uniquely identify individuals in the United States and are, therefore, single-valued attributes.

Multivalued Attributes

Multivalued attributes can have more than one value. For example, a person may have multiple email addresses or phone numbers. Multivalued attributes in DBMS are often used to store information about relationships between entities. For instance, an employee entity might have a multivalued attribute called "dependents" that stores the names of the employee's dependents. Multivalued attributes can also be used to represent hierarchical data.

Derived Attributes

Derived attributes are based on other attributes and are not stored directly in the database.

For example: Consider a database of employees. Each employee has a date of birth, and we might want to calculate their age. However, age is a derived attribute because it can be determined from the date of birth. As such, it would not make sense to store it directly in the database. Here is an example diagram of a derived attribute in DBMS:

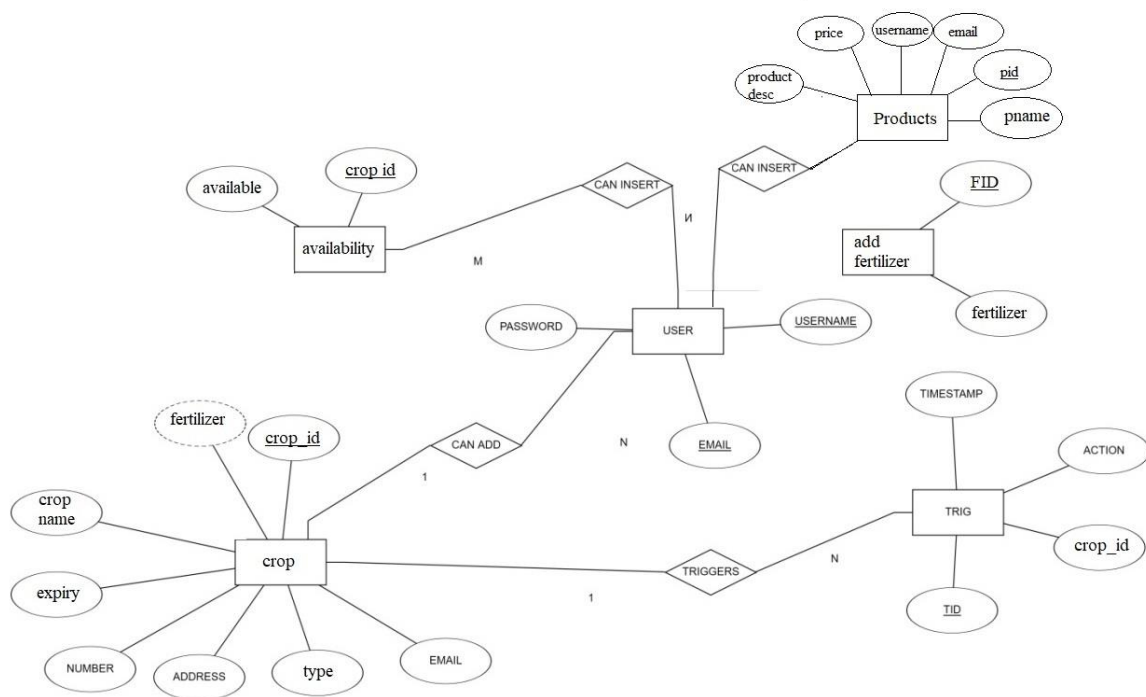
Complex Attributes

The complex attribute in DBMS involves both multivalued and composite attributes. For example, someone might have more than one house, and each house might have more than one phone. The phone is then considered a complex attribute.

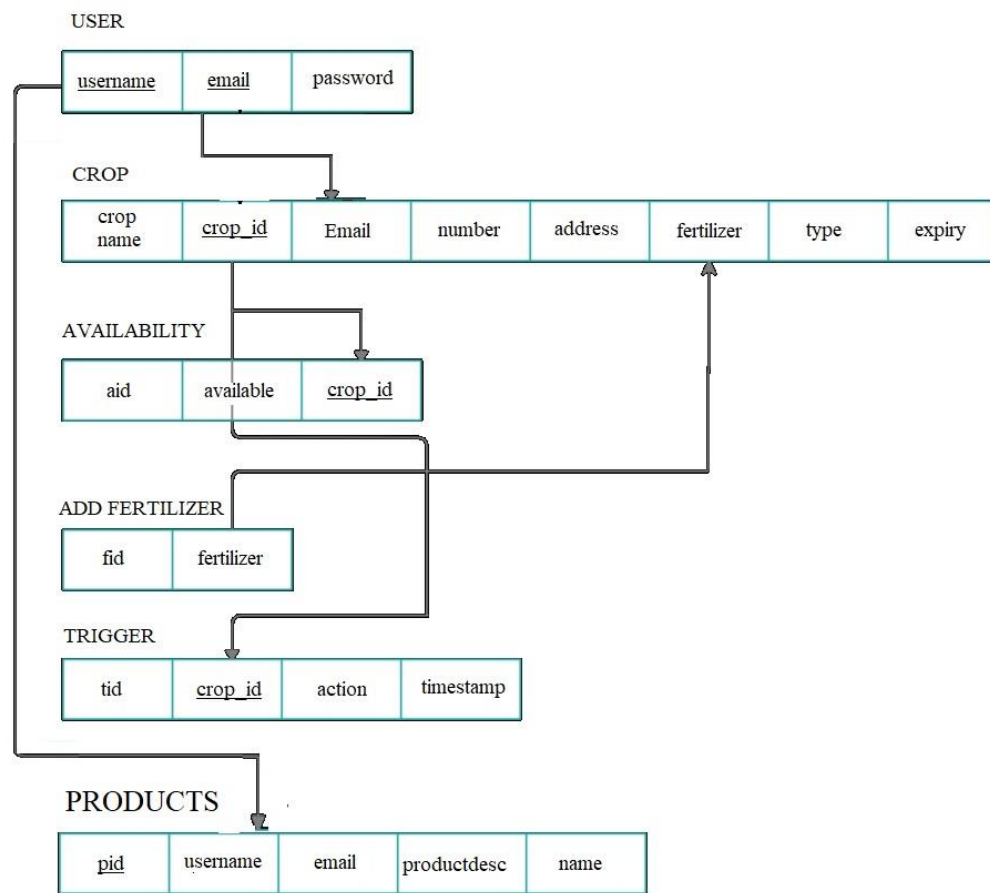
In the example above, the phone number is a composite attribute of the area code, exchange, and line number. Complex attributes are often used in database design to represent relationships between entities. Here is a complex attribute example explained in the form of a diagram.

CONCEPTUAL DESIGN

2.4-E.R DIAGRAM



2.5-SCHEMA DIAGRAM:



2.6 LIST OF TABLES

- 1.USER
- 2.CROP
- 3.AVAILABILITY
- 4.ADD FERTILIZER
- 5.TRIGGER

CHAPTER 3: IMPLEMENTATON

3.1 CREATE TABLE STATEMENTS

Database: `crops`

-- Table structure for table `Availability`

```
CREATE TABLE `Availability` (  
  `aid` int(11) NOT NULL,  
  `cropid` varchar(20) NOT NULL,  
  `availability` int(100) NOT NULL  
)
```

-- Table structure for table `department`

```
CREATE TABLE `fertilizer` (  
  `fid` int(11) NOT NULL,  
  `fertilizer` varchar(50) NOT NULL  
)
```

-- Table structure for table `crop`

```
CREATE TABLE `crop` (  
  `cropid` varchar(20) NOT NULL,  
  `cropname` varchar(50) NOT NULL,  
  `availability` int(20) NOT NULL,  
  `type` varchar(50) NOT NULL,  
  `fertilizer` varchar(50) NOT NULL,  
  `email` varchar(50) NOT NULL,  
  `number` varchar(12) NOT NULL,  
  `address` text NOT NULL  
)
```



```
-- Table structure for table `test`
```

```
CREATE TABLE `test` (  
  `id` int(11) NOT NULL,  
  `name` varchar(52) NOT NULL,  
  `email` varchar(50) NOT NULL  
)
```

```
-- Dumping data for table `test`
```

```
INSERT INTO `test` (`id`, `name`, `email`) VALUES
```

```
CREATE TABLE `trig` (  
  `tid` int(11) NOT NULL,  
  `cropid` varchar(50) NOT NULL,  
  `action` varchar(50) NOT NULL,  
  `timestamp` datetime NOT NULL  
)
```

```
CREATE TABLE `user` (  
  `id` int(11) NOT NULL,  
  `username` varchar(50) NOT NULL,  
  `email` varchar(50) NOT NULL,  
  `password` varchar(500) NOT NULL  
)
```

```
-- Indexes for table `attendance`
```

```
ALTER TABLE `Availability`  
ADD PRIMARY KEY (`aid`);
```

```
-- Indexes for table `Fertilizer`
```

```
ALTER TABLE `Fertilizer`  
ADD PRIMARY KEY (`fid`);
```

```
-- Indexes for table `crop`
```

```
ALTER TABLE `crop`  
ADD PRIMARY KEY (`cropid`);
```

```
-- Indexes for table `test`
```

```
ALTER TABLE `test`  
ADD PRIMARY KEY (`id`);
```

```
-- Indexes for table `trig`
```

```
ALTER TABLE `trig`  
ADD PRIMARY KEY (`tid`);
```

```
-- Indexes for table `user`
```

```
ALTER TABLE `user`  
ADD PRIMARY KEY (`id`);
```

```
-- AUTO_INCREMENT for dumped tables
```

```
-- AUTO_INCREMENT for table `availability`
```

```
ALTER TABLE `Availability`  
MODIFY `aid` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7;  
-- AUTO_INCREMENT for table `Fertilizer`
```

```
ALTER TABLE `fertilizer`  
MODIFY `fid` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=9;  
  
-- AUTO_INCREMENT for table `crop`  
ALTER TABLE `crop`  
MODIFY `cropid` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7;  
  
-- AUTO_INCREMENT for table `test`  
ALTER TABLE `test`  
MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;  
  
-- AUTO_INCREMENT for table `trig`  
ALTER TABLE `trig`  
MODIFY `tid` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=10;  
  
-- AUTO_INCREMENT for table `user`  
ALTER TABLE `user`  
MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;
```

3.2 INSERT INTO VALUES

INSERT INTO AVAILABILITY

```
INSERT INTO `Availability` (`aid`, `cropid`, `availability`) VALUES(6, '18pot11', 12);
```

INSERT INTO FERTILIZER

```
INSERT INTO `fertilizer` (`fid`, `fertilizer`) VALUES
```

```
(2, 'Ammoniacal Fertilizer'),
```

```
(3, 'Sodium Nitrate'),
```

```
(4, 'Calcium Ammonium Nitrate'),
```

```
(5, 'Urea '),
```

```
(7, 'Cyanamide'),
```

```
(8, 'Citrate');
```

TRIGGERS:

```
Triggers `crop`
```

```
--
```

```
DELIMITER $$
```

```
CREATE TRIGGER `DELETE` BEFORE DELETE ON `crop` FOR EACH ROW INSERT  
INTO trig VALUES(null,OLD.rollno,'CROP DELETED',NOW())
```

```
$crop
```

```
DELIMITER ;
```

```
DELIMITER $$
```

```
CREATE TRIGGER `Insert` AFTER INSERT ON `crop` FOR EACH ROW INSERT INTO  
trig VALUES(null,NEW.rollno,'CROP INSERTED',NOW())
```

```
$$
```

```
DELIMITER ;
```

```
DELIMITER $$
```

```
CREATE TRIGGER `UPDATE` AFTER UPDATE ON `crop` FOR EACH ROW INSERT  
INTO trig VALUES(null,NEW.rollno,'CROP UPDATED',NOW())
```

```
$$
```

DELIMITER ;

INSERT INTO TEST

```
INSERT INTO `test` (`id`, `name`, `email`) VALUES  
(1, 'aaa', 'aaa@gmail.com');
```

INSERT INTO TRIGGER

```
INSERT INTO `trig` (`tid`, `cropid`, `action`, `timestamp`) VALUES  
(7, '18pot11', 'CROP INSERTED', '2021-01-10 19:19:56'),  
(8, '18pot12', 'CROP UPDATED', '2021-01-10 19:20:31'),  
(9, '18pot13', 'CROP DELETED', '2021-01-10 19:21:23');
```

INSERT INTO USER

```
INSERT INTO `user` (`id`, `username`, `email`, `password`) VALUES  
(4, 'anees', 'anees@gmail.com', '123456');
```

3.3 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  
fromwerkzeug.securityimport 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  
import json
```

```
# MY db connection

local_server= True

app = Flask(__name__)

app.secret_key='Shri'

# this is for getting unique user access student

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/cropm'

app.config['SQLALCHEMY_DATABASE_URI']='mysql://root:@localhost/cropm'

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))

    email=db.Column(db.String(100))


class fertilizer(db.Model):

    fid=db.Column(db.Integer,primary_key=True)

    fertilizer=db.Column(db.String(100))


class Available(db.Model):

    cropid=db.Column(db.Integer,primary_key=True)

    availability=db.Column(db.Integer())
```

```
class Trig(db.Model):
    tid=db.Column(db.Integer,primary_key=True)
    cropid=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 Add Products(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 Crop(db.Model):
    cropid=db.Column(db.Integer,primary_key=True)
    cropname=db.Column(db.String(50))
    availability=db.Column(db.Integer)
    type=db.Column(db.String(50))
    fertilizer=db.Column(db.String(50))
    email=db.Column(db.String(50))
    number=db.Column(db.String(12))
    address=db.Column(db.String(100))
```

```
@app.route('/')
def index():
    return render_template('index.html')

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

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

@app.route('/product',methods=['POST','GET'])
@login_required
def product():
    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=productdesc,price=price)

        db.session.add(products)
        db.session.commit()
        flash("Product Added","info")
        return redirect('/agroproducts')

    return render_template('addagroproducts.html')

@app.route('/triggers')
```



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

@app.route('/fertilizer',methods=['POST','GET'])
def fertilizer():
    if request.method=="POST":
        dept=request.form.get('dept')
        query=fertilizer.query.filter_by(branch=dept).first()
        if query:
            flash("This Fertilizer Already Exist","warning")
            return redirect('/fertilizer')
        dep=fertilizer(branch=dept)
        db.session.add(dep)
        db.session.commit()
        flash("Fertilizer Added","success")
    return render_template('fertilizer.html')

@app.route('/addavailability',methods=['POST','GET'])
def addavailability():
    query=db.engine.execute(f"SELECT * FROM `crop`")
    if request.method=="POST":
        rollno=request.form.get('cropid')
        attend=request.form.get('availability')
        print(availability,cropid)
        atte=availability(cropid=cropid,availability=availability)
        db.session.add(atte)
        db.session.commit()
        flash("Record added","warning")
    return render_template('percent.html',query=query)

@app.route('/search',methods=['POST','GET'])
```

```

def search():
    if request.method=="POST":
        cropid=request.form.get('id')
        bio=Student.query.filter_by(cropid=cropid).first()
        attend=Availabilty.query.filter_by(cropid=cropid).first()
        return render_template('search.html',bio=bio,attend=attend)
    return render_template('search.html')
@app.route("/delete/<string:id>",methods=['POST','GET'])
@login_required
def delete(id):
    db.engine.execute(f"DELETE FROM `crop` WHERE `crop`.`id`={id}")
    flash(" Deleted Successful","danger")
    return redirect('/cropdetails')
@app.route("/edit/<string:id>",methods=['POST','GET'])
@login_required
def edit(id):
    dept=db.engine.execute("SELECT * FROM `fertilizer`")
    posts=Student.query.filter_by(id=id).first()
    if request.method=="POST":
        cropid=request.form.get('cropid')
        cropname=request.form.get('cropname')
        availability=request.form.get('availability')
        typr=request.form.get('type')
        fertilizer=request.form.get('fertilizer')
        email=request.form.get('email')
        num=request.form.get('num')
        address=request.form.get('address')
        query=db.engine.execute(f"UPDATE `crop` SET
`cropid`='{cropid}',`cropname`='{cropname}',`availabiliy`='{availability}',`type`='{type}',`fe
rtilizer`='{fertilizer}',`email`='{email}',`number`='{num}',`address`='{address}'")
        flash("Crop is Updated","success")

```

```
        return redirect('/cropdetails')

    return render_template('edit.html',posts=posts,dept=dept)
@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')
        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('/addcrop',methods=['POST','GET'])
@login_required
def addcrop():
    fertilizer=db.engine.execute("SELECT * FROM `fertilizer`")
    if request.method=="POST":
        cropid=request.form.get('cropid')
        cropname=request.form.get('cropname')
        availability=request.form.get('availabilty')
        type=request.form.get('type')
        fertilizer=request.form.get('fertilizer')
        email=request.form.get('email')
        num=request.form.get('num')
```

```
        address=request.form.get('address')

        query=db.engine.execute(f"INSERT INTO `crop`
(`cropid`,`cropname`,`availability`,`typr`,`fertilizer`,`email`,`number`,`address`) VALUES
('{cropid}','{cropname}','{availability}','{typr}','{fertilizer}','{email}','{number}','{address}'))")

        flash("Booking Confirmed","info")

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

@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)
```

3.4 LABEL DESCRIPTIONS

SELECT* FROM TABLES

USER

Server: 127.0.0.1 » Database: cropm » Table: user

Showing rows 0 - 12 (13 total, Query took 0.0003 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

Extra options

	id	username	email	password
<input type="checkbox"/>	4	anees	anees@gmail.com	pbkdf2:sha256:150000\$1CSLss89\$ef995dfc48121768b207...
<input type="checkbox"/>	5	see	1j120ai041@jyothyit.ac.in	pbkdf2:sha256:260000\$1K8fczITRI5kt5WS\$94a855ead355...
<input type="checkbox"/>	6	King	queen@gmail.com	pbkdf2:sha256:260000\$ZeMlsntJQ5QQuaf\$82a5c736ee11...
<input type="checkbox"/>	7	kick	kck@gmail.com	pbkdf2:sha256:260000\$9AVFE4j9TRmZDU4j\$b7deffa9a8f0...
<input type="checkbox"/>	8	hi	a@gmail.com	pbkdf2:sha256:260000\$55na3XZeghmZixSS\$7df9fc140b8b...
<input type="checkbox"/>	9	Shrisha	12082002srishaudupa@gmail.com	pbkdf2:sha256:260000\$bWlNonI9JW8q5CxK\$98209afd9ca2...
<input type="checkbox"/>	10	sri	sri@gmail.com	pbkdf2:sha256:260000\$MjNc30ottISUpH6A5\$079e6ae13f8a...
<input type="checkbox"/>	11	swamy	tdhgjgd@gmail.com	pbkdf2:sha256:260000\$8Ke0YDJFZhSchMg\$4ae77f06b158...
<input type="checkbox"/>	12	ranga	d@gmail.com	pbkdf2:sha256:260000\$kn8EoYFVKaUTvLke\$2640e434989f...
<input type="checkbox"/>	13	shrisha	test@gmail.com	pbkdf2:sha256:260000\$BwR87DcdSOxHQzW8\$42ba61c8356b...
<input type="checkbox"/>	14	di	q@gmail.com	pbkdf2:sha256:260000\$ROBa5GW5tBe2GJ0Z\$70b8dd415366...

Console

Figure 3.4.1

CROP

Server: 127.0.0.1 » Database: cropm » Table: student

Showing rows 0 - 3 (4 total, Query took 0.0003 seconds.)

`SELECT * FROM `crop``

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

Extra options

	id	cropid	cropname	type	fertilizer	email	number	address
<input type="checkbox"/>	22	12	qwerty	3 Vegetable	Sodium Nitrate	1@h	111	dd
<input type="checkbox"/>	25	12	qwerty	3 Vegetable	Sodium Nitrate	1@h	111	dd
<input type="checkbox"/>	26	12	qwerty	3 Vegetable	Sodium Nitrate	1@h	111	dd
<input type="checkbox"/>	27	111	onion	3 Vegetable	Cyanamide	a@gmail.com	12345678	bangalore

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

Bookmark this SQL query

Console

Figure 3.4.2

AVAILABILITY

Server: 127.0.0.1 » Database: cropm » Table: attendance

	aid	cropid	availability
<input type="checkbox"/> Edit Copy Delete	12	11	9
<input type="checkbox"/> Edit Copy Delete	13	11	9
<input type="checkbox"/> Edit Copy Delete	14	11	1000
<input type="checkbox"/> Edit Copy Delete	15	11	1000
<input type="checkbox"/> Edit Copy Delete	16	11	99
<input type="checkbox"/> Edit Copy Delete	17	11	231
<input type="checkbox"/> Edit Copy Delete	18	1	21
<input type="checkbox"/> Edit Copy Delete	19	1	4
<input type="checkbox"/> Edit Copy Delete	20	12pot90	12
<input type="checkbox"/> Edit Copy Delete	21	1234	9
<input type="checkbox"/> Edit Copy Delete	22	111	40
<input type="checkbox"/> Edit Copy Delete	23	111	30
<input type="checkbox"/> Edit Copy Delete	24	12	6888
<input type="checkbox"/> Edit Copy Delete	25	111	20

☐ 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

Figure 3.4.3

FERTILIZER

Server: 127.0.0.1 » Database: cropm » Table: department

	cid	fertilizer
<input type="checkbox"/> Edit Copy Delete	2	Ammoniacal Fertilizer
<input type="checkbox"/> Edit Copy Delete	3	Sodium Nitrate
<input type="checkbox"/> Edit Copy Delete	4	Calcium Ammonium Nitrate
<input type="checkbox"/> Edit Copy Delete	5	Urea
<input type="checkbox"/> Edit Copy Delete	7	Cyanamide
<input type="checkbox"/> Edit Copy Delete	8	Citrate
<input type="checkbox"/> Edit Copy Delete	9	Bone meal
<input type="checkbox"/> Edit Copy Delete	10	organic manure

☐ 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

☐ Bookmark this SQL query

Console

Figure 3.4.4

PRODUCT

Showing rows 0 - 6 (7 total, Query took 0.0003 seconds.)

`SELECT * FROM `addagroproducts``

☐ Profiling [[Edit inline](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

☐ Show all | Number of rows: 25 | Filter rows: | Sort by key: None

Extra options

	username	email	pid	productname	productdesc	price	Availability
<input type="checkbox"/> Edit Copy Delete	test	test@gmail.com	1	GIRIJA CAULIFLOWER	Tips for Growing Cauliflower. Well drained medium...	520	0
<input type="checkbox"/> Edit Copy Delete	test	test@gmail.com	2	COTTON	Cotton is a soft, fluffy staple fiber that grows i...	563	0
<input type="checkbox"/> Edit Copy Delete	arkpro	arkpro@gmail.com	3	silk	silk is best business developed from cocoon for sa...	582	0
<input type="checkbox"/> Edit Copy Delete	shri	test@gmail.com	4	ddd	ddd	300	0
<input type="checkbox"/> Edit Copy Delete	manju	q@gmail.com	5	ddd	sss	3	0
<input type="checkbox"/> Edit Copy Delete	swamy	a@gmail.com	6	onion	citrus	30	0
<input type="checkbox"/> Edit Copy Delete	farmer	farmer@gmail.com	7	onion	ripe	40	0

☐ Check all | With selected: [Edit](#) [Copy](#) [Delete](#) [Export](#)

☐ Show all | Number of rows: 25 | Filter rows: | Sort by key: None

Figure 3.4.5

TRIGGER

Server: 127.0.0.1 » Database: cropm » Table: trig

	tid	cropid	action	timestamp
<input type="checkbox"/> Edit Copy Delete	45	1	CROP DELETED	2023-01-18 21:53:26
<input type="checkbox"/> Edit Copy Delete	46	1	CROP DELETED	2023-01-18 21:53:29
<input type="checkbox"/> Edit Copy Delete	47	1	CROP DELETED	2023-01-18 21:53:31
<input type="checkbox"/> Edit Copy Delete	48	1	CROP DELETED	2023-01-18 21:53:34
<input type="checkbox"/> Edit Copy Delete	49	1	CROP DELETED	2023-01-18 21:54:03
<input type="checkbox"/> Edit Copy Delete	50	1	CROP INSERTED	2023-01-18 21:54:43
<input type="checkbox"/> Edit Copy Delete	51	1	CROP UPDATED	2023-01-18 21:55:38
<input type="checkbox"/> Edit Copy Delete	52	1	CROP DELETED	2023-01-19 11:43:35
<input type="checkbox"/> Edit Copy Delete	53	12pot90	CROP INSERTED	2023-01-19 12:36:16
<input type="checkbox"/> Edit Copy Delete	54	12pot90	CROP UPDATED	2023-01-19 12:37:00
<input type="checkbox"/> Edit Copy Delete	55	12pot90	CROP UPDATED	2023-01-19 12:37:36
<input type="checkbox"/> Edit Copy Delete	56	12pot90	CROP UPDATED	2023-01-19 12:43:52
<input type="checkbox"/> Edit Copy Delete	57	12pot90	CROP UPDATED	2023-01-19 12:43:59
<input type="checkbox"/> Edit Copy Delete	58	111	CROP INSERTED	2023-01-19 12:47:01
<input type="checkbox"/> Edit Copy Delete	59	111	CROP UPDATED	2023-01-19 12:47:50
<input type="checkbox"/> Edit Copy Delete	60	111	CROP UPDATED	2023-01-19 12:47:50
<input type="checkbox"/> Edit Copy Delete	61	111	CROP UPDATED	2023-01-19 12:49:39
<input type="checkbox"/> Edit Copy Delete	62	111	CROP UPDATED	2023-01-19 12:49:39

Console

Figure 3.4.6

CHAPTER 4: RESULT AND SNAPSHOT

HOME PAGE

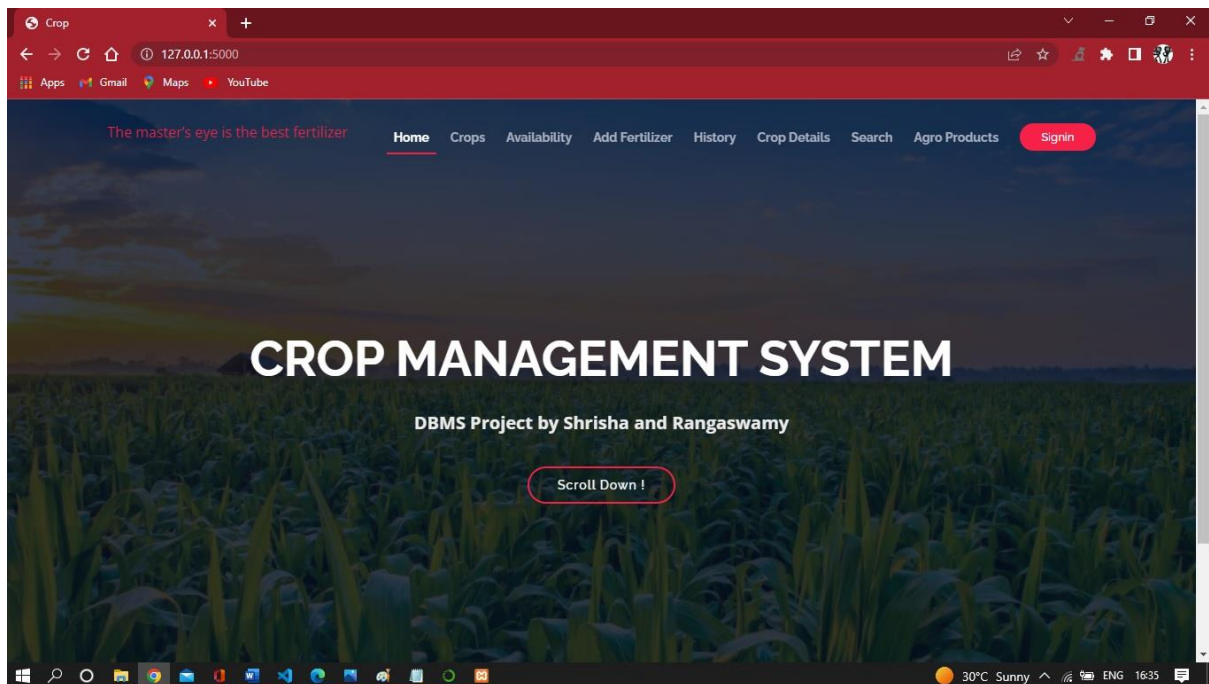


Figure 4.1

SIGNUP PAGE

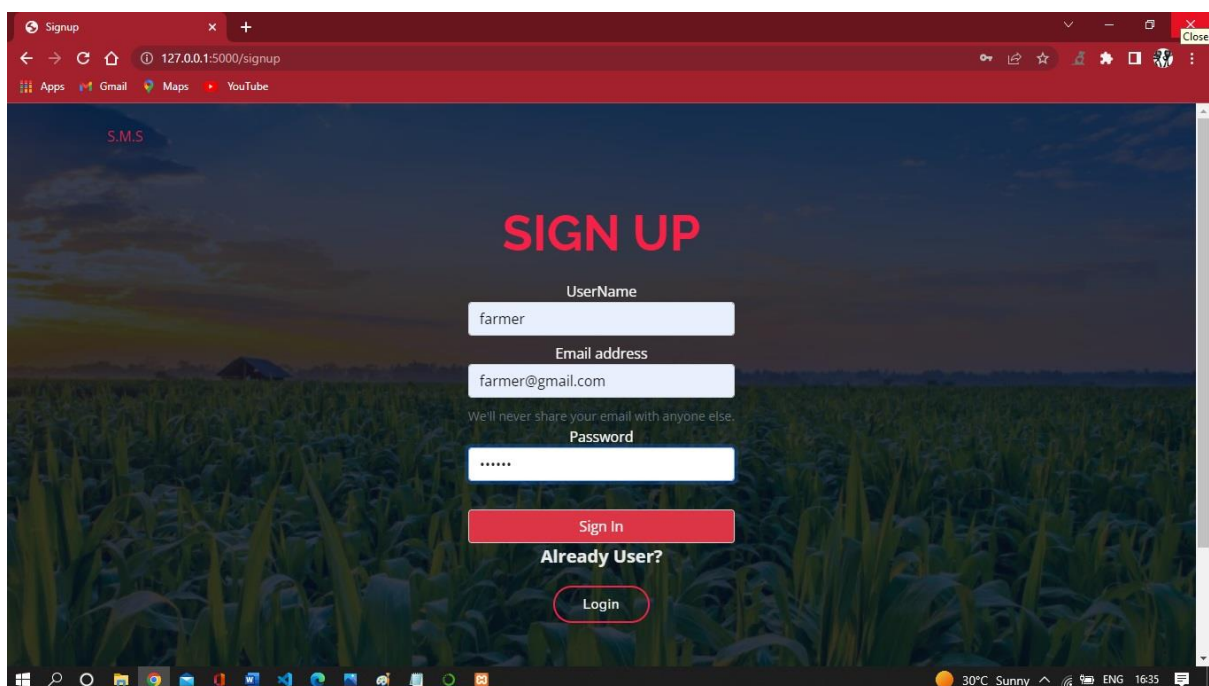
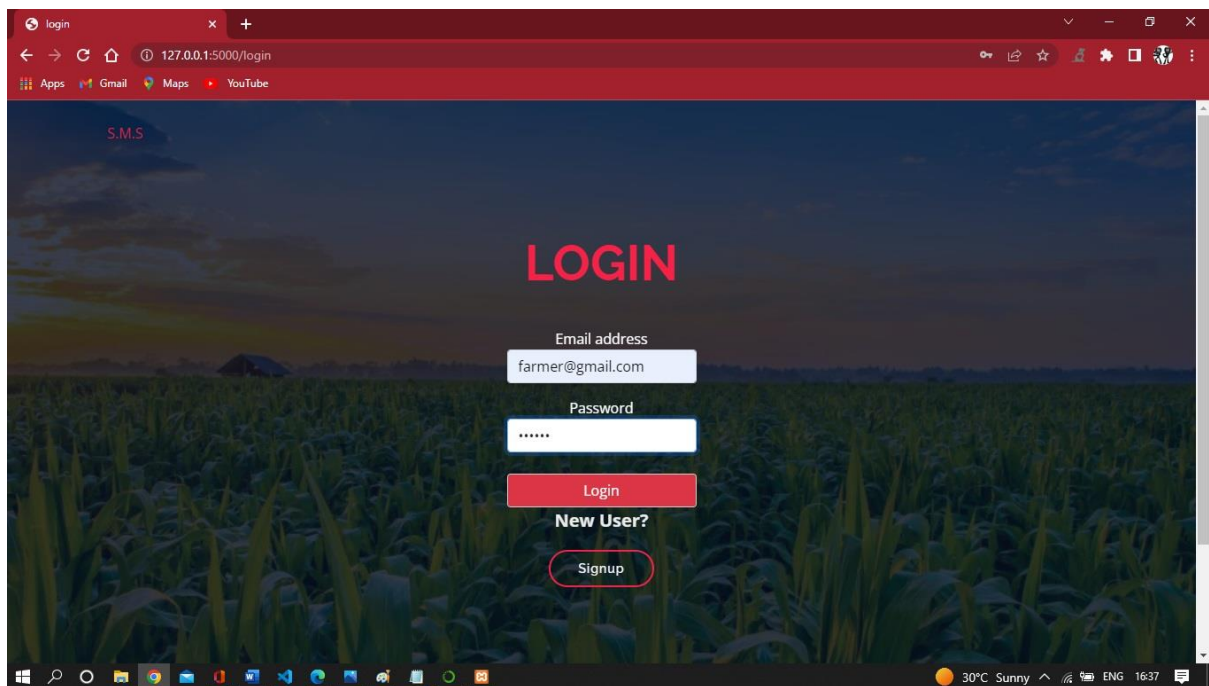


Figure 4.2

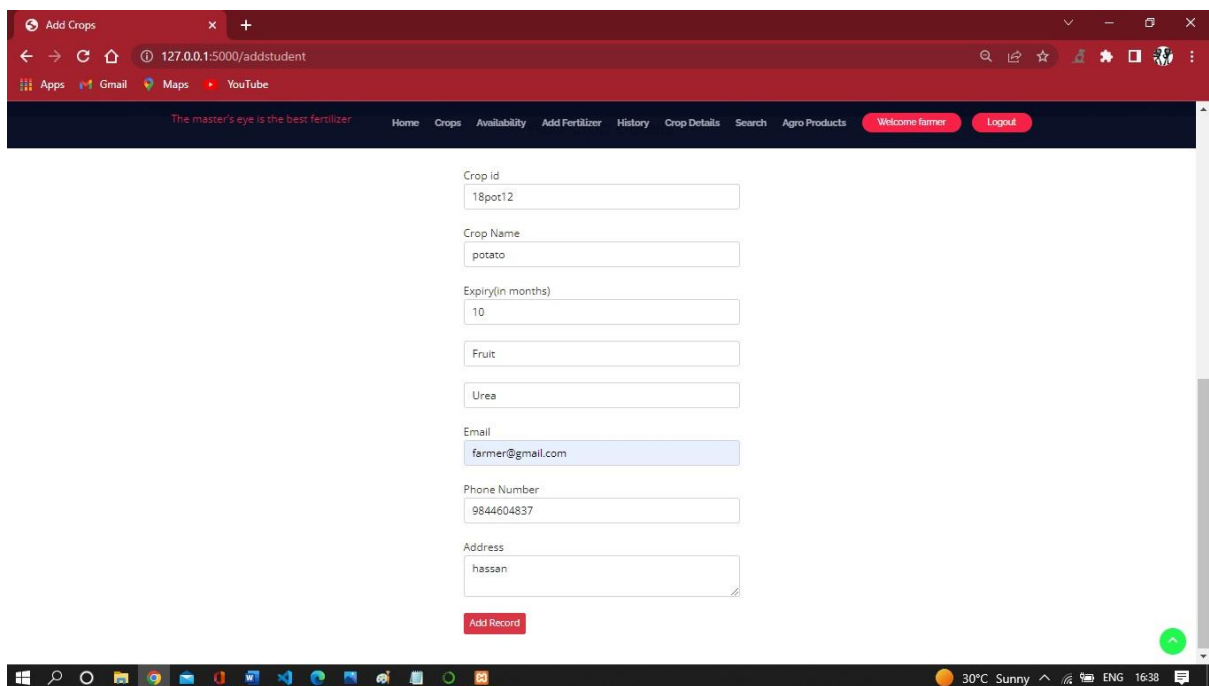
LOGIN PAGE



The screenshot shows a web browser window with the URL `127.0.0.1:5000/login`. The page has a dark blue header with the text "S.M.S" in red. The main content area features a large "LOGIN" title in red. Below the title, there are two input fields: "Email address" with the value "farmer@gmail.com" and "Password" with masked characters "*****". A red "Login" button is positioned below the password field. Below the "Login" button, there is a link "New User?" and a red "Signup" button. The background of the page is a blurred image of a cornfield. The Windows taskbar at the bottom shows the system time as 16:37 and the weather as 30°C Sunny.

Figure 4.3

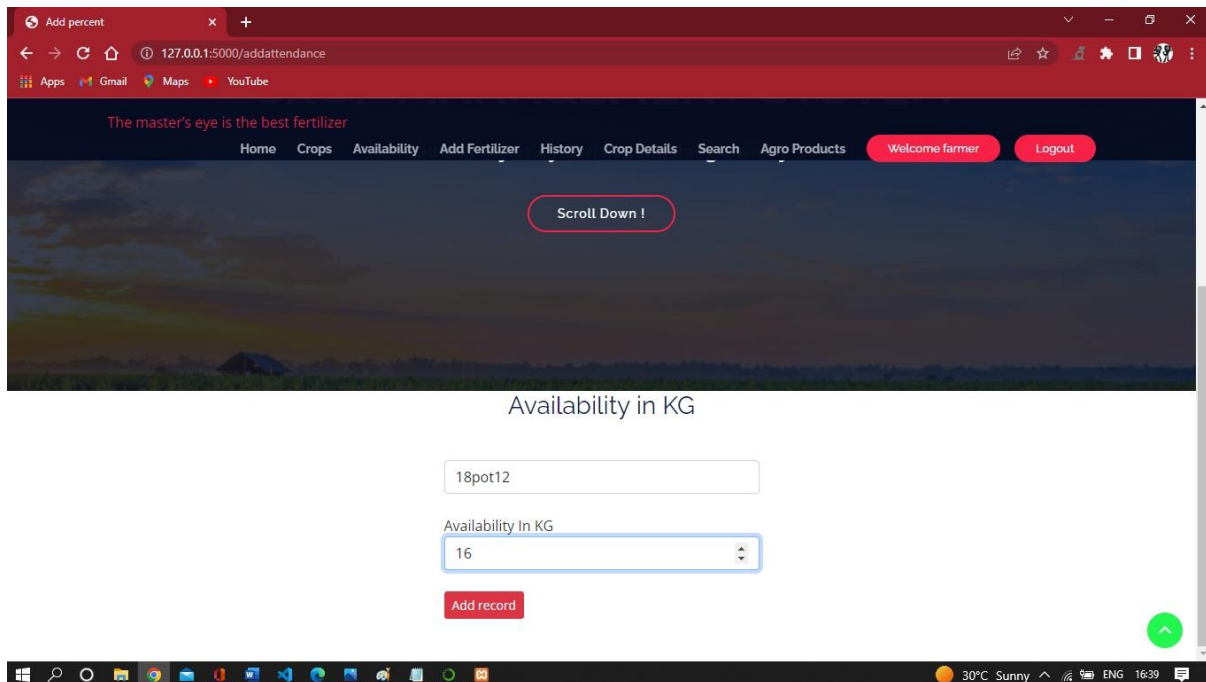
CROP TABLE



The screenshot shows a web browser window with the URL `127.0.0.1:5000/addstudent`. The page has a dark blue header with the text "The master's eye is the best fertilizer" and a navigation bar with links: Home, Crops, Availability, Add Fertilizer, History, Crop Details, Search, Agro Products, Welcome farmer, and Logout. The main content area contains a form with the following fields: "Crop id" (18pot12), "Crop Name" (potato), "Expiry(in months)" (10), "Fruit", "Urea", "Email" (farmer@gmail.com), "Phone Number" (9844604837), and "Address" (hassan). A red "Add Record" button is located at the bottom of the form. The Windows taskbar at the bottom shows the system time as 16:38 and the weather as 30°C Sunny.

Figure 4.4

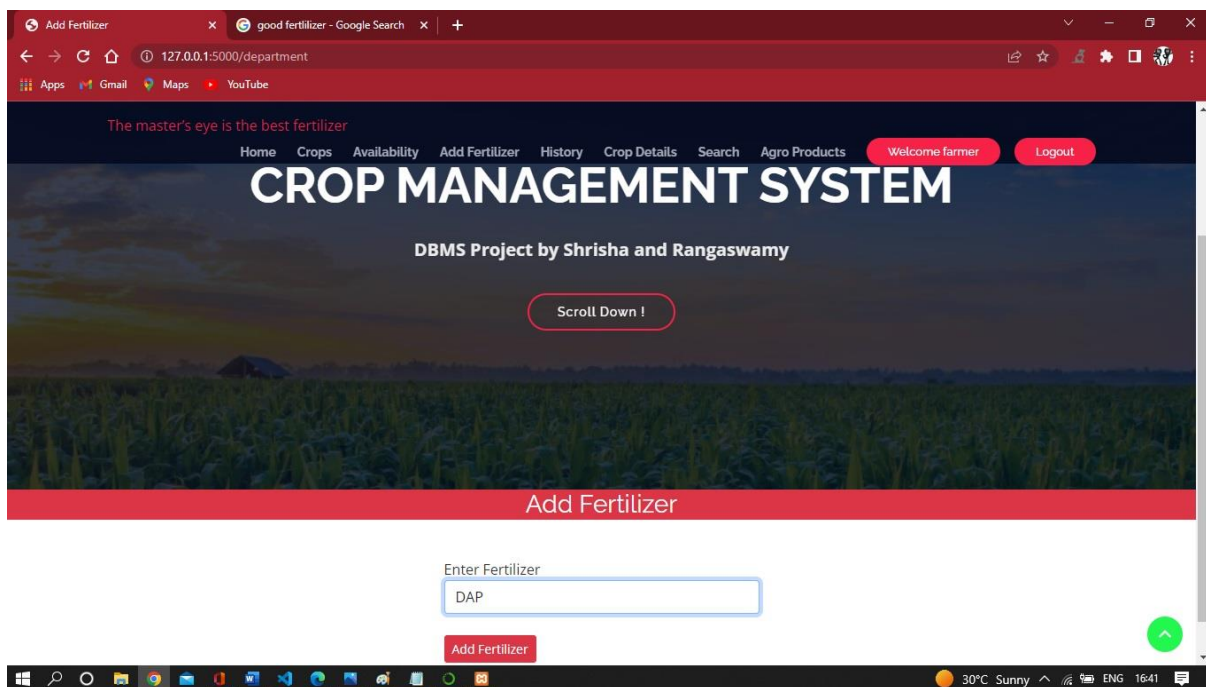
ADD AVAILABILITY TABLE



The screenshot shows a web browser window with the URL `127.0.0.1:5000/addattendance`. The page features a dark blue header with the text "The master's eye is the best fertilizer" and a navigation menu with links: Home, Crops, Availability, Add Fertilizer, History, Crop Details, Search, Agro Products, Welcome farmer, and Logout. A "Scroll Down !" button is centered on the page. Below this, the section "Availability in KG" contains a text input field with the value "18pot12", a dropdown menu for "Availability In KG" with the value "16" selected, and a red "Add record" button. A green up arrow button is located in the bottom right corner of the page.

Figure 4.5

FERTILIZER



The screenshot shows a web browser window with the URL `127.0.0.1:5000/department`. The page features a dark blue header with the text "The master's eye is the best fertilizer" and a navigation menu with links: Home, Crops, Availability, Add Fertilizer, History, Crop Details, Search, Agro Products, Welcome farmer, and Logout. A "Scroll Down !" button is centered on the page. Below this, the section "CROP MANAGEMENT SYSTEM" is displayed, followed by "DBMS Project by Shrisha and Rangaswamy". The "Add Fertilizer" section contains a text input field with the value "DAP" and a red "Add Fertilizer" button. A green up arrow button is located in the bottom right corner of the page.

Figure 4.6

CROP DETAILS

SID	CROP ID	CROP NAME	EXPIRY	TYPE	FERTILIZER USED	EMAIL	NUMBER	ADDRESS	EDIT	DELETE	ADD AGRO PRODUCT
22	12	qwerty	3	Vegetable	Sodium Nitrate	1@h	111	dd	Edit	Delete	ADD
25	12	qwerty	3	Vegetable	Sodium Nitrate	1@h	111	dd	Edit	Delete	ADD
26	12	qwerty	3	Vegetable	Sodium Nitrate	1@h	111	dd	Edit	Delete	ADD
27	111	onion	3	Vegetable	Cyanamide	a@gmail.com	12345678	bangalore	Edit	Delete	ADD
28	18pot12	Potato	12	Fruit	Urea	farmer@gmail.com	9844726558	hassan	Edit	Delete	ADD

Figure 4.7

HISTORY

TID	CROP ID	ACTION	TIMESTAMP
7	12pot24	CROP INSERTED	2022-12-22 20:19:56
8	12oni24	CROP UPDATED	2022-12-21 19:20:31
9	12rad24	CROP DELETED	2022-12-21 19:21:23
44	1	CROP DELETED	2023-01-18 21:53:23
45	1	CROP DELETED	2023-01-18 21:53:26
46	1	CROP DELETED	2023-01-18 21:53:29
47	1	CROP DELETED	2023-01-18 21:53:31
48	1	CROP DELETED	2023-01-18 21:53:34
49	1	CROP DELETED	2023-01-18 21:54:03

Figure 4.8

PRODUCTS

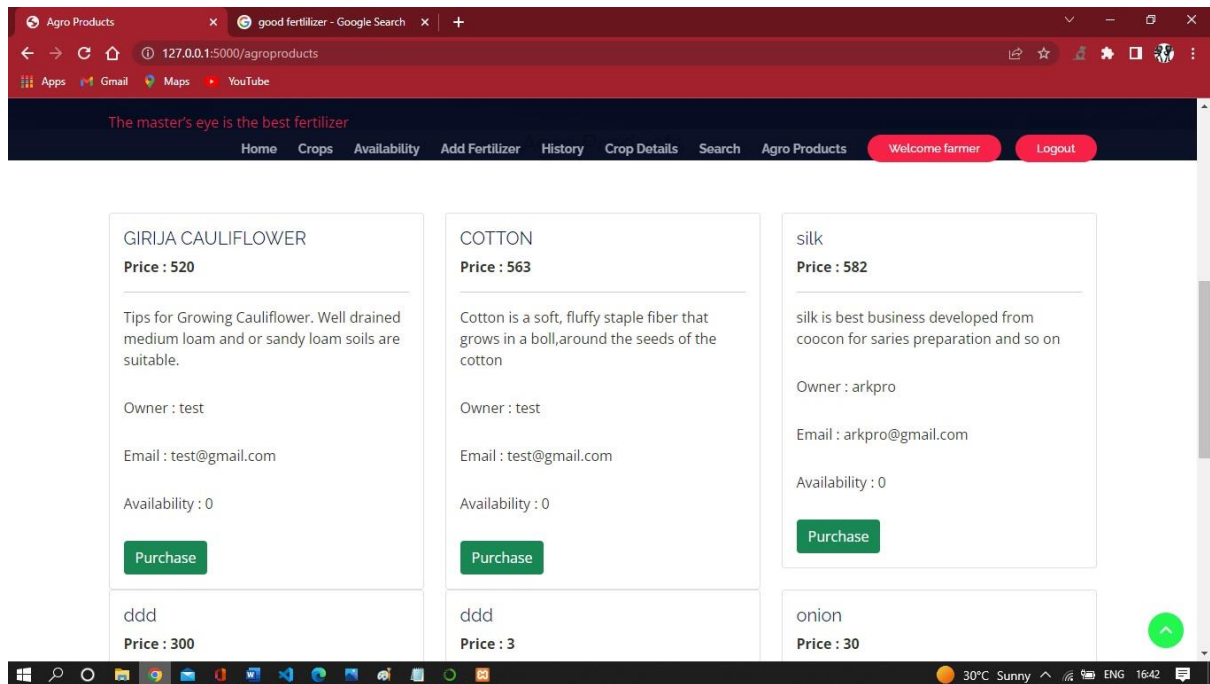


Figure 4.9

CHAPTER 5:

CONCLUSION

We have successfully implemented the CROP MANAGEMENT SYSTEM DATABASE which makes easier for user to manage crop management for the farmers

View tables are used to display all the components of different entities user needs. One can just select the buttons and modify the data as per requirements

We have successfully used various functionalities of HTML,CSS and PYTHON and created the fully functional database system

Crop management system can be used to manage the Crop handling and facilitate a simple transaction in the end helping farmers to sell to customer who orders through Email

In this project many farmer can add the crop into the database easily and sell any crop he needs

5.1 FEATURES

User can update the crop inside the database and also update the availability of the crop

If a Fertilizer used is not already present user can add the fertilizer used

User can edit the crop if the information updated is wrong or needs any changing

User can delete the crop whenever required

A customer can purchase the crop directly by sending the Email to the farmer giving the address

5.2 REFERENCES

- 1.- Fundamentals of Database Systems, RamezElmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson
- 2- Database management systems, Ramakrishnan, and Gehrke, 3rd Edition, 2014, McGraw Hill
- 3-SilberschatzKorth and Sudharshan, Database System Concepts, 6th Edition, Mc-GrawHill, 2013.
- 4- Coronel, Morris, and Rob, Database Principles Fundamentals of Design, Implementation and Management, Cengage Learning 2012
- 5- www.youtube.com
- 6- www.github.com
- 7- www.w3schools.com
- 8- Python IDLE:
<https://www.python.org/downloads/>
- 9- www.javatpoint.com/xampp
- 10- <https://www.g2.com/categories/crop-management>
- 11- www.getbootstrap.com

5.3 SOFTWARE USED

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

5.4 HARDWARE USED

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

5.5 FUTURE ENHANCEMENTS

Enhanced database storage facility

Enhanced user friendly GUI

more advanced results systems

online payments