INDEX

PROBLEM STATEMENT

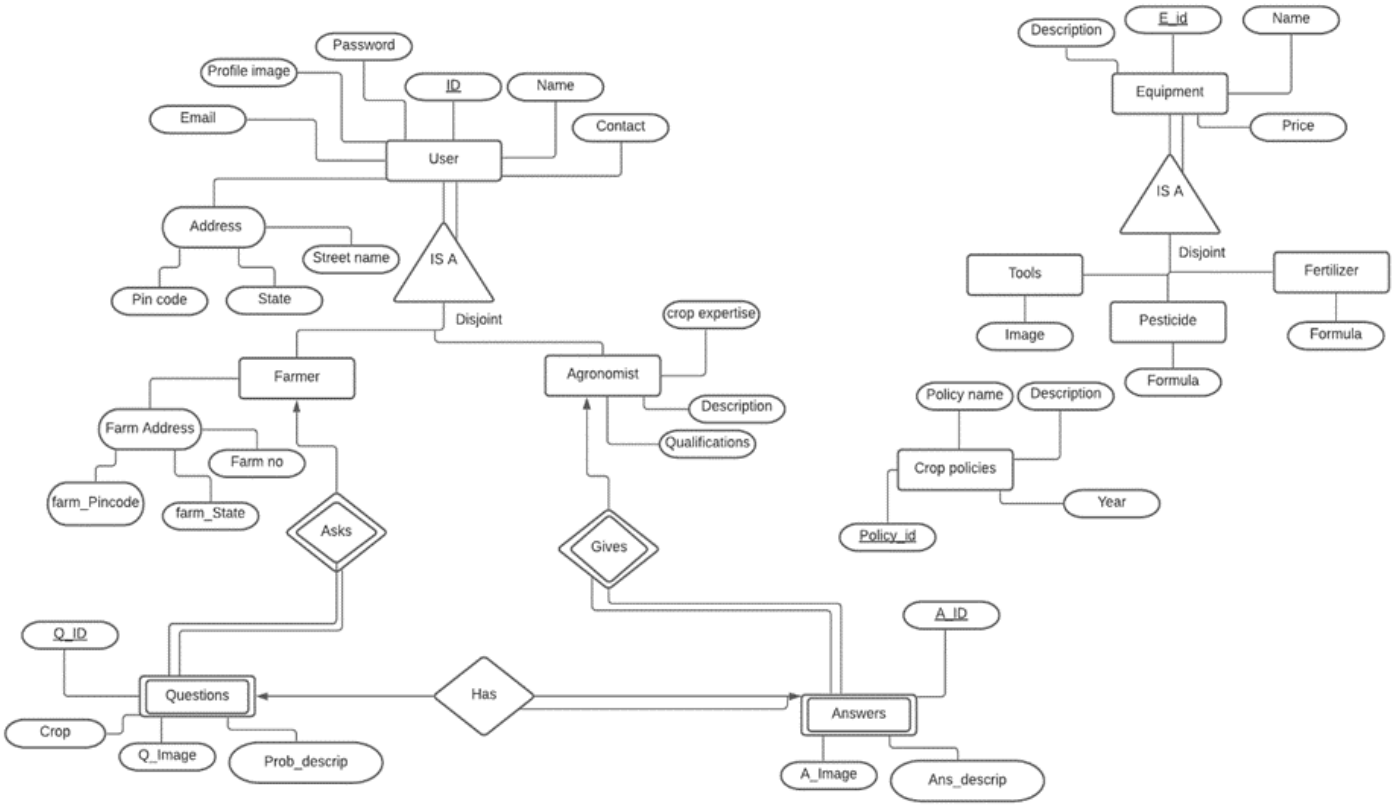
In current scenario, Farmers in our country face a lot of challenges which are harbored by their lack of education and lack of a facility to educate themselves. Farming is affected due to various factors such as change in weather, soil type, irrigation facilities etc. Apart from this, farmers have other difficulties such as market price drop, crop quality check, no information on export parameters of a crop, government aid and support and more. This creates a requirement of platform where the marginal farmers can ask their doubts to agriculture experts and work upon the solutions to improve the situation. Our Integrated Crop Management system caters to this problem.

INTRODUCTION TO THE PROJECT

Integrated Crop Management system will have users mainly in the form of Farmers and Agronomists. This system will connect farmers with agriculture experts who can mentor and solve problems of marginal farmers and their farm. Farmers can ask their questions from the platform. This system will also enable farmers and agronomists to view lists of fertilizers, pesticides, tools and government policies relating to farming. There can be many more features, but only above-mentioned core functionalities will be built as part of this project. Detailed requirements include:

1. Farmers and Agronomists will be able to view list of equipments which will include Fertilizers, Pesticides and Tools with images from database.
2. Farmer and Agronomists will be able to view list of crop policies, which will have attributes, name, year, description and link to the official policy websites.
3. Farmers will be able to ask questions concerning farming activities and more. Questions will include their crop name, problem description and question image.
4. Users will be able to see previously asked questions, and their respective answers with images if applicable. All questions and answers will be visible to all so that doubts and queries can be solved.

ER MODEL DESCRIPTION



	character string	alpha numeric string	image	number
Farmer	Name	email	profile image	ID
	state			contact
	street name			password
	farm no			pincode
	farm state			farm pincode
Agronomist	Name	email	profile image	ID
	state			contact
	street name			password
	Qualification			pincode
	description			
	crop Expertise			
Question	crop	problem description	ques image	Q_id
Has (Relationship Set)		Answer description	Ans image	A_id
				Q_id
Answer		answer description	ans image	A_id
Pesticides	name	formula		id

		description		price
Fertilizers	name	formula		id
		description		price
Tools	name	description	image	id
Crop policies	name	description		id
				year

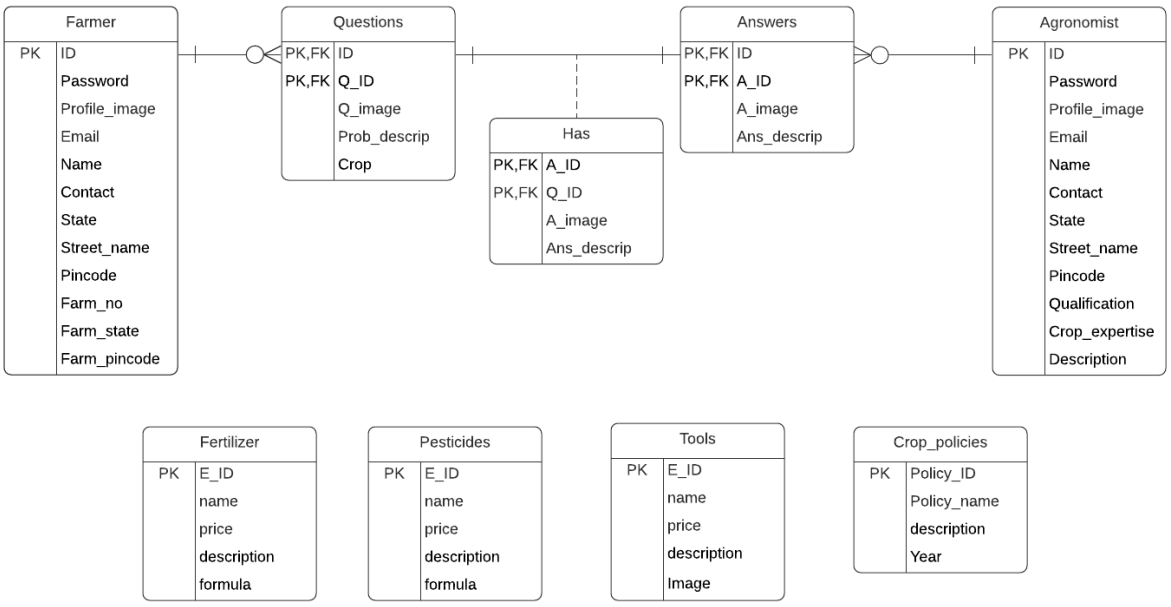
Relationship Sets are: Asks, Gives and Has.
We have considered Questions and Answers are weak entity sets because, Questions exist only when a farmer asks them, and Answers exist only when an Agronomist answers and there is a question to answer.

Cardinality

- Gives
  One agronomist can give multiple answers.
  Each and every answer is given by one agronomist.
- Has
  One question will have one answer.
  Every answer will have a question.
- Ask
  One farmer can ask multiple question.
  Every question will be asked by only one farmer.

RELATIONAL MODEL

Schema Diagram



- Farmer(ID ,email, profile_image, password, name, contact, state, street name, pincode, farm_no, farm_state, farm_pincode)

- Agronomist(ID ,email, profile_image, password, name, contact, state, street name, pincode, qualification, crop_expertise, description)
- Questions(ID, Q ID, Q_image, Prob_descrip, crop)
- Answers(ID, A_ID, A_Image, Ans_descrip)
- Has(A_ID, A_Image, Ans_descrip, Q_ID)
- Crop_Policies(Policy_ID, Policy_name, description, year)
- Tools(E ID, name, price, description, image)
- Pesticide(E ID, name, price, description, formula)
- Fertilizer(E ID, name, price, description, formula)

List of tables with attributes, data types and keys

As part of this project, only the above core functionalities are implemented. The ER Model is made in such a way that the project can be scalable in future. Answers relation now includes q_id as foreign key instead of creating a HAS mapping table in order to simplify database processing in the current pilot project. For the functionalities present in the project, only following tables are created:

Relation	Attribute	Data Type	Primary Key	Foreign Key	Not null	Unique
Question	q_id	int, auto increment	✓	-	✓	✓
	q_image	longblob	-	-	-	-
	crop	char	-	-	-	-
	q_description	varchar	-	-	-	-
Answer	ans_id	int, auto increment	✓	-	✓	✓
	ans_image	longblob	-	-	-	-
	ans_description	varchar	-	-	-	-
	q_id	int	-	✓	-	-
crop_policy	policy_id	int	✓	-	✓	✓
	policy_name	char	-	-	-	-
	policy_year	int	-	-	-	-
	policy_desc	varchar	-	-	-	-
	link	varchar	-	-	-	-
fertilizer	f_id	int	✓	-	✓	✓
	f_name	char	-	-	-	-
	price	int	-	-	-	-
	f_desc	varchar	-	-	-	-
	formula	varchar	-	-	-	-
pesticide	p_id	int	✓	-	✓	✓
	p_name	char	-	-	-	-
	price	int	-	-	-	-
	p_desc	varchar	-	-	-	-
	formula	varchar	-	-	-	-
tools	t_id	int	✓	-	✓	✓

	t_name	char	-	-	-	-
	price	int	-	-	-	-
	t_desc	varchar	-	-	-	-
	tool_image	longblob	-	-	-	-

NORMALIZATION

Normalization is the process of minimizing redundancy from a relation or set of relations. Redundancy in relation may cause insertion, deletion and updation anomalies. So, it helps to minimize the redundancy in relations. Normal forms are used to eliminate or reduce redundancy in database tables.

1NF

If a relation contain composite or multi-valued attribute, it violates first normal form or a relation is in first normal form if it does not contain any composite or multi-valued attribute. A relation is in first normal form if every attribute in that relation is singled valued attribute.

2NF

To be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency. A relation is in 2NF if it has No Partial Dependency, i.e., no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table.

3NF

A relation is in third normal form, if there is no transitive dependency for non-prime attributes as well as it is in second normal form. A relation is in 3NF if at least one of the following condition holds in every non-trivial function dependency X –> Y

BCNF

A relation R is in BCNF if R is in Third Normal Form and for every FD, LHS is super key. A relation is in BCNF if in every non-trivial functional dependency X –> Y, X is a super key.

1. Crop_Policies(Policy_ID, Policy_name, description, year)
    - All attributes are single value, so it satisfies 1NF
    - Policy_name, description, year functionally dependent on policy_ID, so it satisfies 2NF
    - Transitive doesn’t exist, so it satisfies 3NF
    - Policy_ID is superkey, so it satisfies BCNF
2. Tools(E ID, name, price, description, image)
    - All attributes are single value, so it satisfies 1NF
    - name, price, description, image functionally dependent on policy_ID, so it satisfies 2NF
    - Transitive doesn’t exist, so it satisfies 3NF
    - E ID is superkey, so it satisfies BCNF

3. Pesticide(E ID, name, price, description, formula)
    - All attributes are single value, so it satisfies 1NF
    - name, price, description, formula functionally dependent on policy_ID, so it satisfies 2NF
    - Transitive doesn’t exist, so it satisfies 3NF
    - E ID is superkey, so it satisfies BCNF

4. Fertilizer(E ID, name, price, description, formula)
    - All attributes are single value, so it satisfies 1NF
    - name, price, description, formula functionally dependent on policy_ID, so it satisfies 2NF
    - Transitive doesn't exist, so it satisfies 3NF
    - E ID is superkey, so it satisfies BCNF

5. Questions (Q ID, Q_image, Prob_descrip, crop)
    - Q_image, Prob_descrip, crop store atomic data and therefore questions is in 1NF.
    - Q_ID is the primary key. Q_image, Prob_descrip, crop attributes fully functionally depend on Q_Id therefore satisfies 2NF.
    - There is no transitivity, so it satisfies 3NF.
    - Q_ID is the superkey, so it satisfies BCNF.
6. Answer (A_ID, A_Image, Ans_descrip, Q_ID)
    - A_ID is the primary key.
    - A_Image, Ans_descrip, Q_id store atomic data and therefore Answer is in 1NF.
    - One Question can have multiple possible answers, however for every answer there is just one question. So A_ID fully functionally determines Q_ID. A_Image and Ans_descrip also fully functionally depend on A_ID. Hence Answer is in 2NF.
    - Transitivity doesn't exist as A_ID only has functional dependency to all attributes, no other attribute functionally determines any other attribute. Therefore is in 3NF.
    - A_ID is the superkey, so it satisfies BCNF.

IMPLEMENTATION

1. Frontend
    - HTML and CSS has been used to implement frontend.
    - These are easy to implement and integrate with backend.

dbms project > static > # style.css > .background

```css
@import url('https://fonts.googleapis.com/css2?family=Roboto:wght@300&display=swap');
*{
    margin: 0;
    padding: 0;
}

html{
    scroll-behavior: smooth;
}
.logo{
    width: 20%;
    display: flex;
    justify-content: center;
    align-items: center;
}

.logo img{
    width: 33%;
    border: 3px solid black;
    border-radius: 50px;
```

```
dbms project > templates > index.html > html > body
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel= "stylesheet" type= "text/css" href="{{ url_for('static',filename='style.css') }}">
    <!--<link rel="stylesheet" href="style.css">-->
    <title>Integrated crop management</title>
</head>
<body>
    <nav class="navbar background">
        <ul class="nav-list">
            <div class="logo"><!--<img src="http://localhost/static/logo.jpg" alt="logo">--></div>
            <li><a href="#home">Home</a></li>
            <li><a href="#about">About</a></li>
            <li><a href="#ourteam">Our Team</a></li>
            <div class="dropdown">
            <div class="dropbtn">Services</div>
            <div class="dropdown-content">
            <a href="http://localhost/app?doAction=showfertilizers">Fertilizers</a>
            <a href="http://localhost/app?doAction=showpesticides">Pesticides</a>
            <a href="http://localhost/app?doAction=showtools">Tools</a>
            <a href="http://localhost/app?doAction=showpolicies">Crop Policies</a>
            </div>
            </div>
            <li><a href="#askaquestion">Ask a question</a></li>
            <li><a href="http://localhost/app?doAction=showquestions">Previously asked questions</a></li>
        </ul>
        <div class="rightNav">
        <input type="text" name= "search" id="search">
        <button class="btn btn-sm">Search</button>
        </div>
    </nav>

    <section class="background firstSection">
        <div class="box-main">
```

2. Backend
   - MYSQL is used for backend database.
   - MYSQL is a relational database management system based on SQL and is used for a wide range of purposes including data warehousing, e-commerce and logging applications.

   Following are the table creation and dummy data insertion queries.
   - create table question (q_id int not null auto_increment, q_image longblob, crop char(30), q_description varchar(255), primary key(q_id));

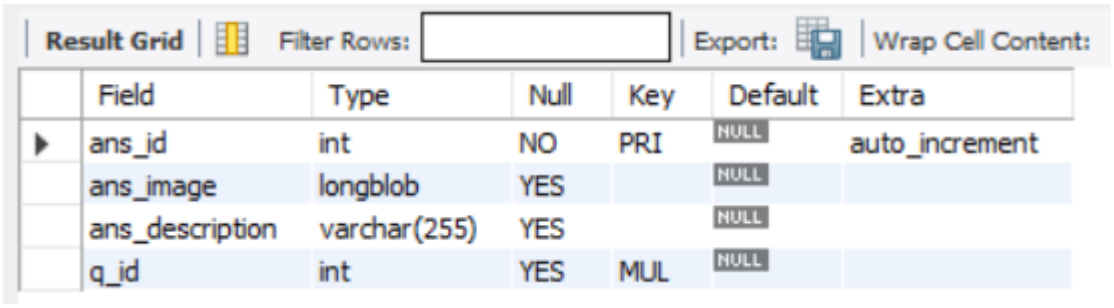
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
q_id	int	NO	PRI	NULL	auto_increment
q_image	longblob	YES		NULL	

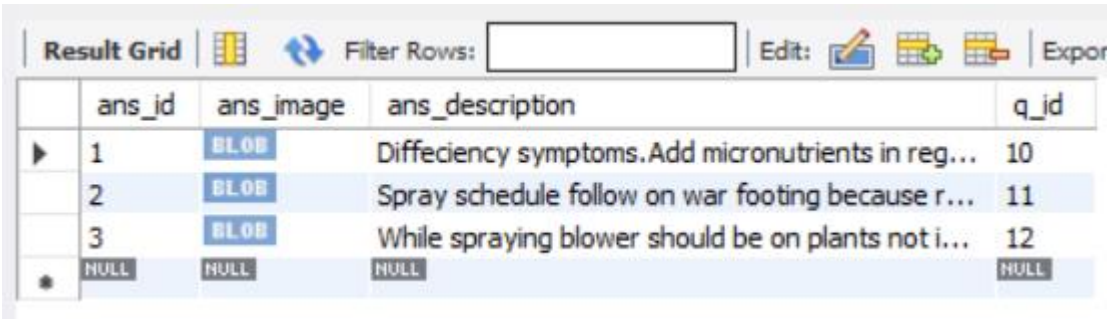
Result Grid | Filter Rows: | Edit: | Expo

q_id	q_image	crop	q_description
10	BLOB	Wheat	The dosage of fertilizers exceeded desired limits...
11	BLOB	Rice	Heavy Rainfall in the region. How to manage th...
12	BLOB	Potato	Potato prices dropping in market. How to delay ...
NULL	NULL	NULL	NULL

- create table answer (ans_id int not null auto_increment, ans_image longblob, ans_description varchar(255), q_id int, foreign key(q_id) references question(q_id), primary key(ans_id));

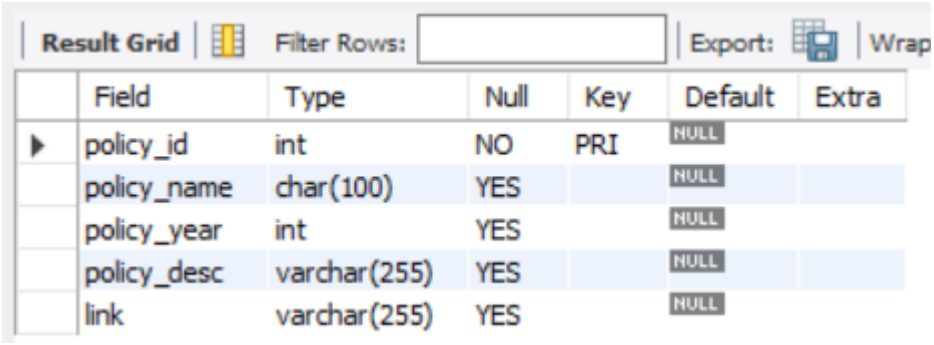


Field	Type	Null	Key	Default	Extra
ans_id	int	NO	PRI	NULL	auto_increment
ans_image	longblob	YES		NULL	
ans_description	varchar(255)	YES		NULL	
q_id	int	YES	MUL	NULL	

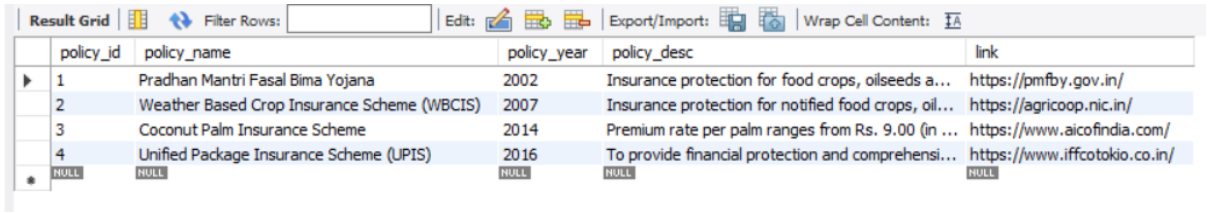


ans_id	ans_image	ans_description	q_id
1	BLOB	Diffeciency symptoms.Add micronutrients in reg...	10
2	BLOB	Spray schedule follow on war footing because r...	11
3	BLOB	While spraying blower should be on plants not i...	12
NULL	NULL	NULL	NULL

- create table crop_policy(policy_id int, policy_name char(100), policy_year int, policy_desc varchar(255), link varchar(255),primary key(policy_id));
- insert into crop_policy values(1,'Pradhan Mantri Fasal Bima Yojana',2002,"Insurance protection for food crops, oilseeds and annual horticultural/commercial crops notified by state government.","https://pmfby.gov.in/");



Field	Type	Null	Key	Default	Extra
policy_id	int	NO	PRI	NULL	
policy_name	char(100)	YES		NULL	
policy_year	int	YES		NULL	
policy_desc	varchar(255)	YES		NULL	
link	varchar(255)	YES		NULL	



policy_id	policy_name	policy_year	policy_desc	link
1	Pradhan Mantri Fasal Bima Yojana	2002	Insurance protection for food crops, oilseeds a...	https://pmfby.gov.in/
2	Weather Based Crop Insurance Scheme (WBCIS)	2007	Insurance protection for notified food crops, oil...	https://agricoop.nic.in/
3	Coconut Palm Insurance Scheme	2014	Premium rate per palm ranges from Rs. 9.00 (in ...	https://www.aicofindia.com/
4	Unified Package Insurance Scheme (UPIS)	2016	To provide financial protection and comprehensi...	https://www.iffcotokio.co.in/
NULL	NULL	NULL	NULL	NULL

- create table fertilizer(f_id int, f_name char(30),price int,f_desc varchar(255),formula varchar(10), primary key(f_id));

- insert into fertilizer values(1,'Ammonium sulphate',100,"It can be applied before sowing, at the time of sowing or as a top-dressing to the growing crop.",'(NH4)2 S04');

Result Grid | Filter Rows: | Export:

Field	Type	Null	Key	Default	Extra
f_id	int	NO	PRI	NULL	
f_name	char(30)	YES		NULL	
price	int	YES		NULL	
f_desc	varchar(255)	YES		NULL	
formula	varchar(10)	YES		NULL	

Result Grid | Filter Rows: | Edit: | Export/Import:

f_id	f_name	price	f_desc	formula
1	Ammonium sulphate	100	It can be applied before sowing, at the time of ...	(NH4)2 S04
2	Sodium nitrate	200	Sodium nitrate is particularly useful for acidic soils	NaNO3
3	Ammonium nitrate	300	This fertilizer is quick-acting, but highly hygrosc...	NH4N03
4	Urea	400	It is suitable for most crops and can be applied t...	CO(NH2)2
5	Super phosphate	200	Phosphatic fertilizer hardly moves in the soil and...	Ca(H2PO4)2
NULL	NULL	NULL	NULL	NULL

- create table pesticide(p_id int, p_name char(30),price int,f_desc varchar(255),formula varchar(10), primary key(p_id));
- insert into pesticide values(1,'Carbofuran',100,"Insecticide for potato, corn, soybean, and ornamentals.",'C12H15NO3');

Result Grid | Filter Rows: | Export:

Field	Type	Null	Key	Default	Extra
p_id	int	NO	PRI	NULL	
p_name	char(30)	YES		NULL	
price	int	YES		NULL	
f_desc	varchar(255)	YES		NULL	
formula	varchar(10)	YES		NULL	

Result Grid | Filter Rows: | Edit: | Export/Import:

p_id	p_name	price	f_desc	formula
1	Carbofuran	100	Insecticide for potato, corn, soybean, and orna...	C12H15NO3
2	Ethylene Dibromiden	120	oil and grain fumigant used on vegetable and gr...	C2H4Br2
3	Endrin	150	Insecticide/Pesticide used on Field crops.	C12H8Cl6O
4	Hexachlorobenzene	200	Formed as the byproduct of the manufacture of...	C6Cl6
5	Oxamyl	140	Used on many field crops, fruits and vegetables	C7H13N3O3S
NULL	NULL	NULL	NULL	NULL

- create table tools(t_id int, t_name char(20),price int,t_desc varchar(255),tool_image longblob, primary key(t_id));

Result Grid | Filter Rows: | Export:

Field	Type	Null	Key	Default	Extra
t_id	int	NO	PRI	NULL	
t_name	char(20)	YES		NULL	
price	int	YES		NULL	
t_desc	varchar(255)	YES		NULL	
tool_image	longblob	YES		NULL	

Result Grid | Filter Rows: | Edit: | Export/Import:

t_id	t_name	price	t_desc	tool_image
1	Tractor	300000	Its power and size allow it to work in rough terr...	BLOB
2	Motocultor	100000	The tiller is a type of agricultural machinery of a...	BLOB
3	Rake	1200	The main function of this toothed bar is to loose...	BLOB
4	Machete	5000	It is used for mowing the grass, cutting or pruni...	BLOB
5	Shovel	11000	The general usage of this equipment is to dig th...	BLOB
NULL	NULL	NULL	NULL	NULL

3. Connectivity

- Python flask is used to connect frontend with backend MYSQL database.
- Flask is a Python web framework or a third-party Python library used for developing web applications.
- We import mysql.connector module for connection.
- To interact with database tables, cursor is needed.

```
cnx = mysql.connector.connect(user='mitali', password='[redacted]',
                              host='localhost',
                              database='farmers')
print("Connecting to database")
cursor = cnx.cursor()
```

- Following modules are needed in the project

```
from flask import Flask, render_template, request
import mysql.connector
from mysql.connector import errorcode
import base64
from PIL import Image
import io
```

- Flask application is as follows:

```
app = Flask(__name__)
@app.route('/app', methods=['GET', 'POST'])
def index():
    print(request.path)
```

```
    items = list()
    templateName = 'index.html'
    if request.args.get('doAction') == 'showfertilizers':
        templateName = '/showfertilizers.html'
        items = fertilizer_db()
        print(items)
    if request.args.get('doAction') == 'showpesticides':
        templateName = '/showpesticides.html'
        items = pesticides_db()
        print(items)
    if request.args.get('doAction') == 'showpolicies':
        templateName = '/showpolicies.html'
        items = policies_db()
        print(items)
    if request.args.get('doAction') == 'showtools':
        templateName = '/showtools.html'
        items = tools_db()
        print(items)
    if request.args.get('doAction') == 'saveQuestion':

        if request.method == 'POST':
            question = request.form.get('questionText')
            db_cropname = request.form.get('cropname')
            db_image = request.form.get('image')
            print(question)
            print(db_cropname)
            f = request.files['image']
            saveQuestionToDB(question,db_cropname,f)
    if request.args.get('doAction') == 'showquestions':
        templateName = '/showquestions.html'
        items = showquestions_db()
        print(items)
    if request.args.get('doAction') == 'showanswers':
        templateName = '/showanswers.html'
        q_id = request.args.get('qid')
        items = showanswers_db(q_id)
        print(items)

    templateName = render_template(templateName, items=items,
static_folder='/static')


    return templateName
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=80)
```

Here, GET and POST methods are applicable, with output port 80. Request.args.get() is used to get particular functionality call from frontend. According to this, various python functions are called where database connection is made, cursor executes queries, data is appended in list, which is then passed to frontend for display.

- For flask, the frontend images and CSS needs to be in a folder called static.
- All the HTML files need to be in Templates folder.
- FERTILIZERS

```
query = "select * from fertilizer"
cursor.execute(query)
items = list()
rows = cursor.fetchall()
for row in rows:
    items.append(row)
    print(row)

cursor.close()
return items
```

HTML: showfertilizers.html

```
        <tr>
          <th>ID</th>
          <th>Name</th>
          <th>Price</th>
          <th>Description</th>
          <th>Formula</th>
        </tr>
        {% for item in items %}
        <tr>
          <td>{{ item[0] }}</td>
          <td>{{ item[1] }}</td>
                <td>{{ item[2] }}</td>
                <td>{{ item[3] }}</td>
                <td>{{ item[4] }}</td>
        </tr>
        {% endfor %}
    </table>
  </body>
</html>
```

- PESTICIDES

```
query = "select * from pesticide"
        cursor.execute(query)
        items = list()
        rows = cursor.fetchall()
        for row in rows:
            items.append(row)
            print(row)

        cursor.close()
        return items
```

- TOOLS

Images are inserted into database outside of the frontend application by encoding the image to get base64 string using base64.b64encode() passing the file read from file location.

Tools.py

```python
from flask import Flask, render_template, request
import mysql.connector
from mysql.connector import errorcode
import base64
from PIL import Image
import io
try:
    cnx = mysql.connector.connect(user='mitali', password='',
                                  host='localhost',
                                  database='farmers')
    print("Connecting to database")
    cursor = cnx.cursor()


    # Open a file in binary mode
    file = open('C:/Users/prash/Desktop/NMIMS MPSTME NEW/flask/tools image/Shovel.jpg','rb').read()

    # We must encode the file to get base64 string
    file = base64.b64encode(file)

    # Sample data to be inserted
    args = ('5','Shovel','11000',"The general usage of this equipment is to dig the soil. It is very versatile and can get us out of a hurry at

    # Prepare a query
    query = 'INSERT INTO TOOLS VALUES(%s, %s, %s, %s, %s)'

    # Execute the query and commit the database.
    cursor.execute(query,args)
    cnx.commit()
except mysql.connector.Error as err:
        if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
            print("Something is wrong with your user name or password")
        elif err.errno == errorcode.ER_BAD_DB_ERROR:
            print("Database does not exist")
        else:
            print(err)
```

Tools_db() function

```python
query = "select t_id,t_name,price, t_desc from tools"
#query1 = "select tool_image from tools"
cursor.execute(query)
items = list()
rows = cursor.fetchall()
for row in rows:

    items.append(row)
    print(row)
for row in rows:
    getImagesFromDB(row)
cursor.close()
return items
```

GetImagesFromDB() function

```python
def getImagesFromDB(id):
    # Get images from DB
    # itereate over items of blob
    # for each item, save the blob as image at folder location with .jpg extension
    try:
        cnx = mysql.connector.connect(user='mitali', password='',
                                      host='localhost',
                                      database='farmers')
        print("Connecting to database")
        cursor = cnx.cursor()



        query = "select t_id,tool_image from tools where t_id="+str(id[0])
        #query1 = "select tool_image from tools"
        cursor.execute(query)
        #items = list()
        rows = cursor.fetchall()
        for row in rows:
            #print("Id = ", row[0], )
            #print("Name = ", row[1])
            image = row[1]
            fileName = row[0]
            binary_data = base64.b64decode(image)
            #print("Storing image disk \n")
            write_file(binary_data, "C:/Users/prash/Desktop/NMIMS MPSTME NEW/flask/dbms project (1)/dbms project/static/"+str(fileName)+".jpg")

        cursor.close()
        #return items
```

Base64 image string is fetched from database and decoded. It is stored in the given location on disk with filename as id.jpg with tools id coming from database.

```
    <tr>
      <th>ID</th>
      <th>Name</th>
      <th>Price</th>
      <th>Description</th>
      <th>Image</th>
    </tr>
    {% for item in items %}
    {% set my_string = item[0] | string() + '.jpg' %}
    <tr>
      <td>{{ item[0] }}</td>
      <td>{{ item[1] }}</td>
                <td>{{ item[2] }}</td>
                <td>{{ item[3] }}</td>
                <td class="secondHalf">
                    <img src="{{ url_for('static',filename=my_string ) }}" alt="tools image" class="imgFluid">
                </td>
    </tr>
    {% endfor %}
  </table>
 </body>
</html>
```

- CROP POLICY

```
query = "select * from crop_policy"
#query1 = "select tool_image from tools"
cursor.execute(query)
items = list()
rows = cursor.fetchall()
for row in rows:
    items.append(row)
    print(row)

cursor.close()
return items
```

```
        <th>ID</th>
        <th>Name</th>
        <th>Year</th>
        <th>Description</th>
        <th>Link</th>
      </tr>
      {% for item in items %}
      <tr>
        <td>{{ item[0] }}</td>
        <td>{{ item[1] }}</td>
                <td>{{ item[2] }}</td>
                <td>{{ item[3] }}</td>
                <td>
                    <a href= "{{ item[4] }}">
                        Link
                    </a>
                </td>
      </tr>
      {% endfor %}
    </table>
  </body>
</html>
```

- ASK QUESTION

```
if request.args.get('doAction') == 'saveQuestion':

    if request.method == 'POST':
        question = request.form.get('questionText')
        db_cropname = request.form.get('cropname')
        db_image = request.form.get('image')
        print(question)
        print(db_cropname)
        f = request.files['image']
        saveQuestionToDB(question,db_cropname,f)
```

```
file = base64.b64encode(db_image.read())

# Sample data to be inserted
args = (file, db_cropname, question)

# Prepare a query
query = 'INSERT INTO QUESTION(q_image,crop,q_description) VALUES(%s, %s, %s)'


# Execute the query and commit the database.
cursor.execute(query,args)
cnx.commit()
cursor.close()
```

```
<div class="form">
    <form action = "http://localhost/app?doAction=saveQuestion" enctype="multipart/form-data" method="post">
        <input class="form-input" type="text" name="fullname" id="fullname" placeholder="Enter your full name" >
        <input class="form-input" type="text" name="phone" id="phone" placeholder="Enter your contact number" >
        <input class="form-input" type="text" name="cropname" id="cropname" placeholder="Enter the crop name">
        <input class="form-input" type="email" name="email" id="email" placeholder="Enter your email id">
            <textarea class="form-input" name="questionText" id="questionText" cols="30" rows="10" placeholder="Elaborate your concern
        <input class="form-input" type="file" name="image" id="image" placeholder="upload the picture">
        <!--<submit class="btn-sm btn-dark">Submit</submit>-->
        <input type = "submit" value="Upload">
    </form>
</div>
```

Question data is taken from HTML form, passed to flask app. The flask connector inserts this data into questions table in database. The image is first encoded to base64 string.

- SHOW ANSWERS with PREVIOUSLY ASKED QUESTIONS

Showing previously asked questions is similar to tools with question images displaying along with description and crop name. In addition to that, q_id is also present with link to the question's answer page.

```
query = "select ans_id,ans_description from answer where q_id ="+str(id)
cursor.execute(query)
items = list()
rows = cursor.fetchall()
for row in rows:

    items.append(row)
    print(row)
for row in rows:
    getAnsImagesFromDB(id)
cursor.close()
return items
```

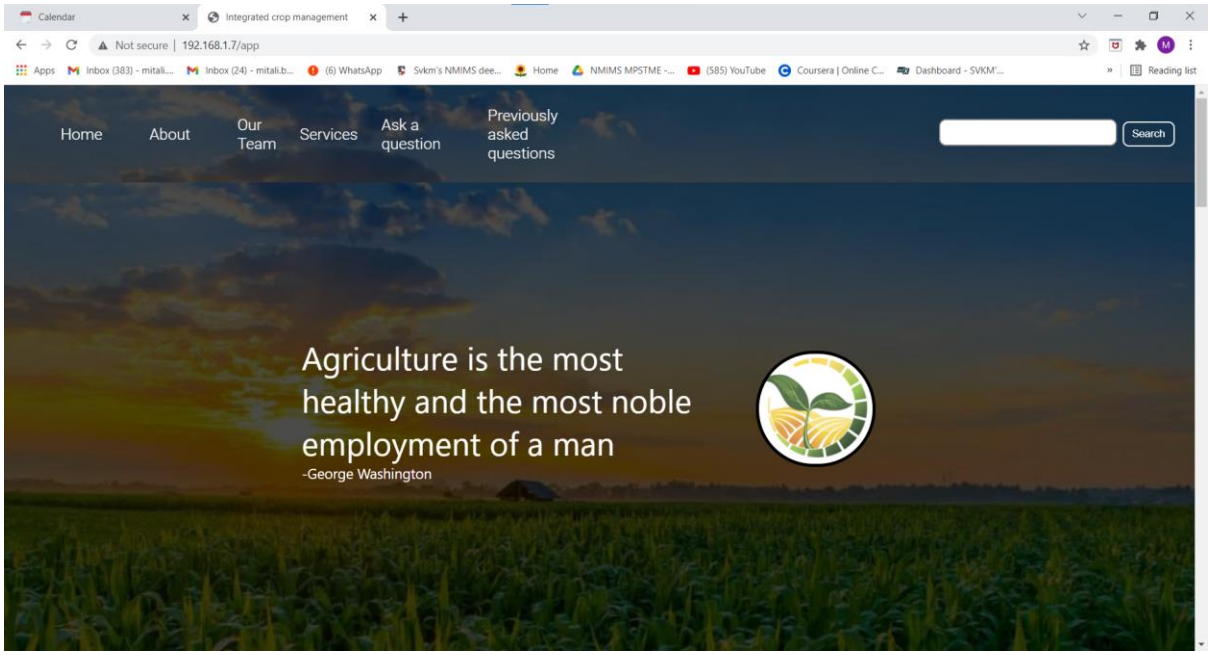
Answers are again displayed, with query being executed by appending where q_id = with str(id) passed from flask main function and html. The answer images are also retrieved in the same manner as tools images.

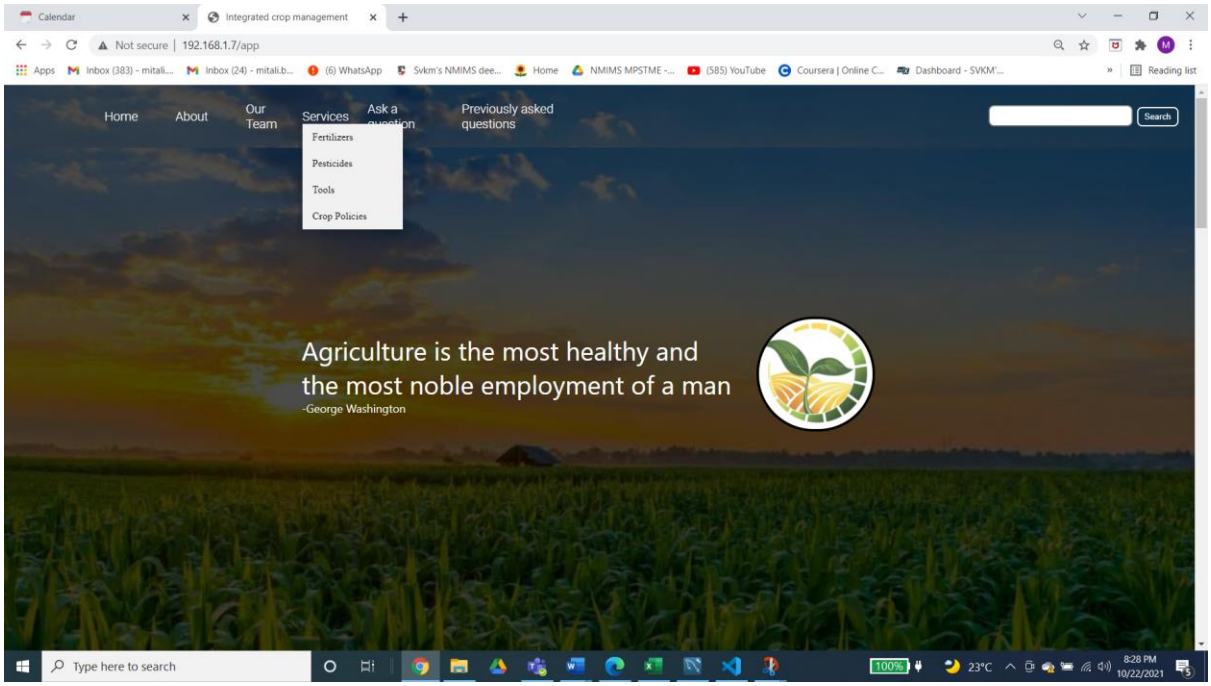
```
  <tr>
    <th>ID</th>
    <th>Crop name</th>
    <th>Description</th>
    <th>Image</th>
  </tr>
  {% for item in items %}
  {% set my_string = item[0] | string() + 'quest.jpg' %}
  <tr>
    <td><a href="http://localhost/app?doAction=showanswers&qid={{item[0]}}">{{ item[0] }}</a></td>
    <td>{{ item[1] }}</td>
    <td>{{ item[2] }}</td>
    <td class="secondHalf">
        <img src="{{ url_for('static',filename=my_string ) }}" alt="question image" class="imgFluid">
    </td>

  </tr>
  {% endfor %}
</table>
```

```
  <tr>
    <th>ID</th>
    <th>Description</th>
    <th>Image</th>
  </tr>
  {% for item in items %}
  {% set my_string = item[0] | string() + 'ans.jpg' %}
  <tr>
    <td>{{ item[0] }}</td>
    <td>{{ item[1] }}</td>
    <td class="secondHalf">
        <img src="{{ url_for('static',filename=my_string ) }}" alt="answer image" class="imgFluid">
    </td>

  </tr>
  {% endfor %}
</table>
```

SCREENSHOTS OF WORKING MODEL

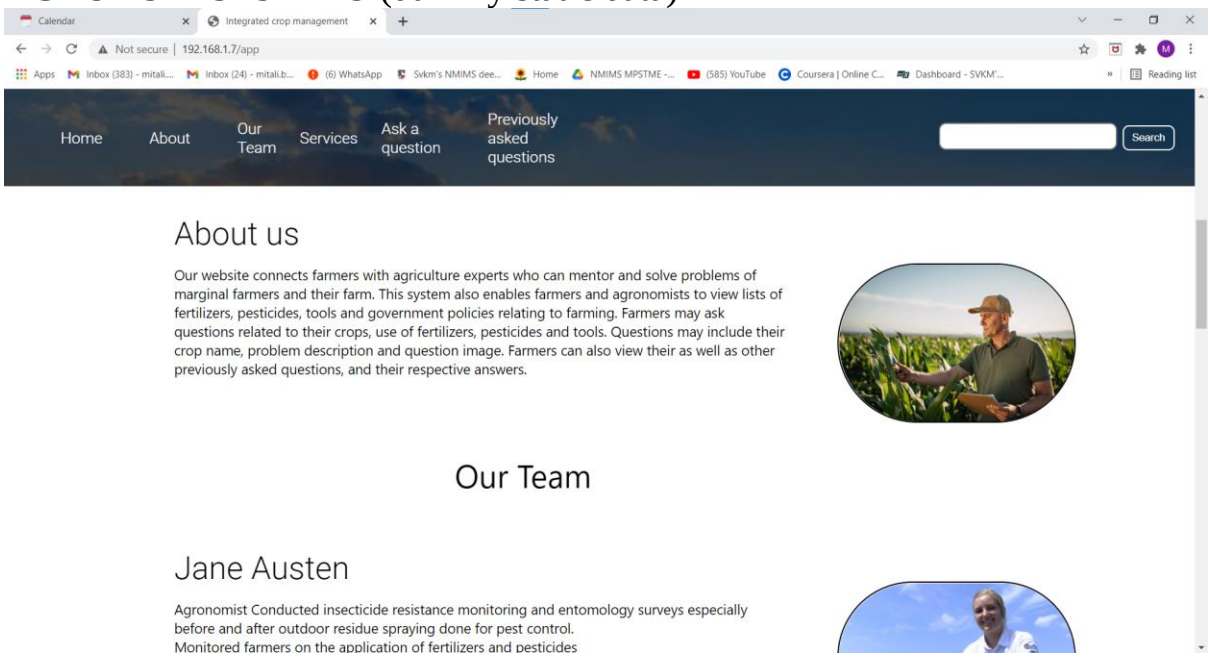
HOME PAGE



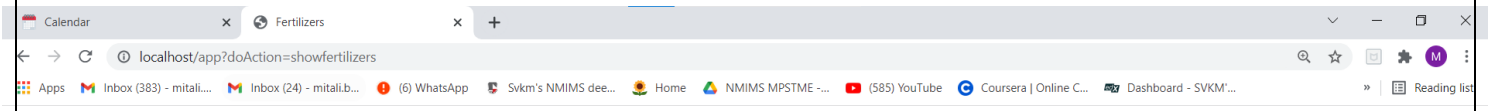


The menu has links to Services which on hover include Fertlizers, Pesticides, Tools and Crop policies page links.

AGRONOMISTS INFO (dummy static data)



FERTILIZERS

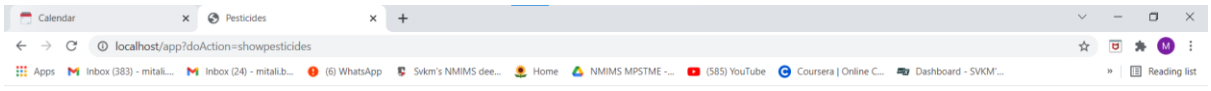


Fertilizers

ID	Name	Price	Description	Formula
1	Ammonium sulphate	100	It can be applied before sowing, at the time of sowing or as a top-dressing to the growing crop.	(NH4)2 S04
2	Sodium nitrate	200	Sodium nitrate is particularly useful for acidic soils	NaNO3
3	Ammonium nitrate	300	This fertilizer is quick-acting, but highly hygroscopic and not fit for storage.	NH4N03
4	Urea	400	It is suitable for most crops and can be applied to all soils.	CO(NH2)2
5	Super phosphate	200	Phosphatic fertilizer hardly moves in the soil and hence they are placed in the, root zone.	Ca(H2PO4)2

Displays a table showing lists of fertilizers retrieved from database.

PESTICIDES

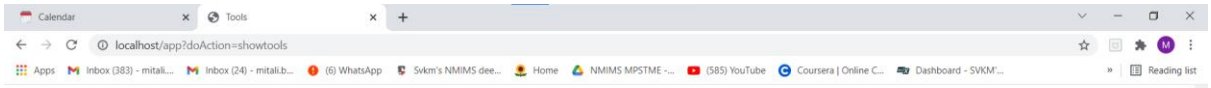


Pesticides

ID	Name	Price	Description	Formula
1	Carbofuran	100	Insecticide for potato, corn, soybean, and ornamentals.	C12H15NO3
2	Ethylene Dibromiden	120	oil and grain fumigant used on vegetable and grain crops	C2H4Br2
3	Endrin	150	Insecticide/Pesticide used on Field crops.	C12H8Cl6O
4	Hexachlorobenzene	200	Formed as the byproduct of the manufacture of other chemicals Attenuation in soil environments	C6Cl6
5	Oxamyl	140	Used on many field crops, fruits and vegetables	C7H13N3O3S

Displays a table showing lists of pesticides retrieved from database.

TOOLS

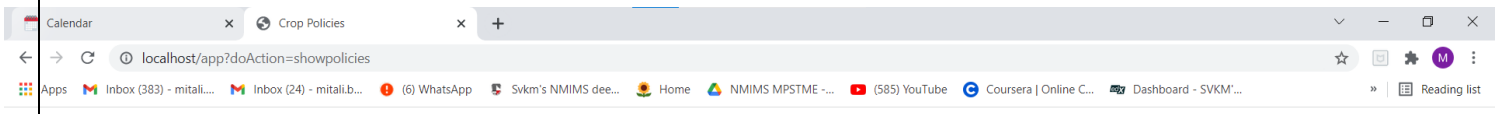


Tools

ID	Name	Price	Description	Image
1	Tractor	300000	Its power and size allow it to work in rough terrain, dragging or towing agricultural implements.	
2	Motocultor	100000	The tiller is a type of agricultural machinery of a single axis.	
3	Rake	1200	The main function of this toothed bar is to loosen and level the ground. The traditional rake has evolved and from being manual or pulled by beasts has happened to fix the tractors.	

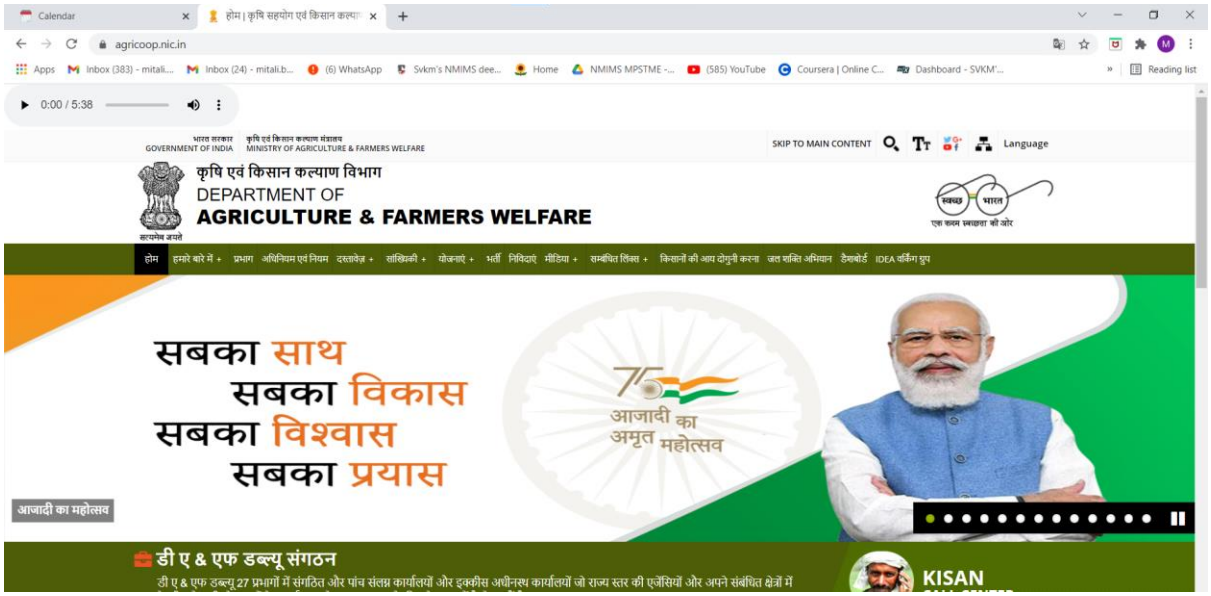
Shows contents of Tools as retrieved with images from database.

CROP POLICIES



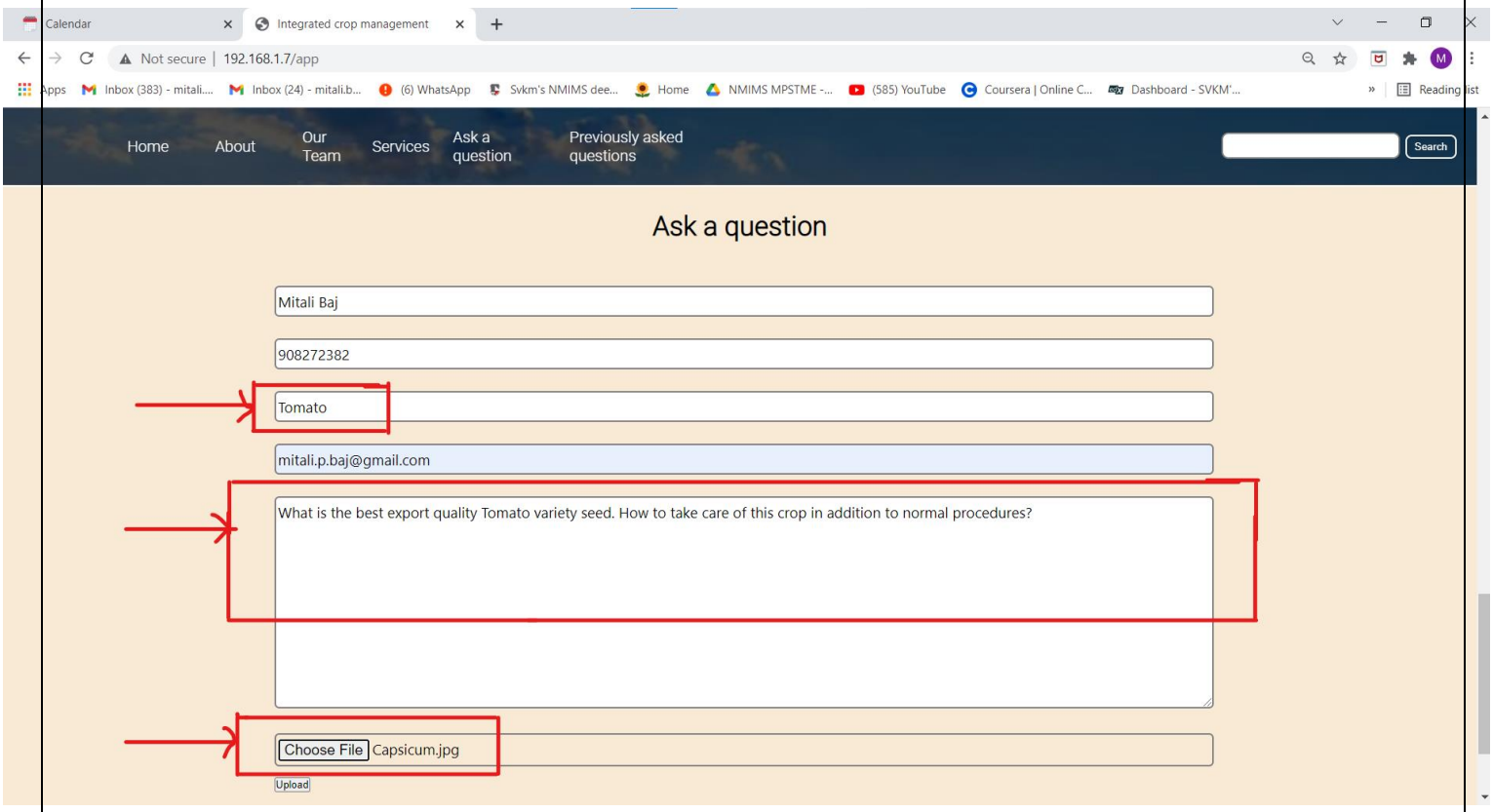
Crop policies

ID	Name	Year	Description	Link
1	Pradhan Mantri Fasal Bima Yojana	2002	Insurance protection for food crops, oilseeds and annual horticultural/commercial crops notified by state government.	Link
2	Weather Based Crop Insurance Scheme (WBCIS)	2007	Insurance protection for notified food crops, oilseeds and horticultural /commercial crops.	Link
3	Coconut Palm Insurance Scheme	2014	Premium rate per palm ranges from Rs. 9.00 (in the plant age group of 4 to 15 years) to Rs. 14.00 (in the plant age group of 16-60 years).	Link
4	Unified Package Insurance Scheme (UPIS)	2016	To provide financial protection and comprehensive risk coverage of crops, assets, life and student safety to farmers.	Link



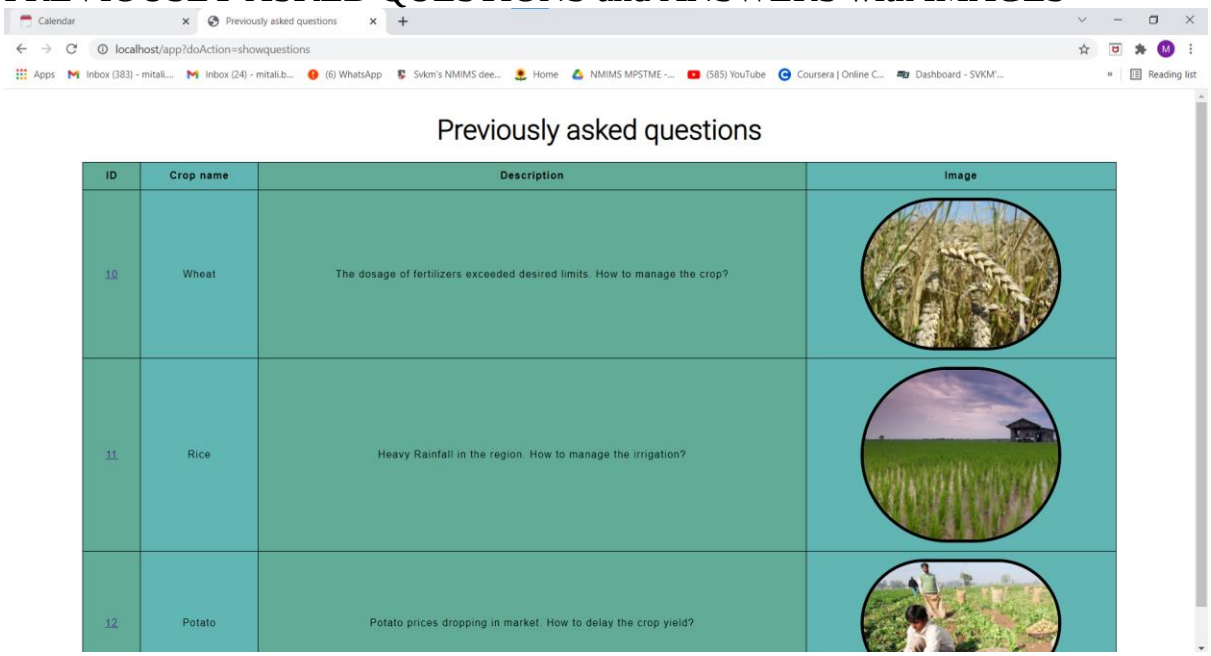
Details of crop policies are shown from database, with links to navigate to official policy websites.

ASK QUESTION



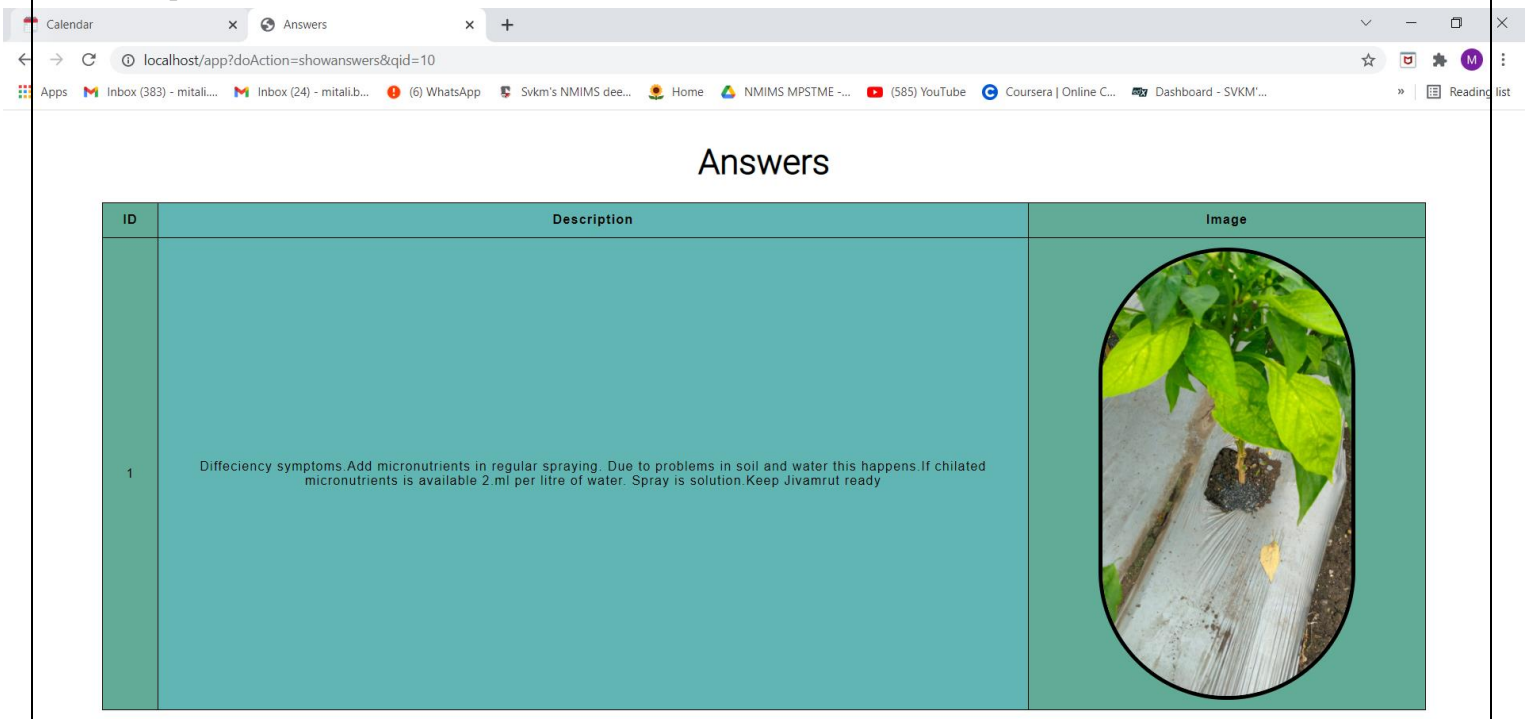
Crop name, Problem description and image can be uploaded for asking a question.

PREVIOUSLY ASKED QUESTIONS and ANSWERS with IMAGES



Questions input from html form are stored in the database, and displayed in the form of table when clicked on Previously asked questions. Each question has a

q_id link for displaying answers if present which are again retrieved from questions table.



CONCLUSION & LEARNING

In current scenario, Farmers face a lot of difficulties due to lack of knowledge and platform to raise their questions. Our project, Integrated Crop Management System, aims to solve these difficulties in the field of Aggrotech. This website displays lists of fertilizers, pesticides, tools with images and crop policies. It also provides functionality for users to ask questions with crop name, problem description along with image if necessary. Previously asked questions with solutions from agronomists are displayed. This is built using HTML, CSS in frontend, MYSQL with PYTHON FLASK as backend and connector. With successful completion of this project, we now have a clear understanding of building an ER Model, transforming it to Relational and Schema model and optimizing the relations with normalization. We also have a clear view of how database management systems are essential to user applications and how stack development takes place.