Contents

List of Tables

List of Figures

1. INTRODUCTION

1.1 Problem Statement

In agricultural communities worldwide, farmers face persistent challenges in accessing timely and accurate information crucial for decision-making regarding crop planting, harvesting, and sales. Despite advancements in technology, many farmers, particularly those in rural areas, struggle with limited access to reliable data, hindering their ability to optimize their agricultural practices and maximize their profits. One critical aspect of this information deficit is the lack of accessibility to up-to-date crop price information, which significantly impacts farmers' economic well-being.

1.2 Objectives

- Develop a system capable of providing real-time updates on crop prices to farmers, traders, and other stakeholders.
- Develop simple models to forecast types of crops based on historical data.
- 3.Gather historical data on crops, weather patterns, soil conditions, and other relevant factors.

1.3 Scope

This project aims to develop a simple web-based application that displays current crop prices and predicts the crops. The prediction model will utilize historical data and basic machine learning techniques to predict the crop.

2. REQUIREMENT SPECIFICATIONS

2.1 Software Requirements:

- Visual Studio Code or any preferred IDE for code development.
- Python, as it is the primary language for the project.
- SQLite for data storage.
- Scikit-learn for implementing machine learning algorithms.
- Django or any other web framework for building web applications.

2.2 Hardware Requirements

- Processor 12th Gen Intel(R) Core (TM) i5-1235U 1.30 GHz
- Installed RAM16.0 GB (15.7 GB usable)
- Windows 11 OS

3. TOOLS USED

In our project, we utilize a combination of powerful tools to streamline and enhance the development process. Visual Studio Code, a popular integrated development environment (IDE), proves invaluable for web development, particularly in creating web applications. Python serves as the project's backbone, providing essential tools, libraries, and a supportive community for the development of robust conversational interfaces capable of handling diverse language interactions. Scikit-learn, a renowned Python machine learning library, offers simplicity and effectiveness, making it a versatile resource for various machine learning tasks. Additionally, Django, a high-level web framework, empowers us to build efficient and scalable web applications using the Python programming language. We provide the description about the tools in Table 1.

Table 1:List of Tools

Tool	**Description**
VISUAL STUDIO CODE	Visual Studio Code is a popular integrated development environment (IDE) that is used for web development, including the creation of web applications. There are several ways in which VS Code can be beneficial in the development of this project.
PYTHON	Python serves as the backbone for web development, providing the tools, libraries, and community support necessary to create robust and versatile conversational interfaces capable of handling interactions in multiple languages.
SCIKIT-LEARN	Scikit-learn is a Python machine learning library renowned for its simplicity and effectiveness. It provides a comprehensive set of tools for various machine learning tasks, offering user-friendly interfaces and integration with other Python libraries. With support for diverse algorithms and functionalities, it serves as a versatile and widely adopted resource for both beginners and experienced practitioners in the field of machine learning.
DJANGO	Django is a high-level web framework for building web applications using the Python programming language.

4. SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

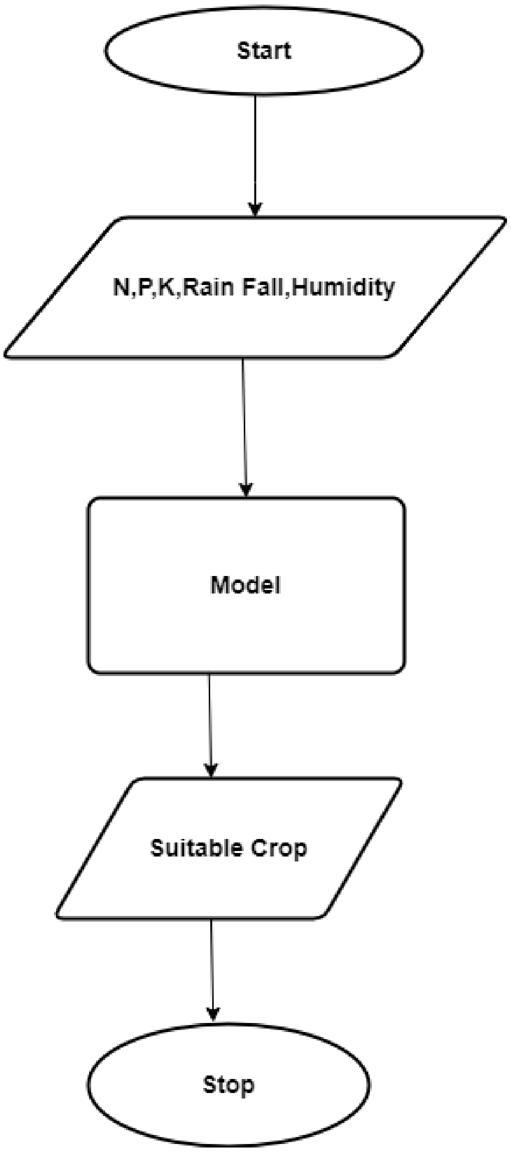


Figure 1:Crop Prediction

In this project, the flow chart begins with data collection, encompassing environmental factors like temperature, humidity, and soil nutrient levels. The dataset is then prepared by separating features and the target variable—suitable crop. Next, the dataset undergoes a split into training and testing sets. The RandomForestClassifier is initialized, trained on the dataset, and fine-tuned for optimal performance. Once satisfied, the model is deployed to predict suitable crops based on new environmental inputs.

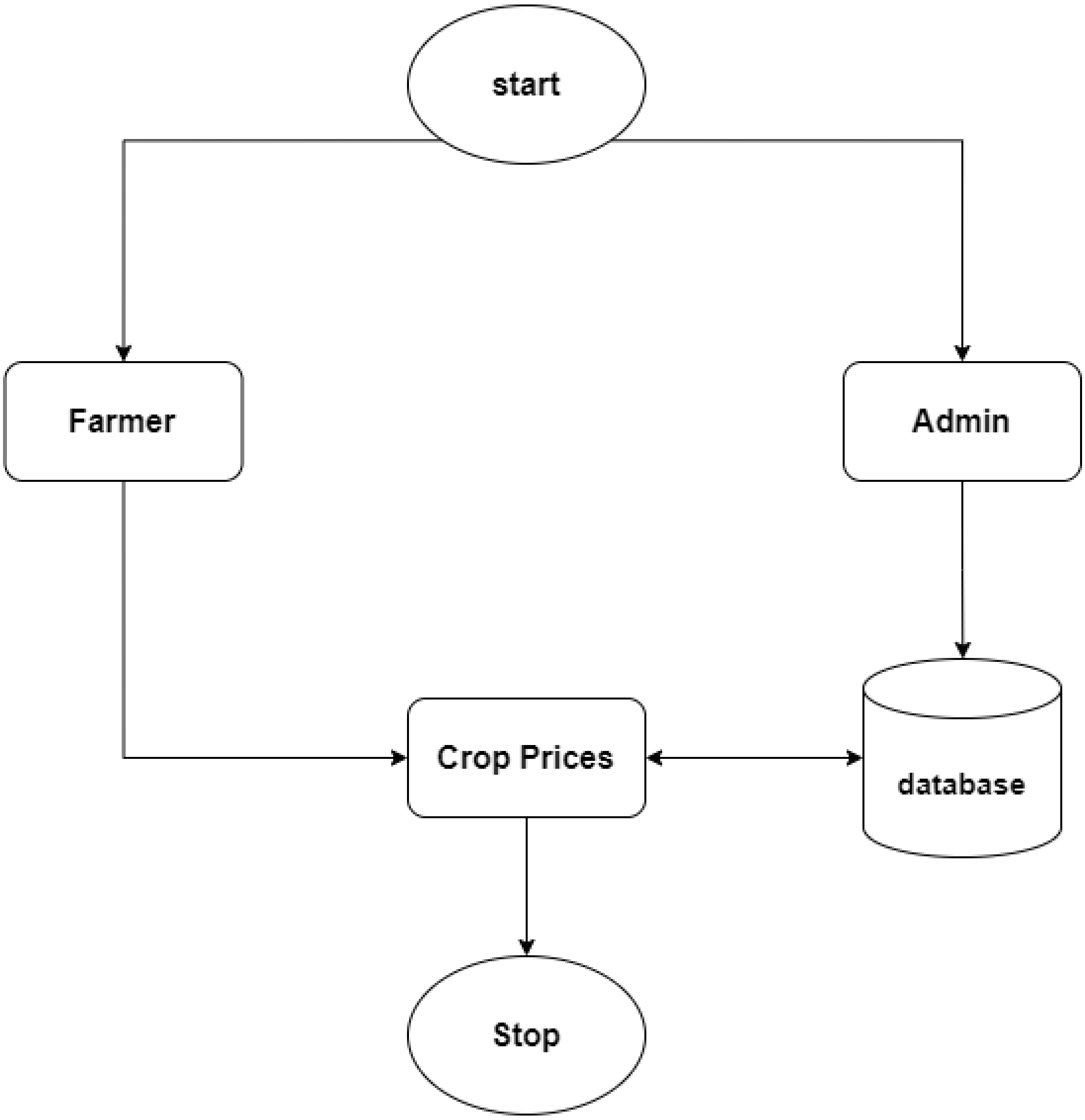


Figure 2:CropPrices

The crop price updation module integrates real-time market data, utilizes machine learning for price prediction, and provides a user-friendly interface for decision-making. This essential component bridges the gap between farmers and market trends, fostering a resilient and efficient agricultural ecosystem.

5.IMPLEMENTATION

5.1 Crop Recommendation Dataset

This dataset contains information about the levels of nitrogen, phosphorus, and potassium in soil, along with temperature, humidity, pH, and rainfall. It also includes details on how these factors impact the growth of crops. The data serves as a foundation for making data-driven recommendations to achieve optimal nutrient and environmental conditions, ultimately enhancing crop yield.

Below is an example of how a crop recommendation dataset might look in a tabular form. Please note that this is a simplified representation, and in a real-world scenario, a dataset would likely include many more features and samples.

Table 2:Sample Instance of Dataset

Temperature (°C)	**Humidity (%)**	**pH Level**	**Rainfall (mm)**	**Nitrogen Level**	**Phosphorus Level**	**Potassium Level**	**Suitable Crop**
25	70	6.5	100	High	Medium	Low	Rice
30	60	7	150	Medium	High	Medium	Wheat
28	75	6.8	120	Low	Low	High	Maize
22	80	6	80	High	High	Medium	Tomato
35	50	7.5	200	Low	Medium	High	Cotton

In this hypothetical example, each row represents a sample with various environmental factors such as temperature, humidity, pH level, and rainfall, along with soil nutrient levels of nitrogen, phosphorus, and potassium. The "Suitable Crop" column indicates the recommended crop based on these conditions. The dataset can be used to train a machine learning model for crop recommendation.

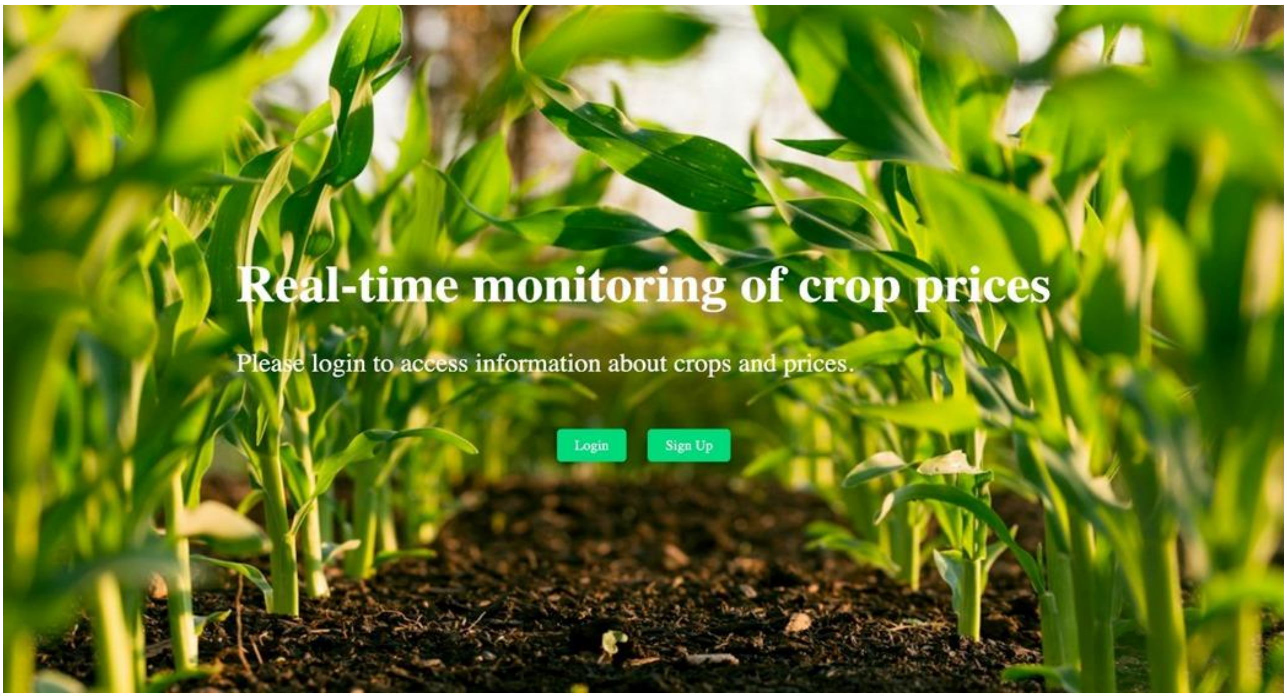
5.2 Module Description

To utilize the RandomForestClassifier for predicting suitable crops in a crop recommendation system dataset, follow these steps. First, import necessary libraries such as scikit-learn's RandomForestClassifier and tools for model evaluation. Next, prepare the dataset by separating features (temperature, humidity, pH, etc.) and the target variable (suitable crop). Split the dataset into training and testing sets using train_test_split. Initialize the RandomForestClassifier with an appropriate number of estimators and random state, then fit it to the training data. Make predictions on the test set and assess the model's performance using metrics like accuracy and classification report. Optionally, fine-tune hyperparameters for optimal results. Once satisfied, deploy the trained model to predict suitable crops based on new environmental inputs, providing valuable insights for crop recommendations in agriculture.

The module for crop price updation in our project serves as a pivotal component, ensuring timely and accurate information for farmers and stakeholders. This module integrates real-time market data, leveraging APIs and data scraping techniques to fetch the latest crop prices.

Utilizing machine learning algorithms, it analyzes historical pricing trends to predict future fluctuations. The module's user-friendly interface enables seamless access to up-to-date pricing information, empowering farmers to make informed decisions. Additionally, it facilitates data visualization, aiding users in comprehending market dynamics. This essential module bridges the gap between farmers and market trends, fostering a more resilient and efficient agricultural ecosystem.

6. RESULT AND DISCUSSIONS



Real-time monitoring of crop prices

Please login to access information about crops and prices.

Figure 3:Home page

Home page, your gateway to a seamless agricultural experience. Here, you can access our platform's full potential by logging in or signing up. As a registered user, you'll unlock a personalized journey tailored to your agricultural needs.

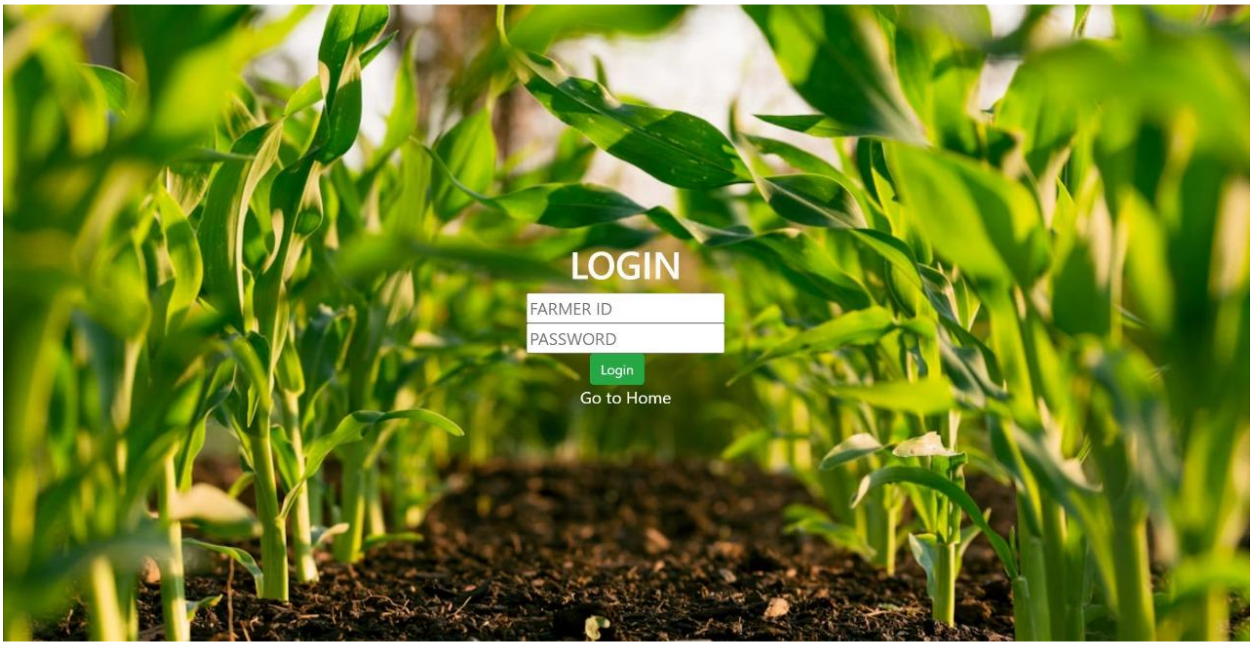
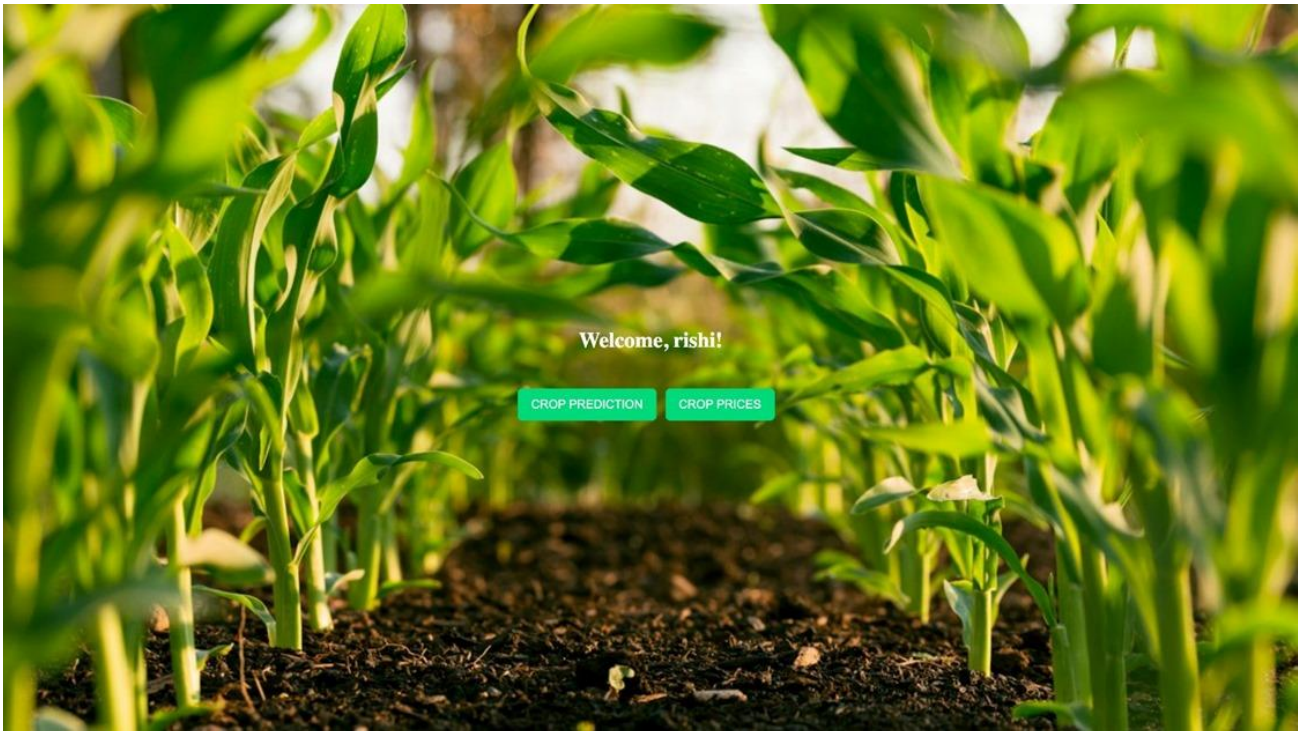


Figure 4:Login Page

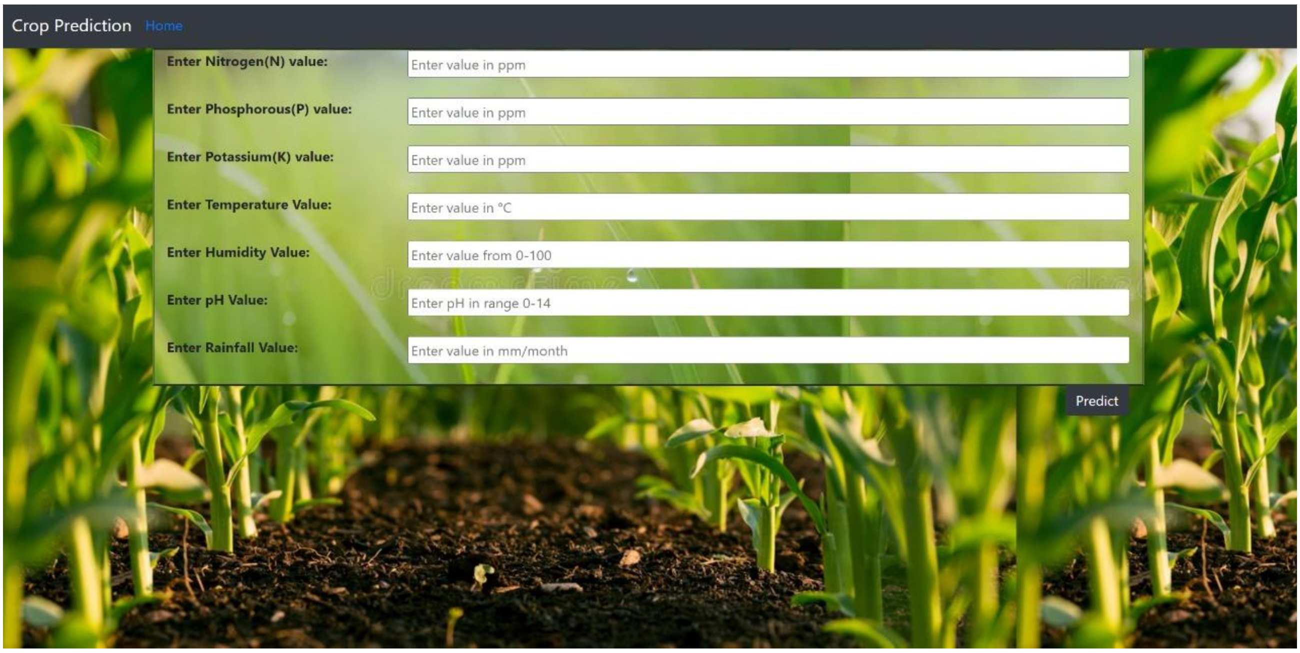
Login page: Enter your credentials to access personalized features and harness the full potential of our platform. Your agricultural journey begins with a simple login.quickly sign up to join our thriving community.



Welcome, rishi!

Figure 5:Dash board

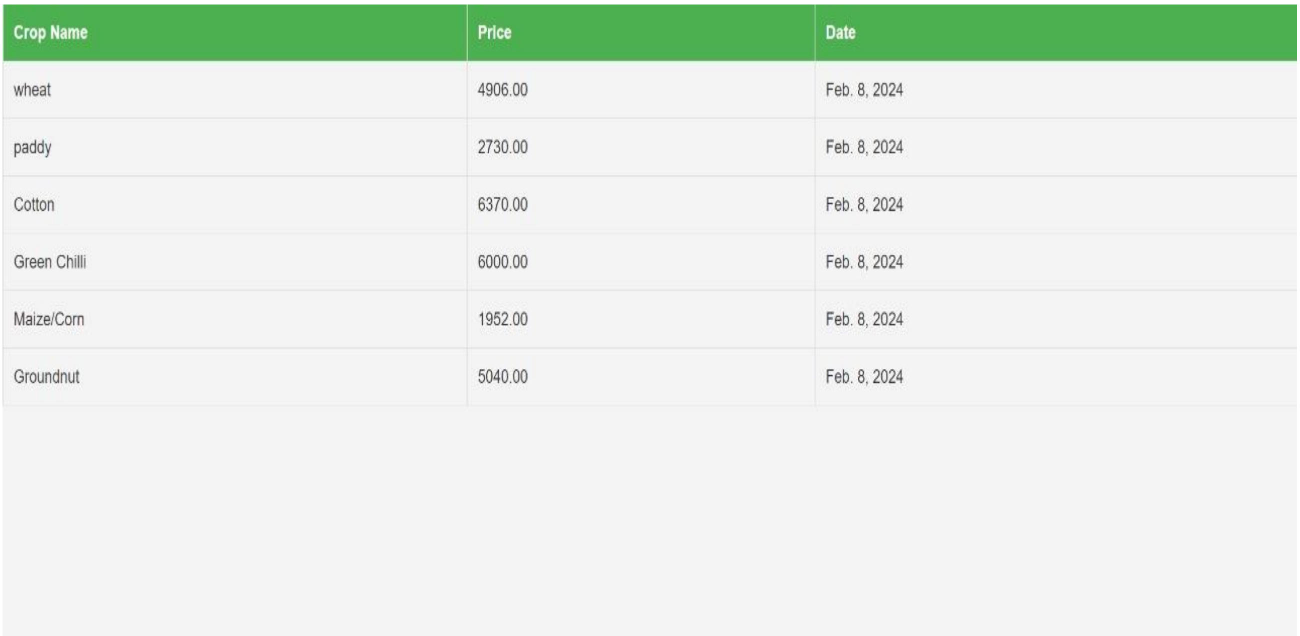
Dashboard:your hub for agricultural insights. Predict crop outcomes with our Crop Prediction tool or stay updated on market dynamics through Crop Prices Update. Empower your decisions effortlessly.



Enter Nitrogen(N) value:	Enter value in ppm
Enter Phosphorous(P) value:	Enter value in ppm
Enter Potassium(K) value:	Enter value in ppm
Enter Temperature Value:	Enter value in °C
Enter Humidity Value:	Enter value from 0-100
Enter pH Value:	Enter pH in range 0-14
Enter Rainfall Value:	Enter value in mm/month

Figure 6:Crop Prediction

Crop Recommendation Page: your guide to optimal crops. Input environmental factors such as Nitrogen, Phosphorus, Potassium, Humidity, and Rainfall. Receive personalized recommendations for crops, ensuring a flourishing agricultural journey.



Crop Name	Price	Date
wheat	4906.00	Feb. 8, 2024
paddy	2730.00	Feb. 8, 2024
Cotton	6370.00	Feb. 8, 2024
Green Chilli	6000.00	Feb. 8, 2024
Maize/Corn	1952.00	Feb. 8, 2024
Groundnut	5040.00	Feb. 8, 2024

Figure 7:Crop prices

Crop Prices Page:your window into real-time market dynamics. Stay informed about the latest trends in crop prices.

7. CONCLUSION

Utilize the RandomForestClassifier from scikit-learn for crop prediction by preparing the dataset. The crop price updation module integrates real-time market data, uses machine learning to predict prices, and offers a user-friendly interface for informed decision-making. This pivotal component strengthens the connection between farmers and market trends, fostering agricultural efficiency.

REFERENCES

1. https://www.python.org/

2. https://scikit-learn.org/stable/modules/ensemble.html#random-forests-and-other-randomized-tree-ensembles

3. https://www.djangoproject.com/

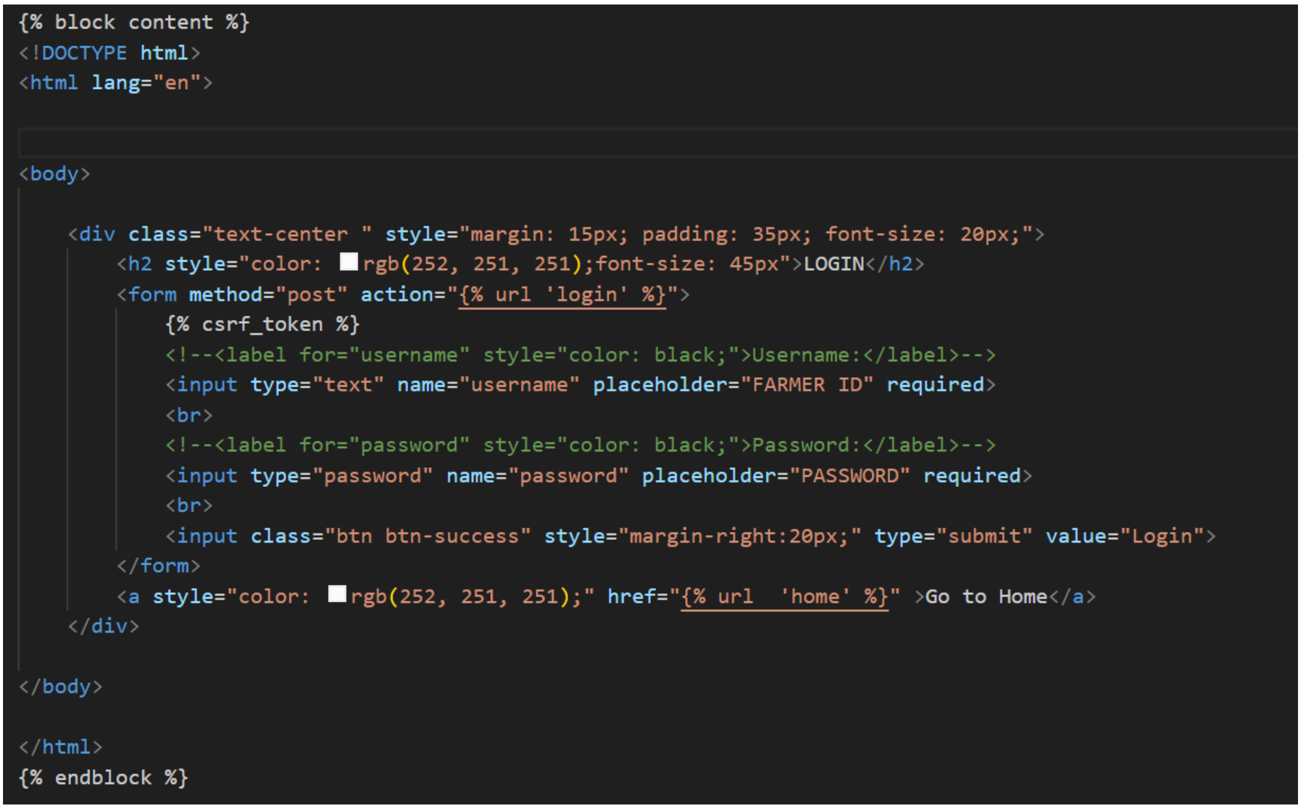
4. https://www.zappycode.com/

5. https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/4GBWFV

APENDEX

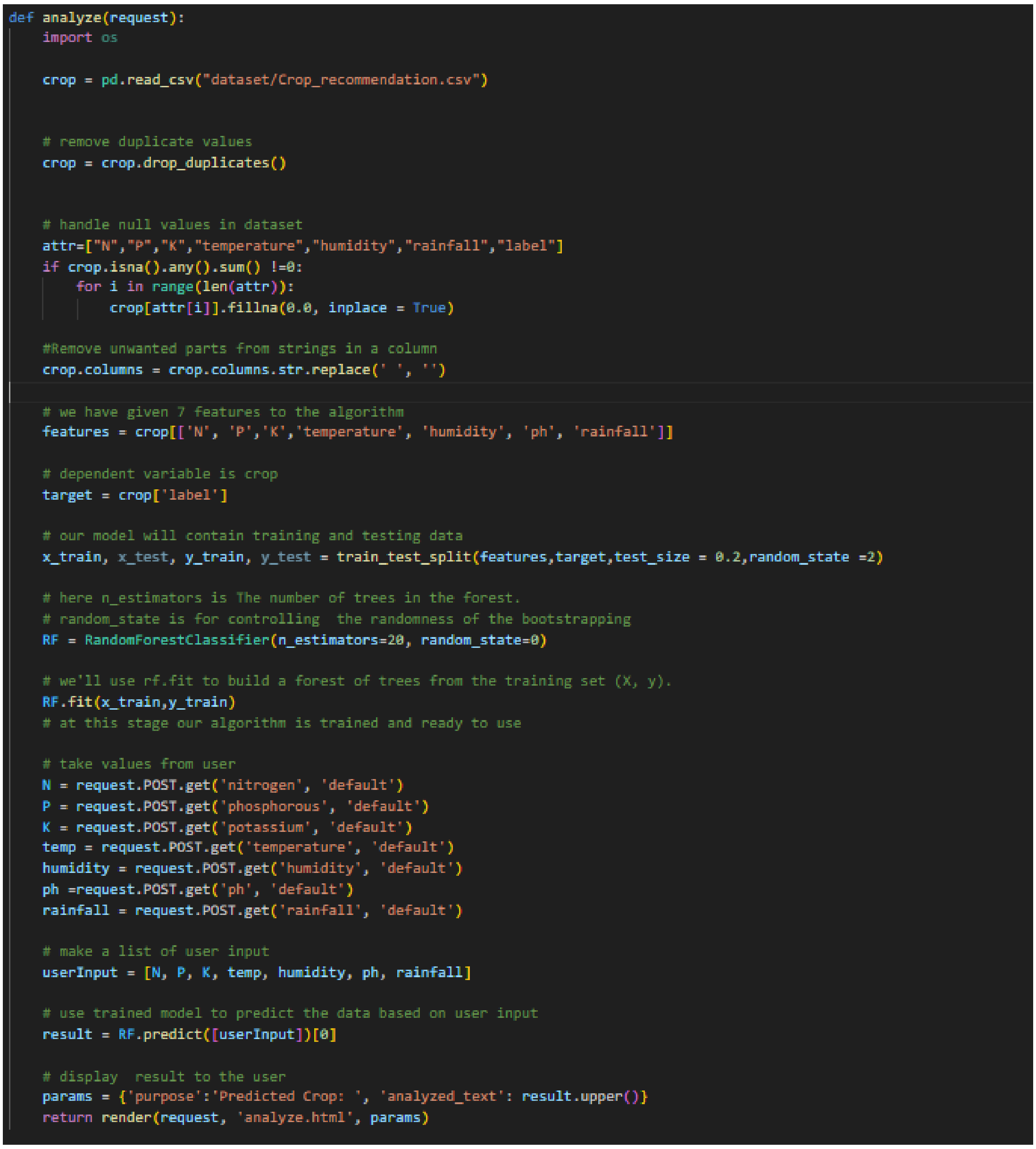
Sample Code

Login page



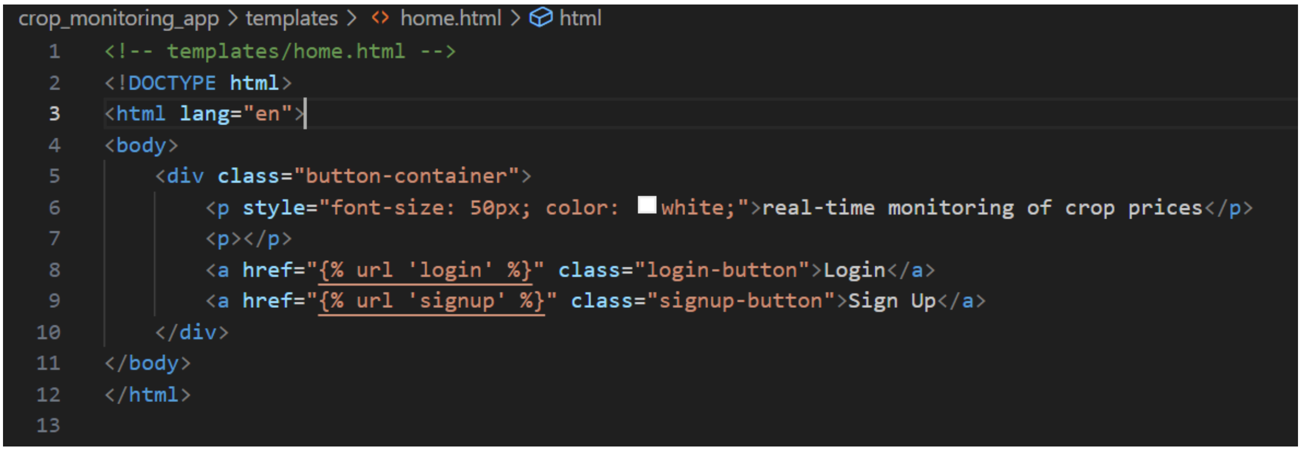
```
{% block content %}
<!DOCTYPE html>
<html lang="en">


<body>

    <div class="text-center " style="margin: 15px; padding: 35px; font-size: 20px;">
        <h2 style="color: □rgb(252, 251, 251);font-size: 45px">LOGIN</h2>
        <form method="post" action="{% url 'login' %}">
            {% csrf_token %}
            <!--<label for="username" style="color: black;">Username:</label>-->
            <input type="text" name="username" placeholder="FARMER ID" required>
            <br>
            <!--<label for="password" style="color: black;">Password:</label>-->
            <input type="password" name="password" placeholder="PASSWORD" required>
            <br>
            <input class="btn btn-success" style="margin-right:20px;" type="submit" value="Login">
        </form>
        <a style="color: □rgb(252, 251, 251);" href="{% url  'home' %}" >Go to Home</a>
    </div>

</body>

</html>
{% endblock %}
```

Analyze page



```
def analyze(request):
    import os

    crop = pd.read_csv("dataset/Crop_recommendation.csv")


    # remove duplicate values
    crop = crop.drop_duplicates()


    # handle null values in dataset
    attr=["N","P","K","temperature","humidity","rainfall","label"]
    if crop.isna().any().sum() !=0:
        for i in range(len(attr)):
            crop[attr[i]].fillna(0.0, inplace = True)

    #Remove unwanted parts from strings in a column
    crop.columns = crop.columns.str.replace(' ', '')

    # we have given 7 features to the algorithm
    features = crop[['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']]

    # dependent variable is crop
    target = crop['label']

    # our model will contain training and testing data
    x_train, x_test, y_train, y_test = train_test_split(features,target,test_size = 0.2,random_state =2)

    # here n_estimators is The number of trees in the forest.
    # random_state is for controlling  the randomness of the bootstrapping
    RF = RandomForestClassifier(n_estimators=20, random_state=0)

    # we'll use rf.fit to build a forest of trees from the training set (X, y).
    RF.fit(x_train,y_train)
    # at this stage our algorithm is trained and ready to use

    # take values from user
    N = request.POST.get('nitrogen', 'default')
    P = request.POST.get('phosphorous', 'default')
    K = request.POST.get('potassium', 'default')
    temp = request.POST.get('temperature', 'default')
    humidity = request.POST.get('humidity', 'default')
    ph =request.POST.get('ph', 'default')
    rainfall = request.POST.get('rainfall', 'default')

    # make a list of user input
    userInput = [N, P, K, temp, humidity, ph, rainfall]

    # use trained model to predict the data based on user input
    result = RF.predict([userInput])[0]

    # display  result to the user
    params = {'purpose':'Predicted Crop: ', 'analyzed_text': result.upper()}
    return render(request, 'analyze.html', params)
```

Home page



```
<!-- templates/home.html -->
<!DOCTYPE html>
<html lang="en">
<body>
    <div class="button-container">
        <p style="font-size: 50px; color: white;">real-time monitoring of crop prices</p>
        <p></p>
        <a href="{% url 'login' %}" class="login-button">Login</a>
        <a href="{% url 'signup' %}" class="signup-button">Sign Up</a>
    </div>
</body>
</html>

```