VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI-590018

Mini-Project
On

"Crop Recommendation System"

A Mini project report submitted in partial fulfillment of the requirements for the 6th semester of ***Bachelor of Engineering in Computer Science Engineering*** *of Visvesvaraya Technological University, Belagavi*

Submitted by:
Darshan R. 1ST21CS057

Under the Guidance of:
Prof. Anitha K.
Assistant Professor
Dept. of CSE, SaIT

SAMBHRAM
INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SAMBHRAM INSTITUTE OF TECHNOLOGY

M.S. PALYA, JALAHALLI, BENGALURU – 5560097

2023-2024

SAMBHRAM
INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the mini project work entitled **"Crop Recommendation System"** has been successfully carried out by Darshan R. bearing 1ST21CS057, bonafide student of **Sambhram Institute of Technology** in partial fulfillment of the requirements for the **6th semester** of **Bachelor of Engineering** in **Computer Science and Engineering** of **Visvesvaraya Technological University**, Belagavi, during academic year 2023-2024. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The Mini Project report has been approved as it satisfies the laboratory requirements of 6th semester BE, CSE.

Prof. Anitha K.
Assistant Professor
Dept. of CSE, SaIT

Dr. T John Peter
HOD-CSE
Dept. of CSE, SaIT

Internal Viva:

Name of the Examiners **Signature with Date**

1.
2.

ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped me in carrying out this project work. We would like to take this opportunity to thank them all.

I would like to thank **Dr. H.G. Chandrakanth**, Principal, SaIT, Bangalore, for his moral support towards completing my project.

I would like to thank **Dr.T. John Peter**, Prof & Head, Department of Computer Science & Engineering, SaIT, Bangalore, for his valuable suggestions and expert advice.

I deeply express my sincere gratitude to my guide to **Prof. Anitha K.** Assistant Professor, SaIT, Bangalore, for their able guidance, regular source of encouragement and assistance throughout this project.

I would like to thank all the teaching and non-teaching staff of Department of Computer Science & Engineering, SaIT, Bengaluru for their constant support and encouragement.

Date: 05/August/2024 **Darshan R.**

Place: Bengaluru

ABSTRACT

The Crop Recommendation System is an advanced, user-friendly platform designed to assist farmers and agricultural professionals in making informed decisions regarding crop selection based on specific soil characteristics. Leveraging the power of Machine Learning (ML) and Python Django, this system takes into account critical soil parameters such as Nitrogen, Potassium, Phosphorous levels, Rainfall, Humidity, pH, and Temperature to provide precise crop suggestions tailored to the user's soil conditions. The backend infrastructure, built on the Python Django framework, ensures robustness and scalability, while the frontend, developed using HTML and CSS, offers a responsive and intuitive user interface. An SQL database is employed for persistent data storage and management, maintaining user details and crop recommendation history, thus facilitating seamless interaction and easy retrieval of historical data. The ML component uses a Random Forest Classifier trained on a comprehensive dataset, enabling high accuracy in predicting the most suitable crop for given soil conditions.

The Crop Recommendation System enhances the decision-making process for crop selection, optimizing yield and sustainability. Users input their soil details through a straightforward form, and the system processes these inputs to provide recommendations based on the trained ML model. The system records each user's interaction and recommendation history in the SQL database, allowing for better crop management practices. This innovative solution significantly benefits the agricultural sector by improving crop yield, resource management, and overall productivity. Future enhancements could include additional soil parameters, a larger dataset, advanced ML algorithms, and a mobile application version to broaden accessibility. In conclusion, this project exemplifies the integration of technology in agriculture, paving the way for more sophisticated and intelligent farming solutions.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 Overview

The Crop Recommendation System represents a significant advancement in agricultural technology, designed to address the pressing need for optimized crop selection in the face of changing environmental conditions and increasing demand for food production. The primary goal of this system is to assist farmers and agricultural professionals in making informed decisions regarding crop cultivation, thereby enhancing productivity and sustainability. By leveraging Machine Learning (ML) and a robust backend infrastructure built on Python Django, this system analyzes various soil parameters, including Nitrogen, Potassium, Phosphorous levels, Rainfall, Humidity, pH, and Temperature, to provide tailored crop recommendations. These parameters are critical for determining soil health and suitability for different crops, ensuring that the recommendations are precise and reliable.

The backbone of the Crop Recommendation System is its architecture, which seamlessly integrates a user-friendly frontend with a powerful backend. The frontend, developed using HTML and CSS, offers an intuitive interface that allows users to input their soil details easily. The backend, powered by the Python Django framework, ensures robustness and scalability, enabling the system to handle numerous user interactions and data processing tasks efficiently. An SQL database is utilized for data management, storing user information and their crop recommendation history. This structure not only facilitates seamless user interaction but also allows for easy retrieval and analysis of historical data, providing users with a comprehensive tool for crop management. By maintaining a detailed record of each user's interaction, the system can offer personalized recommendations and track trends over time.

At the core of the system's recommendation engine is the Random Forest Classifier, a sophisticated ML algorithm known for its high accuracy and robustness in classification tasks. The model is trained on a comprehensive dataset comprising various soil attributes and corresponding optimal crops. By analyzing the input parameters provided by the user, the Random Forest Classifier predicts the most suitable crops for the given soil conditions. This predictive capability enables users to make data-driven decisions, potentially leading to improved crop yields and better resource management. The user experience is designed to be straightforward and efficient: users input their soil details through a simple form, the system processes the information, and generates

crop recommendations based on the ML model's predictions. These recommendations are then presented in an easily understandable format, enhancing the decision-making process for the user. The system also records each recommendation in the SQL database, allowing users to access their recommendation history and make comparisons over time, thus fostering better crop management practices.

1.2 Problem Statement

Agriculture remains the backbone of many economies worldwide, particularly in developing regions where it constitutes a significant portion of employment and livelihoods. However, farmers frequently face challenges related to soil health and crop selection, which can directly impact yield and sustainability. Traditional methods of crop selection are often based on historical practices and anecdotal evidence, which may not account for the specific soil characteristics and current environmental conditions. This lack of precision can lead to suboptimal crop choices, resulting in poor yields, inefficient use of resources, and economic losses for farmers. The need for a more scientific, data-driven approach to crop selection is therefore evident, to enhance productivity and ensure sustainable agricultural practices.

The complexity of soil health assessment further exacerbates the problem. Soil health is influenced by numerous parameters such as Nitrogen, Potassium, Phosphorous levels, Rainfall, Humidity, pH, and Temperature, each playing a crucial role in determining the suitability of the soil for different crops. Farmers and agricultural professionals often lack the tools and expertise to analyze these parameters comprehensively. Consequently, there is a gap between the available scientific knowledge on soil health and its practical application in the field. This gap can hinder efforts to optimize crop selection and maximize agricultural output. Without a reliable system to process and interpret soil data, farmers may continue to make uninformed decisions, adversely affecting their productivity and economic stability.

To address these challenges, there is a pressing need for an advanced, user-friendly system that can provide precise crop recommendations based on detailed soil analysis. Such a system would empower farmers by bridging the gap between scientific soil health assessment and practical crop selection. The Crop Recommendation System aims to fulfill this need by leveraging Machine Learning and a robust technological infrastructure to deliver accurate, data-driven crop suggestions. By inputting soil parameters into the system, users can receive tailored crop recommendations that

consider the unique characteristics of their soil. This approach not only enhances the decision-making process but also promotes sustainable farming practices, ultimately leading to improved crop yields and better resource management. The system's ability to store and analyze historical data further adds value, allowing users to track trends and make more informed decisions over time.

1.3 Objectives

- **Enhance Agricultural Decision-Making:** Develop a system that assists farmers and agricultural professionals in making informed crop selection decisions based on detailed soil analysis.
- **Leverage Machine Learning:** Utilize a Random Forest Classifier to analyze soil parameters and provide accurate crop recommendations tailored to specific soil conditions.
- **User-Friendly Interface:** Create an intuitive and responsive frontend using HTML and CSS, enabling users to input soil details easily and interact with the system seamlessly.
- **Robust Backend Infrastructure:** Implement a scalable and reliable backend using Python Django to handle multiple user interactions and data processing tasks efficiently.
- **Comprehensive Data Management:** Employ an SQL database to store user information and crop recommendation history, facilitating easy retrieval and analysis of historical data.
- **Promote Sustainable Farming Practices:** Provide data-driven crop suggestions that optimize resource use and enhance crop yield, contributing to sustainable agricultural practices.
- **Personalized Recommendations:** Maintain a detailed record of user interactions to offer personalized crop recommendations and allow users to track trends over time.
- **Future Enhancements:** Plan for future improvements such as integrating additional soil parameters, expanding the training dataset, incorporating advanced ML algorithms, and developing a mobile application version for broader accessibility.

CHAPTER 2

LITERATURE SURVEY

2.1 Tools and Technologies

2.1.1 Python Django

Python Django is a high-level web framework that facilitates the rapid development of robust and scalable web applications. It is the backbone of the Crop Recommendation System's backend infrastructure. Django's primary features include its built-in administrative interface, which simplifies content management and user administration, and its adherence to the "don't repeat yourself" (DRY) principle, which promotes code reusability and efficiency. Additionally, Django provides a robust security framework, protecting the application from common web vulnerabilities. Its Object-Relational Mapping (ORM) system streamlines database interactions, allowing developers to work with database records as Python objects without writing raw SQL queries. This feature enhances the system's scalability and maintainability, making Django an ideal choice for building the backend of the Crop Recommendation System.

2.1.2 HTML and CSS

HTML (HyperText Markup Language) and CSS (Cascading Style Sheets) are fundamental technologies used for creating and styling the frontend of web applications. HTML provides the structural framework of the web pages in the Crop Recommendation System, defining elements such as forms, buttons, and input fields. CSS is used to enhance the visual appearance of these elements, ensuring a responsive and user-friendly interface. Features of CSS include layout design capabilities, such as Flexbox and Grid, which enable the creation of dynamic and adaptable layouts. CSS also supports various styling options, including color schemes, typography, and animations, which contribute to a visually appealing and engaging user experience. Together, HTML and CSS ensure that the frontend of the Crop Recommendation System is both functional and aesthetically pleasing.

2.1.3 SQL Database

An SQL (Structured Query Language) database is utilized for data management in the Crop Recommendation System. SQL databases are renowned for their ability to handle complex queries and manage large volumes of structured data efficiently. Key features include robust data integrity, support

for transactions, and advanced querying capabilities. In this project, the SQL database stores critical information such as user details, soil parameters, and crop recommendation history. The database schema is designed to accommodate the specific needs of the system, ensuring efficient data retrieval and management. SQL's ability to handle relational data and enforce data constraints makes it a suitable choice for maintaining the system's data accuracy and consistency.

2.1.4 Random Forest Classifier

The Random Forest Classifier is a Machine Learning algorithm employed for predicting crop recommendations based on soil parameters. This ensemble learning method constructs multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees. The Random Forest Classifier is valued for its high accuracy, robustness against overfitting, and ability to handle large datasets with numerous features. It excels in managing complex interactions between variables, which is crucial for accurately predicting the suitability of different crops based on various soil attributes. By analyzing input parameters such as Nitrogen, Potassium, Phosphorous levels, Rainfall, Humidity, pH, and Temperature, the Random Forest Classifier provides precise and reliable crop recommendations.

Each of these software components plays a vital role in the overall functionality of the Crop Recommendation System, contributing to its effectiveness and efficiency. Python Django forms the backbone of the backend infrastructure, HTML and CSS shape the user experience, SQL handles data management, and the Random Forest Classifier drives the predictive capabilities of the system. Together, they create a comprehensive solution for optimizing crop selection and enhancing agricultural productivity.

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 Functional Requirements

3.1.1 User Authentication Module

The User Authentication Module is a critical component of the Crop Recommendation System, designed to ensure that user access is secure and well-managed. This module handles the processes for user registration, login, and profile management. When users first interact with the system, they can register by providing a username, password, and optionally an email address. This registration data is securely stored in an SQL database, where passwords are encrypted to protect against unauthorized access.

The login functionality of this module verifies user credentials against the database records. Successful authentication grants users access to their personalized dashboard, which includes access to crop recommendations, user profile settings, and historical data. If the provided credentials are incorrect, the system generates an error message prompting the user to re-enter their information. Additionally, the module supports secure logout functionality, ensuring that users can safely end their sessions. Users can also update their profiles to reflect changes in their farming practices and preferences, allowing the system to provide more accurate and personalized recommendations based on updated data.

3.1.2 Crop Recommendation User Input Page

The Crop Recommendation User Input Page is designed to collect comprehensive soil and environmental data from users, which is essential for generating accurate crop recommendations. This module features a form-based interface where users input critical parameters including Nitrogen, Potassium, Phosphorous levels, Rainfall, Humidity, pH, and Temperature. The form validation process ensures that all entries are accurate and complete before the data is submitted.

Data entered into this form is then transmitted to the backend of the system where it is processed and formatted for analysis. The data collection is facilitated through forms.py, which includes various input fields and validation rules to ensure the integrity of the data. The secure transmission of this data to the backend is crucial for maintaining the confidentiality of user information. Once the

data is collected and validated, it is prepared for the Crop Recommendation Engine, which will use this information to generate tailored crop recommendations.

3.1.3 Crop Recommendation Engine

The Crop Recommendation Engine is the core component responsible for analyzing user input and providing crop recommendations. This engine utilizes the Random Forest Classifier, a powerful machine learning algorithm known for its accuracy and ability to handle complex datasets. The engine processes the soil parameters provided by users, comparing them against a pre-trained model to predict the most suitable crops for the given conditions.

The Random Forest Classifier works by constructing multiple decision trees during training and outputting the mode of the classes (classification) or mean prediction (regression) of the individual trees. For crop recommendations, the engine evaluates various soil attributes and uses the trained model to generate a list of recommended crops. Each recommendation includes detailed information such as expected yield, optimal growth conditions, and market value. This functionality empowers users to make informed decisions about which crops are best suited to their specific soil conditions, potentially improving their agricultural outcomes.

3.1.4 Crop Recommendation Results/History Page

The Crop Recommendation Results/History Page provides users with a view of their crop recommendations and maintains a record of past recommendations. This module is designed to present the results of the recommendations generated by the Crop Recommendation Engine in a clear and user-friendly format. Users can review the suggested crops and assess how well they align with their soil conditions.

In addition to displaying current recommendations, this module offers a historical overview of previous recommendations. Users can track and analyze past recommendations, compare different suggestions over time, and observe any trends or patterns. This historical perspective aids users in understanding how their choices have evolved and how different crops have performed under varying conditions. Such insights are valuable for making data-driven decisions about future crop selections.

3.1.5 Admin Module

The Admin Module is designed to provide administrative oversight and management capabilities for the Crop Recommendation System. Administrators use this module to view and manage user data, including profiles and historical recommendations. This functionality is critical for maintaining the integrity of the system and ensuring that user data is accurately reflected.

The Admin Module allows administrators to monitor user activities, handle queries, and perform routine maintenance tasks. It includes tools for viewing user records, updating information, and managing system operations. Additionally, the module supports monitoring user interactions and system performance, which helps in identifying and addressing any issues that may arise. Ensuring that the system operates smoothly and efficiently is a key responsibility of the administrative functions provided by this module.

3.2 Non-Functional Requirements

3.2.1 Software Requirements

- **Operating System:** The Crop Recommendation System is designed to be compatible with multiple operating systems including Windows, macOS, and Linux. This ensures that users from different environments can access and utilize the system effectively.
- **Database:** The system uses a SQL-based relational database management system, such as MySQL or PostgreSQL. This choice ensures reliable data storage and efficient management of user details and crop recommendation history. SQL databases are known for their robustness and support for complex queries and transactions.
- **Programming Languages:** Python is used as the primary programming language, leveraging Django as the web framework for backend development. Django provides a high-level, secure, and scalable environment for building web applications, while Python's versatility and ease of use facilitate efficient coding and integration.
- **User Interface:** The frontend of the system is developed using HTML, CSS, and JavaScript. These technologies create a responsive and interactive user interface, ensuring that users can easily navigate the system and access its features. HTML structures the content, CSS styles the elements, and JavaScript enhances user interactions and dynamic content.
- **Performance:** The system is engineered to handle large volumes of data and deliver quick responses to user queries. Performance optimization techniques are applied to ensure that the system operates efficiently, even under high load conditions.

- **Security:** The system incorporates comprehensive security measures to protect user data, including encryption of sensitive information and secure data transmission protocols. These measures are essential for maintaining user privacy and safeguarding against potential threats.
- **Frontend Frameworks:** Bootstrap is used to ensure a responsive and visually appealing design, while jQuery enhances user interactions by providing advanced functionalities and dynamic content updates. These frameworks contribute to a modern and engaging user experience.
- **Development Environment:** Visual Studio Code is the chosen integrated development environment (IDE) for coding and debugging. It offers a range of features, including syntax highlighting, debugging tools, and extensions, which facilitate efficient development and maintenance of the system.

3.2.2 Hardware Requirements

- **Processor:** The system requires a minimum of a 2GHz processor to handle processing tasks efficiently. Adequate processing power is essential for running the system smoothly and managing data processing operations.
- **Storage:** A minimum of 30GB of storage space is needed to accommodate system files, software installations, and data storage. Sufficient storage ensures that the system can handle large datasets and maintain performance over time.
- **Network:** A stable internet connection is necessary for remote access and data transfer. Both wired and wireless connections are supported, ensuring flexibility in accessing the system from various locations.
- **Display:** The system should be compatible with monitors that have a resolution of at least 1024x768. This ensures that the user interface is displayed correctly and that users can interact with the system effectively.
- **Peripherals:** Standard input devices such as a keyboard and mouse are required for interacting with the system. Additional peripherals, like a scanner, may be needed depending on specific user requirements.

3.3 Database Requirements

- **Database Engine:** The Crop Recommendation System utilizes a relational database management system (RDBMS) such as PostgreSQL or MySQL. These databases are chosen for their ability to manage complex data relationships and support advanced querying capabilities.
- **ACID Compliance:** The database adheres to ACID (Atomicity, Consistency, Isolation, Durability) properties, which are crucial for maintaining data integrity and reliability. ACID compliance ensures that transactions are processed accurately and that data remains consistent and protected against corruption.
- **Performance:** The database schema and queries are optimized to ensure efficient data retrieval and processing. Performance tuning is applied to minimize latency and improve the speed of data operations, providing a smooth user experience.

- **Scalability:** The database is designed to scale both horizontally (by adding more servers) and vertically (by increasing server capacity) to accommodate future data growth and increased user load. This scalability ensures that the system can handle expanding data and user demands.
- **Backup and Recovery:** Regular automated backups are implemented to prevent data loss and facilitate quick recovery in case of hardware failures or errors. Backup strategies are designed to ensure that data can be restored efficiently and with minimal disruption.
- **Security:** Database-level security measures are applied to protect sensitive data from unauthorized access and breaches. This includes encryption of sensitive data fields, strict access controls, and regular security audits to safeguard data integrity and confidentiality.

CHAPTER 4

SYSTEM DESIGN

4.1 Architecture Overview

The architecture of the Crop Recommendation System is meticulously designed to support scalability, security, and efficient management of crop recommendations based on user-provided soil data. This system leverages a multi-tier architecture, structured as follows:

- **Presentation Layer:** This layer is responsible for delivering a user-friendly interface and handling user interactions. It includes the design and layout of web pages using HTML for structure, CSS for styling, and JavaScript for dynamic content. This layer ensures that users can easily input their soil parameters, view crop recommendations, and manage their profiles. The user interface is built to be intuitive and responsive, accommodating different devices and screen sizes.
- **Security Layer:** The security layer is crucial for protecting user data and ensuring secure access to the system. It utilizes Django's built-in authentication framework to handle user registration, login, and profile management. This layer employs secure hashing for password storage and manages user sessions to prevent unauthorized access. It ensures that sensitive user information is encrypted and safeguarded against potential threats.
- **Data Layer:** The data layer manages the storage and retrieval of data, including user information, soil parameters, and crop recommendations. It utilizes a relational database management system (RDBMS) such as MySQL or PostgreSQL to handle data persistence. This layer ensures data integrity and efficient querying by maintaining a well-structured database schema and implementing optimization techniques for fast data access.

4.2 Components

1. **Frontend**
    - **HTML/CSS/JavaScript:** These core technologies are used to build the user interface of the Crop Recommendation System. HTML provides the foundational structure of web pages, CSS is used to create visually appealing layouts and responsive designs, and JavaScript adds interactivity and dynamic functionality. Together, they create a seamless and engaging experience for users, allowing them to easily navigate the system, input data, and view results.

- **Django Templates:** Django's templating engine dynamically generates HTML pages based on data from the backend. Templates are used to render various components of the system, such as user input forms, crop recommendation results, and user profile pages. This approach allows for consistent design and efficient rendering of content, as templates can be reused and customized based on user data and application logic.

2. **Backend**
   - **Django Framework:** The Django framework serves as the backbone of the Crop Recommendation System, providing a robust environment for developing the application's backend logic. It handles request processing, data validation, and interaction with the database. Django's built-in features, such as the ORM (Object-Relational Mapping) and middleware, facilitate the development of scalable and maintainable applications.
   - **Django Models:** Django models define the data structure for the application, representing entities such as users, soil parameters, and crop recommendations. Each model corresponds to a database table, with fields representing the attributes of the entity. Models provide an abstraction layer for database operations, allowing developers to interact with the database using Python code rather than SQL queries.
   - **Django Views:** Views are responsible for processing incoming requests, interacting with models, and rendering the appropriate templates. They handle the core application logic, such as retrieving soil data, running the crop recommendation algorithm, and displaying results. Views act as intermediaries between the user interface and the data layer, ensuring that user inputs are processed correctly and results are presented in a user-friendly manner.
   - **Django Forms:** Django forms manage user input and validation. They ensure that data submitted through web forms is accurate and complete before it is processed by the backend. Forms handle tasks such as input validation, error messaging, and data cleaning, ensuring that the data entered by users meets the system's requirements.
3. **Database**
   - **MySQL/PostgreSQL:** These relational database management systems (RDBMS) are used to store and manage user data, soil parameters, and crop recommendations. The choice of RDBMS ensures reliable data storage and retrieval, with support for complex queries and transactions. The database schema is designed to accommodate the various data types and relationships required by the system, and indexing is employed to optimize query performance.
4. **Authentication System**

- **Django Authentication:** Django's authentication system manages user login, logout, and registration processes. It provides features for secure password handling, user session management, and role-based access control. The system ensures that only authenticated users can access their personal data and recommendations, and it handles tasks such as password reset and account recovery.

5. **Analytics and Reporting Layer**
   - **Dashboards:** Dashboards provide real-time insights into crop recommendations and user interactions. They display key metrics such as current recommendations, historical data, and user activity. Dashboards are designed to be interactive and visually appealing, using charts and graphs to help users understand trends and make data-driven decisions.
   - **Reports:** The reporting functionality generates detailed reports on crop recommendations, user behavior, and system performance. These reports are used for administrative purposes and to evaluate the effectiveness of the recommendations provided by the system. Reports can be customized to include various metrics and data points, helping administrators and users make informed decisions.

4.3 Crop Recommendation Workflow

1. **User Authentication**
   - Users initiate their session by logging in through the Django authentication system. Upon successful authentication, users are redirected to their personalized dashboard. This dashboard provides access to the crop recommendation features, user profile management, and historical data. The authentication system ensures that user sessions are secure and that access to sensitive data is restricted to authorized individuals.
2. **User Input Collection**
   - Users input their soil parameters, including Nitrogen, Potassium, Phosphorous levels, Rainfall, Humidity, pH, and Temperature, using a user-friendly form interface. Django forms validate and clean this data before sending it to the backend for processing. The data collection process is designed to be straightforward and efficient, ensuring that users can easily provide accurate information.
3. **Crop Recommendation Generation**
   - The Crop Recommendation Engine processes the collected soil data using the Random Forest Classifier algorithm. This algorithm evaluates the soil parameters against a trained dataset to predict the most suitable crops. The system generates a list of recommended crops, including details such as expected yield, optimal growth conditions, and potential

market value. Users can view these recommendations on the Crop Recommendation Results Page.

4. **History Tracking**
   - The system maintains a history of past crop recommendations, which users can access through the Crop Recommendation Results/History Page. This functionality allows users to review previous recommendations, analyze trends, and compare different recommendations over time. Historical data helps users understand the effectiveness of past recommendations and make more informed decisions.
5. **Admin Management**
   - Administrators access the Admin Module to manage user records, monitor user activity, and perform system maintenance tasks. This module provides tools for viewing and updating user data, handling user queries, and ensuring data integrity. Administrators can also manage the system's overall health and efficiency, ensuring that the Crop Recommendation System operates smoothly and effectively.

CHAPTER 5

IMPLEMENTATION

5.1 Frontend

#Recommender/templates/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Welcome</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-size: cover;
            margin: 0;
            padding: 0;
            color: #fff;
            display: flex;
            justify-content: center;
            align-items: center;
            flex-direction: column;
            height: 100vh;
        }
        .container {
            text-align: center;
            background: rgba(0, 0, 0, 0.7);
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0 0 15px rgba(0, 255, 255, 0.5);
            backdrop-filter: blur(10px);
            width: 80%;
            max-width: 600px;
        }
        h1 {
            color: #0ff;
```

```
            margin-bottom: 20px;
        }
        p {
            font-size: 18px;
            margin: 10px 0;
        }
        button {
            background: #0ff;
            color: #000;
            border: none;
            padding: 10px 20px;
            font-size: 16px;
            border-radius: 5px;
            cursor: pointer;
            margin-top: 20px;
        }
        button:hover {
            background: #00bcd4;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Welcome to the Agricultural Hub</h1>
        <p>"The farmer has to be an optimist or he wouldn’t still be a farmer." – Will Rogers</p>
        <p>"Agriculture is the most healthful, most useful, and most noble employment of man." – George Washington</p>
        <p>"Farming is a profession of hope." – Brian Brett</p>
        {% if user.is_authenticated %}
            <button><a href="{% url 'about_us' %}">about us</a></button>
            <button ><a href="{% url 'recommend_crop' %}">Crop Recommend</a></button>
            <button ><a href="{% url 'history' %}">History</a></button>
            <button ><a href="{% url 'logout' %}">logout</a></button>
        {% else %}
            <button><a href="{% url 'about_us' %}">about us</a></button>
            <button ><a href="{% url 'signup' %}">Signup & Login</a></button>
            <button ><a href="{% url 'login' %}">Login</a></button>
        {% endif %}
```

```
    </div>
</body>
</html>
```

#Recommender/templates/login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Log In</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-size: cover;
            margin: 0;
            padding: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
        }
        .container {
            background: rgba(255, 255, 255, 0.1);
            border-radius: 10px;
            box-shadow: 0 0 15px rgba(0, 255, 255, 0.5);
            backdrop-filter: blur(10px);
            padding: 20px;
            width: 80%;
            max-width: 400px;
            text-align: center;
        }
        h2 {
            color: #0ff;
            margin-bottom: 20px;
        }
        form {
```

```
            display: flex;
            flex-direction: column;
            align-items: center;
        }
        form input, form button {
            width: 100%;
            padding: 10px;
            margin: 10px 0;
            border-radius: 5px;
            border: none;
        }
        form input {
            background: rgba(255, 255, 255, 0.7);
            font-size: 16px;
        }
        form button {
            background: #0ff;
            color: #000;
            font-size: 16px;
            cursor: pointer;
        }
        form button:hover {
            background: #00bcd4;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Log In</h2>
        <form method="post">
            {% csrf_token %}
            {{ form.as_p }}
            <button type="submit">Log In</button>
        </form>
    </div>
</body>
</html>
```

#Recommender/templates/signup.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sign Up</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-size: cover;
            margin: 0;
            padding: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
        }
        .container {
            background: rgba(255, 255, 255, 0.1);
            border-radius: 10px;
            box-shadow: 0 0 15px rgba(0, 255, 255, 0.5);
            backdrop-filter: blur(10px);
            padding: 20px;
            width: 80%;
            max-width: 400px;
            text-align: center;
        }
        h2 {
            color: #0ff;
            margin-bottom: 20px;
        }
        form {
            display: flex;
            flex-direction: column;
            align-items: center;
        }
        form input, form button {
```

```
            width: 100%;
            padding: 10px;
            margin: 10px 0;
            border-radius: 5px;
            border: none;
        }
        form input {
            background: rgba(255, 255, 255, 0.7);
            font-size: 16px;
        }
        form button {
            background: #0ff;
            color: #000;
            font-size: 16px;
            cursor: pointer;
        }
        form button:hover {
            background: #00bcd4;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Sign Up</h2>
        <form method="post">
            {% csrf_token %}
            {{ form.as_p }}
            <button type="submit">Sign Up</button>
        </form>
    </div>
</body>
</html>
```

#Recommender/templates/recommend_crop.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Crop Recommendation</title>
<style>
    body {
        font-family: Arial, sans-serif;
        background-size: cover;
        margin: 0;
        padding: 0;
        color: #ffffff;
    }
    header {
        background: rgba(0, 0, 0, 0.7);
        padding: 10px 0;
        text-align: center;
        position: sticky;
        top: 0;
        z-index: 1000;
    }
    nav a {
        color: #fff;
        text-decoration: none;
        padding: 10px 20px;
        display: inline-block;
    }
    nav a:hover {
        background: rgba(255, 255, 255, 0.2);
    }
    .container {
        display: flex;
        justify-content: center;
        align-items: center;
        height: 30cm;
        padding: 20px;
    }
    .form-container {
        background: rgba(0, 0, 0, 0.5);
        border-radius: 10px;
        padding: 20px;
```

```
            box-shadow: 0 0 15px rgba(0, 255, 255, 0.5);
            backdrop-filter: blur(10px);
            width: 100%;
            max-width: 500px;
        }
        h2 {
            color: #0ff;
            text-align: center;
        }
        form {
            display: flex;
            flex-direction: column;
        }
        form input, form button {
            margin: 10px 0;
            padding: 10px;
            border: none;
            border-radius: 5px;
            font-size: 16px;
        }
        form input {
            background: rgba(255, 255, 255, 0.2);
            color: #fff;
        }
        form button {
            background: #0ff;
            color: #000;
            cursor: pointer;
        }
        form button:hover {
            background: #00bcd4;
        }
    </style>
</head>
<body>
    <header>
        <nav>
            <a href="/">Home</a>
```

```
            <a href="{% url 'about_us' %}">About us</a>
            <a href="#">Crop Recommend</a>
            <a href="{% url 'history' %}">History</a>
            <a href="{% url 'logout' %}">Logout</a>
        </nav>
    </header>
    <div class="container">
        <div class="form-container">
            <h2>Crop Recommendation Form</h2>
            <form method="post">
                {% csrf_token %}
                {{ form.as_p }}
                <button type="submit">Submit</button>
            </form>
        </div>
    </div>
</body>
</html>
```

#Recommender/templates/history.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Recommendation Result</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-size: cover;
            margin: 0;
            padding: 0;
            color: #fff;
        }
        header {
            background: rgba(0, 0, 0, 0.7);
            padding: 10px 0;
```

```
        text-align: center;
        position: sticky;
        top: 0;
        z-index: 1000;
    }
    nav a {
        color: #fff;
        text-decoration: none;
        padding: 10px 20px;
        display: inline-block;
    }
    nav a:hover {
        background: rgba(255, 255, 255, 0.2);
    }
    .container {
        display: flex;
        justify-content: center;
        align-items: center;
        height: 100vh;
        padding: 20px;
    }
    .result-container {
        background: rgba(0, 0, 0, 0.7);
        border-radius: 10px;
        padding: 20px;
        box-shadow: 0 0 15px rgba(0, 255, 255, 0.5);
        backdrop-filter: blur(10px);
        text-align: center;
        max-width: 600px;
        width: 100%;
    }
    h2 {
        color: #0ff;
        margin-bottom: 20px;
    }
    p {
        font-size: 24px;
        font-weight: bold;
```

```
            color: #0ff;
            margin: 0;
        }
    </style>
</head>
<body>
    <header>
        <nav>
            <a href="/">Home</a>
            <a href="{% url 'about_us' %}">About Us</a>
            <a href="{% url 'recommend_crop' %}">Crop Recommend</a>
            <a href="#">History</a>
            <a href="{% url 'logout' %}">Logout</a>
        </nav>
    </header>
    <div class="container">
        <div class="result-container">
            <h2>Recommended Crop</h2>
            <p><ul>
                {% for recommendation in recommendations %}
                <p>  </p>
                <li>{{ recommendation.user }} <p> : </p> {{ recommendation.timestamp }} <p> : </p> {{
recommendation.recommended_crop }}</li>
                <p>  </p>
                {% endfor %}
            </ul></p>
        </div>
    </div>
</body>
</html>
```

#Recommender/templates/about_us.html

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

```
<title>About Us</title>
<style>
    body {
        font-family: Arial, sans-serif;
        background-size: cover;
        margin: 0;
        padding: 0;
        color: #fff;
    }
    header {
        background: rgba(0, 0, 0, 0.7);
        padding: 10px 0;
        text-align: center;
        position: sticky;
        top: 0;
        z-index: 1000;
    }
    nav a {
        color: #fff;
        text-decoration: none;
        padding: 10px 20px;
        display: inline-block;
    }
    nav a:hover {
        background: rgba(255, 255, 255, 0.2);
    }
    .container {
        padding: 20px;
        display: flex;
        flex-direction: column;
        align-items: center;
        text-align: center;
    }
    .team-member {
        background: rgba(0, 0, 0, 0.7);
        padding: 20px;
        border-radius: 10px;
        margin: 10px 0;
```

```
            width: 80%;
            max-width: 600px;
            box-shadow: 0 0 15px rgba(0, 255, 255, 0.5);
            backdrop-filter: blur(10px);
        }
        .team-member h3 {
            color: #0ff;
            margin-bottom: 10px;
        }
        .team-member p {
            font-size: 16px;
            margin: 10px 0;
        }
        .social-icons {
            margin-top: 10px;
        }
        .social-icons a {
            color: #fff;
            margin: 0 10px;
            text-decoration: none;
            font-size: 18px;
        }
        .social-icons a:hover {
            color: #00bcd4;
        }
        .social-icons img {
            width: 24px;
            height: 24px;
        }
    </style>
</head>
<body>
    <header>
        <nav>
            <a href="/">Home</a>
            <a href="#">About Us</a>
            {% if user.is_authenticated %}
                <a href="{% url 'recommend_crop'%}">Crop Recommend</a>
```

```
            <a href="{% url 'history' %}">History</a>
            <a href="{% url 'logout' %}">Logout</a>
        {% else %}
            <a href="{% url 'login' %}">Login</a>
            <a href="{% url 'signup' %}">Signup & Login</a>
        {% endif %}

    </nav>
</header>
<div class="container">
    <h1>About Us</h1>

    <div class="team-member">
        <h3>Darshan Rajanna</h3>
        <p>Lead Developer</p>
        <div class="social-icons">
            <a href="#">darshidr59328@hmail.com</a>
            <a href="#">SaIT, CSE, A section</a>
        </div>
    </div>

    <div class="team-member">
        <h3>Bhavan B</h3>
        <p>Project Manager</p>
        <div class="social-icons">
            <a href="#">bhavanraj68@gmail.com</a>
            <a href="#">SaIT, CSE, A section</a>
        </div>
    </div>

    <div class="team-member">
        <h3>Dileep Kumar K</h3>
        <p>Data Scientist</p>
        <div class="social-icons">
            <a href="#">dileepkumar3603@gmail.com</a>
            <a href="#">SaIT, CSE, A section</a>
        </div>
    </div>
```

```
        <div class="team-member">
            <h3>Bhuvan G R</h3>
            <p>UI/UX Designer</p>
            <div class="social-icons">
                <a href="#">bhuvangujjarbhuvan@gmail.com</a>
                <a href="#">SaIT, CSE, A section</a>
            </div>
        </div>
    </div>
</body>
</html>
```

5.2 Backend

#Recommender/views.py

```
from django.shortcuts import render, redirect
from django.contrib.auth import login, authenticate
from django.contrib.auth.decorators import login_required
from .forms import SignUpForm, CropRecommendationForm
from .models import CropHistory

# Load the model
with open('C:/Users/admin/OneDrive/Desktop/New/myenv/CropRecomendation/Recomender/RandomForest.pkl',
'rb') as file:
    model = pickle.load(file)

def index(request):
    return render(request, 'Recomender/index.html' )

def about(request):
    return render(request, 'Recomender/about_us.html' )

def signup(request):
    if request.method == 'POST':
        form = SignUpForm(request.POST)
        if form.is_valid():
            form.save()
```

```
            username = form.cleaned_data.get('username')
            raw_password = form.cleaned_data.get('password1')
            user = authenticate(username=username, password=raw_password)
            login(request, user)
            return redirect('index')
    else:
        form = SignUpForm()
    return render(request, 'Recomender/signup.html', {'form': form})

@login_required
def recommend_crop(request):
    if request.method == 'POST':
        form = CropRecommendationForm(request.POST)
        if form.is_valid():
            instance = form.save(commit=False)
            instance.user = request.user
            user_data = [instance.N, instance.P, instance.K, instance.temperature, instance.humidity, instance.ph,
instance.rainfall]
            predicted_crop = model.predict([user_data])[0]
            instance.recommended_crop = predicted_crop
            instance.save()
            return redirect('history')
    else:
        form = CropRecommendationForm()
    return render(request, 'Recomender/recommend_crop.html', {'form': form})

@login_required
def history(request):
    recommendations = CropHistory.objects.filter(user=request.user)
    return render(request, 'Recomender/history.html', {'recommendations': recommendations})

@login_required
def user_logout(request):
    logout(request)
    return redirect('index')


from django.contrib.auth import authenticate, login
from django.shortcuts import render, redirect
```

```
from .forms import LoginForm

def login_view(request):
    if request.method == 'POST':
        form = LoginForm(request.POST)
        if form.is_valid():
            username = form.cleaned_data.get('username')
            password = form.cleaned_data.get('password')
            user = authenticate(request, username=username, password=password)
            if user is not None:
                login(request, user)
                return redirect('index')  # Redirect to a homepage or dashboard after login
    else:
        form = LoginForm()
    return render(request, 'Recomender/login.html', {'form': form})
```

#Recommender/models.py

```
from django.db import models
from django.contrib.auth.models import User
from django.utils import timezone

class CropHistory(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    N = models.FloatField()
    P = models.FloatField()
    K = models.FloatField()
    temperature = models.FloatField()
    humidity = models.FloatField()
    ph = models.FloatField()
    rainfall = models.FloatField()
    recommended_crop = models.CharField(max_length=100)
    timestamp = models.DateTimeField(default=timezone.now)

    def __str__(self):
        return f"{self.user.username} - {self.recommended_crop} on {self.timestamp}"
```

#Recommender/forms.py

```
from django import forms
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User
from .models import CropHistory
from django import forms

class LoginForm(forms.Form):
    username = forms.CharField(max_length=150, widget=forms.TextInput(attrs={'placeholder': 'Username'}))
    password = forms.CharField(widget=forms.PasswordInput(attrs={'placeholder': 'Password'}))

class SignUpForm(UserCreationForm):
    class Meta:
        model = User
        fields = ['username', 'password1', 'password2']

class CropRecommendationForm(forms.ModelForm):
    class Meta:
        model = CropHistory
        fields = ['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall']
```

#Recommender/urls.py

```
from django.urls import path
from . import views


urlpatterns = [
    path('', views.index, name='index'),
    path('signup/', views.signup, name='signup'),
    path('login/', views.login_view, name='login'),
    path('recommend_crop/', views.recommend_crop, name='recommend_crop'),
    path('about_us/', views.about, name='about_us'),
    path('history/', views.history, name='history'),
    path('logout/', views.user_logout, name='logout'),
]
```

#main /urls.py

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('Recomender.urls')),
]
```

CHAPTER 6

SNAPSHOTS

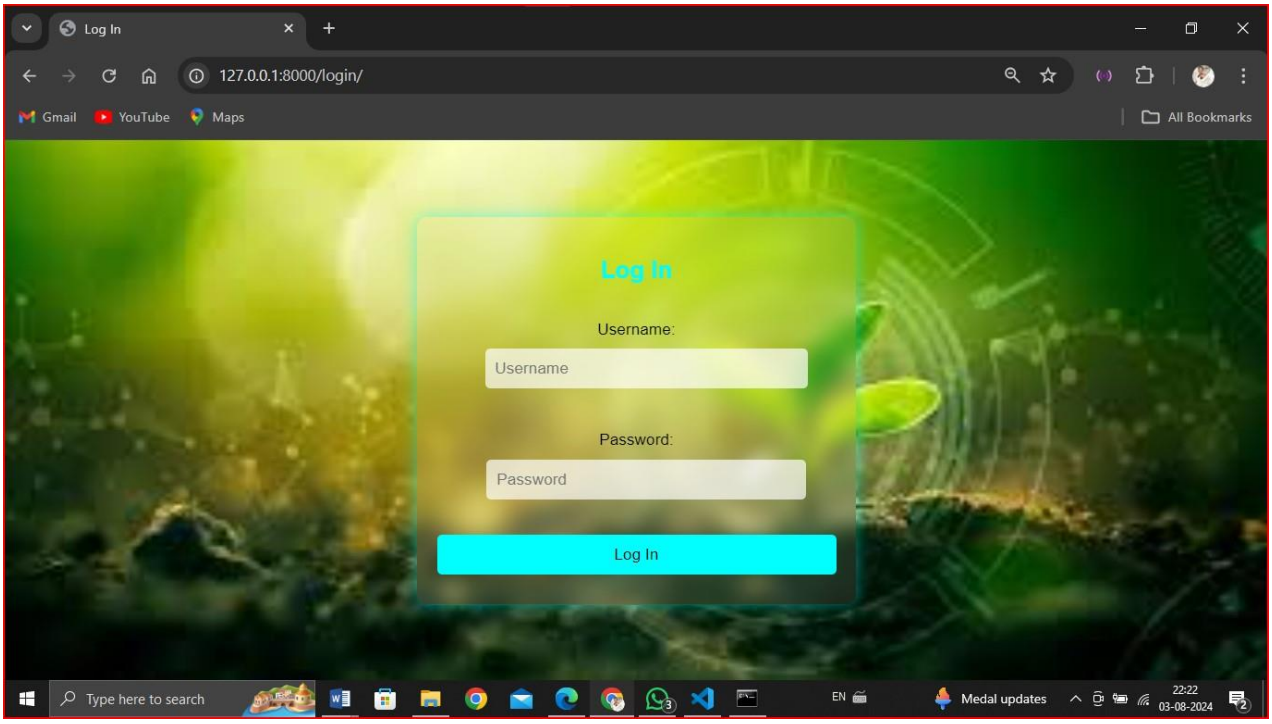

Fig 6.1: Login page

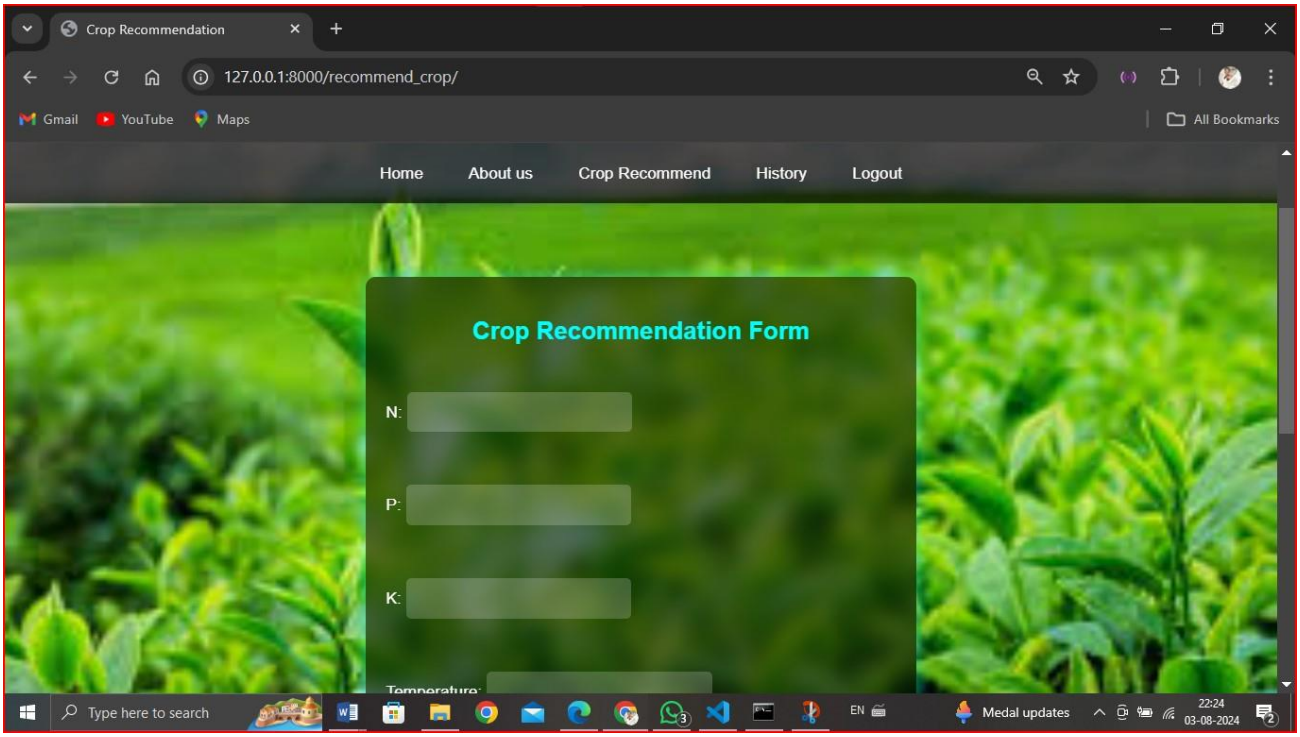

Fig 6.2: Crop Recommendation Page

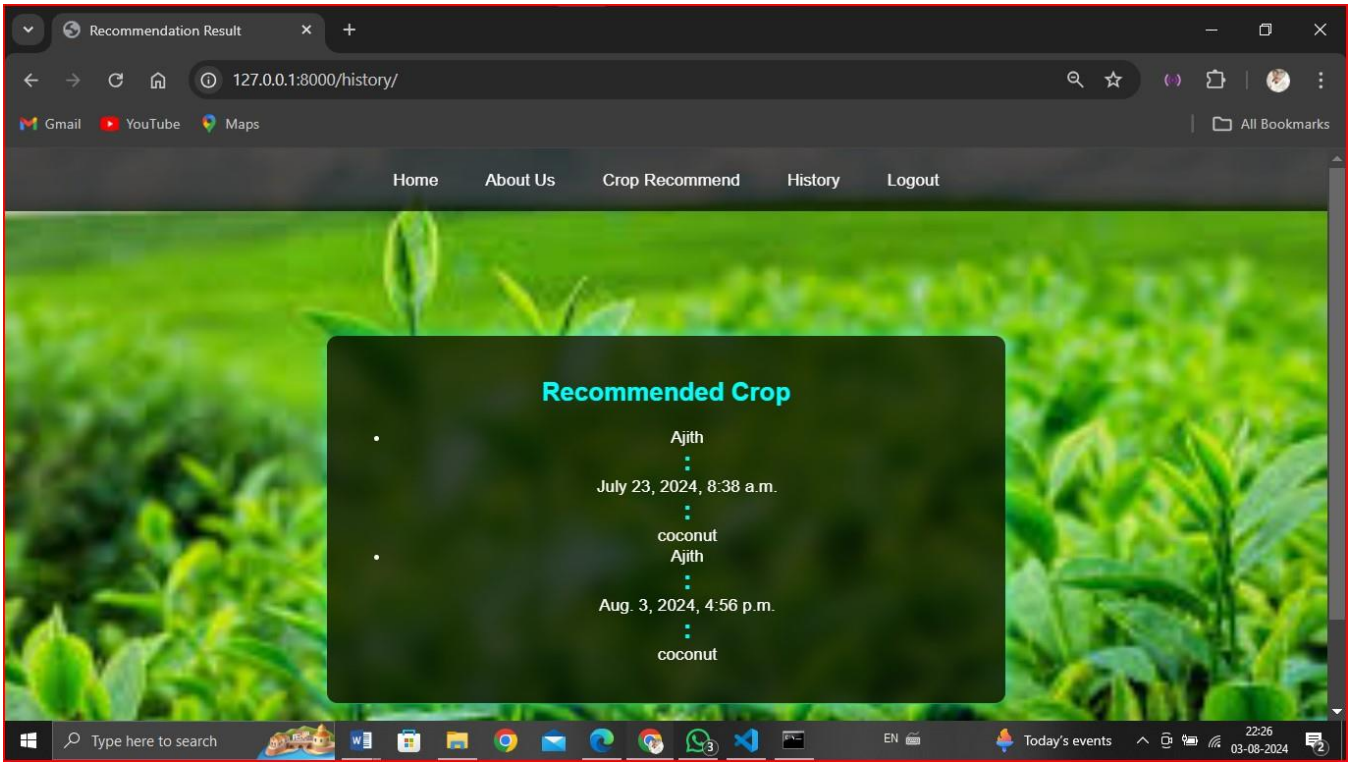

Fig 6.3: History Page

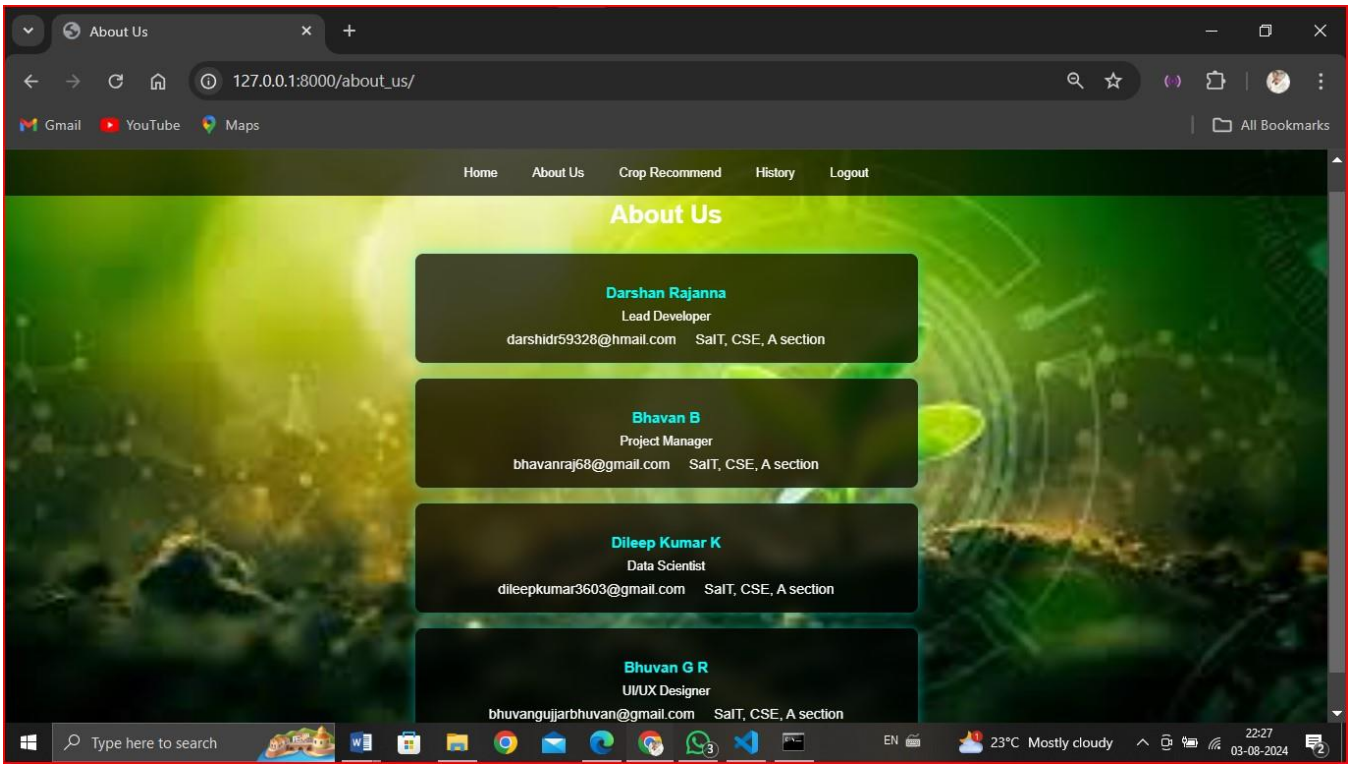

Fig 6.4: About_us Page

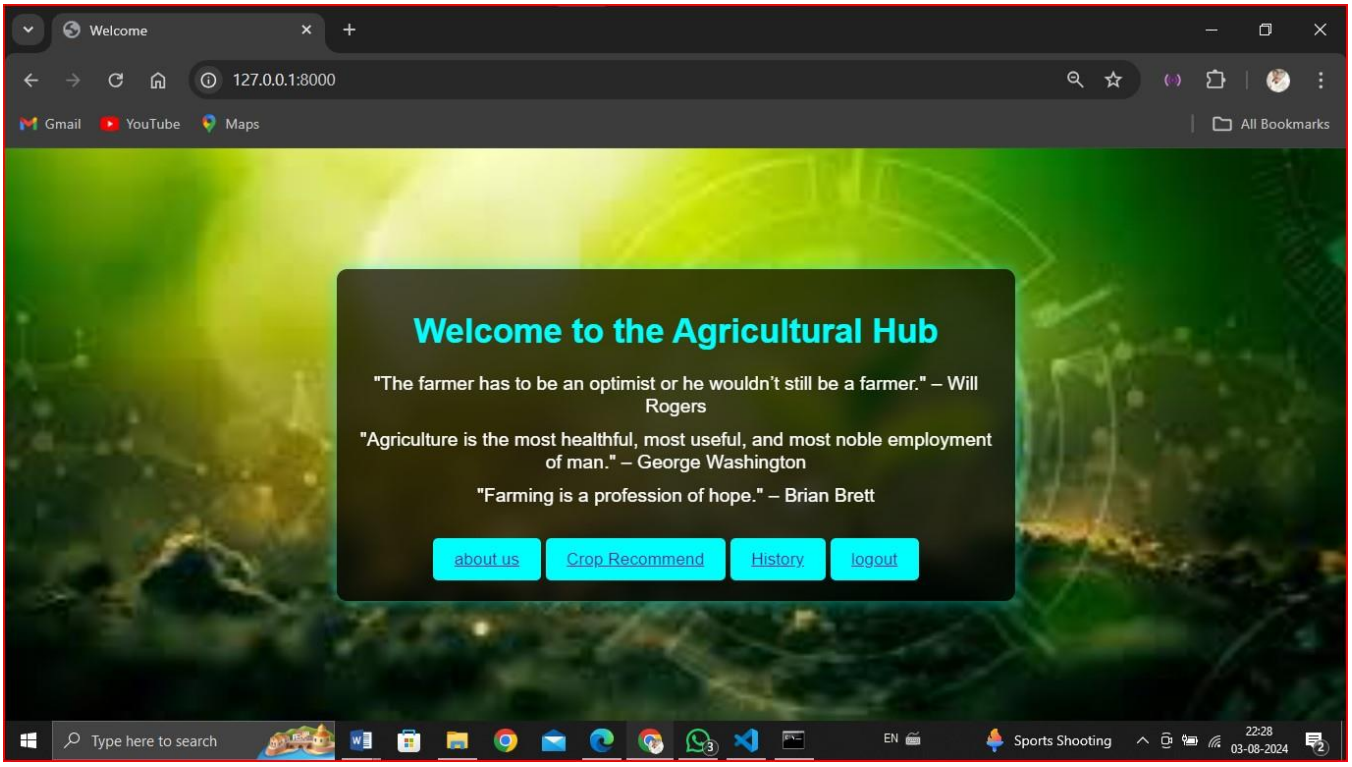

Fig 6.5: Home Page

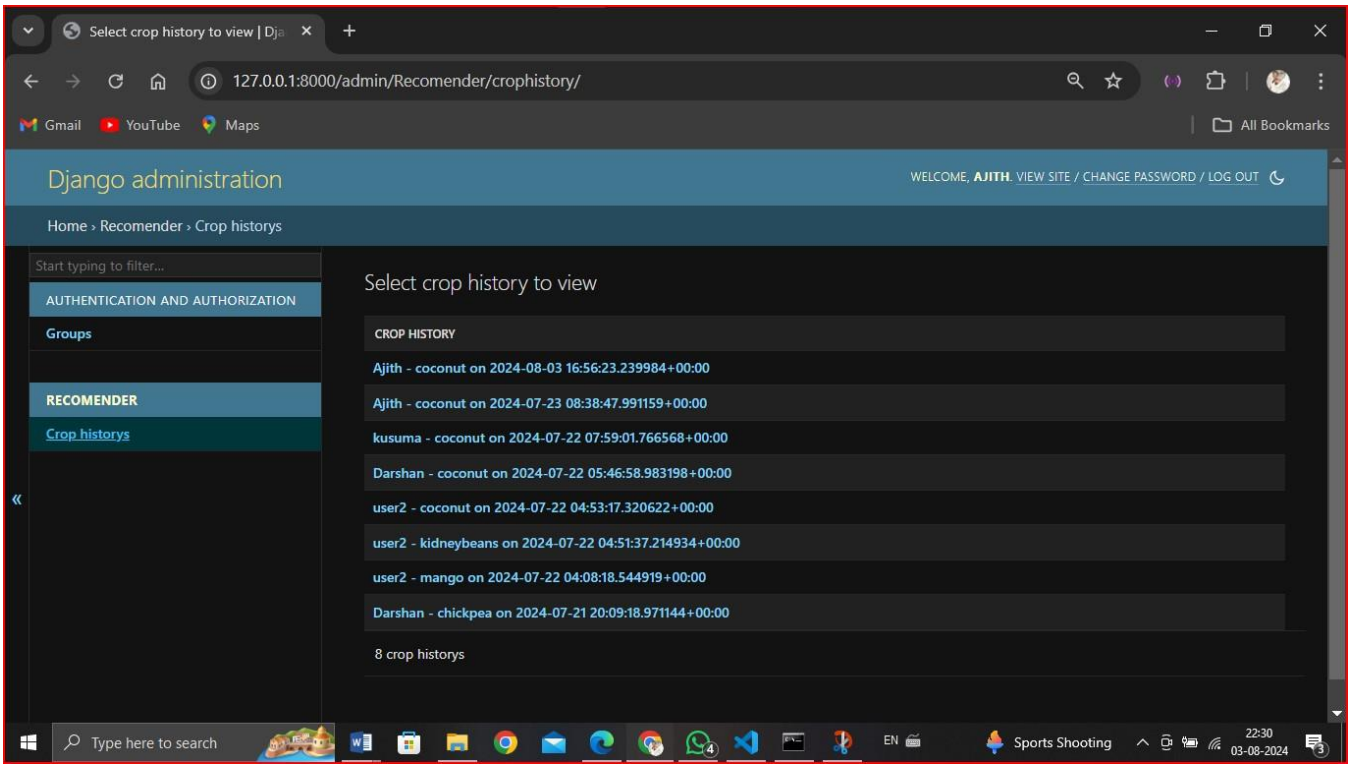

Fig 6.6: Admin History page

CONCLUSION & FUTURE WORK

The Crop Recommendation System, developed using Django, SQL, and machine learning techniques, represents a significant advancement in precision agriculture by providing an advanced solution for optimizing crop selection based on detailed soil and environmental conditions. The system's architecture integrates a user-friendly frontend, a secure backend, and a robust data management layer, ensuring an efficient and seamless experience. Django's powerful framework handles application logic and form management, while an RDBMS like MySQL or PostgreSQL maintains reliable data storage. The inclusion of the Random Forest Classifier algorithm enables the system to analyze complex soil data and generate accurate, tailored crop recommendations. Data preprocessing within Django ensures that user inputs are accurately processed and formatted, enhancing the model's effectiveness. Users benefit from the ability to input soil parameters, receive personalized crop recommendations, and review historical data, which supports informed decision-making and efficient crop management. Additionally, the admin module facilitates system oversight and data integrity, while real-time dashboards and detailed reporting provide valuable insights into crop recommendations and user activity. Overall, this system not only enhances the efficiency of crop selection but also empowers users with data-driven guidance, contributing to more productive and sustainable agricultural practices.

Looking ahead, the future work for the Crop Recommendation System encompasses several enhancements aimed at improving its accuracy, user experience, and adaptability to evolving agricultural practices. One significant area of development is the integration of additional data sources, such as satellite imagery and real-time weather data, to provide more comprehensive and dynamic recommendations. By incorporating advanced techniques such as deep learning and neural networks, the system could further refine its predictive capabilities, enabling it to handle more complex data sets and deliver even more precise crop suggestions. Enhancing the user interface with features like interactive maps, predictive analytics, and personalized recommendations based on historical performance could greatly improve user engagement and decision-making. Additionally, expanding the system to support a wider range of crops and regional farming practices would increase its applicability and value to a global audience. Implementing features such as automated alerts for optimal planting and harvesting times, as well as integration with farm management software, could provide users with actionable insights and streamline their agricultural operations. Furthermore, exploring the use of blockchain technology for secure data management and traceability could enhance the system's credibility and reliability. Overall, these future developments aim to elevate the Crop Recommendation System's functionality and impact, driving innovation in precision agriculture and contributing to more sustainable and efficient farming practices.

REFERENCES

[1] Aruvansh Nigam, Saksham Garg, Archit Agrawal ―Crop Yield Prediction using ML Algorithms ―, 2019 Fifth International Conference on Image Information Processing (ICIIP)

[2] LeoBrieman, ―Random Forests‖, 2019 IEEE Transactions on Instrumentation and Measurement

[3] Priya, P., Muthaiah, U., Balamurugan, M.‖ Predicting Yield of the Crop Using Machine Learning Algorithm‖,2020 International Journal of Engineering Sciences & Research Technology (IJESRT)

[4] Mishra, S., Mishra, D., Santra, G. H.,―Applications of machine learning techniques in agricultural crop production‖,2020 Indian Journal of Science and Technology

[5] Dr.Y Jeevan Kumar,‖Supervised Learning Approach for Crop Production‖, 2020 5th International Conference on Communication and Electronics Systems (ICCES)

[6] RameshMedar,Vijay S, Shweta, ―Crop Yield Prediction using Machine Learning Techniques‖, 2018 International Journal of Engineering Sciences & Research Technology (IJESRT)

[7] Ranjini B Guruprasad, Kumar Saurav, Sukanya Randhawa,‖Machine Learning Methodologies for Paddy Yield Estimation in India: A CASE STUDY‖, IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium

[8] https://en.wikipedia.org/wiki/Crop_yield

[9] https://en.wikipedia.org/wiki/Random_forest