SOIL QUALITY DETECTION AND CROP RECOMMENDATION SYSTEM

A PROJECT REPORT

Submitted by

DEVADHARSHINI.G

DHARANYA.T

MALINI.S

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

P. A. COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution)

Pollachi, Coimbatore Dt. 642002

APRIL 2025

P.A. COLLEGE OF ENGINEERING AND TECHNOLOGY

BONAFIDE CERTIFICATE

Certified that this project report **"SOIL QUALITY DETECTION AND CROP RECOMMENDATION SYSTEM"** is the bonafide work of **"DEVADHARSHINI.G(721721104015),DHARANYA.T(721721104018), MALINI.S(721721104057)"**who carried out the project work under my supervision.

----------	----------
SIGNATURE	**SIGNATURE**
Dr. D. CHITRA M.E., Ph.D.,	**Mrs. E. JANANANDHINI M.E.,**
Professor	Assistant Professor
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Department of Computer Science and Engineering	Department of Computer Science and Engineering
P. A. College of Engineering and Technology	P. A. College of Engineering and Technology
Pollachi-642 002.	Pollachi-642 002.

Submitted to the Viva-Voce Examination held on ……

……	…….
INTERNAL EXAMINER	**EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

First and foremost, we thank the **GOD ALMIGHTY** for blessing us with the mental strength that was needed to carry out the project work. We thank our Chairman **Dr. P. APPUKUTTY M.E, FIE, FIV.,** for his extensive support to successfully carry out this project.

We take privilege in expressing our sincere and heartfelt thanks to our beloved Principal **Dr. T. MANIGANDAN M.E., Ph.D.,** for providing us an opportunity to carry out this project work.

We express our sincere thanks to **Dr. D. CHITRA M.E., Ph.D.,** Professor and Head, Department of Computer Science and Engineering for her technical guidance and support.

We express our gratitude and sincere thanks to our Project Guide **Mrs. E. JANANANDHINI M.E.,** Assistant Professor, Department of Computer Science and Engineering, for her technical guidance, constructive criticism and many valuable suggestions provided throughout the project work.

We also express our gratitude and sincere thanks to our Project Coordinator **Mr. T. DINESH KUMAR M. Tech.,** Assistant Professor, Department of Computer Science and Engineering and all teaching and non- teaching staffs of CSE department, P. A. College of Engineering and Technology, Pollachi, for their encouragement and valuable suggestions.

We take this opportunity to express our indebted gratitude to our parents, friends, family and other members whose belongings and love has always been with us.

ABSTRACT

Soil quality assessment plays a crucial role in modern agriculture, influencing crop yield and sustainability. Traditional methods of soil analysis are often time-consuming and labour intensive, prompting the need for advanced computational techniques. Deep learning offers a powerful approach to evaluating soil properties by analysing key parameters such as texture, moisture and nutrient composition. By leveraging machine learning algorithms, it becomes possible to classify soil types and predict their suitability for various crops with high accuracy.

Integrating deep learning models with agricultural data enables accurate crop recommendations based on soil characteristics. These models process vast datasets, learning patterns that correlate soil properties with optimal crop selections. By analysing factors such as pH levels, organic matter content and macronutrient presence, the system suggests crops that are most likely to thrive in given soil conditions. This data-driven method not only improves efficiency but also promotes sustainable farming practices by reducing unnecessary resource usage and preventing soil degradation. The adoption of such intelligent systems in agriculture paves the way for more informed and scientifically backed cultivation strategies.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	**IV**
	LIST OF FIGURES	**VII**
	LIST OF ABBREVIATIONS	**VIII**
1	**INTRODUCTION**	1
	1.1 INTRODUCTION	1
	1.2 PROBLEM STATEMENT	4
	1.3 BACKGROUND OF SOIL AND ITS QUALITY	5
	1.4 CHALLENGES IN TRADITIONAL SOIL QUALITY ANALYZING METHOD	6
	1.5 IMPORTANCE OF ANALYZING SOIL QUALITY	7
	1.6 OBJECTIVES	8
2	**SURVEY ON SOIL QUALITY DETECTION AND CROP RECOMMENDATION SYSTEM**	9
3	**SOIL QUALITY IDENTIFICATION AND MONITORING APPROACH FOR SUGAR-CANE USING ML**	13
	3.1 TECHNOLOGIES USED	13
	3.2 PROGRAMMING LANGUAGES	14
	3.3 PROCESS FLOW	14
	3.4 DRAWBACKS	15

4	**SOIL QUALITY DETECTION AND CROP RECOMMENDATION SYSTEM**	17
	4.1 TECHNOLOGIES USED	17
	4.2 SYSTEM ARCHITECTURE	18
	4.3 FEATURES AND FUNCTIONALITIES	19
	4.3.1 SOIL CLASSIFICATION & Ph PREDICTION	19
	4.3.2 CROP RECOMMENDATION SYSTEM	19
	4.3.3 IMAGE UPLOAD & ANALYSIS	19
	4.3.4 REAL-TIME WEATHER FORECASTING	20
	4.4 IMPLEMENTATION DETAILS	20
	4.4.1 DEEP LEARNING MODEL	20
	4.4.2 MOBILE APP INTERFACE	20
	4.5 PROCESS FLOW	21
	4.6 ADVANTAGES OF THE SYSTEM	21
5	**SYSTEM REQUIREMENTS**	22
6	**IMPLEMENTATION AND RESULT**	25
	6.1 SOURCE CODE	25
	6.2 RESULTS	46
7	**CONCLUSION AND FUTURE ENHANCEMENT**	50
	7.1 CONCLUSION	50
	7.2 FUTURE ENHANCEMENT	50
	REFERENCES	52

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
1.1	Black Soil	3
1.2	Red Soil	3
1.3	Clay Soil	4
1.4	Alluvial Soil	4
4.1	Process Flow Chart	21
6.1	Welcome Screen	46
6.2	Upload Screen	47
6.3	File Selection	47
6.4	Image Preview	48
6.5	Result	48
6.6	Menu Bar	49
6.7	Weather Screen	49

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
CNN	Convolutional Neural Network
ANN	Artificial Neural Network
FC	Fully Connected (Neural Network Layer)
Ph	Potential of Hydrogen
NPK	Nitrogen Phosphorus Potassium
ML	Machine Learning
SVM	Support Vector Machine
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheet
SQL	Structured Query Language
OpenCV	Open Source Computer Vision
IoT	Internet Of Things
RF	Random Forest
CPU	Central Processing Unit
AMD	Advanced Micro Devices
VS CODE	Visual Studio Code
UI	User Interface

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Agriculture is the foundation of food production and economic growth in many regions. Soil quality is one of the most crucial factors influencing crop yield, but soil degradation caused by deforestation, climate change, excessive use of chemical fertilizers and unsustainable farming practices has significantly reduced agricultural productivity. Many farmers rely on traditional soil testing methods that require laboratory analysis, expert knowledge and significant time investment. These conventional approaches are often expensive and inaccessible, particularly for small-scale farmers in rural areas. The lack of accurate and timely soil analysis results in inefficient crop selection, reduced yields, and increased financial losses. The introduction of artificial intelligence and machine learning in agriculture presents an innovative solution to these challenges by providing automated, fast and cost-effective soil analysis and crop recommendations.

The Soil Quality Detection and Crop Recommendation System integrates machine learning techniques to assess soil quality and suggest optimal crops. Convolutional Neural Networks analyse soil images by extracting features such as texture, colour and moisture content to determine soil type. The system first verifies whether the uploaded image contains soil before proceeding with classification. Using deep learning techniques, the CNN model categorizes the soil into different types such as Black, Red, Clay, Alluvial based on its characteristics. The model then evaluates additional parameters, including pH level and nutrient content, to determine the soil's overall health and suitability

for farming. The Random Forest algorithm is employed to analyse these parameters and predict the most suitable crops that can thrive under the given soil conditions. By leveraging machine learning models, the system eliminates guesswork and provides farmers with precise recommendations to enhance crop yield and soil fertility.

To ensure accessibility and usability, the system is developed as a mobile application using the Kivy framework, allowing farmers to capture and upload soil images through their smartphones. The backend implementation is powered by Python, OpenCV and machine learning libraries such as TensorFlow, Keras and Scikit-learn, which enable efficient processing, training, and deployment of the AI models. Unlike traditional soil testing methods that require manual sample collection and laboratory analysis, this system provides real-time, on-the-go soil quality assessment and crop recommendations, empowering farmers with immediate and actionable insights. The integration of AI-driven image processing and predictive analytics enhances the accuracy and reliability of soil classification and crop selection, ensuring that farmers make data-driven decisions for improved agricultural outcomes.

The adoption of AI in precision agriculture introduces multiple benefits, including reduced dependency on expensive soil testing labs, increased efficiency in soil analysis and improved decision-making in crop selection. Traditional farming methods often rely on trial and error, leading to wasted resources and suboptimal crop yields. The system offers a scientific and data-driven approach to soil and crop management, allowing farmers to optimize fertilizer usage, irrigation planning and land management. By providing farmers with easy access to soil quality analysis, the system helps prevent soil depletion, promotes sustainable farming and supports long-term agricultural productivity. The use of AI in agriculture bridges the gap between traditional

knowledge and modern technology, ensuring that farmers, regardless of their technical expertise, can benefit from smart farming solutions.

The Soil Quality Detection and Crop Recommendation System represents a significant advancement in agricultural technology, environmental conservation and food security. As climate change and population growth continue to strain agricultural resources, adopting AI-powered solutions will be critical in ensuring efficient land use and maximizing crop yield. By harnessing the potential of deep learning and machine learning, this system contributes to the future of precision agriculture, empowering farmers with the tools they need to make informed decisions, reduce costs, and improve overall farm efficiency. As AI-driven innovations continue to evolve, integrating them into farming practices will be essential for creating a sustainable, productive and technology-driven agricultural ecosystem.

Figure 1.1 Black soil

Figure 1.2 Red Soil

Rich and fertile black soil, ideal for cotton and crop cultivation.Nutritious red soil with high iron content, perfect for cultivating pulses, groundnuts, and millets.

Figure 1.3 Clay Soil

Figure 1.4 Alluvial Soil

Dense and moisture-retentive clay soil, ideal for crops like paddy, wheat and vegetables requiring consistent water supply.Fertile alluvial soil, rich in minerals and ideal for growing rice, wheat, sugarcane.

1.2 PROBLEM STATEMENT

Soil quality plays a vital role in agriculture as it directly affects crop yield and overall farm productivity. Farmers often struggle to determine the exact condition of the soil due to the lack of efficient and accessible soil testing methods. Traditional soil testing techniques require laboratory analysis which is time-consuming and costly. Without accurate knowledge of soil properties such as texture, moisture and pH level, improper selection of crops and fertilizers leads to decreased productivity and long-term soil degradation. The excessive use of chemical fertilizers and pesticides further deteriorates soil health affecting both the environment and human health. The absence of a reliable and user-friendly soil quality detection system makes it difficult for farmers to make informed decisions about crop selection and soil management. An effective solution is needed to provide accurate soil analysis and crop recommendations using advanced technologies to ensure sustainable agricultural practices and higher crop yields.

1.3 BACKGROUND OF SOIL AND ITS QUALITY

Soil is the foundation of all terrestrial ecosystems and plays a crucial role in sustaining life by providing essential nutrients and support for plant growth. It is a dynamic and complex natural resource formed through the weathering of rocks and the decomposition of organic materials over thousands of years. The physical, chemical and biological properties of soil determine its ability to support plant life, regulate water flow and maintain ecological balance. Soil structure, texture, porosity and nutrient content influence its fertility and productivity, making it an essential component of agricultural sustainability. The presence of microorganisms, organic matter and minerals contributes to soil health by enhancing nutrient cycling and improving water retention, which directly impacts plant growth and crop yield.

Soil quality is a key factor in determining agricultural success and environmental stability. High-quality soil ensures the availability of essential nutrients, proper aeration, and water retention, leading to healthy crop development. Soil degradation due to erosion, salinity, compaction and pollution reduces its ability to sustain plant life, affecting agricultural output and food security. Traditional farming methods relied on natural indicators such as plant growth patterns, soil colour and moisture levels to assess soil quality.

Modern agriculture integrates scientific methods to assess and improve soil quality. Advanced soil testing techniques analyse pH levels, organic matter content and nutrient composition, helping farmers make informed decisions about soil management. Precision agriculture, remote sensing and data analytics enhance soil monitoring by providing real-time information on soil conditions, enabling better resource allocation and sustainable farming practices. Despite these advancements, challenges such as soil erosion, deforestation and land degradation continue to threaten soil health, making it

essential to adopt conservation practices. Sustainable soil management, including organic farming, minimal tillage, afforestation and the use of biofertilizers, helps restore soil quality and preserve its long-term productivity.

The role of technology in soil quality assessment has expanded with the introduction of artificial intelligence and machine learning. Image-based classification and predictive models enable accurate identification of soil types, nutrient deficiencies and potential risks to soil health. Machine learning algorithms analyse soil properties and recommend suitable crops based on soil conditions, optimizing agricultural efficiency and reducing resource wastage. The integration of smart sensors, IoT devices and automated systems further enhances soil monitoring, ensuring precise application of fertilizers and irrigation to improve soil quality.

1.4 CHALLENGES IN TRADITIONAL SOIL QUALITY ANALYZING METHOD

Traditional soil quality analysers face several challenges that hinder their effectiveness and accessibility. First, soil composition is complex, making it difficult to assess all the necessary factors such as texture, nutrients, and microbial activity through conventional methods. The process itself is also time-consuming, as soil samples often need to be sent to laboratories for analysis, resulting in delays that can affect timely decision-making for farmers. Moreover, laboratory testing can be expensive, which makes it less accessible for small-scale farmers.

Traditional methods are prone to inaccuracies due to sampling errors, which can lead to unreliable results. Furthermore, these methods often lack the ability to monitor soil quality in real-time, limiting farmers' ability to respond quickly to changes in soil conditions. The process is also labour intensive and

requires skilled personnel, which may not always be available, especially in rural areas.

Farmers may struggle to obtain precise and timely information, hindering their ability to optimize soil health and crop production. Traditional analysers generally do not provide tailored crop recommendations, forcing farmers to rely on their own knowledge or consult experts. At last the data generated can be difficult to interpret without specialized expertise, further complicating the decision-making process. These limitations highlight the need for more accessible, efficient and accurate methods for soil analysis.

1.5 IMPORTANCE OF ANALYZING SOIL QUALITY

Analysing soil quality is crucial for ensuring sustainable agricultural practices and optimizing crop production. Soil is the foundation of agriculture, and its health directly impacts plant growth, yield and overall ecosystem balance. By analysing soil quality, farmers can gain valuable insights into various factors such as pH levels, nutrient content, moisture and texture, which influence plant health. This helps in determining the right type of soil amendments, fertilizers and irrigation methods required for optimal crop growth. Regular soil analysis can also help identify nutrient deficiencies or imbalances, preventing issues like stunted growth, poor yields, or crop diseases.

Understanding soil quality is essential for soil conservation and long-term land management. It aids in preventing soil degradation, erosion, and nutrient loss, ensuring that the soil remains fertile and productive for future generations. Soil quality analysis also allows farmers to make data-driven decisions on crop rotation and land use, reducing the risk of overexploitation and enhancing biodiversity.

1.6 OBJECTIVE

The Soil Quality Detection and Crop Recommendation System aims to provide farmers with a comprehensive solution to assess and improve soil health for optimal agricultural productivity. Its primary objective is to analyse key soil parameters such as pH, texture, moisture content and nutrient levels to determine the overall quality of the soil. By classifying soil types and categorizing soil quality as good, moderate or bad the system enables farmers to take timely and appropriate actions to enhance soil fertility. The system offers crop recommendations based on the specific soil characteristics, ensuring higher yields and better crop suitability.

With a user-friendly interface, farmers can easily upload soil images or input data to receive real-time analysis and personalized recommendations, making the process accessible and efficient, even for those with limited technical knowledge. By promoting sustainable agriculture practices, the system helps farmers reduce the use of chemical fertilizers and pesticides while maximizing the potential of their land. The system empowers farmers to make data-driven decisions, improving soil health, crop yield and the long-term sustainability of their farming practices.

CHAPTER 2

SURVEY ON SOIL QUALITY DETECTION AND CROP RECOMMENDATION SYSTEM

P. Singh et al. (2020) investigates the application of fuzzy logic systems in agriculture to improve decision-making for tasks like irrigation, pest management, and crop health monitoring. Fuzzy logic is particularly suitable for agricultural environments, which are inherently uncertain due to factors like weather, soil properties, and plant growth patterns. The authors present a smart agriculture monitoring system that integrates fuzzy logic with real-time data collected from sensors on temperature, soil moisture, humidity, and other environmental parameters. The fuzzy system allows for the dynamic adjustment of farming practices based on imprecise and variable data, enabling better management of agricultural resources. For example, it can adjust irrigation schedules based on fluctuating soil moisture levels, reducing water wastage while ensuring optimal crop growth. The paper also discusses how the system helps in maintaining a balance between crop health and resource conservation.

R. Sharma et al. (2020) focuses on monitoring soil quality specifically for sugarcane cultivation. Sugarcane requires high-quality soil for optimal growth, and soil health significantly impacts its yield. The authors propose a machine learning-based approach using the Random Forest (RF) algorithm to assess soil quality. Data collected from various sugarcane fields is used to train the machine learning model, which analyzes key soil parameters such as pH, texture, moisture, and organic matter content. The trained model then predicts the quality of the soil and recommends the appropriate type of fertilizers and

amendments. The authors argue that this approach provides more accurate and faster soil quality assessments compared to traditional methods. Furthermore, it helps optimize fertilization practices, leading to improved crop yield and more sustainable farming practices.

M. Gupta et al. (2020) focuses on the development of a smart irrigation system that uses IoT sensors and machine learning algorithms to optimize irrigation schedules in precision agriculture. Traditional irrigation practices often lead to water wastage and suboptimal crop growth. The proposed system utilizes IoT devices to monitor real-time data on soil moisture, temperature, and environmental conditions. Machine learning algorithms then analyze this data to predict the optimal time and amount of water needed for irrigation. The system automatically adjusts irrigation schedules based on these predictions, ensuring that crops receive the right amount of water at the right time. The paper demonstrates the system's ability to conserve water while improving crop yields.

A. Kumar et al. (2021) explores a novel method for soil moisture detection and irrigation management using a non-contact, low-cost approach. The authors propose the use of a standard RGB camera to capture soil images and an ANN to process these images. Soil moisture content is often challenging to monitor using traditional methods like soil sampling, as they are invasive and labour intensive. The proposed system overcomes these limitations by using colour analysis of the soil, which changes with moisture content. The ANN is trained to classify the soil based on these colour changes, making it possible to predict irrigation needs. The pilot study showed that this system could reliably predict soil moisture, offering a scalable and efficient solution for smart irrigation. The paper emphasizes the cost-effectiveness of this technology, which could be implemented in agriculture for large-scale soil moisture monitoring.

S. K. Gupta et al. (2021) focus on the integration of AI and IoT for improving soil fertility and recommending the most suitable crops for specific regions. Traditional soil fertility assessments are often slow, expensive, and require laboratory testing. The proposed system uses IoT devices to collect soil data in real-time, including parameters like soil pH, nutrient content, and moisture levels. This data is then analysed using AI algorithms, specifically machine learning models, which provide recommendations for crop types that are most likely to thrive based on the current soil conditions. The system can also offer suggestions for optimal fertilization, crop rotation, and sustainable farming practices. The paper demonstrates the potential for this AI-powered system to increase crop yields while reducing resource consumption, making agriculture more efficient and sustainable.

C. Martin et al. (2021) focuses on providing farmers with easy-to-use AI tools that can analyse soil health without the need for expensive laboratory tests. By integrating data from IoT sensors and remote sensing technologies, the system provides real-time soil health analysis. The AI algorithms process this data to give farmers insights into soil fertility, nutrient levels, and potential issues like soil erosion or acidification. The project aims to reduce the reliance on traditional soil testing methods and promote more sustainable farming practices by offering cost-effective, accessible tools for soil management.

R. Kumar et al. (2021) introduces an AI-driven framework that combines crop prediction and soil fertility monitoring to enhance sustainable agriculture. By analysing historical climate data, soil health parameters and farming practices, the system predicts the most suitable crops for specific regions. It also uses machine learning algorithms to monitor soil fertility over time, recommending interventions like fertilization and crop rotation to improve soil quality. The paper emphasizes that this AI-based approach allows farmers to make data-driven decisions, leading to higher yields, reduced resource usage,

and more sustainable farming practices. The authors argue that this approach is not only cost-effective but also scalable for use in different agricultural regions.

S. Jain et al. (2021) explores the use of AI to predict soil moisture levels for precision irrigation in agriculture. Efficient water management is critical for sustainable farming, especially in areas facing water scarcity. The authors propose a machine learning-based system that analyzes environmental variables such as temperature, humidity, and precipitation to predict soil moisture levels. The system uses various machine learning models like decision trees and SVM to make accurate predictions, which are then used to optimize irrigation schedules. By predicting soil moisture in advance, the system ensures that crops receive adequate water while minimizing waste, making it ideal for precision irrigation applications.

L. Zhang et al. (2022) discusses the use of deep learning models, specifically CNNs, for soil classification in precision agriculture. Soil classification is a crucial step in precision agriculture as it directly impacts crop management, fertilization, and irrigation strategies. Traditional soil classification methods require manual sampling and laboratory analysis, which are time-consuming and costly. The authors propose a deep learning approach where CNNs are trained on soil images captured through various imaging techniques (such as RGB, infrared, etc.). The model is capable of classifying soil types with high accuracy, which can then be used to determine the best farming practices for each soil type. This approach eliminates the need for traditional soil testing, making it faster and more scalable for large-scale agricultural operations.

CHAPTER 3

SOIL QUALITY IDENTIFYING AND MONITORING APPROACH FOR SUGARCANE USING ML

The soil quality identifying and monitoring approach for sugarcane using ML for soil quality monitoring in sugarcane farming relies on traditional soil testing methods and basic machine learning models to classify soil quality and recommend fertilizers. The system applies Random Forest and Decision Tree algorithms to analyse soil parameters such as pH level, nitrogen, phosphorus, potassium and micronutrient availability. Based on these values it determines whether the soil is suitable for sugarcane cultivation.

3.1 TECHNOLOGIES USED

1. **Machine learning algorithms** – The Random Forest classifier is used for soil classification analysing multiple soil parameters and predicting soil fertility.
2. **Dataset-based analysis** – Pre collected soil data is processed using machine learning models to determine soil health.
3. **Statistical soil testing** – The system relies on traditional lab testing where soil samples are collected and analysed using chemical tests before inputting the values into the model.
4. **Fertilizer recommendation system** – Based on soil test results the system suggests appropriate fertilizers and nutrients to improve soil quality.

5. **Rule-based decision making** – Instead of deep learning the system follows fixed thresholds (e.g. if pH is below 5.5 it suggests lime treatment).

3.2 PROGRAMMING LANGUAGES

The existing system primarily uses Python as the core programming language for implementing machine learning models and processing soil data. Python is widely chosen for artificial intelligence and machine learning applications due to its simplicity, extensive library support and efficient handling of large datasets. In the system, Python is used for data preprocessing, feature extraction and model training. The Random Forest and Decision Tree algorithms are implemented using the Scikit-learn library to classify soil quality and provide recommendations for fertilizers and crop rotation. Additionally, Pandas and NumPy are utilized to manage and manipulate soil datasets, ensuring accurate data analysis.

For database management, SQL is used to store soil test results, classification outputs and fertilizer recommendations. This allows the system to maintain historical soil data, which can be referenced for future predictions and analysis. If the system includes a web-based interface, technologies like JavaScript, HTML and CSS may be used for frontend development. JavaScript helps in creating interactive dashboards where users can view soil reports and recommendations, while HTML and CSS structure and style the web pages for better usability.

3.3 PROCESS FLOW

Soil sample collection: Farmers manually collect soil samples from their fields.

1. **Lab-based soil testing**: The samples are sent to a laboratory for chemical analysis to measure pH, NPK levels and micronutrients.
2. **Data input into ML model**: The results from the lab test are manually entered into the machine learning model.
3. **Soil classification**: The Random Forest model predicts whether the soil is suitable, moderately suitable or unsuitable for sugarcane farming.
4. **Fertilizer suggestion**: Based on classification results the system recommends organic or chemical fertilizers to improve soil fertility.

3.4 DRAWBACKS

1. **Dependence on manual soil testing**
    - Farmers must collect and send soil samples to a lab leading to delays in obtaining results.
    - Chemical-based testing is time-consuming and expensive.
2. **Lack of real-time monitoring**
    - The system does not use IoT sensors to collect soil data instantly.
    - Changes in soil conditions are not continuously tracked making it difficult to adjust farming practices dynamically.
3. **Limited accuracy of predictions**
    - Machine learning models rely on pre-collected datasets which may not reflect real-time soil variations.
    - The system does not incorporate deep learning for enhanced image-based soil analysis.
4. **Fixed threshold-based recommendations**
    - The system follows static rules rather than dynamically adapting recommendations based on historical trends and weather conditions.

- It does not consider external factors like rainfall, soil erosion or microbial activity.

5. **Lack of user-friendly mobile integration**
    - Farmers must access the system via computers or print reports making it less accessible.
    - There is no mobile app to instantly upload soil images and get real-time recommendations.

CHAPTER 4

SOIL QUALITY DETECTION AND CROP RECOMMENDATION SYSTEM

Agriculture plays a crucial role in food security and economic stability. However, soil degradation and improper crop selection often lead to reduced yields and environmental damage. The project presents a machine learning-based soil quality detection and crop recommendation system, which utilizes deep learning for soil classification, pH level prediction and K-Means clustering for soil categorization. The system is integrated into a mobile application built using Kivy, allowing farmers to scan soil images, receive instant analysis and get suitable crop recommendations based on soil type and pH value.

4.1 TECHNOLOGIES USED

The project combines artificial intelligence, deep learning, and mobile application development to deliver an efficient, user-friendly solution.

- **PyTorch:** Used for developing the deep learning model, which includes a modified ResNet18 for soil classification and pH level prediction.
- **Random Forest:** Implemented in earlier models for soil classification based on predefined datasets.
- **K-Means Clustering:** Used to categorize soil samples into acidic, neutral, and alkaline groups based on extracted features.

- **Torchvision & OpenCV:** Applied for image transformations, resizing, normalization, and feature extraction.
- **PIL (Python Imaging Library):** Handles image loading and conversion before prediction.
- **Kivy & KivyMD:** Used to build an interactive mobile application where users can upload soil images, receive soil analysis, and view crop recommendations.
- **OpenWeather API:** Integrated into the system to provide real-time weather updates, helping farmers make informed decisions based on climate conditions.

4.2 SYSTEM ARCHITECTURE

- **Image Processing & Classification:** The user uploads a soil image, which undergoes preprocessing (resizing, normalization) before being analysed by the ResNet18 model to classify the soil type and estimate its pH level.
- **Soil pH Categorization:** The K-Means clustering algorithm categorizes the soil into Acidic (pH < 6), Neutral (pH ~ 7), or Alkaline (pH > 7.5) based on extracted features.
- **Crop Recommendation System:** Based on the identified soil type and pH level, the system suggests suitable crops using a predefined mapping of crops to optimal soil conditions.
- **Weather Forecasting:** The system fetches real-time weather data (temperature, humidity, wind speed) via the OpenWeather API to help farmers plan planting and irrigation.

4.3 FEATURES & FUNCTIONALITIES

4.3.1 Soil Classification & pH Prediction

Uses ResNet18 deep learning model to classify soil into four types:

- Alluvial
- Black
- Clay
- Red

Predicts the soil's pH level to determine its suitability for different crops.

4.3.2 Crop Recommendation System

- Provides crop recommendations based on the soil type and pH value.
- Example recommendations:
  - Alluvial (Neutral pH): Rice, Wheat, Sugarcane
  - Black (Acidic pH): Cotton, Soybean, Tobacco
  - Clay (Alkaline pH): Lentils, Peas, Mustard

4.3.3 Image Upload & Analysis

- Users can upload a soil image, and the model will analyze it within seconds.
- The result includes soil type, pH level, and crop recommendations.

4.3.4 Real-Time Weather Forecasting

- Fetches current weather conditions such as temperature, humidity, and wind speed based on the user's location.

4.4 IMPLEMENTATION DETAILS

4.4.1 Deep Learning Model (train.py)

- The train.py script defines and trains the SoilClassificationWithPH model using ResNet18.
- It includes two outputs:
    1. Soil classification (4 categories)
    2. pH level regression
- The model is trained for 50 epochs using an Adam optimizer and CrossEntropyLoss for classification and MSELoss for pH regression.
- After training, the model is saved as "soil_ph_classification_model.pth" for later use.

4.4.2 Mobile App Interface (main.py)

- The main.py script initializes the Kivy mobile application.
- It includes four screens:
    1. Welcome Screen – Displays an introduction to the app.
    2. Upload Screen – Allows users to upload a soil image for analysis.
    3. Results Screen – Displays soil classification, pH level, and recommended crops.
    4. Weather Screen – Provides real-time weather updates.

- The app loads the trained deep learning model and predicts soil type and pH value for uploaded images.

4.5 PROCESS FLOW

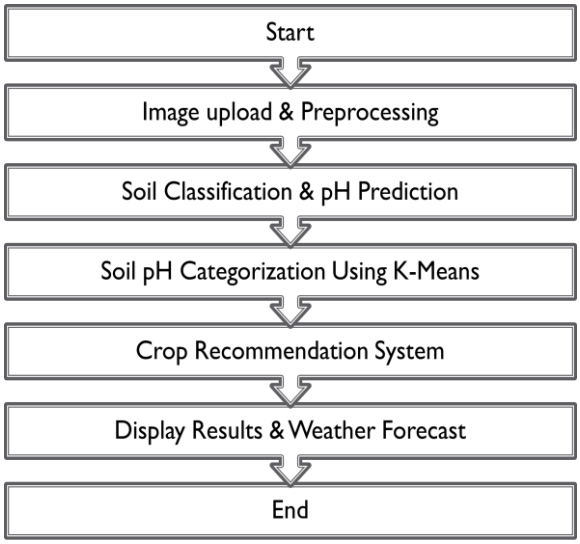

Figure 4.1 Process Flow Chart

The above figure shows the process of the application

4.6 ADVANTAGES OF THE SYSTEM

1.Instant Soil Analysis

Farmers can get real time insights.

2.AI-Powered Crop Recommendation

Suggests optimal crops based on soil type and pH category.

3.Mobile Accessibility

Farmers can use system anytime, anywhere via mobile app.

4.Weather Forecasting Integration

Helps optimize irrigation schedules and planting times.

CHAPTER 5

SYSTEM REQUIREMENTS

COMPONENTS		DESCRIPTIONS
Operating System	:	Windows 10/11, Ubuntu 20.04+, macOS 10.15+
Processor	:	Intel Core i5/i7 (8th Gen or newer) / AMD Ryzen 5+
RAM	:	Minimum 8GB, Recommended 16GB+
Storage	:	Minimum 50GB, Recommended 100GB+ SSD
GPU	:	NVIDIA GTX 1650 / RTX 2060+ (for ML model training)
Python Version	:	Python 3.8+
Libraries& Frameworks	:	TorchVision,PyTorch,OpenCV,Scikit-learn, Pandas, NumPy
Development Tools	:	Jupyter Notebook, PyCharm, VS Code

PROCESSOR (CPU)

A CPU is the brain of any computing system. It executes all the instructions and computations required for running software applications. For project, a multi-core processor such as an Intel Core i5 (8th Gen or later) or AMD Ryzen 5 is required, with a higher-end processor like Intel Core i7/Ryzen 7 recommended for better efficiency.

NEED OF PROCESSOR

CNN for soil classification, require significant computational power.

The CPU handles data preprocessing, model inference and feature extraction, ensuring smooth operation.

PYTHON & LIBRARIES

The primary programming language for this project is Python 3.8+ because of its extensive support for artificial intelligence, machine learning, and image processing. Python is widely used in AI-based applications due to its rich ecosystem of libraries, ease of use and strong community support. In this project, Python plays a crucial role in developing the deep learning model for soil classification, implementing machine learning for crop recommendation, and creating a mobile-friendly user interface**.**

Kivy, a Python framework, is used for mobile application development to ensure compatibility with Android devices. This allows farmers and agricultural professionals to use the application on their smartphones for real-time soil analysis. Torchvision and pyTorch are essential for building and training the CNN model that classifies soil images based on their texture, moisture, and colour. OpenCV is used for image processing, enabling feature extraction from soil images to improve classification accuracy.

Scikit-learn is another important library that helps implement the Random Forest algorithm for crop recommendation. It processes extracted soil data and predicts suitable crops based on soil quality, pH level. NumPy and Pandas are required for handling large datasets efficiently, ensuring smooth processing of soil information.

DEVELOPMENT TOOL

VS Code is a lightweight, open-source code editor developed by Microsoft. It is widely used for programming due to its extensibility, powerful debugging tools, and built-in Git integration. For the Soil Quality Detection and

Crop Recommendation project, VS Code serves as the primary development environment because of its seamless support for Python, Kivy, Torch and OpenCV.

NEED OF VSCODE

VS Code provides an efficient and developer-friendly environment for writing, debugging, and testing the machine learning models and mobile application. It offers syntax highlighting, IntelliSense (code completion), and built-in terminal support, which simplifies coding and execution. The availability of numerous extensions, such as Python extension pack, Jupyter Notebook support and Kivy plugin, enhances the development workflow.

VS Code will be used to write and test Python scripts for deep learning-based soil classification and crop recommendation. It helps in debugging CNN models, handling image processing with OpenCV and optimizing machine learning algorithms. Additionally, for mobile app development, the Kivy framework can be integrated into VS Code, allowing developers to design the UI and test the application directly within the editor. The built-in Git support also enables version control, ensuring smooth collaboration and code management.

CHAPTER 6

IMPLEMENTATION AND RESULT

6.1 SOURCE CODE

train.py:

```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision.transforms as transforms
import torchvision.datasets as datasets
from torch.utils.data import DataLoader
from torchvision import models
from sklearn.cluster import KMeans
import numpy as np
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
```

```
transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])])
data_dir = "./SoilDatasets/train_data"  # Update path to dataset location
train_dataset = datasets.ImageFolder(root=data_dir, transform=transform)
train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
model = models.resnet18(pretrained=True)
num_ftrs = model.fc.in_features
class SoilClassificationWithPH(nn.Module):
  def __init__(self, base_model, num_classes=4):
    super(SoilClassificationWithPH, self).__init__()
    self.base_model = nn.Sequential(*list(base_model.children())[:-1])
    self.fc_class = nn.Linear(num_ftrs, num_classes)  # Soil type classification
    self.fc_regress = nn.Linear(num_ftrs, 1)  # pH level regression
  def forward(self, x):
    features = self.base_model(x)  # Extract features
    features = torch.flatten(features, 1)  # Flatten output
    class_output = self.fc_class(features)  # Soil type prediction
    ph_output = self.fc_regress(features)  # pH level prediction
    return class_output, ph_output
model = SoilClassificationWithPH(model).to(device)
criterion_class = nn.CrossEntropyLoss()  # Classification Loss
```

```
criterion_regress = nn.MSELoss()  # Regression Loss (for pH values)
optimizer = optim.Adam(model.parameters(), lr=0.001)
epochs = 50
for epoch in range(epochs):
    model.train()
    running_loss = 0.0
    correct = 0
    total = 0
    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)
        optimizer.zero_grad()
        class_outputs, ph_outputs = model(images)
                ph_labels = torch.tensor([5.5 + i for i in labels],
 dtype=torch.float32).to(device)
        loss_class = criterion_class(class_outputs, labels)
        loss_regress = criterion_regress(ph_outputs.squeeze(), ph_labels)
        loss = loss_class + loss_regress
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
        _, predicted = class_outputs.max(1)
        total += labels.size(0)
```

```
        correct += predicted.eq(labels).sum().item()
                print(f"Epoch {epoch+1}/{epochs}, Loss: {running_loss/len(train_loader):.4f}, Accuracy: {100 * correct / total:.2f}%")
torch.save(model.state_dict(), "soil_ph_classification_model.pth")
print("Model training complete and saved!")
features = []
with torch.no_grad():
    for images, _ in train_loader:
        images = images.to(device)
        encoded_features, _ = model(images)
        encoded_features = encoded_features.cpu().numpy()
        features.append(encoded_features)
features = np.vstack(features)
num_clusters = 3  # Acidic, Neutral, Alkaline
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
clusters = kmeans.fit_predict(features)
ph_labels = {
    0: "Acidic (pH < 6)",
    1: "Neutral (pH ~ 7)",
    2: "Alkaline (pH > 7.5);}
for i, label in enumerate(clusters[:10]):
    print(f"Image {i}: Cluster {label} → {ph_labels[label]}")
```

main.py:

```
from kivy.lang import Builder
from kivy.core.window import Window
from kivymd.app import MDApp
from kivymd.uix.screen import Screen
from kivymd.uix.filemanager import MDFileManager
from kivymd.uix.button import MDRaisedButton
from kivymd.uix.label import MDLabel
from kivymd.uix.menu import MDDropdownMenu
from kivy.clock import Clock
import torch
import torch.nn as nn
import torchvision.transforms as transforms
from torchvision import models, datasets
from torch.utils.data import DataLoader
from PIL import Image
import os
import requests
import numpy as np
from sklearn.cluster import KMeans
Window.size = (412, 917)
```

```
KV = """MDScreenManager:
    WelcomeScreen:
    UploadScreen:
    ResultsScreen:
    WeatherScreen:
<WelcomeScreen>:
    name: "welcome"
    MDFloatLayout:
        Image:
            source: "background.jpg"
            size: self.size
            allow_stretch: True
            keep_ratio: False
            opacity: 0.5
        MDLabel:
            text: "Welcome to\\n[font=Roboto-Bold]Soil Scan[/font]"
            markup: True
            halign: "center"
            font_style: "H4"
            pos_hint: {"center_y": 0.6}
        MDRaisedButton:
```

```
            text: "Continue"
            pos_hint: {"center_x": 0.5, "center_y": 0.4}
            size_hint: None, None
            size: 180, 50
            on_release: root.manager.current = "upload"
<UploadScreen>:
    name: "upload"
    MDFloatLayout:
        Image:
            source: "background.jpg"
            size: self.size
            allow_stretch: True
            keep_ratio: False
            opacity: 0.5
     MDIconButton:
            icon: "menu"
            id:open_menu_button
            pos_hint: {"top": 1, "left": 0.2}
            size_hint: None, None
            size: "55dp", "55dp"
            on_release: root.open_menu()
```

```
MDCard:
        id: upload_card
        orientation: "vertical"
        size_hint: None, None
        size: "200dp", "200dp"
        pos_hint: {"center_x": 0.5, "center_y": 0.7}
        elevation: 10
        padding: "10dp"
        spacing: "10dp"
        opacity: 0
        Image:
            id: uploaded_image
            source: "default.png"
            size_hint: None, None
            size: 150, 150
            pos_hint: {"center_x": 0.5, "center_y": 1}
      MDRaisedButton:
        text: "Upload Soil Image"
        pos_hint: {"center_x": 0.5, "center_y": 0.5}
        size_hint: None, None
        size: 200, 50
```

```
            on_release: root.open_file_manager()
        MDRaisedButton:
            text: "Scan"
            pos_hint: {"center_x": 0.5, "center_y": 0.4}
            size_hint: None, None
            size: 180, 50
            on_release: root.analyze_image()
<ResultsScreen>:
    name: "results"
    MDFloatLayout:
        Image:
            source: "background.jpg"
            size: self.size
            allow_stretch: True
            keep_ratio: False
            opacity: 0.5
        MDIconButton:
            icon: "arrow-left"
            pos_hint: {"top": 1, "left": 0}
            size_hint: None, None
            size: "55dp", "55dp"
```

```
    on_release: app.go_back_to_upload()
MDLabel:
    id: result_label
    text: ""
    theme_text_color: "Custom"
    text_color: 0, 0, 0, 1
    font_style: "H4"
    halign: "center"
    pos_hint: {"center_x": 0.5, "center_y": 0.7}
    size_hint: None, None
    width: 300
MDLabel:
    id: ph_label
    text: ""
    halign: "center"
    font_style: "H5"
    pos_hint: {"center_y": 0.6}
MDLabel:
    id: crop_label
    text: ""
    halign: "center"
```

```
            font_style: "H5"
            pos_hint: {"center_y": 0.6}
        MDLabel:
            id: crop_label
            text: "Recommended Crops: "
            font_style:"H5"
            color: 0, 0, 0, 1  # Green color
            pos_hint: {"center_y": 0.5}
<WeatherScreen>:
    name:"weather"
    MDFloatLayout:
        Image:
            source: "background.jpg"
            size: self.size
            allow_stretch: True
            keep_ratio: False
            opacity: 0.5
        MDIconButton:
            icon: "menu"
            id:open_menu_button
            pos_hint: {"top": 1, "left": 0.2}
```

```
    size_hint: None, None
    size: "55dp", "55dp"
    on_release: root.open_menu()
MDLabel:
    text: "Weather Information"
    font_size: "20sp"
    size_hint_y: None
    height: "40dp"
    halign: "center"
    pos_hint: {"center_x": 0.5,"center_y": 0.8}
MDTextField:
    id: city_input
    hint_text: "Enter City"
    size_hint_y: None
    size_hint_x: None
    height: "20dp"
    width: "100dp"
    pos_hint: {"center_x": 0.5,"center_y": 0.6}
MDRaisedButton:
    text: "Get Weather"
    size_hint: None, None
```

```
            size: 200, 50
            pos_hint: {"center_x": 0.5,"center_y": 0.5}
            on_release: root.fetch_weather()
        MDLabel:
            id: weather_label
            text: ""
            size_hint_y: None
            height: "80dp"
            halign: "center"
            pos_hint: {"center_x": 0.5,"center_y": 0.4"""
class SoilClassificationWithPH(nn.Module):
    def __init__(self, base_model, num_classes=4):
        super(SoilClassificationWithPH, self).__init__()
            self.base_model = nn.Sequential(*list(base_model.children())[:-1])  # Remove final FC layer
        self.fc_class = nn.Linear(base_model.fc.in_features, num_classes)  # Soil type classification
            self.fc_regress = nn.Linear(base_model.fc.in_features, 1)  # pH level regression
    def forward(self, x):
        features = self.base_model(x)  # Extract features
        features = torch.flatten(features, 1)  # Flatten output
```

```
        class_output = self.fc_class(features)  # Soil type prediction
        ph_output = self.fc_regress(features)  # pH level prediction
        return class_output, ph_output, features
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
base_model = models.resnet18(pretrained=False)
model = SoilClassificationWithPH(base_model)
model.load_state_dict(torch.load("soil_ph_classification_model.pth", map_location=device))
model = model.to(device)
model.eval()
soil_classes = ["Alluvial", "Black", "Clay", "Red"]
ph_labels = {
    0: "Acidic",
    1: "Neutral",
    2: "Alkaline"}
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
      transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])])
def train_kmeans(data_path, num_clusters=3):
    dataset = datasets.ImageFolder(root=data_path, transform=transform)
```

```
    data_loader = DataLoader(dataset, batch_size=32, shuffle=False)
    feature_list = []
    with torch.no_grad():
        for images, _ in data_loader:
            images = images.to(device)
            _, _, features = model(images)
            feature_list.append(features.cpu().numpy())
    feature_matrix = np.vstack(feature_list)  # Convert to NumPy array
    kmeans = KMeans(n_clusters=num_clusters, random_state=42, n_init=10)
    kmeans.fit(feature_matrix)  # Train K-Means
    return kmeans
kmeans = train_kmeans("./SoilDatasets/train_data")
crop_recommendations = {
    "Alluvial": {"Acidic": ["Tea", "Rubber", "Pineapple"],
        "Neutral": ["Rice", "Wheat", "Sugarcane"],
        "Alkaline": ["Barley", "Oats", "Peas"]},
    "Black": {"Acidic": ["Cotton", "Soybean", "Tobacco"],
        "Neutral": ["Maize", "Groundnut", "Sunflower"],
        "Alkaline": ["Sorghum", "Millets", "Sesame"]},
    "Clay": {"Acidic": ["Coffee", "Cocoa", "Spices"],
        "Neutral": ["Paddy", "Sugarcane", "Jute"],
```

```
        "Alkaline": ["Lentils", "Peas", "Mustard"]},
    "Red": {"Acidic": ["Tapioca", "Pineapple", "Citrus"],
        "Neutral": ["Tomato", "Chili", "Onion"],
        "Alkaline": ["Groundnut", "Ragi", "Castor"]}}
def predict_soil_and_ph(image_path):
    image = Image.open(image_path).convert("RGB")
    image = transform(image).unsqueeze(0).to(device)
    with torch.no_grad():
        class_output, ph_output, features = model(image)
        class_pred = torch.argmax(class_output, dim=1).item()
        soil_type = soil_classes[class_pred]
        ph_value = round(ph_output.item(), 2)
        feature_vector = features.cpu().numpy().reshape(1, -1)
        cluster_label = kmeans.predict(feature_vector)[0]
        ph_category = ph_labels[cluster_label]  # Acidic, Neutral, Alkaline
        recommended_crops = crop_recommendations[soil_type][ph_category]
    return soil_type, ph_value, ph_category, recommended_crops
class WelcomeScreen(Screen):
    pass
class UploadScreen(Screen):
    def __init__(self, **kwargs):
```

```
        super().__init__(**kwargs)
        self.file_manager = MDFileManager(select_path=self.select_file)
        self.selected_image_path = ""
    def open_file_manager(self):
        self.file_manager.show(os.getcwd())
    def select_file(self, file_path):
        self.selected_image_path = file_path
        print(f"Selected file: {file_path}")
        self.file_manager.close()
        self.ids.uploaded_image.source = file_path
        self.ids.upload_card.opacity = 1
    def analyze_image(self):
        if not self.selected_image_path:
            print("No image selected!")
            return
                soil_type, ph_value, ph_category, recommended_crops =
predict_soil_and_ph(self.selected_image_path)
        results_screen = self.manager.get_screen("results")
                results_screen.update_result(soil_type, ph_value, ph_category,
recommended_crops)
        self.manager.current = "results"
    def open_menu(self):
```

```
        """Opens the dropdown menu with navigation options."""
                menu_items = [{"viewclass": "OneLineListItem", "text": "Scan","on_release": self.navigate_scan},
                {"viewclass": "OneLineListItem", "text": "Weather","on_release": self.navigate_weather}]
        self.menu = MDDropdownMenu(
            caller=self.ids.open_menu_button,
            items=menu_items,
            width_mult=4)
        self.menu.open()
    def navigate_scan(self):
        self.manager.current = "upload"
    def navigate_weather(self):
        self.manager.current = "weather"
class ResultsScreen(Screen):
            def update_result(self, soil_type, ph_value, ph_category, recommended_crops):
        self.ids.result_label.text = f"Soil Type: {soil_type}"
        self.ids.ph_label.text = f"pH Level: {ph_value} ({ph_category})"
                self.ids.crop_label.text = f"Recommended Crops: {', '.join(recommended_crops)}"
```

```
    def open_menu(self):
        """Opens the dropdown menu with navigation options."""
        menu_items = [
                {"viewclass": "OneLineListItem", "text": "Scan","on_release": self.navigate_scan},
                {"viewclass": "OneLineListItem", "text": "Weather","on_release": self.navigate_weather},]
        self.menu = MDDropdownMenu(
            caller=self.ids.open_menu_button,
            items=menu_items,
            width_mult=4)
        self.menu.open()
def navigate_scan(self):
        self.manager.current = "upload"
    def navigate_weather(self):
        self.manager.current = "weather"
class WeatherScreen(Screen):
    def fetch_weather(self):
        """Fetches weather details based on the entered city."""
        city = self.ids.city_input.text.strip()
```

```
    if city:
        api_key = "24d1f55fb007fb74335ecf1fdb9325bc"  # Replace with your working API key url = f"https://api.openweathermap.org/data/2.5/weather?q={city}&appid={api_key}&units=metric"
        # Request weather data from API
        response = requests.get(url)
        if response.status_code == 200:
            data = response.json()
            print(data)
            if "main" in data and "weather" in data:
                temp = data["main"]["temp"]
                description = data["weather"][0]["description"]
                humidity = data["main"]["humidity"]
                wind_speed = data["wind"]["speed"]
                weather_info = f"City: {city}\nTemperature: {temp}°C\nCondition: {description}\nHumidity: {humidity}%\nWind Speed: {wind_speed} m/s"
                # Update the label with the weather info
                self.ids.weather_label.text = weather_info
            else:
                self.ids.weather_label.text = f"Error: Invalid data received for {city}. Please try again."
        else:
```

```
            self.ids.weather_label.text = f"Error: Could not retrieve weather data for {city}. Please try again."
        else:
            self.ids.weather_label.text = "Please enter a valid city."
    def open_menu(self):
        """Opens the dropdown menu with navigation options."""
        menu_items = [
                {"viewclass": "OneLineListItem", "text": "Scan","on_release": self.navigate_scan},
            {"viewclass": "OneLineListItem", "text": "Weather","on_release": self.navigate_weather},]
        self.menu = MDDropdownMenu(
            caller=self.ids.open_menu_button,
            items=menu_items,
            width_mult=4)
        self.menu.open()
    def navigate_scan(self):
        self.manager.current = "upload"
    def navigate_weather(self):
        self.manager.current = "weather"
class SoilScanApp(MDApp):
    def build(self):
```

```
        return Builder.load_string(KV)

    def go_back_to_upload(self):
        self.root.current = "upload"

if __name__ == "__main__":
    SoilScanApp().run()
```

6.2 RESULTS

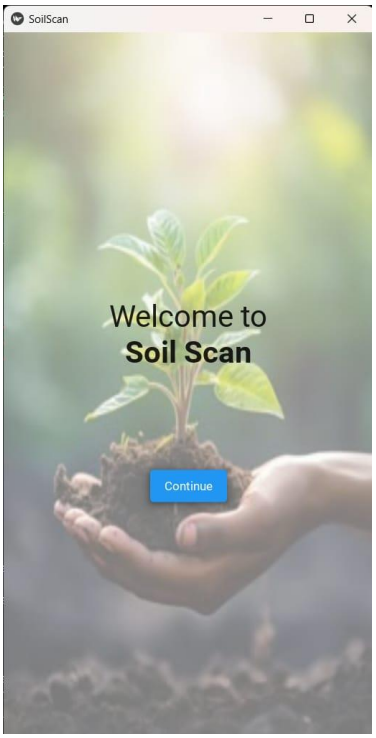

Figure 6.1 Welcome Screen

The above figure is the Welcome Page of Soil Scan, featuring a greeting and a "Continue" button.

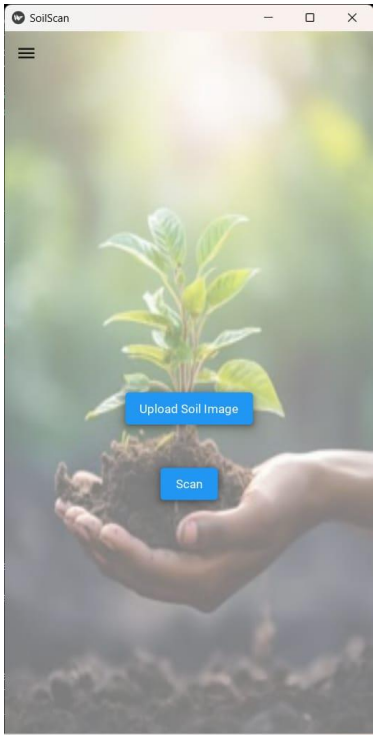

Figure 6.2 Upload Screen

The above figure is the Scanning Page of Soil Scan, allowing users to upload a soil image and scan it.

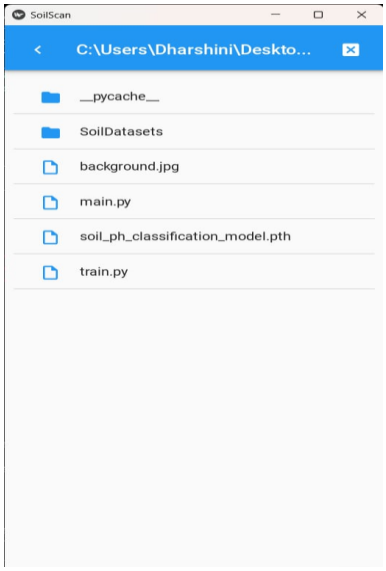

Figure 6.3 File Selection

The above image shows the File Selection Page, it allows users to browse directories and select a file for processing.

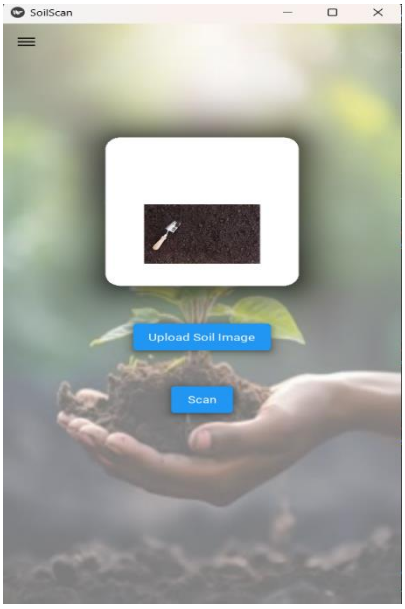

Figure 6.4 Image Preview

The above image shows Image Preview Page it displays the uploaded soil image before scanning.

Figure 6.5 Result

The above image shows the identified soil type, pH level and recommended crops.

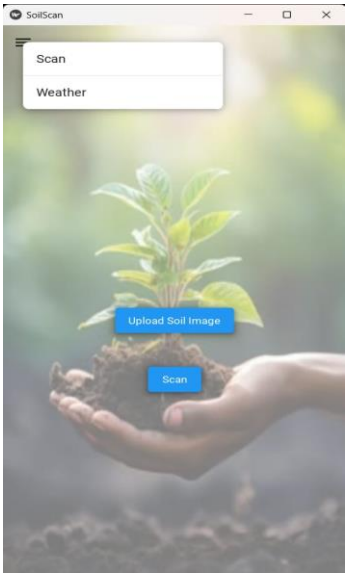

Figure 6.6 Menu Bar

The above image displays a sidebar menu with options for "Scan" and "Weather".

Figure 6.7 Weather Screen

The above image shows Weather Information Screen It allows users to enter a city name and fetch real-time weather.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

The integration of deep learning in soil quality detection and crop recommendation significantly enhances agricultural productivity by providing accurate soil assessments and optimized crop suggestions. By analysing soil properties and patterns, deep learning models can predict suitable crops, reducing resource wastage and improving yield. This approach eliminates the need for extensive manual testing, making precision farming more accessible. With further advancements, incorporating real-time data and adaptive learning techniques can refine predictions, ensuring sustainable and efficient agricultural practices. Future research can focus on expanding datasets and improving model accuracy to support diverse soil conditions globally.

7.2 FUTURE ENHANCEMENT

Future enhancements for soil quality detection and crop recommendation systems can focus on integrating real-time satellite imagery, weather data, and advanced deep learning models for more precise predictions. Enhancing model accuracy with larger, diverse datasets and implementing cloud-based AI solutions can improve accessibility for farmers developing a user-friendly mobile application with multilingual support can increase

adoption among local farmers. Incorporating explainable AI techniques will help users understand recommendations better, ensuring trust and usability. Further research can explore sustainable farming practices by integrating soil health monitoring with eco-friendly fertilization strategies.

REFERENCES

1. Anbarasan, P., et al. "Crop Recommendation and Yield Prediction Using Machine Learning Based Approaches." ResearchGate, 2024.
2. Agarwal, S.**,** M. Ahmad, et al., "Crop Recommendation Based on Soil Properties: A Machine Learning Approach," International Journal of Agricultural Data Science, *vol.* 12, pp. 45-58, 2023.
3. Ahmed, F. U.**,** A. Das, et al., "A Machine Learning Approach for Crop Yield and Disease Prediction Integrating Soil Nutrition and Weather Factors," arXiv preprint arXiv:2403.19273, March 2024.
4. Chana, A. M.**,** A. Thorat, et al., "Real-Time Crop Prediction Based on Soil Fertility and Weather Forecast Using IoT and a Machine Learning Algorithm," Agricultural Sciences, vol. 14, no. 5, pp. 645-660, May 2023.
5. Chen, L.**,** W. Zhang, et al., "CNN-Based Soil Texture Classification for Precision Agriculture," IEEE Access, vol. 10, pp. 14321-14330, 2022.
6. Dasgupta, **S.**, S. Pate, et al., "Soil Fertility Prediction Using Combined USB-microscope Based Soil Image, Auxiliary Variables, and Portable X-Ray Fluorescence Spectrometry,*"* arXiv preprint arXiv:2404.12415, April 2024.
7. Delgadillo-Duran, D. A.**,** C. A. Vargas-García, et al., "Using vis-NIRS and Machine Learning Methods to Diagnose Sugarcane Soil Chemical Properties," arXiv preprint arXiv:2012.12995, December 2020.
8. Gayathiri, B., et al. "Machine Learning Based Crop Suitability Prediction and Fertiliser Recommendation System." ResearchGate, 2023.

9. Kumar, J., R. Singh, et al., "Deep Learning for Soil Nutrient Prediction and Crop Suitability," Computers and Electronics in Agriculture, vol. 190, pp. 105-115, 2022.
10. Li, H., X. Zhou, et al., "Remote Sensing and Deep Learning for Soil Moisture and Fertility Prediction," Remote Sensing in Agriculture, vol. 15, no. 4, pp. 332-345, 2023.
11. Mariappan, A. K., C. Madhumitha, et al., "Crop Recommendation System through Soil Analysis Using Classification in Machine Learning," International Journal of Advanced Science and Technology, vol. 29, no. 3, pp. 12738-12747, March 2020.
12. Madhuri, J., M. Indiramma, et al., "Artificial Neural Networks Based Integrated Crop Recommendation System Using Soil and Climatic Parameters," Indian Journal of Science and Technology, vol. 14, issue 19, pp. 1587-1597, February 2021.
13. Ngo, Q. H., N.-A. Le-Khac, et al., "Predicting Soil pH by Using Nearest Fields," arXiv preprint arXiv:1912.01303, December 2019.
14. Patel, M., V. Gupta, et al., "AI-Based Smart Agriculture: Soil Health Monitoring and Crop Yield Prediction," Agricultural Systems and AI, vol. 17, no. 2, pp. 125-137, 2023.
15. Rao, R. V. R., U. S. Reddy, et al., "Sparse Attention Regression Network Based Soil Fertility Prediction With Ummaso," *arXiv preprint* arXiv:2404.10274, April 2024.
16. Sharma, P., K. Verma, et al., "A Hybrid Deep Learning Approach for Soil Quality Analysis and Crop Selection," Journal of Environmental Informatics, vol. 38, no. 3, pp. 220-230, 2021.
17. Sharma, A., et al. "AI-Farm: A Crop Recommendation System." IEEE ICACC, 2021.

18. Sumathi, P., et al. "Improved Soil Quality Prediction Model Using Deep Learning for Smart Agriculture Systems." TechScience Press, 2023.

19. Sunandini, M., K. Hema Sree, et al., "Smart Soil Fertilizer Monitoring and Crop Recommendation System by Using IoT and Machine Learning Technology," International Journal of Engineering Research & Technology (IJERT), vol. 12, issue 03, May 2024.

20. Verma, K., P. Sharma, et al., "Soil Nutrient Prediction Using Hybrid Machine Learning Models," Journal of Agricultural Informatics, vol. 14, no. 1, pp. 87-99, 2024.

LIST OF PUBLICATIONS

1. Mrs. E. Jananandhini, G. Devadharshini, T. Dharanya, S. Malini has presented A paper entitled “SOIL QUALITY DETECTION AND CROP RECOMMENDATION SYSTEM” in National Conference on AI NEXTGEN 2025: INNOVATIONS & BEYOND - NCAIN 2025 held on 07th March 2025 organized by the Department of Information Technology, Bannari Amman Institute of Technology, Sathyamangalam.