VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI-590014

An Internship Report On

"Soil Quality Prediction"

Submitted By

Mr. Basavaraj Kokane

USN:2VS21CS010

An internship report submitted to the Visvesvaraya Technological University, Belagavi in Partial Fulfillment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

Under the Guidance of

Prof. NABI M

Assistant. Professor, Dept. of Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VSM's SOMASHEKHAR R KOTHIWALE INSTITUTE OF TECHNOLOGY, NIPANI-591237

2024-25

VSM's SOMASHEKHAR R KOTHIWALE INSTITUTE OF TECHNOLOGY, NIPANI-591237

(Affiliated to VTU Belagavi)

CERTIFICATE

This is to certify that, the Internship work entitled "Soil Quality Prediction" carried out by **Mr. Basavaraj Kokane (2VS21CS010)** is Bonafide student of **VSM's SRK Institute of Technology, Nipani** in partial fulfilment for the award of Bachelor of Engineering in **Computer Science and Engineering** of the **Visvesvaraya Technological University, Belagavi** during the year 2024-25. It is certified that all corrections / suggestions indicated during Internal Assessment have been incorporated into the Internship report. The Internship report has been approved as it satisfies the academic requirements in respect of the work prescribed for the said degree.

Prof. Nabi M	**Prof. Neelamma S**
Asst. Prof & Guide	**Internship Coordinator**
Prof. Rahul Palakar	**Dr. Umesh Patil**
Head of Department	**Principal & Director**

Examiners

Name	**Signature with Date**
1.	
2.	

CERTIFICATE

ACKNOWLEDGEMENT

The success of our internship was not due to our internship work alone, but also due to the interest help, and able guidance offered to us by the staff members of the Computer Science and Engineering Department.

It is our duty to acknowledge with gratitude the help rendered to us by these individuals, without whom we would not be able to carry out the Internship to the best of our ability and to the satisfaction of our superiors.

First and foremost, we wish to record our sincere gratitude to our beloved principal, **Dr. Umesh P Patil, VSM's SRKIT, Nipani** for his constant support and encouragement in the preparation of this report and for making available the library and laboratory facilities needed to prepare this report.

We would also like to thank Prof. **Rahul Palakar**, Head of the Computer Science and Engineering Department, VSM's SRKIT, for his valuable support and guidance throughout the period of this Internship.

We also express our gratefulness to our internship coordinator, Prof. **Neelamma Shinannavar** Also, we would like to express our special gratitude and thanks to our internship guide, **Prof. Nabi M** for guiding us in investigations for this Internship and in carrying out experimental work. Our numerous discussions with her were extremely helpful. Their contributions and technical support in preparing this report are greatly acknowledged.

Last, but not least, we wish to thank our parents for financing our studies in this college as well as for constantly encouraging us to learn. Their personal sacrifice in providing this opportunity to learn engineering is gratefully acknowledged.

BASAVARAJ KOKANE
USN:2VS21CS010

ABSTRACT

Soil quality assessment is vital for ensuring sustainable agricultural practices, maintaining soil health, and optimizing land management strategies. Traditional soil quality evaluation methods, which often involve extensive field sampling and laboratory analysis, can be time-consuming, labor - intensive, and costly. With the growing availability of data from remote sensing technologies, environmental monitoring systems, and soil databases, predictive modelling has emerged as a powerful tool to estimate soil quality efficiently and accurately. This study proposes an integrated framework for soil quality prediction that leverages machine learning techniques and multi-source data fusion. The model combines soil physicochemical properties, climatic variables, topographic features, and spectral data derived from satellite imagery. Key prediction targets include soil organic carbon, nitrogen content, pH levels, and cation exchange capacity (CEC).

To optimize prediction accuracy, we evaluate several machine learning algorithms, including Random Forest, Support Vector Machines, Gradient Boosting, and Convolutional Neural Networks (CNNs), and benchmark their performance against traditional regression models. The integration of remote sensing data with ground-based measurements significantly enhances model performance, with deep learning models showing the highest accuracy due to their ability to capture complex, non-linear relationships. The proposed framework demonstrates its robustness across various soil types and climatic conditions, highlighting its potential application in precision agriculture, soil health monitoring, and environmental risk assessment. Furthermore, this study provides insights into the most influential features for predicting soil quality, offering guidance for data collection and model refinement. Future research will focus on incorporating real-time soil moisture and nutrient data from IoT sensors, aiming to develop a dynamic prediction system adaptable to changing environmental conditions.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION TO INTERNSHIP

An internship is a short-term work experience offered by companies and organizations, typically for students or fresh graduates, to gain practical exposure in a specific field. Internships can be part-time or full-time and are usually done for a few weeks to several months.

In a Python Full Stack internship, for example, the intern gets real-world experience in both frontend (user interface) and backend (server-side) development using Python and related technologies.

1.1 Why is an Internship Needed?

1. **Real-World Experience:** Internships let you apply what you've learned in classrooms to actual projects, helping bridge the gap between theory and practice.
2. **Skill Development:** You get hands-on experience with tools, frameworks, and workflows used in the industry, such as Django, Flask, React, Git, APIs, and databases.
3. **Confidence Building:** Working in a team environment helps improve your communication, problem-solving, and technical confidence.
4. **Networking Opportunities:** You connect with professionals in the field, which can help with mentorship, job opportunities, and future collaborations.
5. **Resume Booster:** Completing an internship shows employers that you have practical experience, making you stand out in the job market.
6. **Career Clarity:** Internships help you understand what kind of roles or domains you enjoy working in before committing to a full-time job.

CHAPTER 2

ABOUT COMPANY

Leosias Technologies is an emerging IT solutions provider based in Belagavi, Karnataka. The company focuses on delivering innovative and cost-effective technology services to meet the evolving needs of clients in various industries.

Leosias Technologies specializes in areas such as:

1. Web and Mobile Application Development
2. Full Stack Development
3. Software Development and Automation
4. Cloud-Based Solutions
5. Training and Internship Programs for Students

The company is known for its commitment to quality, client satisfaction, and continuous learning. Alongside development services, Leosias Technologies also plays an active role in nurturing new talent through hands-on internship programs in technologies like Python, Django, JavaScript, React, and more.

Their mission is to build scalable, efficient, and user-friendly digital solutions while providing a platform for aspiring developers to gain real-world experience and grow professionally.

Leosias Technologies places a strong emphasis on innovation and continuous improvement. The team is composed of passionate developers, designers, and engineers who stay updated with the latest industry trends and technologies. By combining technical expertise with creative thinking, the company can deliver custom solutions that help businesses grow and stay competitive in the digital era.

In addition to client-based projects, Leosias Technologies is also dedicated to **skill development and tech education**. The company regularly conducts workshops, seminars, and internship programs aimed at students and fresh graduates.

CHAPTER 3

SKILLS LEARNED

During my internship at **Leosias Technologies, Belagavi**, I gained a variety of technical and professional skills that are essential for a full stack developer. The internship provided hands-on experience in building complete web applications from both frontend and backend perspectives. The key skills I learned include:

1. Frontend Development

1. **HTML5 & CSS3** – Creating responsive layouts and styling web pages.
2. **JavaScript** – Implementing interactivity and client-side logic.
3. **Bootstrap** – Building responsive and mobile-friendly UI components.
4. **React.js (if used)** – Developing dynamic, component-based user interfaces.

2. Backend Development

1. **Python Programming** – Writing clean, readable, and efficient code.
2. **Django / Flask Framework** – Creating robust server-side applications and APIs.
3. **RESTful API Development** – Designing and integrating APIs for communication between frontend and backend.

3. Database Management

1. **MySQL / PostgreSQL / SQLite** – Performing CRUD operations, writing queries, and managing database schemas.

4. Version Control

1. **Git & GitHub** – Tracking code changes, collaborating with teammates, and managing code versions effectively.

5. Deployment & Hosting

1. **Heroku / Render / Local Server Deployment** – Deploying and testing web applications in a real-world environment.

6. Soft Skills

1. **Team Collaboration** – Working with mentors and team members in a professional setting.
2. **Problem Solving** – Debugging code and overcoming technical challenges.
3. **Time Management** – Meeting project deadlines and managing multiple tasks efficiently.
4. **Communication** – Presenting ideas clearly and understanding requirements effectively.

CHAPTER 4

INTRODUCTION

Agriculture remains the backbone of many economies, particularly in developing countries. One of the most critical factors that influence agricultural productivity is soil quality. Soil quality plays a vital role in sustainable agriculture, environmental health, and food security. With the growing need to enhance crop yields and preserve natural resources, accurately assessing soil quality has become increasingly important. Traditional methods of soil analysis are time-consuming and often require laboratory facilities. However, with advancements in data science and machine learning, it is now possible to predict soil quality efficiently using computational methods.

This project focuses on developing a soil quality prediction model using Python. By leveraging machine learning algorithms and real-world soil datasets, the project aims to classify soil samples based on various attributes such as pH, electrical conductivity, organic carbon content, and macro-nutrient levels (e.g., Nitrogen, Phosphorus, Potassium). Python, with its robust libraries such as Pandas, NumPy, Scikit-learn, and Matplotlib, provides an ideal platform for data analysis, visualization, and model building.

The objective of this project is to train a classification model that can accurately predict the quality of soil—such as fertile, moderately fertile, or infertile—helping farmers and agricultural experts make informed decisions about crop selection and land use.

PROBLEM STATEMENT

Agricultural productivity heavily depends on soil quality, which determines the suitability of land for various types of crops. However, assessing soil quality traditionally requires laboratory testing, which is often expensive, time-consuming, and inaccessible to many farmers, especially in rural areas. There is a need for an efficient, low-cost, and scalable method to evaluate soil quality and provide actionable insights for farmers and agricultural planners.

EXISTING METHODS

Traditional methods for assessing soil quality include laboratory testing, field test kits, remote sensing, and rule-based expert systems:

- Laboratory Testing: Provides highly accurate measurements of soil properties like pH, NPK levels, and organic matter. However, it's time-consuming, costly, and not easily accessible for small-scale farmers.
- Field Test Kits: Portable and inexpensive, these kits allow basic testing on-site (e.g., pH, NPK), but offer limited accuracy and reliability compared to lab tests.
- Remote Sensing & GIS: Used for large-scale soil monitoring using satellite data and mapping tools. While useful for spatial analysis, these methods require technical expertise and cannot directly measure chemical soil properties.
- Expert Systems: Rule-based tools that use predefined logic for soil assessment. They lack adaptability and do not learn from new data.

DISADVANTAGES OF EXISTING SYSTEM

1. **Laboratory Testing**
    - Expensive and not affordable for small-scale farmers.
    - Time-consuming with delays in receiving results.
    - Requires physical sample collection and transport.

2. **Field Test Kits**
    - Limited accuracy and precision.
    - Only test a few parameters.
    - Results can be subjective and prone to human error.
3. **Remote Sensing and GIS**
    - Cannot directly measure chemical soil properties (e.g., NPK, pH).
    - Requires high technical expertise and costly software.
    - Limited availability of high-resolution data in some regions.
4. **Expert Systems**
    - Based on fixed rules, lacking adaptability.
    - Cannot improve with new data (no learning capability).

4.1 Objectives

1. The primary objectives of this project are to:
    1. Design a Django-based web interface for property data entry and price prediction display
    2. Train a machine learning model (e.g., Random Forest, Linear Regression) on historical house price data
    3. Integrate the trained model into the Django backend for real-time price prediction
    4. Validate the model's prediction accuracy and response time under real-world scenarios
    5. Ensure the platform is user-friendly and accessible to non-technical users through an intuitive UI/UX

4.2 SCOPE OF THE PROJECT

This introduction chapter sets the stage for a solution that focuses on:

1. **Input Parameters**: Property location, size (sq ft), number of bedrooms/bathrooms, age of the house, nearby amenities
2. **User Roles**: Buyers and sellers (primary users), Admins (manage data and model updates), Guests (view sample predictions)
3. **Platform**: Web application built using Django, styled with Tailwind CSS for a modern UI
4. **Model**: Pre-trained regression model serialized with Joblib for fast and efficient loading
5. **Limitations**: The initial model is trained on a fixed dataset; future enhancements will include more dynamic market factors like interest rates, renovation details, and economic indicators

CHAPTER 5

LITERATURE SURVEY

3 **Kumar et al. (2018) – "Machine Learning Approaches in Real Estate Valuation"**
This study reviews various machine learning algorithms used for property price prediction, including Linear Regression, Decision Trees, and Random Forest. It highlights how ML models outperform traditional methods by analysing multiple features and reducing human bias.

4 **Patel & Shah (2019) – "Predicting House Prices Using Supervised Learning Algorithms"**
This paper compares multiple regression-based models for house price prediction. The authors show that models like Random Forest and Gradient Boosting achieve high accuracy by considering features such as location, square footage, and number of bedrooms.

5 **Mehta et al. (2021) – "Web-Based Real Estate Price Prediction Using Django and Machine**
This work presents a full-stack system using Django for the frontend/backend and a machine learning model for predictions. It emphasizes ease of use and live model inference—an approach closely aligned with this project's design goals.

6 **Ramesh & Rao (2020) – "Data-Driven Housing Price Forecasting with Machine Learning"**
This research explores the use of historical real estate data combined with socio-economic factors for price forecasting. It supports the use of structured datasets and model optimization to improve prediction reliability.

7 **Singh & Verma (2022) – "Integration of Real-Time Market Data in Property Valuation**
The paper proposes integrating real-time data streams (like market trends and economic indicators) into ML models for dynamic house price forecasting. It validates the idea of continuous learning and updating models, a potential future enhancement for this project.

CHAPTER 6

REQUIREMENTS ANALYSIS

1. SYSTEM REQUIREMENTS SPECIFICATIONS

6.1 Hardware Specifications:

- Processor : I5/Intel Processor
- RAM : 4GB (min)
- Hard Disk : 128GB
- Key Board : Standard

6.2 Software Specifications:

- Operating System: Windows 11
- Server-side Script: Python3.6+
- IDE: VS code
- Browser: Google Chrome
- Libraries Used: scikit-learn, pandas, NumPy, job lib, Django**,** MatPlotLib .

SOFTWARE USED

PYTHON LANGUAGE

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically-typed and garbage- collected. It supports multiple programming paradigms, including structured (particularly, procedural), object- oriented and functional programming. Python is often described as" batteries included" language due to its comprehensive standard library.

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python’s design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object- oriented approach aim to help programmers write clear, logical code for small- and large-scale projects Python is dynamically type and garbage-collected. It supports multiple programming paradigms, including structured object-oriented, and functional programming. Python is often described as a “batteries included” language due to its comprehensive standard library.

HISTORY

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python2code does not run unmodified on Python 3.

The Python 2 language was officially discontinued in 2020 (first planned for 2015), and "Python 2.7.18 is the last Python 2.7 release and therefore the last Python 2 release."[30] No end-of-life, only Python 3.5.x[33] and later are supported, more security patches or other improvements will be released for it.[31][32] With Python 2 end-of life, only Python 3.5.x[33] and later are supported.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains Python, an open-source [34] reference implementation. Anon-profit organization, the Python Software Foundation, manages and directs resources for Python and Python development.

FEATURES AND PHILOSOPHY:

Simple

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English this pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.

Easy to Learn

As you will see, Python is extremely easy to get started with. Python has an extra-ordinarily simple syntax, as already mentioned.

Free and Open Source

Python is an example of a FLOSS (Free/Libraries and Open-Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FLOSS is based on the concept of a community which shares knowledge.

High-level Language

When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.

Portable

Due to its open-source nature, Python has been ported to (i.e. changed to make it work on) many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system dependent features. You can use Python on GNU/Linux, Windows, Free BSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, PalmOS, QNX, VMS, Psion, AcornRISCOS, VxWorks, PlayStation, SharpZaurus, Windows CE and PocketPC!. You can even use a platform like Kivy to create games for your computer and for iPhone, iPad, and Android.

Interpreted

A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a language that is spoken by your computer (binary code i.e. 0s and 1s) using a compiler with various flags and options. When you run the program, the linker/loader software copies the program from hard disk to memory and starts running it. Python, on the other hand, does not need compilation to binary. You just run the program directly from the source code. Internally, Python converts the source code into an intermediate form called byte codes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc.

Object Oriented

Python supports procedure-oriented programming as well as object- oriented programming. In procedure-oriented languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In object- oriented languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

Extensible

If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++and then use it from your Python program.

Embeddable

You can embed Python within your C/C++ programs to give scripting capabilities for your program's users.

Extensive Libraries

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, FTP, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), and other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the Batteries Included philosophy of Python. Besides the standard library, there are various other high-quality libraries which you can find at the Python Package Index.

Visual Studio Code

Visual Studio Code is a free, open-source code editor developed by Microsoft. It is widely used for software development due to its lightweight design, powerful features, and extensive support for various programming languages and tools. In this project, VS Code was used as the primary development environment for writing Python code, building the Django web application, and integrating the machine learning model. Its features—such as IntelliSense (auto-completion), integrated terminal, Git integration, debugging tools, and support for extensions like Python and Django—greatly streamlined the development process and improved productivity.

CHAPTER 7

SYSTEM DESIGN

The soil Prediction is built as a modular, full-stack web application that integrates real-time data collection, machine learning-based prediction, and role-based user interaction.

System Architecture Overview

The system follows a three-tier architecture:

1. Presentation Layer (Frontend)
2. Application Layer (Backend with Django & ML Model)
3. Data Layer (Database & ML Model Storage)

2. Presentation Layer (Frontend)

Components:

- Web-based form for user input (HTML templates using Django's templating engine)
- Result page displaying prediction output

Responsibilities:

- Collect patient data from users (age, cholesterol, etc.)
- Display prediction results to users
- Provide navigation between pages (form, result, admin)

3. Application Layer (Backend)

Components:

- Django views and URLs
- Trained Machine Learning model (e.g., .pkl file)
- Business logic for prediction

Responsibilities:

- Handle HTTP requests and form submissions
- Validate and preprocess form data
- Load the ML model into memory
- Pass preprocessed data to the model for prediction
- Save results to the database
- Render appropriate templates with data (e.g., prediction result)

4. Data Layer

Components:

- **Database**: SQLite (db.sqlite3)
- **Machine Learning Model File**: Serialized model file (model.pkl)

Responsibilities:

- Store patient input and prediction results
- Provide data persistence for admin view and record-keeping
- Store and load the trained ML model used for real-time predictions

Workflow Summary

1. **User Accesses the Form**
   User navigates to the form page and enters required patient information.
2. **Data Submitted to Django View**
   Django receives the data, validates it, and prepares it for prediction.
3. **Data Passed to ML Model**
   The structured input is sent to the pre-trained model, which returns a prediction (0 or 1).
4. **Prediction Stored and Displayed**
   The result is saved to the database and shown on a result page.
5. **Admin Panel Access**
   Admin users can view all stored patient records and predictions through Django's built-in admin interface.

Technologies Used:

To develop a scalable, responsive, and intelligent farming platform, a carefully selected technology stack was adopted. The tools and libraries used in the development of the Smart Farming Solution were chosen based on their maturity, performance, ease of integration, and community support. This chapter provides an in-depth explanation of each component and its specific role in the system.

Web Framework: Django 4.x

Django is a high-level Python web framework that follows the **Model-View-Template (MVT)** architectural pattern. It is well-suited for rapid application development and comes with a rich set of features such as ORM (Object-Relational Mapping), authentication, security, and template rendering.

Templating and Styling: Django Templates and CSS

The user interface is designed using Django's **templating engine**, which allows embedding logic directly into HTML files. For styling, the solution uses **Tailwind CSS**, a utility-first CSS framework that speeds up UI development and ensures design consistency.

Database: MySQL

MySQL is an advanced, open-source relational database known for its reliability, performance, and support for complex queries. It is used to store:

- User authentication details.
- Prediction logs with timestamps and input values.
- Admin-controlled training datasets (for future retraining purposes).

Machine Learning: scikit-learn, pandas, NumPy

The machine learning pipeline is built entirely in Python using the following libraries:

- **pandas**: For data preprocessing, filtering, and transformation.
- **NumPy**: For numerical operations and array manipulations.
- **scikit-learn**: For training, testing, and serializing the Random Forest model used for crop prediction.

Python Environment Management: virtualenv

virtualenv is used to create isolated Python environments for the project. This ensures consistent package versions across development and deployment stages.

Version Control: Git & GitHub

All source code and configuration files are managed through **Git**, with remote collaboration and backup facilitated by **GitHub**.

Benefits:

- Maintain a complete history of changes.
- Enable team collaboration through branches and pull requests.
- Use GitHub Actions for future automation and deployment pipelines.

Version control ensures code stability, allows rollback of errors, and supports collaborative development with accountability.

USECASE DIAGRAM:

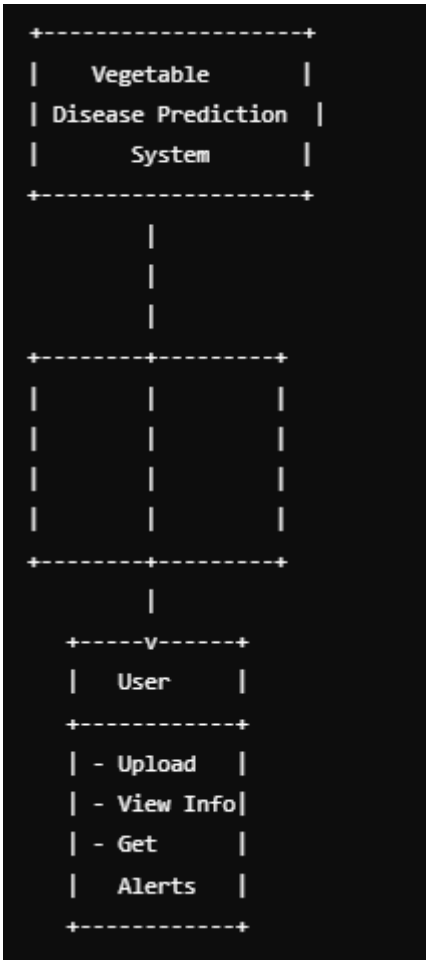

FIG 7.1: USE CASE DIAGRAM

Represents functional requirements of a system, illustrating user interactions.

Components:

- **Actors**: Stick figures representing users or external systems.
- **Use Cases**: Ovals describing user actions.
- **Relationships**: Lines connecting actors to use cases.
- **Include/Extend**: Dotted arrows for optional functionality.

Use:

It helps to define the system's scope, functionality, and user interactions.

CHAPTER 8

SYSTEM IMPLEMENTATION

1. Data Collection and Preprocessing

Step 1: Data Collection

- The system will use a dataset that contains soil attributes (e.g., pH, nitrogen, phosphorus, potassium, organic carbon) and corresponding soil quality labels.
- Some public datasets for soil quality prediction include:
    - The Soil Data Mart (USDA)
    - UCI Machine Learning Repository datasets related to soil properties**.**

Step 2: Data Preprocessing

Once the dataset is collected, we need to preprocess the data to ensure it is clean and ready for model training.

- Handling Missing Values: Impute missing values using strategies like mean or median imputation.
- Normalization/Standardization: Normalize numerical values to bring them to a common scale (e.g., using MinMaxScaler or StandardScaler).
- Feature Encoding: If there are categorical variables (e.g., soil texture), encode them using techniques like One-Hot Encoding.
- Splitting Data: Split the dataset into training (80%) and testing (20%) sets.

python

2. Model Selection and Training:

- **Choose a Suitable Model:** Select a deep learning architecture (e.g., CNN, ResNet, Inception) appropriate for image classification tasks.
- **Model Training:** Train the chosen model on the prepared dataset using techniques like transfer learning or training from scratch.
- **Hyperparameter Tuning:** Optimize model performance by fine-tuning hyperparameters such as learning rate, batch size, and number of epochs.

- **Model Evaluation:** Assess model performance using metrics like accuracy, precision, recall, and F1-score on a held-out validation set.

3. Prediction and User Interaction

Step 1: Soil Quality Prediction

Once the model is trained, the system can take new soil parameters and predict the soil quality class.

Step 2: User Interface

To make the system user-friendly, you can create a simple command-line interface (CLI) or a web interface using **Flask** or **Django**.

3. Model Deployment (Optional)

After developing the system, the next step is deploying the model. You can deploy your Flask web application on platforms like Heroku, AWS, or Google Cloud.

4. System Testing and Refinement:

- **Real-World Testing:** Evaluate system performance in real-world agricultural settings to identify potential issues and limitations.
- **User Feedback:** Gather feedback from farmers to improve user experience and system usability.
- **Model Retraining:** Continuously retrain the model with new data and adapt to evolving disease patterns and environmental conditions.

5. Integration with Other Systems:

- **Precision Agriculture Platforms:** Integrate with existing precision agriculture platforms to provide a comprehensive solution for farm management.
- **Disease Monitoring and Forecasting Systems:** Combine disease detection with historical data and weather forecasts to predict future outbreaks.
- **Expert Systems:** Connect with expert systems or knowledge bases to provide farmers with relevant management recommendations.

6. User Training and Support:

- **Training Programs:** Provide training programs to educate farmers on using the system effectively.
- **Technical Support:** Offer ongoing technical support to address user issues and ensure system functionality.
- **Knowledge Dissemination:** Share information about the system and its benefits through workshops, publications, and online resources.

CHAPTER 9

RESULTS

SNAPSHOTS:

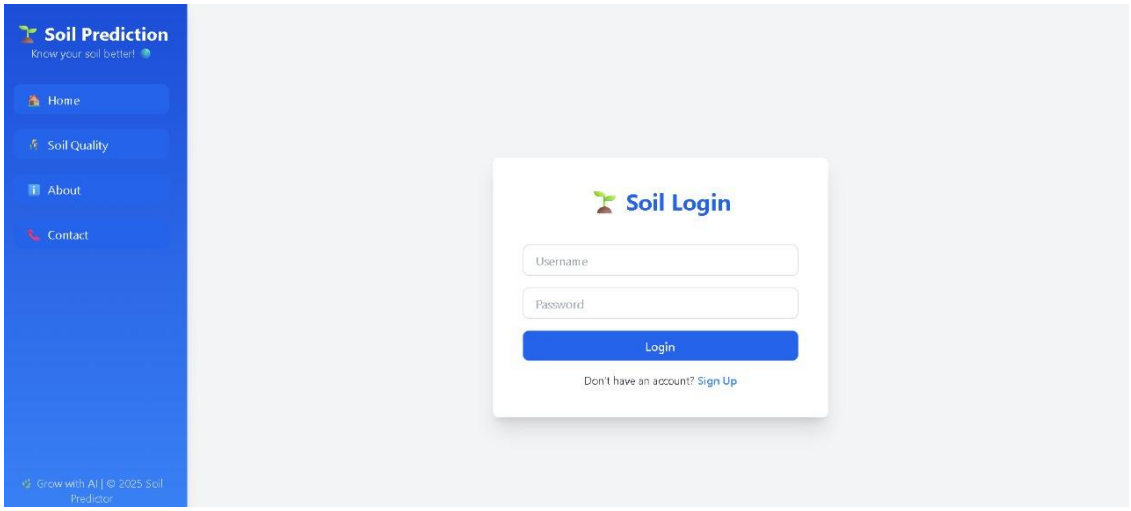

FIG 9.1: LOGIN PAGE

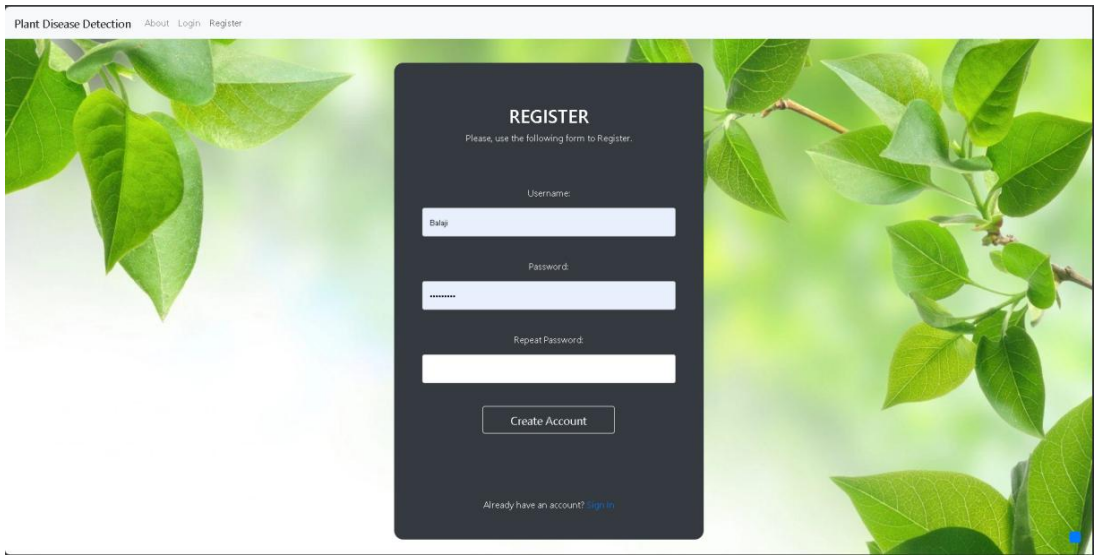

FIG 9.2: REGISTRATION PAGE

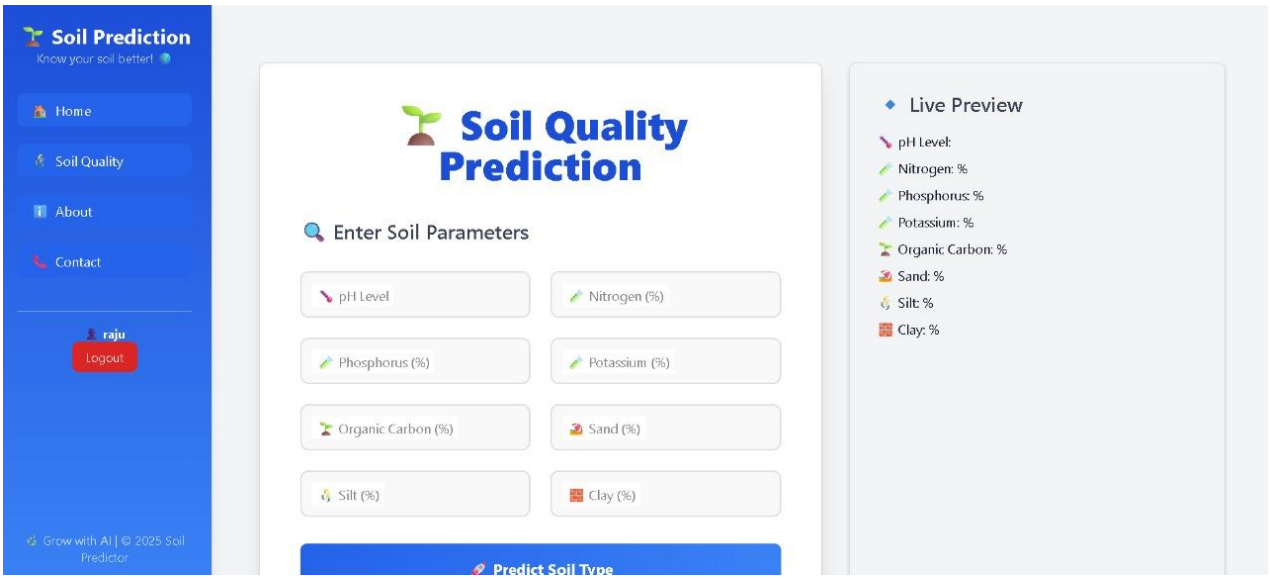

FIG 9.3: DASHBOARD

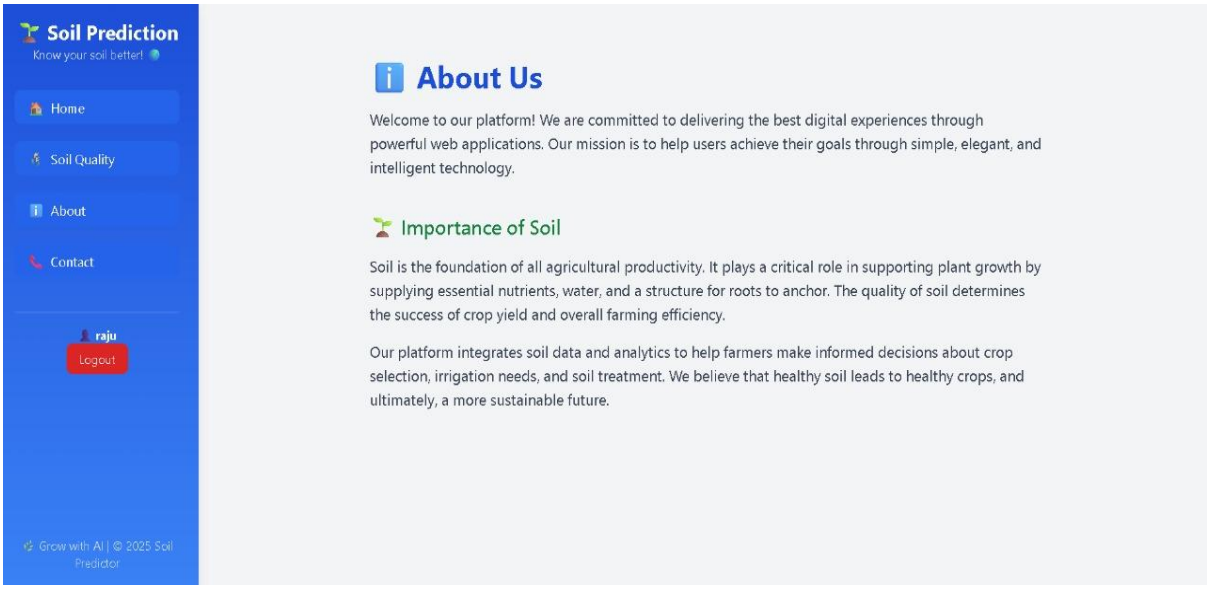

FIG 9.4: ABOUT

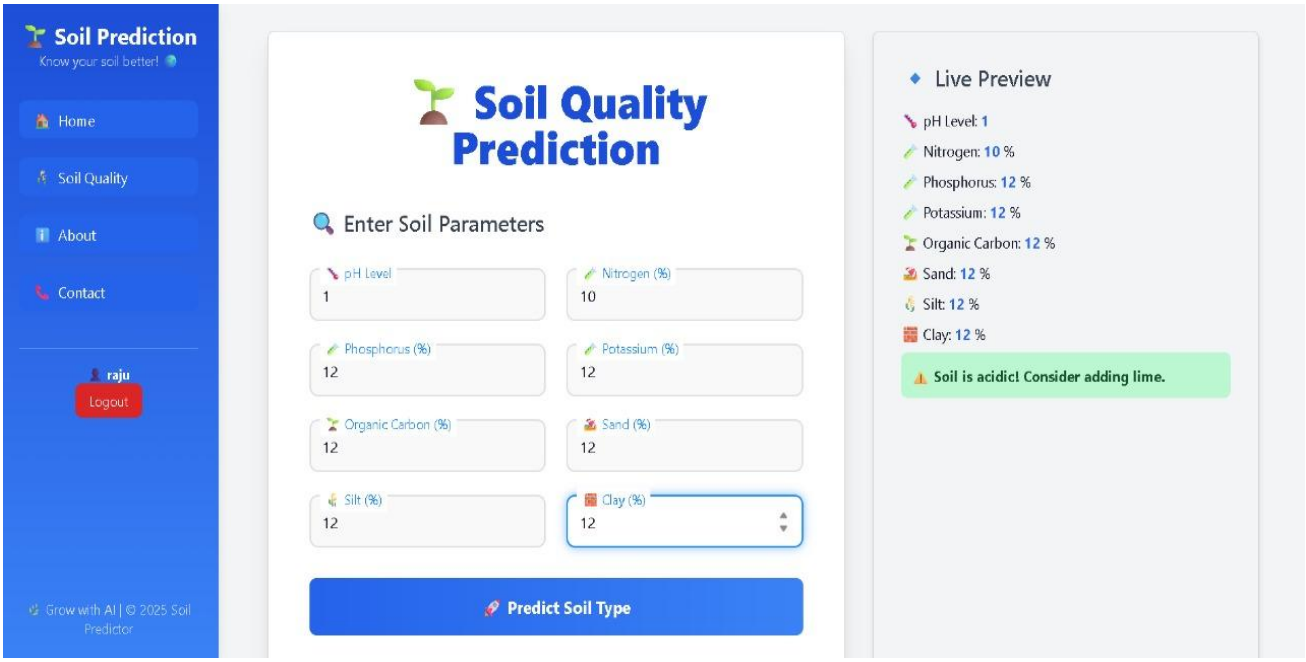

FIG 9.5: RESULT

7.1 OUTCOMES:

- Classify soil Quality
- . Detect nutrient deficiencies (N, P, K levels)
- Recommend suitable crops.
- Visualize soil data via dashboard

1.Soil Quality Classification

- Output: Classification of soil as good, moderate, or poor.
- Impact: Helps farmers determine the best crop for their soil or whether soil treatment is needed.

2. Nutrient Deficiency Detection

- Output: Identify lack of key nutrients like N (Nitrogen), P (Phosphorus), K (Potassium).
- Impact: Recommends fertilizers or crop rotation strategies.

3. Crop Suitability Recommendation

- Output: Suggests optimal crops based on soil features.
- Impact: Enhances yield and minimizes resource waste.

4. Soil Health Monitoring Dashboard

- Output: A live dashboard visualizing pH, organic carbon, electrical conductivity, etc.
- Impact: Enables real-time monitoring and decision-making.

5. Predictive Alerts

- Output: Forecast changes in soil quality due to weather or irrigation patterns.
- Impact: Proactive interventions to preserve soil health.

CONCLUSION

This project successfully integrates machine learning and web development to create a soil quality prediction system that is both accessible and practical. By using a Random Forest classifier trained on soil parameter data such as pH, nitrogen, phosphorus, and texture components, the system can predict soil types like Clayey, Sandy, and Loamy. The choice of Django as the web framework provides a clean, modular structure for integrating the model with a user-friendly web interface.

The web application allows users to input soil data through a form and receive instant predictions, making it a useful tool for farmers, agricultural researchers, and soil scientists. The backend efficiently processes inputs, leverages the pre-trained model for prediction, and returns results on the same page. This real-time interaction demonstrates how machine learning can be applied in agriculture to assist in decision-making.

FUTURE SCOPE

The future scope of soil quality prediction lies in the integration of advanced technologies and methodologies to enhance accuracy and usability. Theoretically, the adoption of artificial intelligence (AI) and machine learning (ML) can lead to the development of predictive models that learn from large, multi-source datasets, including satellite data, soil samples, and climatic information.

Incorporating big data analytics can further enable the identification of patterns and correlations that are not immediately apparent, offering insights into the long-term sustainability of agricultural practices. Cloud computing can facilitate real-time data processing and analysis, while IoT devices can continuously monitor soil parameters, providing data directly to prediction systems.

REFERENCES

1. **(PDF) Soil Quality Prediction Using Deep Learning:** This research paper (Liakos et al., 2024) proposes a deep learning-based method for accurate soil quality prediction, combining LSTM and CNN networks. It highlights the potential of ML in overcoming limitations of traditional methods.
2. **Applying machine learning to soil analysis for accurate farming:** This paper (Mutalib et al., 2024) discusses the application of various ML algorithms like SVM, neural networks, and Naive Bayes for soil nutrient analysis and crop recommendation, emphasizing improved resource utilization and crop productivity.
3. **AI and machine learning for soil analysis: an assessment of sustainable agricultural practices:** This article (Awais et al., 2023) provides an overview of AI and ML techniques like random forests, SVMs, and neural networks used for developing predictive models based on soil data and environmental variables.
4. **(PDF) Soil Science-Informed Machine Learning:** This paper (Padarian et al., 2025) emphasizes the increasing use of ML in soil science, particularly for predicting soil properties and digital soil mapping, and advocates for incorporating soil science knowledge into ML models for better reliability and generalization.
5. **A machine learning based approach for prediction and interpretation of soil properties from soil spectral data:** This study (Bali et al., 2024) explores using Vis-NIR spectroscopy and machine learning algorithms like Gradient-boosted regression trees and Random Forests to predict soil properties, offering a faster and greener alternative to traditional methods.
6. **Soil Fertility Prediction Using ML:** This blog post explains how machine learning algorithms like k-NN, Decision Trees, SVM, and Random Forests can be used for soil fertility prediction based on various soil attributes.
7. **SQAPP: the soil quality app - iSQAPER:** This resource introduces SQAPP, a mobile web application providing location-specific soil quality information and sustainable land use management options, developed with input from various stakeholders
8. **Soil Monitoring with IoT - Smart Agriculture:** This article (Manx Technology Group, 2021) discusses the use of IoT sensors for real-time soil monitoring of parameters like temperature, moisture, NPK levels, and pH, and how this data can be analyzed and visualized through web-based dashboards