

FYP Title: Kaasht



Program Name: Bachelors of Science in Software
Engineering

Session: Session (2021 - 2025)

Authors Name and Roll Number: Submitted By

Ali Sher Khan	10208
Mohsin Waseem	10222

Supervisor

Mr. Mir Jamal-ud-din.

Department Name: Department of Computer Science
Abbottabad University of Science & Technology

Approval Sheet

DEDICATION

In the name of Allah, the Most Gracious, the Most Merciful.

Our way has been enlightened by Allah, the Lord of all worlds, whose boundless mercy, infinite knowledge, and innumerable blessings are worthy of all praise. We have accomplished this important academic milestone because of His will, direction, and power. Above all, this work is dedicated to Him, my Allah , who is our Creator, our Sustainer, the fount of all wisdom, and the source of all achievement. We also dedicate our thesis with great appreciation and respect to our parents, whose prayers, encouragement, and unshakable faith in us have been the cornerstone of our development and willpower. me when I was feeling down, and for having faith in my talents even when I doubted them.

Ali Sher Khan

Mohsin Waseem

ACKNOWLEDGEMENT

Special thanks to our project supervisor Mir Jamal-id-din whose support and guidance played an important role in our achievements. We are also thankful to him for his support to us and also offered us help to make this project successful. His guidance and advice, both mentally and technically has been of great importance to the final outcome of this final thesis. We sincerely appreciate his time and efforts.

PREFACE

This project thesis concerns with the development of “**Kaasht**”. This Project thesis covers the various phases in the development of this project, which are as follow:

CHAPTER 1. INTRODUCTION

This chapter describes a brief introduction about “**Kaasht**” and also about the tools used for the development of this project.

CHAPTER 2. EXISTING SYSTEM

This chapter describes the background (Existing system) and also limitation of “**Existing system**”.

CHAPTER 3. PROPOSED SYSTEM

This chapter describes the proposed “**Kaasht**” and its features.

CHAPTER 4. SYSTEM DESIGN

This chapter describes the system’s design of “**Kaasht**” design phase (architecture) and also about the user interface of the system.

CHAPTER 5. SYSTEM TESTING

This chapter describes the testing phase of the proposed “**Kaasht**”.

CHAPTER 6. CONCLUSION & FUTURE WORK

This chapter describes the conclusion and future work of the proposed “**Kaasht**”.

ABSTRACT

This project thesis concerns with the development of “**Kaasht**”. This Project thesis covers the various phases in the development of this project, which are as follow:

The Kaasht mobile application is developed to revolutionize traditional farming by integrating modern technological advancements such as artificial intelligence and machine learning into agriculture. The primary function of the application is to assist farmers and agricultural professionals in making data-informed decisions by analyzing real-time weather and soil data. By collecting information such as temperature, humidity, rainfall, and critical soil parameters like pH, nitrogen, phosphorus, and potassium levels, the application accurately predicts the most suitable crops for cultivation in a specific region. This helps in enhancing yield, minimizing resource wastage, and fostering sustainable farming practices.

The application leverages machine learning models trained on historical agricultural data to provide crop suitability predictions. It uses real-time inputs from weather APIs and soil sensors to offer dynamic recommendations tailored to the environmental conditions of the user’s location. Users can interact with the app through an intuitive interface, enabling them to view weather forecasts, monitor soil health, export data for analysis, and access detailed reports on crop recommendations. Additionally, the app stores prediction history to help farmers track trends over time and refine future decisions based on past outcomes. Kaasht is designed with a specific focus on the agricultural landscape of Punjab, Pakistan, ensuring that its functionalities and predictions are aligned with local farming practices and climatic conditions. The application is intended not only for farmers but also for agricultural experts, researchers, and government or non-government organizations working in the agriculture sector. By bridging the gap between conventional agricultural knowledge and digital technology, Kaasht aims to improve food security, promote economic growth, and support the adoption of environmentally conscious farming methods.

Table of Contents

Abstract	i
Dedication	ii
Acknowledgment.....	iii
Preface.....	iv
1. Introduction.....	1
1.1 Purpose	4
1.2 Scope of the Project	5
1.3 Definitions, Acronyms, Abbreviations	6
2. Existing System	10
2.1 Literature Review	16
2.2 Limitations of the Existing System.....	21
3. Proposed System.....	27
3.1 Overview	32
3.2 Functional Requirements	33
3.3 Non-Functional Requirements	34
3.4 Chosen Methodology.....	35
4. System Design	36
4.1 System Architecture	37
4.2 Data Flow Diagrams (DFDs).....	39
4.3 Entity-Relationship Diagrams (ERD).....	41
4.4 Use Case Diagrams.....	45
4.5 Activity Diagrams.....	40
4.6 Sequence Diagrams.....	49
4.7 Class Diagram	51
5. System Testing.....	61
5.1 Testing Methodology.....	63
5.2 Test Cases and Results.....	64
6. Conclusion & Future Work.....	65
References	67

List of Figures:

Figure 1 Usecase Diagram Admin.....	10
Figure 2 Usecase Diagram of User	12
Figure 3 Gantt Chart	14
Figure 4 Work Break Down Structure.....	15
Figure 5 Agile Methodology	28
Figure 6 Architecture Diagram	31
Figure 7 Class Diagram	33
Figure 8 Activity Diagram For Admin	34
Figure 9 Activity Diagram for user	35
Figure 10 DATA-FLOW-DIAGRAM LVL-0.....	36
Figure 11 DATA-FLOW-DIAGRAM LVL-1.....	37
Figure 12 Sequence Diagram for Admin.....	38
Figure 13Sequence Diagram For User	39
Figure 14 Entity Relationship Diagram.....	40
Figure 29 Results against model prediction.....	48
Figure 32 Training Results	51
Figure 35 Testing Techniques	60

List of Tables:

Table 1 Definition, Acronyms, abbreviation	9
Table 2 Usage Scenario Table For Admin Usecase	11
Table 3 Usage Scenario for Use Usecase	13
Table 4 Literature Review	18
Table 5 Functional Requirements.....	24
Table 6 Data Processing Of KaaSht Dataset	50
Table 7 Feature Importance Analysis.....	53
Table 8 Working of the API With AI Model.....	54
Table 9 Test Case for User Registration and Authentication	62
Table 10 Test Case For User Profile Management.....	62
Table 11 Test Case for Weather Data Fetching.....	63
Table 12 Test Case For Soil Data Integration	63
Table 13 Test Vase For Crop Prediction	63
Table 14 Test Case For Historical Data Management.....	64
Table 15 Test Case For Notification.....	64
Table 16 Test Case For Location Services	64
Table 17 Test Case For Data Synchronization	65
Table 18 Pathway Table for White Box Testing.....	65
Table 19 Test Case Function	66

CHAPTER 1

INTRODUCTION

1.1.Introduction

The Kaasht application is a mobile-based solution developed to support farmers and agricultural experts in making accurate and informed decisions about crop cultivation based on real-time environmental conditions. It is designed specifically for regions like Punjab, Pakistan, where soil and weather conditions are well-documented and play a significant role in agricultural outcomes. The app collects data from two main sources: weather APIs and soil health sensors. Weather data includes real-time information such as temperature, humidity, and rainfall, while soil sensors capture crucial parameters like pH, nitrogen, phosphorus, and potassium levels. By analyzing these inputs through machine learning algorithms trained on historical agricultural data, the app provides tailored recommendations on the most suitable crops to grow under current conditions. The core purpose is to enhance farming productivity, minimize crop failure risks, and encourage sustainable use of natural resources. Kaasht is developed as a standalone Android application that functions without the need for extensive hardware or integration with other systems, except for its connection to weather APIs and compatible soil sensors. The app features secure user authentication, allowing users to manage their profiles, input region-specific preferences, and access personalized predictions. Users can view detailed weather forecasts and analyze how weather trends influence crop suitability. The app also enables monitoring of real-time soil conditions, presenting data in visual formats like charts and graphs for better understanding. This continuous stream of environmental data is processed to generate crop suitability scores and recommendations, including detailed information about each crop's growth requirements, estimated yield, growth duration, and water usage. Additionally, users can export these results and historical records in CSV or PDF format, aiding in documentation, offline access, and sharing with consultants or institutions. A key focus of Kaasht is on user experience and accessibility. The application is designed to be intuitive and user-friendly, catering to small and medium-scale farmers, many of whom may have limited technological experience. The interface simplifies data input, prediction review, and report generation, ensuring smooth navigation through each feature. Multilingual support is provided to accommodate a diverse range of users in Punjab. The app also maintains a history of past predictions, allowing users to track and compare previous outcomes, which helps them identify long-term patterns in weather and soil behavior and evaluate how accurately past recommendations matched actual results. While the application does not currently support real-time weather alerts or push notifications, it can provide occasional reminders or updates related to crop recommendations and environmental changes. Although the current version of Kaasht excludes features such as integration with smart irrigation systems, market price tracking, and advanced forecasting tools, it establishes a strong foundation for potential future enhancements. It operates under certain constraints, such as compatibility with specific soil sensors and limitation to the Android platform. Assumptions made include the accuracy of input data by users and reliable output from the external APIs. Despite these constraints, Kaasht successfully addresses a critical gap in traditional farming by merging data-driven insights with ease of use, promoting efficient resource utilization, reducing wastage, and supporting long-term sustainability in agriculture. The ultimate aim is to empower farmers, agricultural experts, and institutions by equipping them with actionable insights that drive better crop selection decisions and foster a transition toward smarter, technology-enabled agriculture. Overall, Kaasht's purpose is not just to improve individual crop outcomes.

1.2.Purpose

The purpose of the Kaasht application is to serve as an intelligent decision-support tool that helps farmers, agricultural experts, researchers, and relevant organizations make informed crop cultivation decisions using real-time environmental data. It is developed with the goal of addressing the challenges faced by farmers who often rely on traditional knowledge and guesswork to select crops, without access to scientific insights or updated information about weather and soil health. By integrating artificial intelligence and machine learning, the application analyzes key environmental parameters such as temperature, humidity, rainfall, soil pH, nitrogen, phosphorus, and potassium levels to predict which crops are most suitable for cultivation in a specific region. This minimizes the chances of crop failure, improves overall productivity, and promotes more efficient use of land and agricultural inputs. Furthermore, Kaasht contributes to sustainable agricultural development by encouraging the responsible use of resources such as water, fertilizers, and energy. The application enables users to plan their cultivation practices in harmony with natural conditions, reducing environmental stress and supporting long-term soil health. In doing so, it helps reduce the overuse or misuse of resources that often results from uninformed decisions. The application also supports knowledge transfer to farmers who may have limited exposure to digital tools, by simplifying complex data into user-friendly predictions and recommendations. Overall, Kaasht's purpose is not just to improve individual crop outcomes, but to empower a shift towards smart farming, data-backed decision-making, and environmental stewardship in regions where agriculture is a primary source of livelihood.

1.3.Scope of this project

The scope of the Kaasht project includes the design, development, and deployment of a mobile application that operates on Android devices and supports the prediction of crop suitability based on real-time weather and soil conditions. The application integrates with third-party weather APIs to fetch live weather data and with compatible soil sensors to monitor essential soil nutrients and properties. It features secure user authentication, profile management, data input screens, and result presentation modules. Users can view crop predictions, track historical data, monitor changes in weather and soil parameters, and export data in PDF or CSV formats. The scope also includes creating an intuitive user interface suitable for farmers with limited technological expertise, along with multilingual support to enhance accessibility among users in Punjab, Pakistan. However, the project deliberately excludes several advanced or extended features to maintain a focused and manageable scope in its initial release. It does not support iOS or web platforms, and cloud-based data storage or integration with smart irrigation systems is not included. Real-time weather alerts, SMS/email notifications, agricultural market pricing, and supply chain features are also outside the current scope. The application is developed for use within Punjab, and predictions are tailored to the region's environmental data, meaning the system is not configured for other geographic areas. The application assumes that users will input accurate data and maintain proper sensor usage to ensure effective prediction. By clearly defining these limitations, the scope ensures that the project remains realistic and delivers a reliable, efficient, and localized solution to its intended user base, with a strong foundation for potential enhancements in the future.

1.4.Definition, Acronyms, Abbreviation

Table 1 Definition, Acronyms, abbreviation

S no.	Term	Abbreviation	Definition
1.	Application Programming Interface	API	A set of protocols that allows software applications to communicate with each other.
2.	Potential of Hydrogen	pH	A measure of how acidic or basic the soil is.
3.	Artificial Intelligence	AI	Technology that enables machines to simulate human intelligence.
4.	Comma-Separated Values	CSV	A file format used to store data in tabular form separated by commas.
5.	Portable Document Format	PDF	A file format used to preserve document formatting across different platforms.
6.	Short Message Service	SMS	A mobile service for sending text messages between devices.
7.	Internet of Things	IoT	A network of physical devices connected and sharing data over the internet.
8.	Machine Learning	ML	A subset of AI that enables systems to learn and improve from data.
9.	Operating System	OS	System software that manages computer hardware and software resources.
10.	User Interface/User Experience	UI/UX	UI is how the app looks; UX is how users interact and feel using it.
11.	Global Positioning System	GPS	A satellite-based navigation system used for determining geographic location.
12.	Hypertext Transfer Protocol/Secure	HTTP/HTTPS	Protocols used to transfer data over the web, with HTTPS offering encryption.
13.	Bluetooth Low Energy	BLE	A power-efficient version of Bluetooth for short-range communication.
14.	Random Access Memory	RAM	Temporary memory used by devices to store data for active applications.
15.	Gigahertz	GHz	A unit to measure processor speed, representing billions of cycles per second.
16.	Firebase Cloud Messaging	FCM	A tool for sending push notifications and messages across devices.
17.	Secure Sockets Layer / Transport Layer Security	SSL/TLS	Protocols used for securing data transmission over the internet.
18.	Secure Digital Card	SD Card	A portable memory card used for data storage on mobile devices.
19.	Non-Governmental Organization	NGO	An independent organization working on social, humanitarian, or development goals.

1.5.Usecases & Usage Scenario

1.5.1. Usecase Diagram For Admin

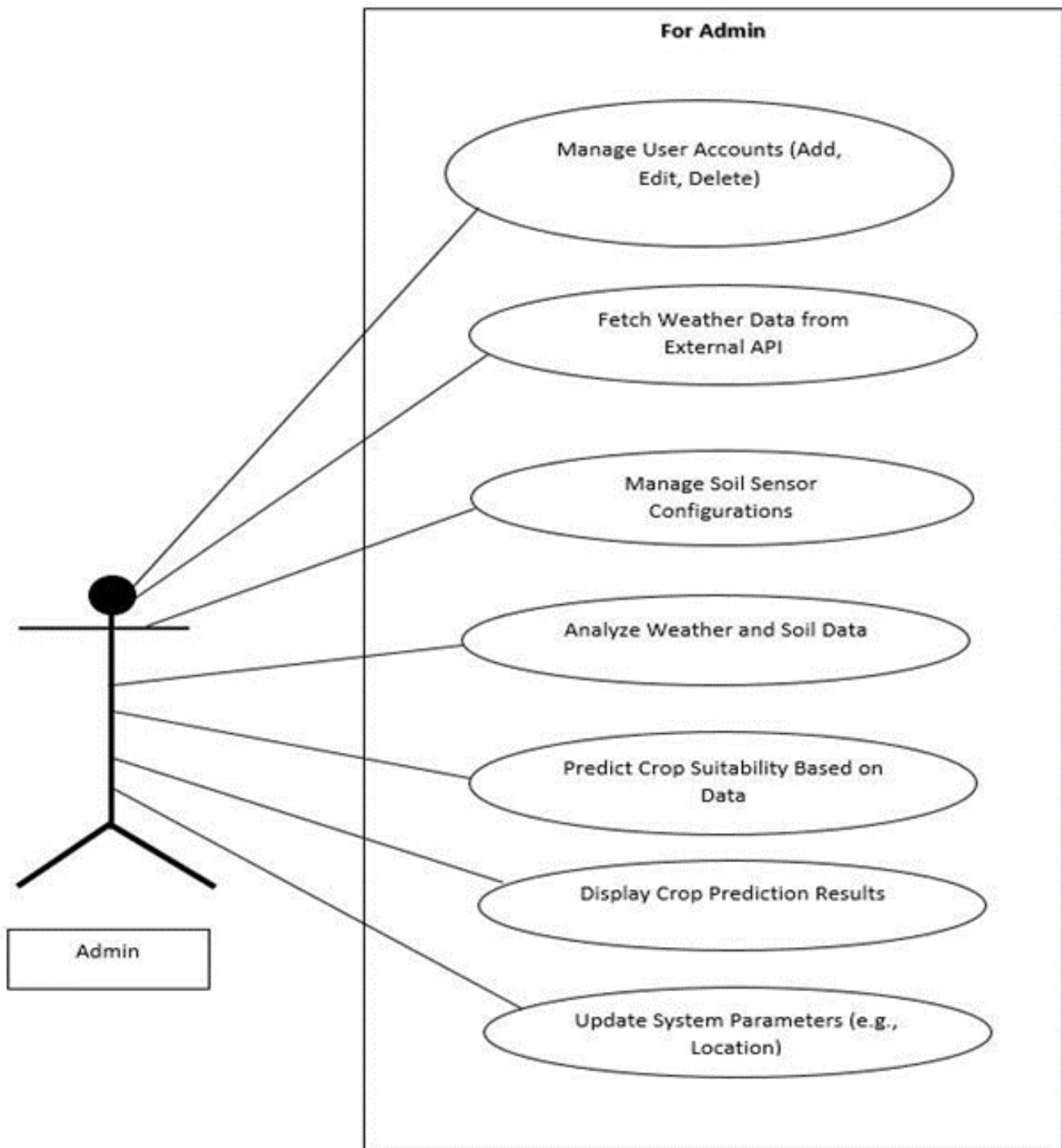


Figure 1 Usecase Diagram Admin

1.5.2. Usage Scenario

Table 2 Usage Scenario Table For Admin Use Case

<ul style="list-style-type: none"> Use Case Title 	<ul style="list-style-type: none"> Manage User Accounts (Add, Edit, Delete)
<ul style="list-style-type: none"> Use Case ID 	<ul style="list-style-type: none"> UC-001
<ul style="list-style-type: none"> Requirement ID 	<ul style="list-style-type: none"> 1.3.1
<ul style="list-style-type: none"> Description: This use case allows the admin to create, update, or delete user accounts to manage system access for farmers and agricultural experts. 	
<ul style="list-style-type: none"> Pre-Condition 	<ul style="list-style-type: none"> Admin must be logged in. User database must be available. System must be operational.
<ul style="list-style-type: none"> Task Sequence 	<ul style="list-style-type: none"> Task Execution
<ul style="list-style-type: none"> Start User Account Management 	<ul style="list-style-type: none"> Admin selects “Manage User Accounts” from the dashboard.
<ul style="list-style-type: none"> Choose Action 	<ul style="list-style-type: none"> Admin chooses to Add, Edit, or Delete a user account.
<ul style="list-style-type: none"> Perform Action 	<ul style="list-style-type: none"> Admin fills in user details or edits/deletes existing ones; system saves changes.
<ul style="list-style-type: none"> Post Condition: User account is successfully created, modified, or removed. 	
<ul style="list-style-type: none"> Unresolved Issues: 	<ul style="list-style-type: none"> None currently identified
<ul style="list-style-type: none"> Authority: Admin, System, 	
<ul style="list-style-type: none"> Modification history: 1.0 Authors: Description: 	

1.5.3 Usecase Diagram for User

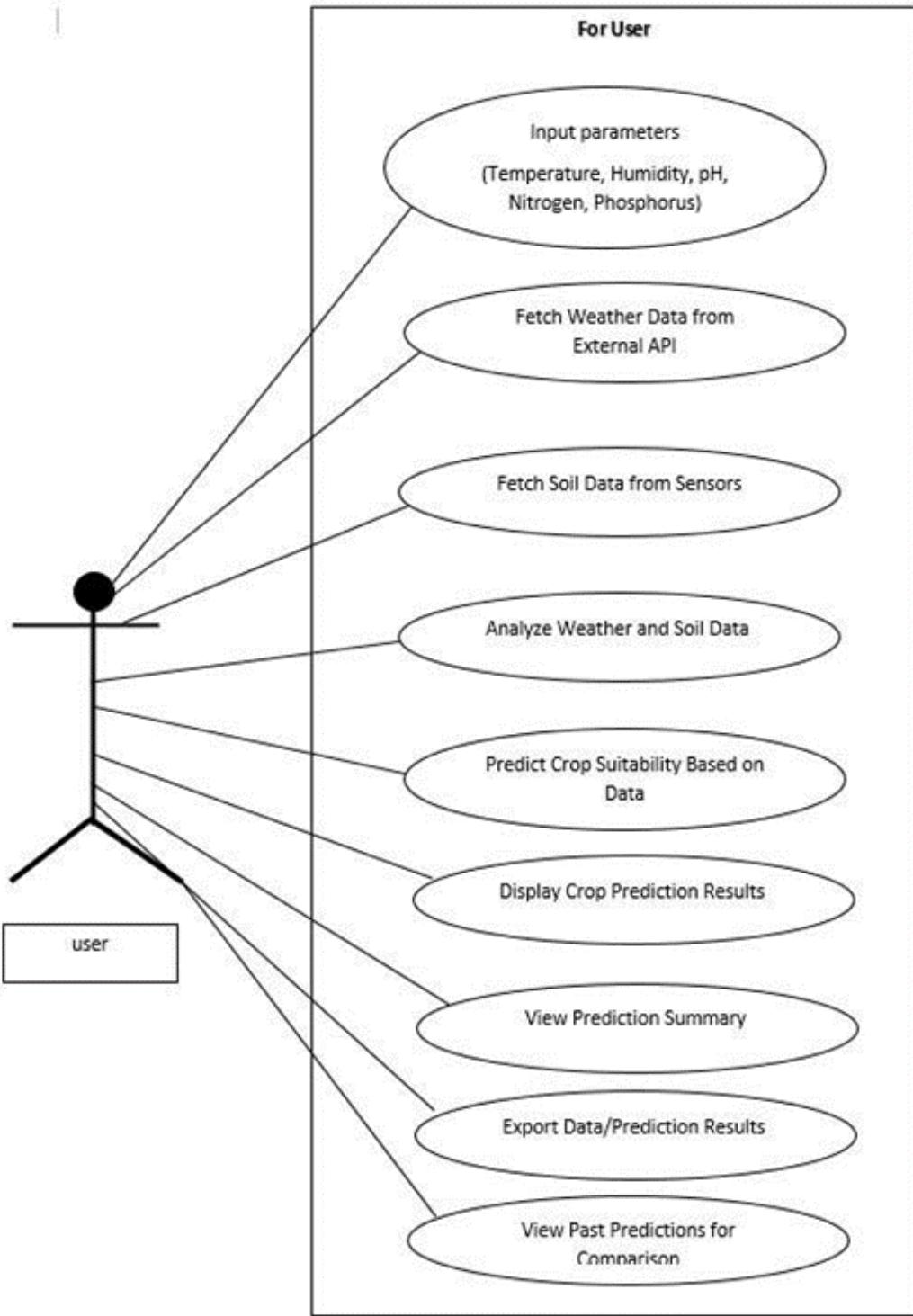


Figure 2 Usecase Diagram of User

1.5.4 Usage Scenario

Table 3 Usage Scenario for Use Usecase

• Use Case Title	• Crop Suitability Prediction
• Use Case ID	• UC-002
• Requirement ID	• 1.3.2
• Description: This use case allows users (farmers) to input environmental data, fetch weather and soil information, and receive crop suitability predictions along with visual summaries.	
• Pre-Condition	
• User must be logged into the system.	
• Soil sensors must be connected and functional.	
• Weather API integration must be active.	
• System must be online.	
• Task Sequence	• Task Execution
• Input Parameters	• User inputs values such as temperature, humidity, pH, nitrogen, and phosphorus.
• Fetch Weather Data	• System fetches real-time weather data using an external API..
• Fetch Soil Data	• System collects data from soil sensors (e.g., pH, nitrogen, phosphorus).
• Analyze Weather and Soil Data	• System processes the gathered weather and soil data for pattern identification and analysis.
• Predict Crop Suitability	• Based on analyzed data, the system uses a trained ML model to predict the most suitable crops.
• Display Crop Prediction Results	• The system displays the list of predicted crops with suitability ratings.
• View Prediction Summary	• The user can see a summarized visual report of predictions.
• Post Condition: The user receives a list of recommended crops based on current weather and soil conditions, with the option to export or compare results.	
• Unresolved Issues:	• Some areas may lack real-time sensor coverage.

1.6.Gantt Chart

The Gantt chart for the Kaasht project played a crucial role in organizing, scheduling, and monitoring each phase of the development lifecycle. The project was divided into clearly defined stages, starting from initial requirement analysis and problem understanding, followed by data collection and preprocessing, model development, API integration, and frontend connectivity. Each task was mapped on a timeline, specifying its estimated duration and expected start and end dates. Dependencies between tasks, such as model training being contingent on completed data preprocessing, were carefully managed to avoid delays. This allowed the team to allocate resources efficiently and ensure that overlapping responsibilities were handled with coordination. Weekly milestones helped track progress and enabled the team to identify any potential bottlenecks early. The Gantt chart not only provided a visual representation of the entire workflow but also served as a communication tool between the team and supervisors for effective project tracking. As the Kaasht system combined machine learning, API development, and real-time prediction mechanisms, the timeline accounted for testing, debugging, and integration challenges, particularly in model optimization and deployment. Buffer periods were intentionally built into the schedule to accommodate unforeseen issues or enhancements based on feedback. Overall, the structured timeline ensured the project stayed on track and each phase transitioned smoothly into the next, ultimately contributing to the successful and timely implementation of the crop prediction system.

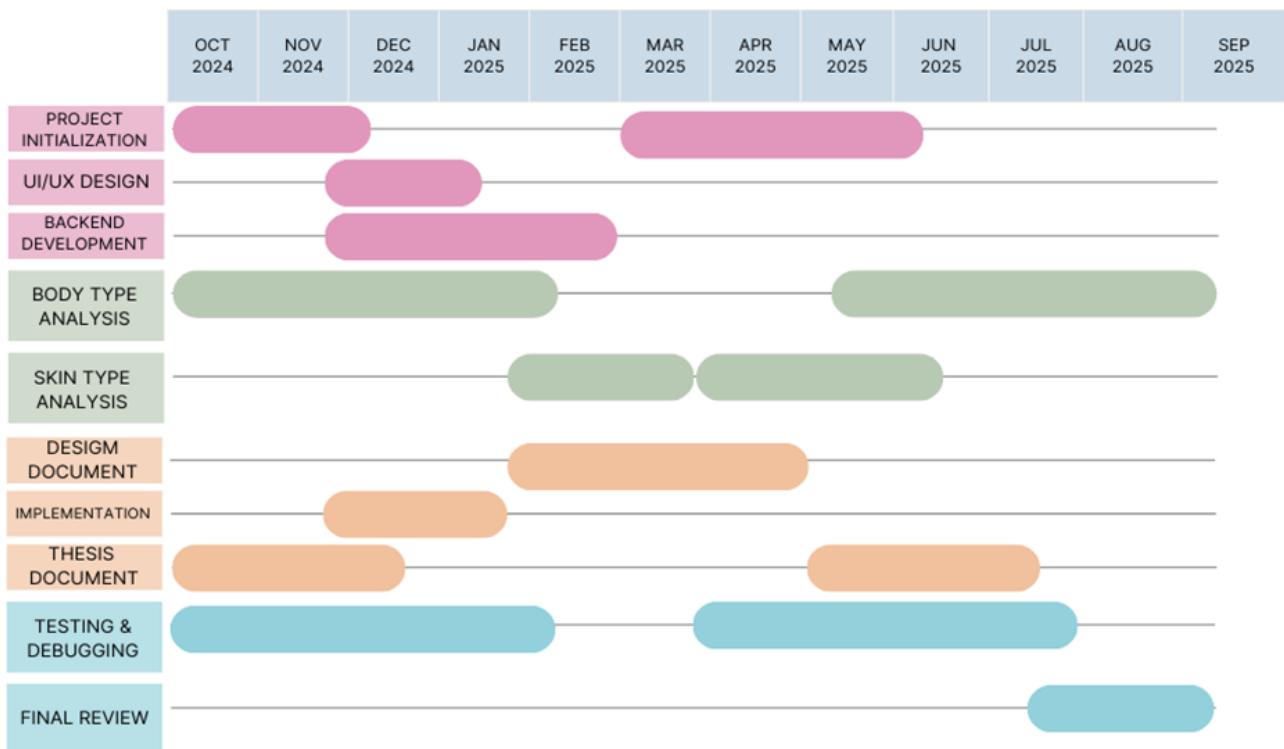


Figure 3 Gantt Chart

1.7.Work Breakdown Structure

The Work Breakdown Structure (WBS) for our Final Year Project "Kaasht – AI-Powered Crop Prediction System" was developed to systematically decompose the entire project into manageable and traceable

components. Our project consisted of major phases such as requirements gathering, dataset preparation, model development, system integration, and testing. Each of these phases was further divided into specific tasks and sub-tasks. For instance, the dataset preparation phase included sourcing the data, cleaning, encoding categorical variables, and performing feature scaling. The model development phase involved experimenting with multiple machine learning algorithms, selecting the Random Forest Classifier, performing hyperparameter tuning using GridSearchCV, and evaluating the model through cross-validation and performance metrics. To ensure efficient progress, responsibilities were divided between the two team members. I primarily focused on the backend architecture, particularly the machine learning pipeline, data preprocessing, and model optimization. I was responsible for feature engineering, model training, accuracy evaluation, and model serialization using Joblib. Meanwhile, my partner took lead on system design, frontend interactions, and the FastAPI-based integration layer. They developed the API endpoints, ensured proper request handling and validation using Pydantic, and managed the deployment architecture including Swagger UI for documentation and Docker containerization for scalability. We both collaborated on debugging and integrating the API with the model, ensuring the endpoints were functional and responsive. Additionally, tasks such as preparing documentation, designing use cases, and creating diagrams like activity, sequence, and data-flow diagrams were divided evenly. Both of us contributed equally during the testing phase, where we validated the system using various input combinations and ensured accurate prediction outputs were generated. Throughout the project, regular coordination, progress reviews, and joint decision-making ensured smooth execution and timely completion of each component in alignment with the Gantt chart. This structured and collaborative approach to the WBS helped us handle a technically diverse project effectively.

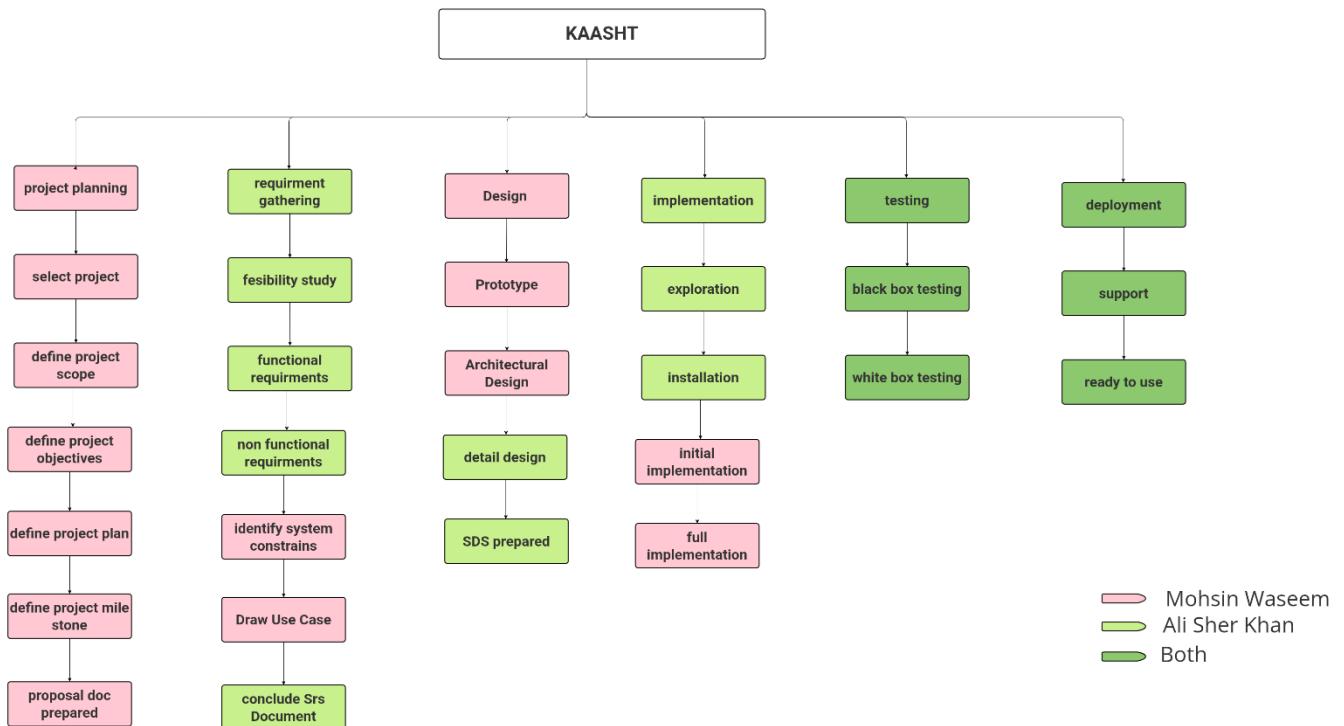


Figure 4 Work Break Down Structure

CHAPTER 2

EXISTING SYSTEM

2.1 Existing system

Over the past decade, a number of mobile and web-based applications have been developed to assist farmers in decision-making using technology. While these systems have helped modernize agriculture, they often lack the integration of real-time localized soil and weather data for crop prediction. Below are nine notable existing systems that relate to agricultural support:

2.1.1 Plantix

Plantix is a popular mobile app that uses image recognition to detect crop diseases and provides recommendations for treatment. However, it focuses primarily on plant health rather than predicting crop suitability based on environmental data.

2.1.2 AgriApp

AgriApp is an agriculture-based application that offers information on crops, fertilizers, and market prices. It helps farmers connect with experts but lacks predictive analytics for crop recommendation.

2.1.3 Krishi Network

This Indian platform allows farmers to access expert advice and market trends. It supports video content and community-based discussions but does not integrate real-time weather or soil analysis.

2.1.4 Kisan Suvidha

Developed by the Indian Ministry of Agriculture, Kisan Suvidha provides weather forecasts, market prices, and expert guidance. The system is comprehensive but lacks AI-driven crop prediction features.

2.1.5 FarmLogs

FarmLogs enables users to monitor their farm's performance and track weather and soil conditions. However, it primarily targets commercial farms in the United States and is not region-specific for South Asian farmers.

2.1.6 SmartFarm

SmartFarm offers monitoring tools and helps manage farm operations using satellite data. It does not include localized soil sensor data or predictive crop recommendations for individual farmers.

2.1.7 Cropin

Cropin offers digital farming solutions to enterprises, focusing on large-scale data analysis. It is enterprise-focused, which makes it less accessible to small-scale farmers.

2.1.8 eKutir

eKutir offers farmer support through advisory services and mobile solutions, but the lack of AI and machine learning integration for crop recommendation limits its effectiveness in precision farming.

2.1.9 Iffco Kisan

This app delivers weather forecasts, mandi prices, and farming tips. While it is highly used in rural areas, it lacks sensor-based soil health tracking or predictive models.

2.2 Literature Review

Table 4 Literature Review

App Name	Year of Release	Main Features	Drawbacks
Plantix	2015	Image-based disease detection, crop health suggestions	No crop suitability prediction; limited to disease diagnostics
AgriApp	2016	Crop database, fertilizer recommendations, marketplace integration	No AI-driven crop prediction; lacks sensor data integration
Krishi Network	2017	Expert advisory, community interaction, video tips	No integration of real-time weather or soil data
Kisan Suvidha	2016	Weather info, market prices, expert contact	No predictive algorithms; generalized information
FarmLogs	2014	Weather monitoring, yield estimation, soil maps	Geared toward large-scale US farms; lacks real-time sensor data
SmartFarm	2018	Satellite imagery, pest/disease tracking	No crop-specific predictions; less localized data
Cropin	2010	Big-data analytics for enterprises, remote monitoring	Not suitable for individual smallholder farmers; complex interface
eKutir	2012	Rural farming support, mobile advisory, crop planning	No AI or ML integration for crop recommendation
Iffco Kisan	2015	Weather, mandi prices, farming content	No prediction features; no soil data analysis

2.3 Limitation of Existing Systems

2.3.1 Plantix

- No crop prediction feature
- No integration with real-time weather or soil data
- Focused only on disease detection

2.3.2 AgriApp

- Lacks AI-based crop recommendation
- No weather or soil sensor integration

- Limited to advisory support

2.3.3 Krishi Network

- No sensor-based soil data
- Lacks crop suitability predictions
- Relies on community input, not AI

2.3.4 Kisan Suvidha

- No machine learning integration
- Cannot predict best-suited crops
- No soil data monitoring

2.3.5 FarmLogs

- Region-limited to the US
- Designed for large-scale farms
- Limited accessibility for smallholders

2.3.6 SmartFarm

- No real-time local sensor integration
- No predictive crop analytics
- Complex interface for small-scale users

2.3.7 Cropin

- Not designed for individual farmer use
- Requires enterprise-level infrastructure
- Lacks affordability for small farmers

2.3.8 eKutir

- No AI-based crop prediction
- Basic advisory functionality
- Limited automation

2.3.9 Iffco Kisan

- No soil sensor data
- No AI-driven recommendations
- Mainly focused on information delivery

CHAPTER 3

PROPOSED SYSTEM

3.1 Introduction

The Kaasht application is a smart agricultural tool designed to revolutionize crop planning and cultivation practices by providing data-driven crop suitability predictions. It leverages machine learning and artificial intelligence to analyze real-time environmental conditions such as weather forecasts and soil health indicators. This mobile-based solution empowers farmers with actionable insights on which crops are most likely to thrive under current conditions, allowing for informed, strategic farming decisions. The application is particularly tailored for regions like Punjab, Pakistan, where localized agricultural support is essential to ensure optimal yields in varying environmental conditions. Kaasht functions by integrating data from two primary sources: external weather APIs and locally installed soil sensors. These inputs capture key parameters such as temperature, humidity, rainfall, soil pH, and nutrient levels (nitrogen, phosphorus, potassium). The system processes this data using AI algorithms trained on historical trends and agronomic patterns, and generates a list of crops that are most compatible with the current climate and soil profile. In addition to predictions, it provides supplementary details such as water needs, estimated yield, and cultivation timelines, helping farmers maximize productivity with minimal resource wastage. The application's interface is designed with user simplicity in mind, offering easy navigation, multilingual support, and minimal input requirements to cater to farmers with limited technical literacy. By reducing the guesswork involved in traditional farming and offering scientifically-backed suggestions, Kaasht contributes to sustainable agriculture. It not only helps reduce overuse of fertilizers and water but also minimizes the risk of crop failure, thereby supporting both environmental protection and economic stability for farming communities. Kaasht fills the gaps left by existing agricultural applications by focusing on personalized, region-specific recommendations through real-time analysis rather than static databases. Unlike many platforms that offer only generic advice or market insights, Kaasht actively processes live data to guide planting decisions. It stands as a bridge between traditional farming methods and modern smart agriculture, offering a scalable model for future development in precision farming and climate-resilient agriculture. Furthermore, Kaasht serves not only as a predictive tool but also as a learning platform for farmers by helping them understand the science behind their crops' success or failure. The application logs each user's interaction and input data, which can be referenced later to evaluate farming outcomes, adjust future strategies, and improve long-term decision-making. By capturing and analyzing trends over time, Kaasht enables farmers to build a digital record of their agricultural activities, providing deeper insights into how changing weather patterns or shifting soil health impacts their yields. This continuous feedback loop enhances the accuracy of future predictions and encourages a more analytical approach to agriculture. Additionally, the modular design of Kaasht allows for easy scalability and integration of future technologies such as irrigation control, pest prediction, or market price forecasts. This adaptability ensures that Kaasht can evolve alongside emerging agricultural challenges, providing ongoing value to users and strengthening the digital ecosystem of smart farming in underserved regions.

3.2 Module od the system

The Kaasht application is structured around six core modules that work together to deliver accurate crop suitability predictions. The weather data module retrieves real-time weather information via external APIs, while the soil data module collects essential soil health metrics through integrated sensors. The crop prediction module processes this data using AI and machine learning models to recommend suitable crops. The user interface module ensures a smooth and accessible experience for farmers with clear visuals and multilingual support. The data storage and management module securely stores historical weather, soil, and prediction data for analysis. Lastly, the location services module uses GPS to provide region-specific insights tailored to the user's farming area.

3.2.1 Weather Data Processing Module

This module is responsible for collecting and analyzing real-time weather information that directly impacts crop suitability. It integrates with a third-party weather API to fetch up-to-date meteorological parameters such as temperature, humidity, rainfall, and wind speed. The data is used as a primary input for the machine learning model to determine which crops are most compatible with the current and forecasted weather conditions. The module ensures that data is refreshed periodically to maintain accuracy and relevance. It also stores weather history to support long-term trend analysis.

3.2.2 Soil Data Analysis Module

The soil data analysis module focuses on collecting and processing real-time soil health metrics through integration with external soil sensors. Parameters like pH level, nitrogen, phosphorus, and potassium content are fetched using Bluetooth or Wi-Fi-enabled sensors. This data is stored and continuously analyzed to assess whether the current soil condition is favorable for certain crops. When combined with weather data, this module plays a crucial role in enhancing the precision of crop prediction, ensuring that recommendations are tailored to actual field conditions.

3.2.3 Machine Learning Prediction Engine

This module is the core intelligence of the Kaasht system. It houses the machine learning model trained on historical agricultural data specific to the Punjab region. The model takes both weather and soil data as inputs and outputs the most suitable crops to cultivate under current conditions. It also provides associated metrics such as expected yield, water requirements, and growth period. The engine is designed to improve over time by incorporating new data and feedback, thus refining its predictions and adapting to evolving climate and soil trends.

3.2.4 User Interface and Interaction Module

The user interface (UI) module manages how users interact with the application. It is designed to be simple, intuitive, and multilingual to accommodate farmers with minimal technical expertise. The interface displays real-time data, prediction results, crop recommendations, and detailed insights in a

visually clear format. It also includes data entry screens for profile management, system settings, and location selection. This module ensures that farmers can easily navigate through the application and make use of its full capabilities without confusion or technical barriers.

3.2.5 User Profile and Authentication Module

This module handles user registration, login, and secure access to the system. Users can sign up using an email address or social media accounts, and once authenticated, they can manage their profiles by updating personal details, farm location, and crop preferences. It ensures data privacy and includes a password recovery feature via email. This module is essential for personalizing the app experience and securing user-specific data such as prediction history and reports.

3.2.6 Historical Data Management Module

The historical data module is designed to maintain a comprehensive local database of past weather conditions, soil readings, and crop prediction outcomes. It enables users to track changes in environmental conditions over time, compare previous recommendations, and analyze long-term farming trends. This module also allows for exporting and sharing data in CSV or PDF formats, supporting data-driven decision-making and documentation for both personal and research purposes.

3.2.7 Notification and Alert Module (Optional)

This optional module sends timely alerts and notifications to users about critical changes in weather conditions or crop suitability. For example, it may notify users of heavy rainfall, extreme temperatures, or sudden soil quality changes. Notifications are delivered as push or local alerts and help farmers take proactive steps to protect their crops or adjust farming activities. Though optional, this module adds significant value by enhancing the app's responsiveness and engagement.

3.2.8 GPS and Location Services Module

The GPS module automatically determines the user's location to fetch localized weather and soil information. When the app is launched for the first time, it requests permission to access the device's GPS. Based on the user's geographical location, it tailors data collection and prediction processes. This ensures that all recommendations are highly relevant to the specific area where the user is operating, particularly important in a geographically diverse region like Punjab.

3.2.9 Cloud Synchronization and Backup Module

This module ensures that user data is not lost by synchronizing it with a secure cloud server. It supports storage of profiles, reports, prediction results, and sensor readings across multiple devices. The system can be configured for automatic or manual syncs at regular intervals. This module provides data resilience,

accessibility across devices, and support for potential future integrations with web-based dashboards or additional services.

3.3 Functional Requirements

Table 5 Functional Requirements

S.No.	FR ID	Functional Requirement Name	Description	Requirements
1	FR-1	User Registration and Authentication	Allow users to register and log in via email or social media	Register using email, password, location- Secure authentication- Password recovery via email
2	FR-2	User Profile Management	Enable users to manage their personal and farm details	Enter name, contact, location- Update profile anytime
3	FR-3	Weather Data Fetching	Fetch and display real-time weather data from an external API	Show temp, humidity, rainfall, wind- Auto fetch on app open- Refresh every 30 mins
4	FR-4	Soil Data Integration (Optional)	Fetch soil metrics using compatible sensors	Connect via Bluetooth/Wi-Fi- Get pH, N, P, K values- View real-time data
5	FR-5	Crop Prediction	Predict suitable crops using weather and soil data	Use ML model with environmental inputs- Show crops, expected yield, suggestions
6	FR-6	Historical Data Management	Store and display past weather, soil, and crop prediction data	Save historical inputs- View reports- Export/share data
7	FR-7	Notifications (Optional)	Notify users about critical weather or crop changes	Alert on rain, heat, drought- Push or local notifications
8	FR-8	Location Services	Use GPS to localize data and predictions	Request GPS permission- Auto-fetch data based on location
9	FR-9	Data Synchronization (Cloud)	Sync data with cloud for access across devices	Sync with Firebase/AWS- Manual or auto sync option

3.4 Non-functional Requirements

3.4.1 Performance

The Kaasht application must deliver a responsive and smooth experience for users, even under varying network conditions and concurrent usage. It is essential that the app fetches weather data within five seconds and completes crop prediction processing within ten seconds. Synchronization tasks such as cloud backups and data updates should not exceed thirty seconds under normal network loads. The system should be optimized to ensure low latency and high throughput, allowing up to 100 users to access services

simultaneously without performance degradation. Efficient data handling for at least six months of historical records must be maintained without causing lag or errors.

3.4.2 Scalability

To support future growth, the application must be designed for scalability. As the number of users, sensors, and data queries increases, the system should be able to expand its resources accordingly. This includes backend infrastructure like databases and cloud storage, which must accommodate large volumes of user data, weather inputs, and crop recommendations. The app should also support the integration of additional features without needing a complete architectural overhaul. The scalability ensures long-term sustainability of the platform as user demand increases.

3.4.3 Data Integrity

The Kaasht system must uphold data accuracy and reliability across all modules. Data retrieved from weather APIs and soil sensors must be validated upon entry to ensure correctness and consistency. Any discrepancies, such as missing or malformed sensor inputs, should trigger alerts for users to correct the issue. This helps prevent inaccurate crop recommendations, which could otherwise negatively impact farming decisions. Strong data validation rules and error-checking mechanisms will be implemented to maintain integrity across the entire system lifecycle.

3.4.4 Security

Security is critical in protecting user data, including personal information, location details, and farming insights. All data transmissions must be encrypted using HTTPS protocols, and personal data must be stored securely using standards like AES encryption. The application must also enforce secure login practices, including email-password combinations or social logins, with encrypted credential storage. Role-based access control must distinguish between regular users and administrators, restricting sensitive actions and features accordingly. Overall, the system must be robust against unauthorized access, data leaks, and malicious attacks.

3.4.5 Reliability

The application must be consistently available and capable of functioning under various conditions without crashing. Kaasht should provide accurate predictions even if some modules (e.g., soil sensors) become temporarily unavailable. In such scenarios, fallback mechanisms such as using last-known values or weather-only predictions must be available. Additionally, the app should implement a monitoring mechanism that detects and recovers from unexpected failures or disruptions to maintain seamless service.

3.4.6 Usability

The user interface should be intuitive and designed with the needs of farmers in mind. Minimal technical knowledge should be required to navigate the application. Visuals, graphs, and recommendations must be easy to interpret. Multilingual support (e.g., Urdu and Punjabi) should be included to ensure accessibility to a wider user base. The system should require minimal user input and offer clear guidance at every step, making it especially helpful for users unfamiliar with digital platforms.

3.4.7 Maintainability

Kaasht must be developed with clean, modular, and well-documented code to support easy updates and maintenance. Future changes, such as adding support for new crop types or integrating more datasets, should not require extensive rewrites of the codebase. A clear separation of concerns among modules and version control for software releases must be practiced to keep the system stable and manageable in the long run.

3.4.8 Compliance

The system must comply with applicable national and international data privacy regulations, such as GDPR. Before collecting or processing any personal or sensitive information, user consent must be obtained explicitly. Transparency about data usage should be provided through a privacy policy, and users should have the ability to manage or delete their data on request. This ensures that Kaasht adheres to ethical standards and legal obligations while protecting user rights.

3.5 Feasibility Study

3.5.1 Technical Feasibility

The Kaasht application is technically feasible as it relies on well-established technologies such as mobile app development (Android), weather APIs, and soil sensors. Machine learning models used for crop prediction can be developed using Python frameworks like TensorFlow or scikit-learn and integrated into the mobile app via cloud-based services. Additionally, the infrastructure for data storage, user authentication, and cloud synchronization can be built using platforms like Firebase or AWS. The availability of these tools and the development team's familiarity with them makes the technical implementation achievable.

3.5.2 Operational Feasibility

The system is operationally feasible due to its simple user interface, low resource requirements, and practical applicability. Farmers will be able to use the mobile app with basic training, and its localized focus on the Punjab region ensures relevance to the end users. The integration of real-time weather and soil data allows farmers to make quick and informed decisions. Community adoption is supported through user-friendly visuals and regional language support, increasing the system's operational success.

3.5.3 Economic Feasibility

Developing the Kaasht system is economically viable due to the use of cost-effective tools and open-source technologies. Most of the machine learning libraries, cloud services, and development environments used are free or have affordable pricing tiers. The return on investment is significant, as the system will help reduce crop failure, improve yield, and minimize unnecessary input costs for farmers. Additionally, because the app can be distributed digitally, deployment and maintenance costs remain low.

3.5.4 Legal Feasibility

The system meets legal feasibility by ensuring user data protection in compliance with data privacy laws such as GDPR. All personal and sensor data is securely stored and processed with user consent. The app will include clear disclaimers and privacy policies to inform users about the data it collects, how it is used, and how they can control it. Since it uses public APIs and licensed data sources, it does not violate intellectual property rights.

3.5.5 Schedule Feasibility

The Kaasht project can be completed within a realistic timeline by dividing it into phases such as requirement analysis, model development, app interface design, integration, testing, and deployment. A typical development schedule for this system would span around four to six months, which is achievable given the project's scope, available resources, and team capabilities. Each module can be developed and tested iteratively to ensure smooth progress and timely delivery.

3.5.6 Environmental Feasibility

The system supports environmental sustainability by promoting the optimal use of natural resources such as water, fertilizers, and land. By recommending suitable crops based on soil and weather conditions, it reduces the risk of over-farming and conserves resources. It also encourages climate-resilient agriculture practices which are critical in regions like Punjab that are sensitive to changing weather patterns.

3.6 Chosen Methodology

For the development of the Kaasht application, the agile methodology was chosen due to its iterative and flexible nature. This methodology allows for continuous feedback, which is crucial in a project like Kaasht that requires frequent testing, real-time data integration, and constant improvement based on user needs. Agile enables the team to break down the development process into smaller, manageable sprints, each focusing on a specific module or functionality—such as weather API integration, soil data processing, or crop prediction modeling. At the end of each sprint, feedback can be gathered from test users, stakeholders, or domain experts to ensure the system is evolving in line with its intended goals. This adaptability is essential in agricultural applications where accuracy and user experience directly impact farming decisions and outcomes.

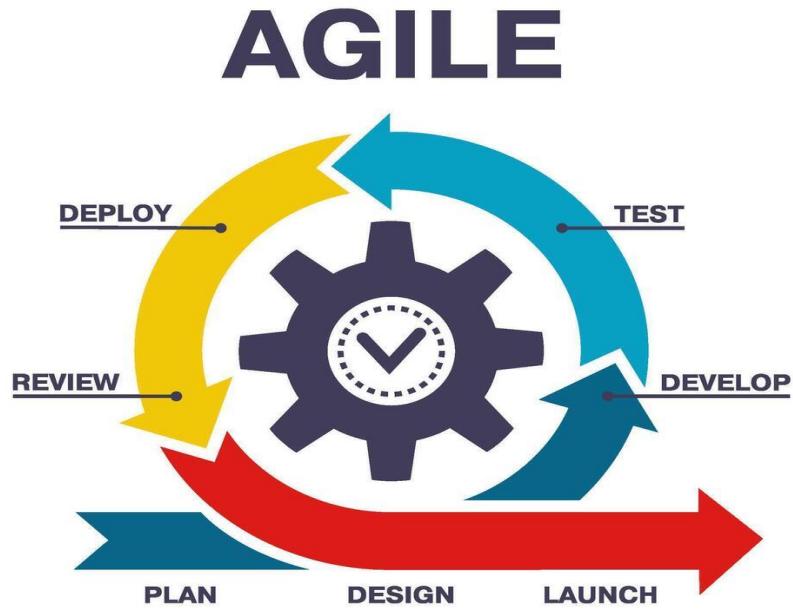


Figure 5 Agile Methodology

Additionally, agile helps in managing risks early in the development cycle by allowing early testing and validation of features like real-time data syncing, ML model performance, and user interface design. It also facilitates collaboration between developers, agronomists, and end users by keeping communication active throughout the project. This approach ensures that any issues or changes in requirements are quickly addressed without disrupting the overall project flow. In the context of Kaasht, where integration of multiple data sources and real-world usability is key, agile provides the structure and flexibility needed to build a scalable, reliable, and user-centric application tailored to the needs of farmers in Punjab.

3.7 Project Structure

As developers of the Kaasht application, I, Ali Sher Khan, along with my teammate Mohsin Waseem, worked collaboratively throughout the development process. From the beginning, we divided our roles based on our strengths—Mohsin focused on the backend development, machine learning model training, and API integration, while I handled the frontend design, user interface, and database management. We followed an agile approach with weekly sprints, ensuring we reviewed progress regularly and adjusted based on feedback. Our team structure was simple but effective: both of us contributed to the core logic, testing, and documentation equally while also consulting with agricultural experts for data accuracy. This teamwork allowed us to successfully build a user-friendly and intelligent crop prediction system tailored for farmers in Punjab. We also held frequent brainstorming sessions to refine system features and align them with real-world farming challenges. During testing, we collaborated closely to debug and optimize performance under various conditions. We divided research tasks, with one of us exploring AI model enhancements while the other worked on sensor data simulation. This balanced collaboration helped us deliver a reliable, practical, and farmer-centric application.

CHAPTER 4

SYSTEM DESIGN

4.1 Introduction

The Kaasht application is a smart farming solution designed to enhance agricultural decision-making by integrating real-time weather and soil data with AI-based crop prediction. Developed specifically for the agricultural landscape of Punjab, Pakistan, this system provides farmers with localized, data-driven insights to determine the most suitable crops for their land at any given time. Through the combination of weather APIs and soil sensor inputs—measuring parameters such as temperature, humidity, rainfall, soil pH, nitrogen, phosphorus, and potassium—the system processes this information using a trained machine learning model. The app then presents accurate crop suitability recommendations, helping reduce guesswork, increase yield potential, and minimize resource waste. Its intuitive Android interface ensures accessibility for users with varying literacy levels, offering multilingual support, simple navigation, and offline data access. By leveraging modern technologies, Kaasht bridges the gap between traditional farming practices and precision agriculture, empowering farmers to make timely, informed, and sustainable farming decisions. By integrating AI and machine learning, the app transforms conventional agriculture into a smarter, more adaptive system. It bridges the gap between technology and rural farming, ensuring long-term growth and food security.

4.2 Purpose

The purpose of the Kaasht application is to empower farmers with intelligent, data-driven tools that enhance agricultural productivity and decision-making. By utilizing real-time weather conditions and soil health data, the app aims to predict the most suitable crops for specific farming conditions, thereby minimizing risks associated with poor crop selection. It is designed to address the challenges faced by traditional farming methods, such as reliance on outdated practices, limited access to expert advice, and lack of awareness about changing environmental factors. Kaasht provides localized, scientifically backed crop recommendations that not only improve yield but also promote efficient use of resources like water and fertilizers. The ultimate goal is to support sustainable farming practices and ensure economic stability for small to medium-scale farmers, especially in regions like Punjab, Pakistan. The *Kaasht* application is developed with the primary purpose of transforming traditional farming into a data-driven, intelligent decision-making process by integrating real-time weather forecasts, soil health analysis, and machine learning-powered crop prediction. Aimed specifically at the agricultural conditions of Punjab, Pakistan, this system empowers farmers to make informed decisions about crop cultivation based on accurate and timely environmental data. The introduction of Kaasht marks a shift from conventional guess-based farming to precision agriculture, where each decision is backed by data insights gathered from external weather APIs and local soil sensors. These tools collect and analyze key factors like temperature, rainfall, humidity, soil pH, and essential nutrients (nitrogen, phosphorus, potassium), allowing the system to predict crop suitability with high accuracy. Beyond basic predictions, Kaasht also offers detailed recommendations on crop care, estimated yield, and resource management, ultimately helping farmers reduce costs, prevent crop failure, and increase productivity. The intuitive interface is designed with simplicity in mind, ensuring ease of use even for farmers with limited technical skills, while features like multi-language support, offline accessibility, and personalized dashboards enhance user experience.

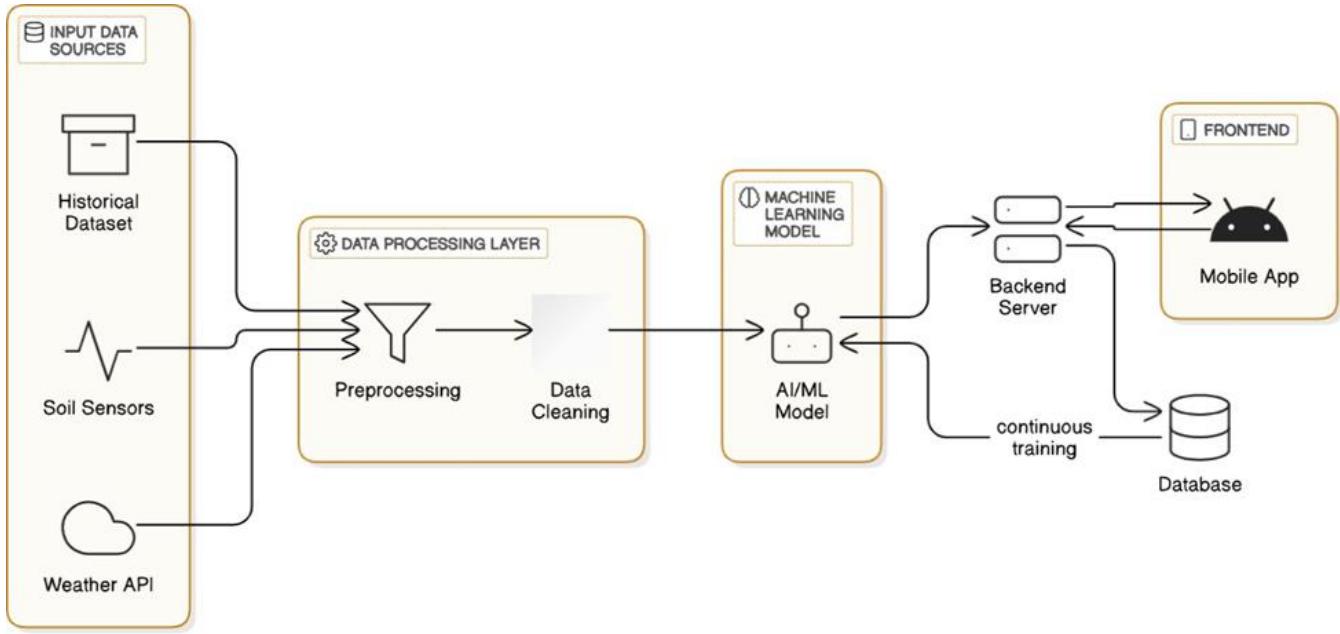


Figure 6 Architecture Diagram

In essence, Kaasht serves as a comprehensive agricultural assistant, aligning with the broader goal of sustainable farming by minimizing resource wastage and maximizing output. It bridges the gap between rural agriculture and cutting-edge technology, offering a scalable, adaptable, and user-friendly solution that addresses the modern challenges of climate change, unpredictable weather, and resource scarcity.

4.3 Scope

The Kaasht application is designed to operate as a smart crop prediction system that leverages real-time data and artificial intelligence to support farmers in making informed crop planning decisions. The scope of the system covers the development of a mobile-based platform that integrates weather APIs and soil sensors to gather accurate environmental data from the user's specific geographical location, particularly targeting the Punjab region of Pakistan. The application is structured to process weather metrics such as temperature, humidity, and rainfall, along with soil parameters like pH, nitrogen, phosphorus, and potassium levels, to predict the most suitable crops for cultivation. This ensures that farmers are provided with actionable insights specific to their farmland's current condition, enhancing their decision-making ability in a region where agriculture is deeply dependent on environmental factors. The scope further extends to include machine learning models trained on historical weather and crop data from Punjab. These models will process real-time inputs and generate predictions with a focus on accuracy, relevance, and timeliness. Additionally, the application will provide a simple and intuitive user interface to make it accessible for farmers with varying levels of digital literacy. It will support features like multilingual display, graphical weather representation, visual guides for crop selection, and offline access to previously fetched data. By doing so, Kaasht aims to reduce the technological barrier and enable farmers from rural and underserved communities to use data-driven tools to improve yield and sustainability. Another key

aspect of the project's scope is the data storage and cloud synchronization module. The application will not only provide real-time analysis but also maintain a database of historical user data, including past predictions, environmental readings, and farming activities. This allows for longitudinal study and improvements in prediction accuracy over time. The system also aims to ensure secure access, scalability for handling multiple users simultaneously, and the capability to expand to include more data sources, crop types, or even pest and disease prediction in future versions. The flexible architecture is intended to accommodate future technological integrations such as drone-based monitoring or satellite data inputs.

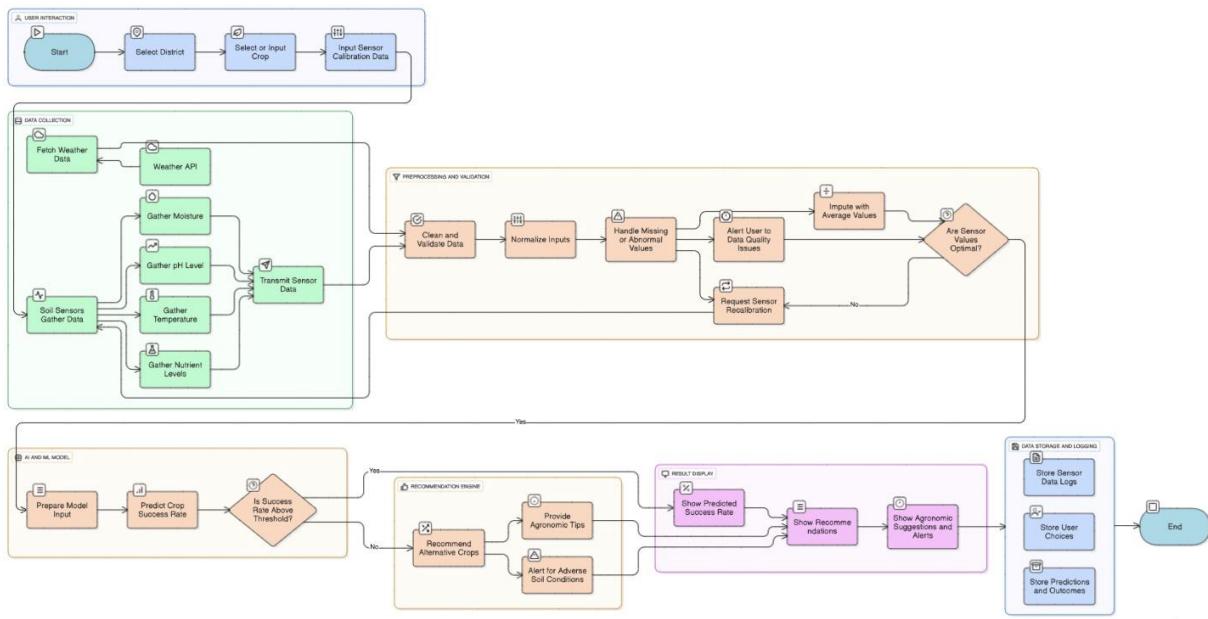


Figure 7 Flowchart

Lastly, the scope includes facilitating a sustainable and resource-efficient agricultural practice by minimizing fertilizer overuse, preventing crop failure, and guiding optimal water usage. The app not only targets individual productivity improvements but also aligns with broader environmental and socio-economic goals such as climate resilience, food security, and economic upliftment of small-scale farmers. By creating a digital ecosystem tailored to the specific climatic and soil conditions of Punjab, the Kaasht system goes beyond a traditional advisory tool — it becomes a personalized farming assistant built for accuracy, usability, and future adaptability.

4.4 Class Diagram

The class diagram outlines the core structure of the crop prediction system, featuring six main classes that work together to enable data-driven agricultural decision-making. The User class represents farmers or agronomists using the system, with essential attributes including username for identification, email for communication, and role to distinguish access levels. The Admin class extends User with specialized capabilities like managing user accounts and adjusting system-wide parameters. At the heart of the system is the CropPredictionSystem class, which coordinates the prediction workflow. It stores location to tailor

recommendations geographically, tracks when data was last updated via lastUpdated, and specifies which algorithm drives predictions through predictionModel. For environmental data, the system integrates with a WeatherAPI class requiring an apiKey for secure access to real-time climate information, while SoilSensor devices each have a unique sensorID and calibrationDate to ensure accurate soil health measurements. All prediction results and user data persist in the Database class, which handles storage and retrieval operations. Key relationships show Users initiating predictions through the CropPredictionSystem, which then pulls weather data from the API, reads soil metrics from Sensors, and archives everything in the Database. The diagram uses light-blue rectangles for classes connected by solid and dashed arrows representing associations and dependencies respectively, providing a clear visual flow of how data moves through the system from user input to actionable crop recommendations. The class diagram of this project represents the structural design of the Kaasht application by outlining the system's main classes, their attributes, and the relationships between them. It includes essential components such as user, crop, weather data, soil data, prediction engine, and database manager. Each class encapsulates specific functionalities and properties, such as user details, sensor readings, API responses, and prediction logic. The diagram also shows how these classes interact, for example, the user class connecting with both the crop prediction and data history classes. This provides a clear blueprint for object-oriented development and helps maintain modularity and scalability in the system.

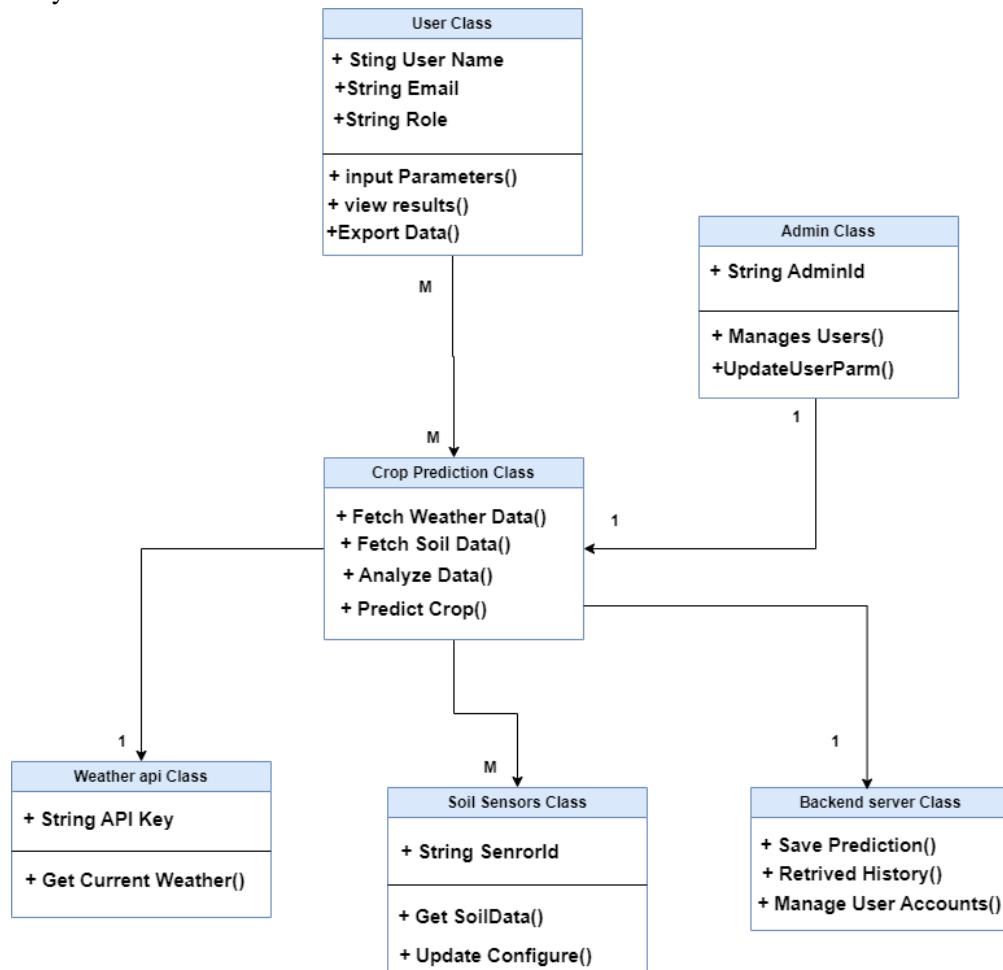


Figure 8 Class Diagram

4.5 Activity Diagram

The activity diagram of the Kaasht system represents the flow of user interactions and system processes from registration to crop prediction. It begins with user login or sign-up, followed by location detection and real-time data fetching from weather APIs and soil sensors. The system then processes this data using a machine learning model to generate suitable crop predictions. Finally, results are displayed to the user with additional details like expected yield, recommended practices, and historical data access.

4.5.1 Activity Diagram for Admin Functionalities

The Admin workflow begins with account management, where admins add, edit, or delete user accounts through a dashboard interface, with all changes being saved to the system database. Next, the admin can fetch real-time weather data by connecting to an external API using secure authentication, which retrieves parameters like temperature and rainfall for analysis. The admin then configures soil sensors by adjusting calibration settings or adding new devices, ensuring accurate soil health measurements. The system automatically analyzes the combined weather and soil data using predefined algorithms, then generates crop suitability predictions displayed to the admin in a report format. Finally, the admin can update critical system parameters such as location settings or prediction models, with all modifications logged in the database.

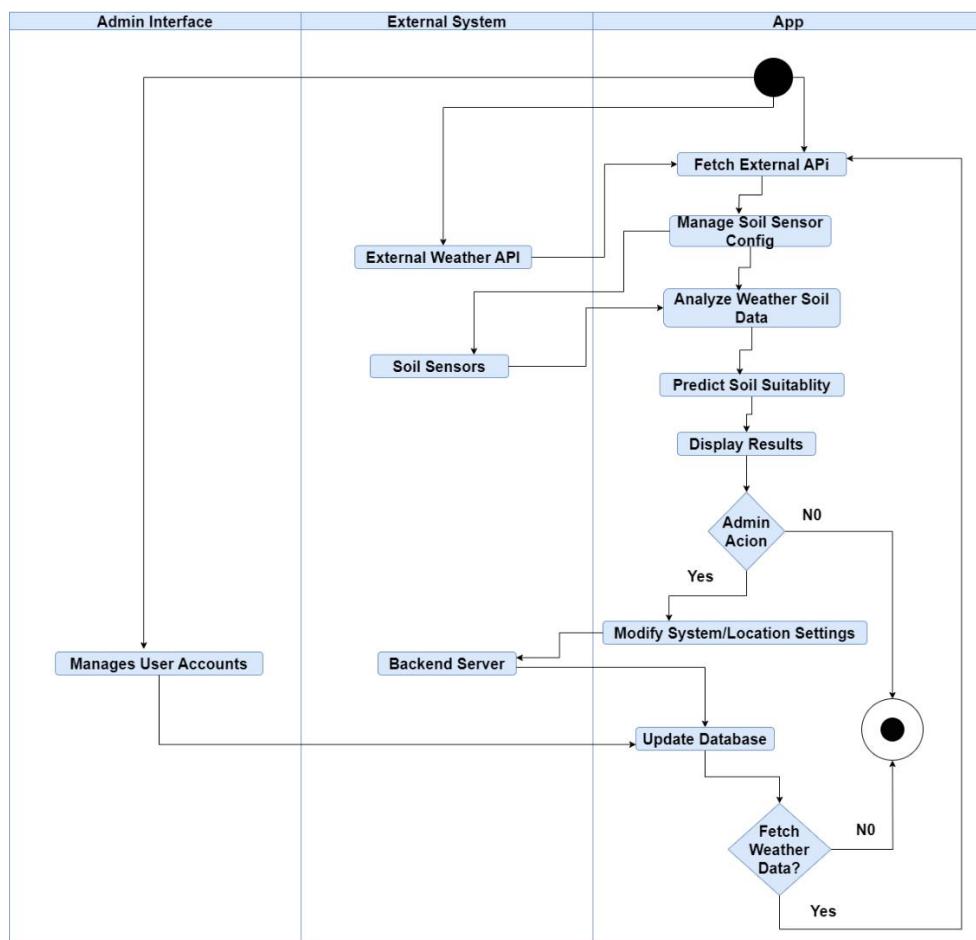


Figure 9 Activity Diagram For Admin

4.5.2 Activity Diagram for User Functionalities

The user workflow begins when farmers or agronomists input key agricultural parameters including temperature, humidity, pH levels, nitrogen, and phosphorus values through an intuitive interface. The system then automatically fetches real-time weather data from an external API and collects soil measurements from connected sensors, creating a comprehensive environmental profile. These datasets are analyzed using machine learning algorithms to evaluate soil-weather compatibility and predict optimal crop selections. The system displays these predictions in an easy-to-understand format, allowing users to review a summary of recommendations. Users can export the results as reports for record-keeping or agricultural planning purposes, with all data securely stored in the system's database. Additionally, the workflow enables users to access and compare their current predictions with historical data, providing valuable insights into changing soil conditions and crop performance trends over time. This end-to-end process combines real-time data collection with predictive analytics to deliver actionable farming recommendations while maintaining user-friendly interaction at every stage. The system ensures data integrity by validating all inputs and maintaining secure connections with external data sources.

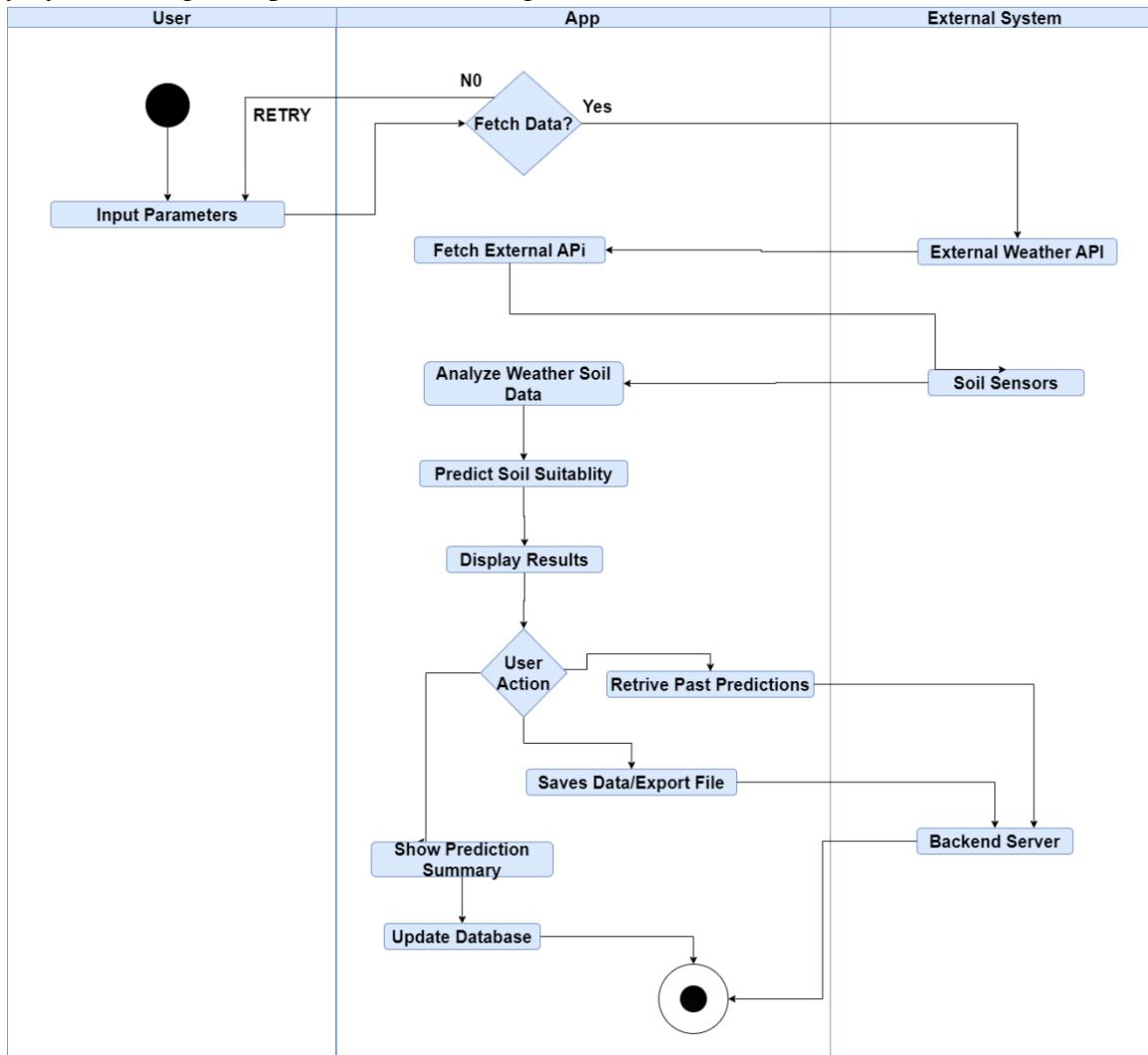


Figure 10 Activity Diagram for user

4.6 Data-Flow-Diagram

The data flow diagram (DFD) of the Kaasht system illustrates the movement of data between users, external APIs, sensors, and the internal processing modules. It begins with user input and location detection, which triggers data requests to external weather APIs and connected soil sensors. The collected environmental data flows into the prediction engine, where it is processed using trained machine learning models. The results, including suitable crop recommendations and additional insights, are then sent back to the user interface. Simultaneously, historical data is stored in the cloud database for future reference and analysis. The DFD clearly shows the interactions between each component and how data is processed, stored, and presented within the system. The data flow diagram of the Kaasht system illustrates how data moves through different components of the application. It visually represents the flow between the user, the weather API, soil sensors, prediction engine, and the database. Each process in the diagram shows how inputs like weather and soil data are transformed into meaningful outputs, such as crop suggestions. The DFD helps in understanding the logical flow of data without diving into system code, making it useful for both developers and stakeholders to grasp the core functionalities of the system.

4.6.1 Data-Flow-Diagram Level 0

The Level 0 Data Flow Diagram (DFD) of the Kaasht system provides a high-level overview of the entire application. It shows the user interacting with the system to input their location and receive crop recommendations. The system fetches weather and soil data, processes it using a machine learning model, and outputs suitable crop predictions.

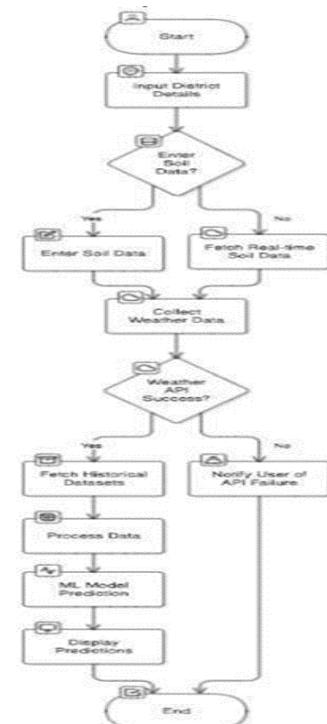


Figure 11 DATA-FLOW-DIAGRAM LVL-0

4.6.2 Data-Flow-Diagram Level 1

The Level 1 Data Flow Diagram of the Kaasht system breaks down the main process into detailed subprocesses, including user registration, weather and soil data collection, machine learning-based crop prediction, and result display. It highlights data flow between the user, external APIs, soil sensors, the prediction engine, and the result interface.

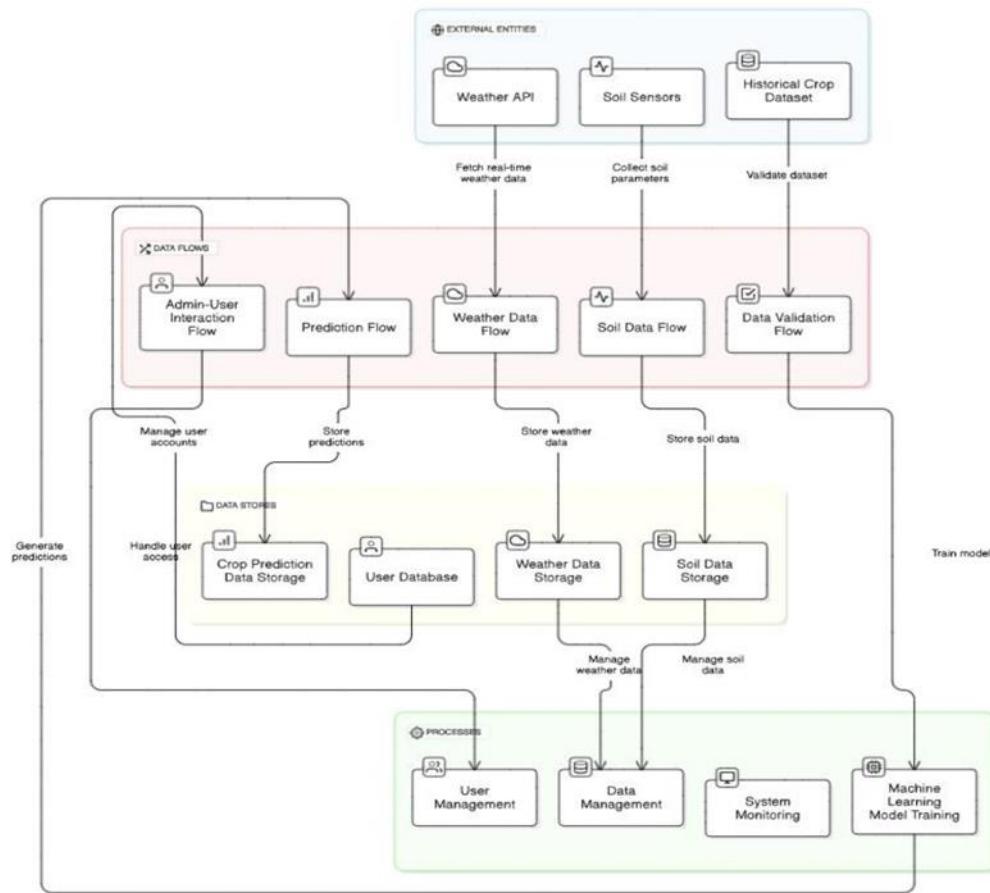


Figure 12 DATA-FLOW-DIAGRAM LVL-1

4.7 Sequence Diagram:

The sequence diagram of the Kaasht application illustrates the step-by-step interaction between the user, system components, external weather API, and soil sensors for the crop prediction process. It begins when a user launches the app and logs in or registers through the authentication module. Upon successful login, the system prompts the user to fetch real-time weather and soil data. The app sends a request to the external weather API to retrieve current weather conditions such as temperature, humidity, and rainfall, while simultaneously retrieving soil data through connected sensors (if available). Once both data sources are successfully collected, they are passed to the machine learning prediction engine, which processes the input parameters using a trained model. The system then generates a list of suitable crops along with

additional insights such as estimated yield and cultivation timelines. Finally, the results are displayed on the user interface, where the user can view, save, or further explore the recommendations. This sequence ensures a smooth, interactive, and real-time decision-support process for farmers using the Kaasht app.

4.7.1 Sequence Diagram For Admin Functionalities

The sequence diagram for admin functionalities in the Kaasht application outlines the interaction between the admin and the system during core management tasks. When the admin logs in, the system verifies credentials and grants access to the admin dashboard. From there, the admin can perform actions such as managing user accounts, updating crop prediction models, and reviewing system logs.

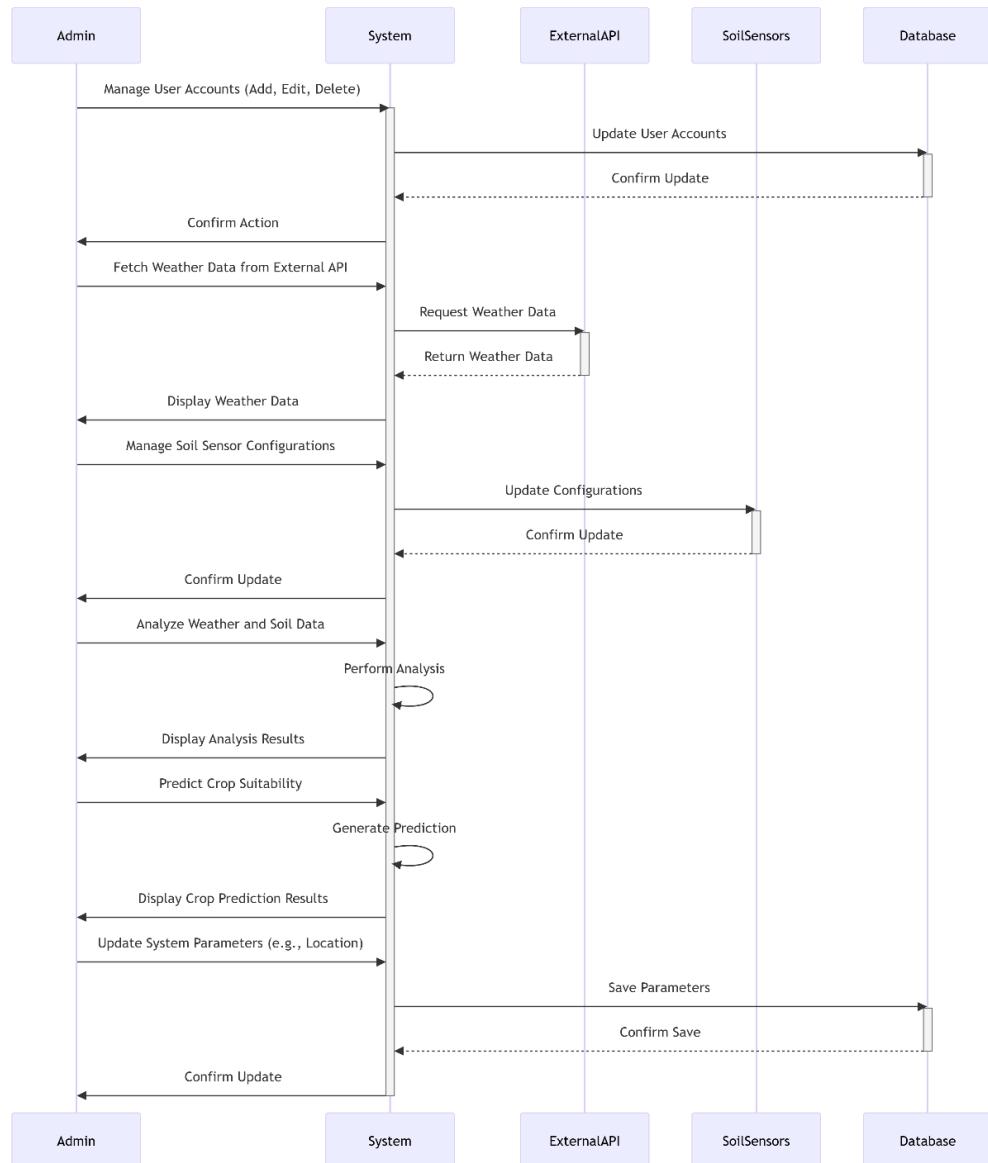


Figure 13 Sequence Diagram for Admin

4.7.2 Sequence Diagram For User Functionalities

The sequence diagram for user functionalities in the Kaasht application outlines the structured flow of interactions between the user and system components. When a user opens the app, they are first prompted to log in or register. Once authenticated, the user gains access to the dashboard, where they can choose from options like viewing real-time weather data, checking soil sensor inputs, or initiating the crop prediction module. Upon selecting crop prediction, the system fetches the latest environmental data, processes it through the trained machine learning model, and returns a list of suitable crops along with yield estimates. The user can also view detailed analytics, save predictions, and access historical data. This interaction ensures a smooth and intelligent farming experience, driven by responsive backend communication and real-time decision support.

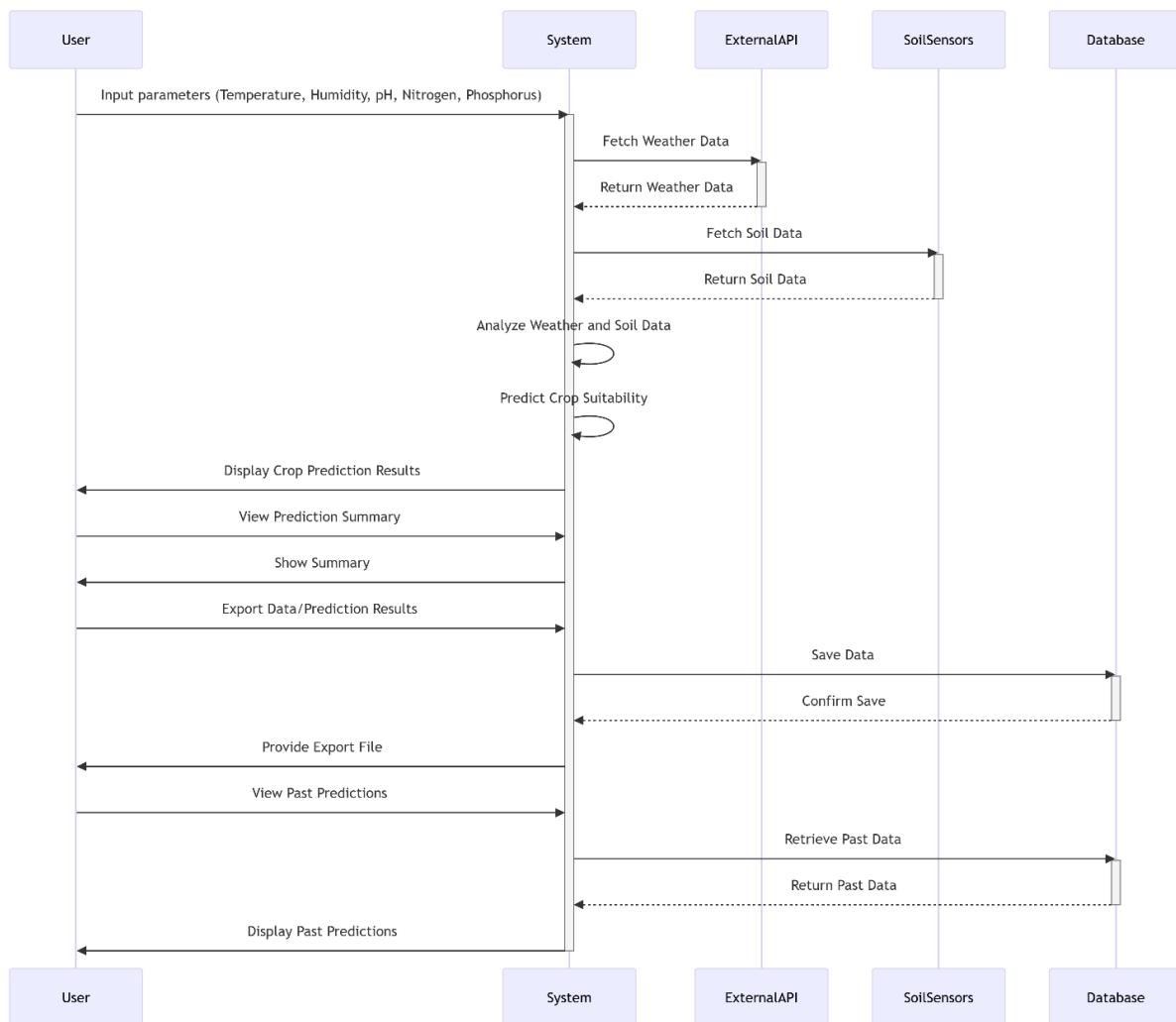


Figure 14 Sequence Diagram For User

4.8 Entity Relationship Diagram

The entity relationship diagram (ERD) for the Kaasht application outlines the structural design of the database, illustrating how various entities interact and maintain relational integrity within the system. At the center of the ERD is the user entity, which stores key personal details such as user ID, name, email, password, and location. Each user can be associated with multiple crop prediction records, which include attributes like predicted crop, estimated yield, prediction timestamp, and relevant environmental parameters. The weather data entity records parameters such as temperature, humidity, rainfall, and wind speed, linked to specific locations for accurate and localized predictions. Soil data is also stored as a separate entity, capturing metrics such as pH level, nitrogen, phosphorus, and potassium values, which can be manually entered or fetched via sensor devices. Another important entity is the sensor entity, which stores sensor-specific metadata including sensor ID, type, and operational status, and is linked to the soil data it collects. The crop information entity maintains a library of various crops along with their optimal environmental conditions and cultivation requirements, which serves as a reference during the prediction process. Notifications are handled through a dedicated notification entity that stores alerts or updates based on dynamic changes in weather or crop predictions. The admin entity manages system-level users with privileges to monitor, manage, and update user data, predictions, and other backend operations.

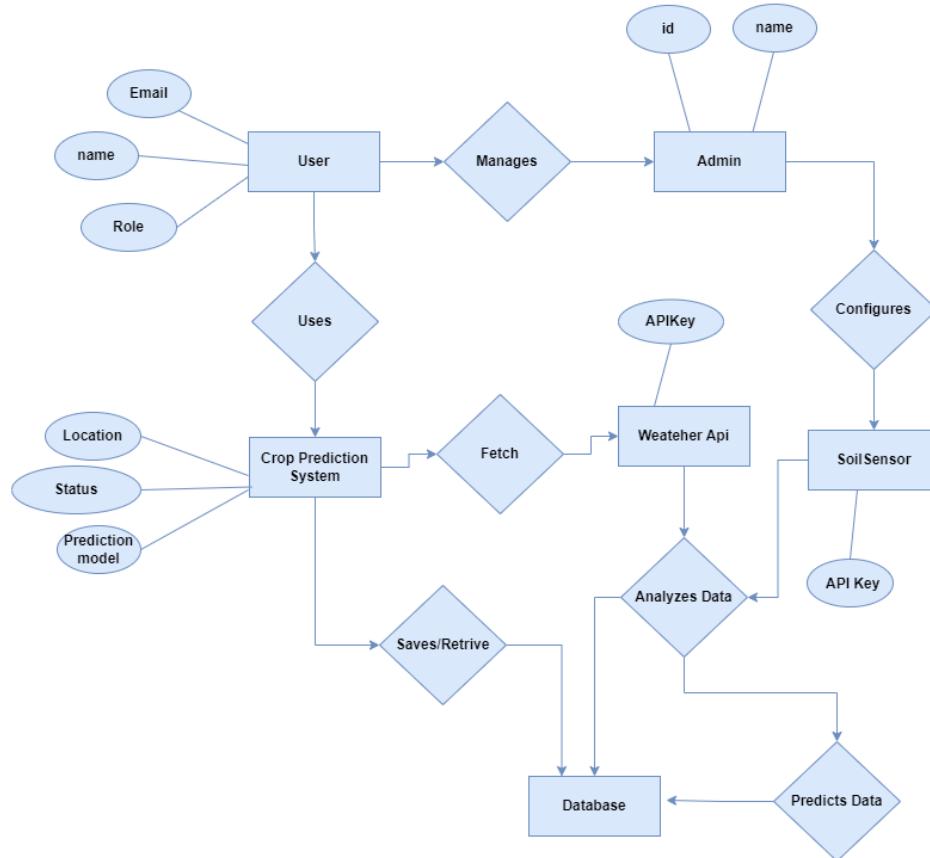


Figure 15 Entity Relationship Diagram

4.9 Screen Images

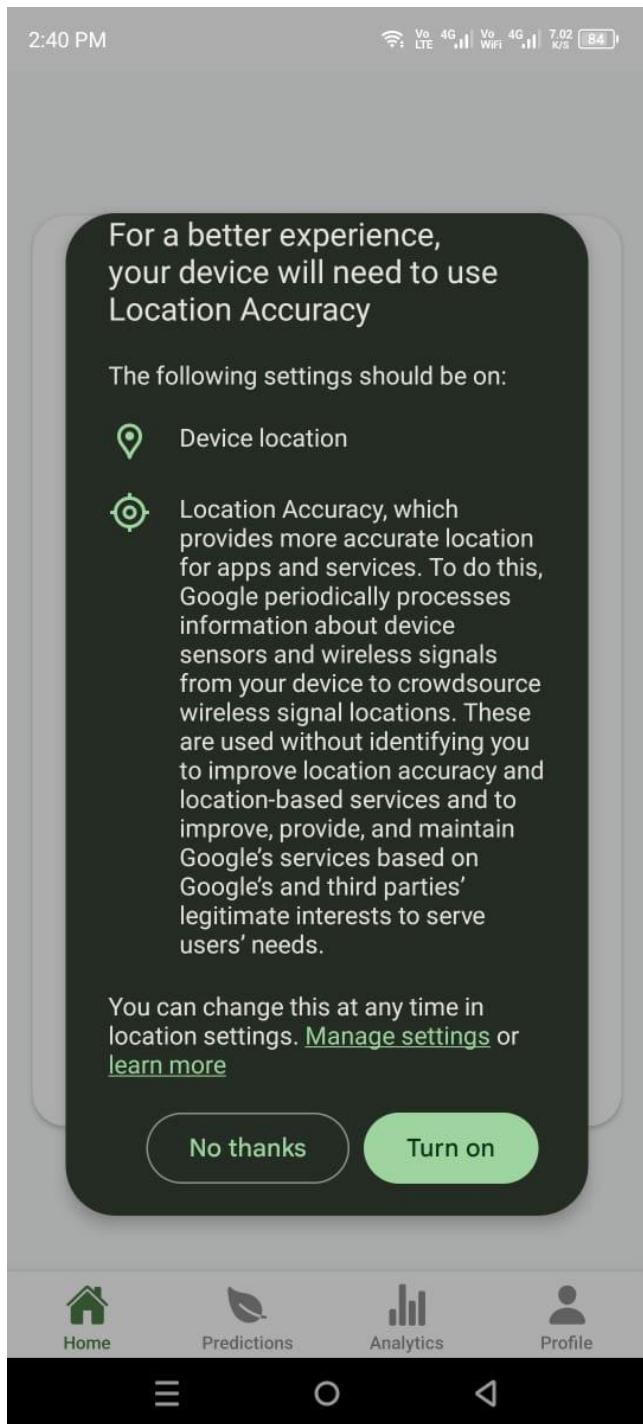


Figure 16 Location Permission

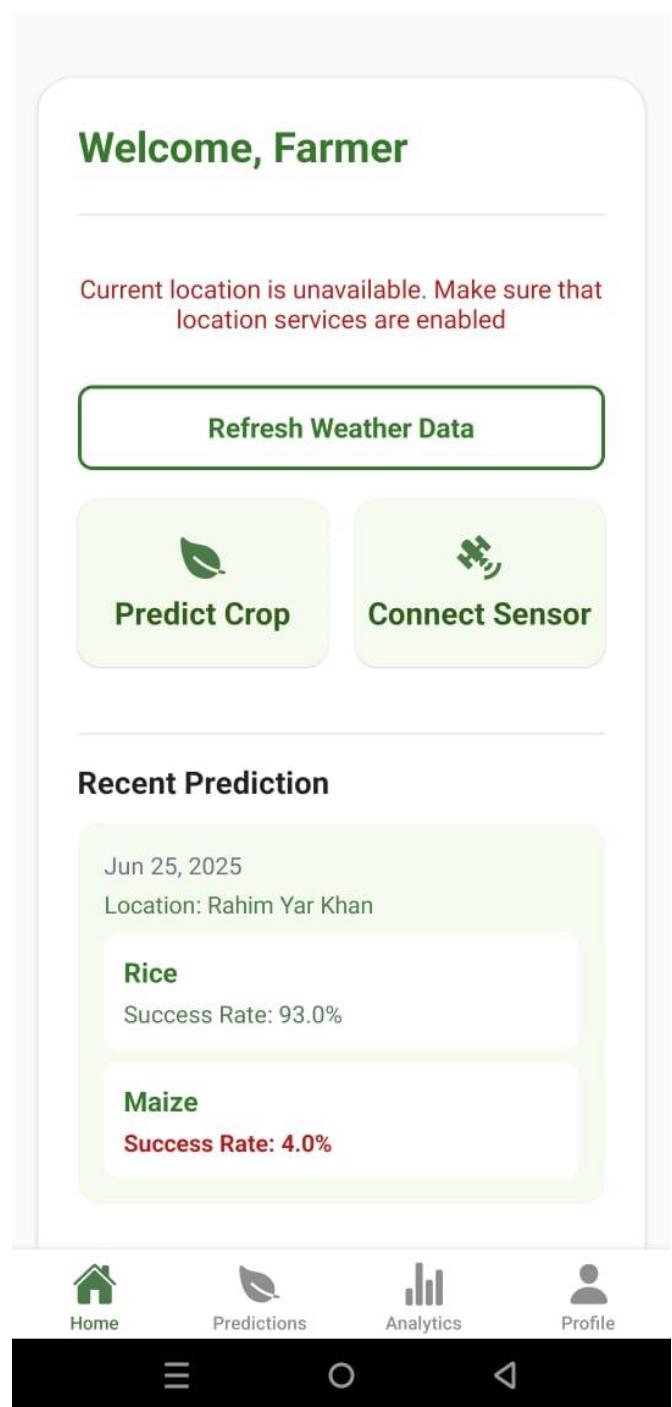


Figure 17 First>Welcome Page

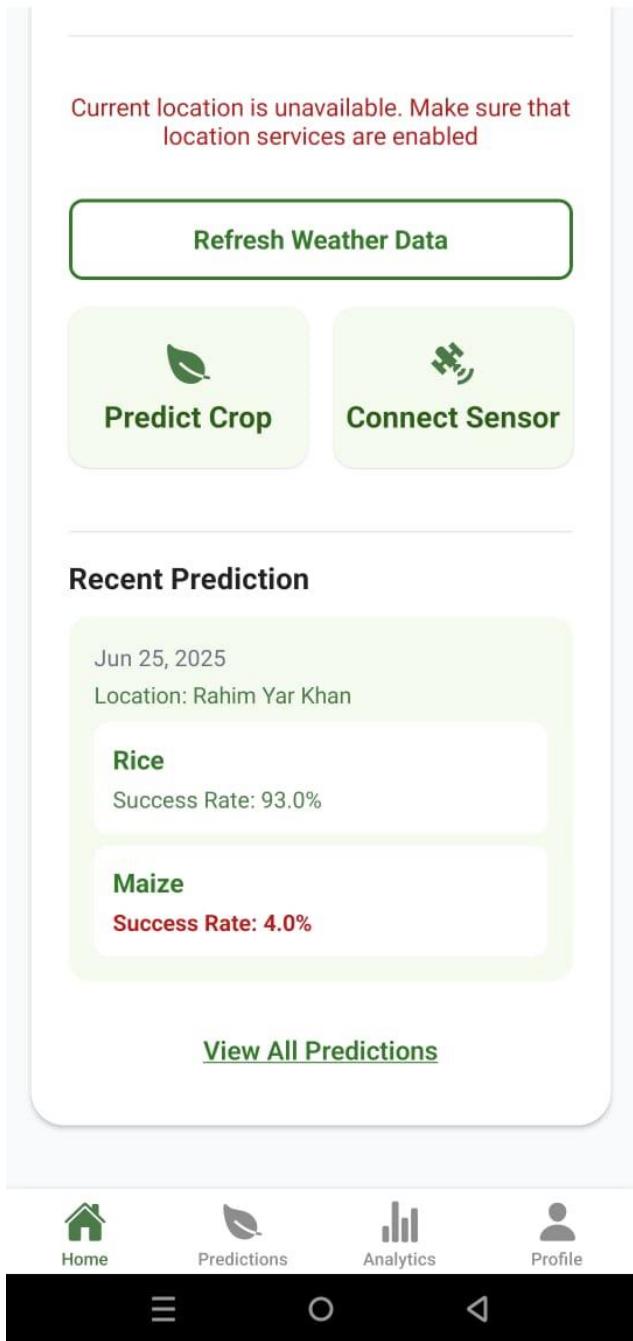


Figure 18 Enter Data of Weather

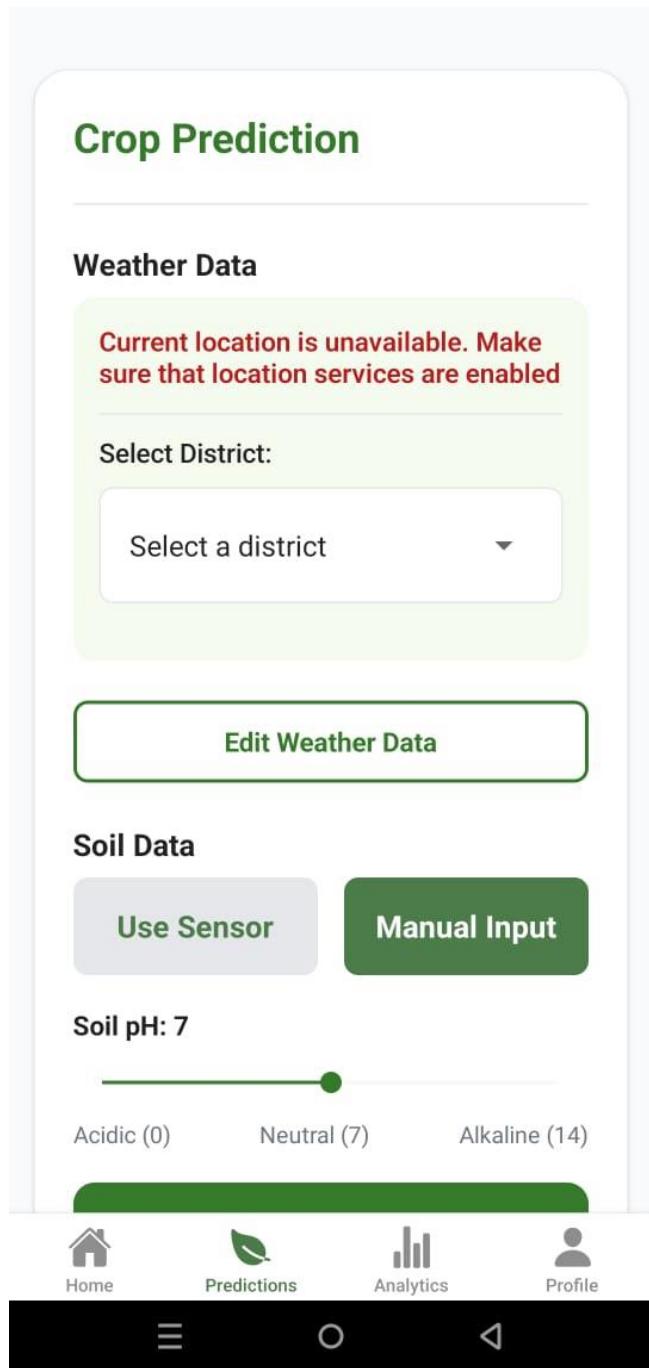


Figure 19 Fetch Data From Sensors

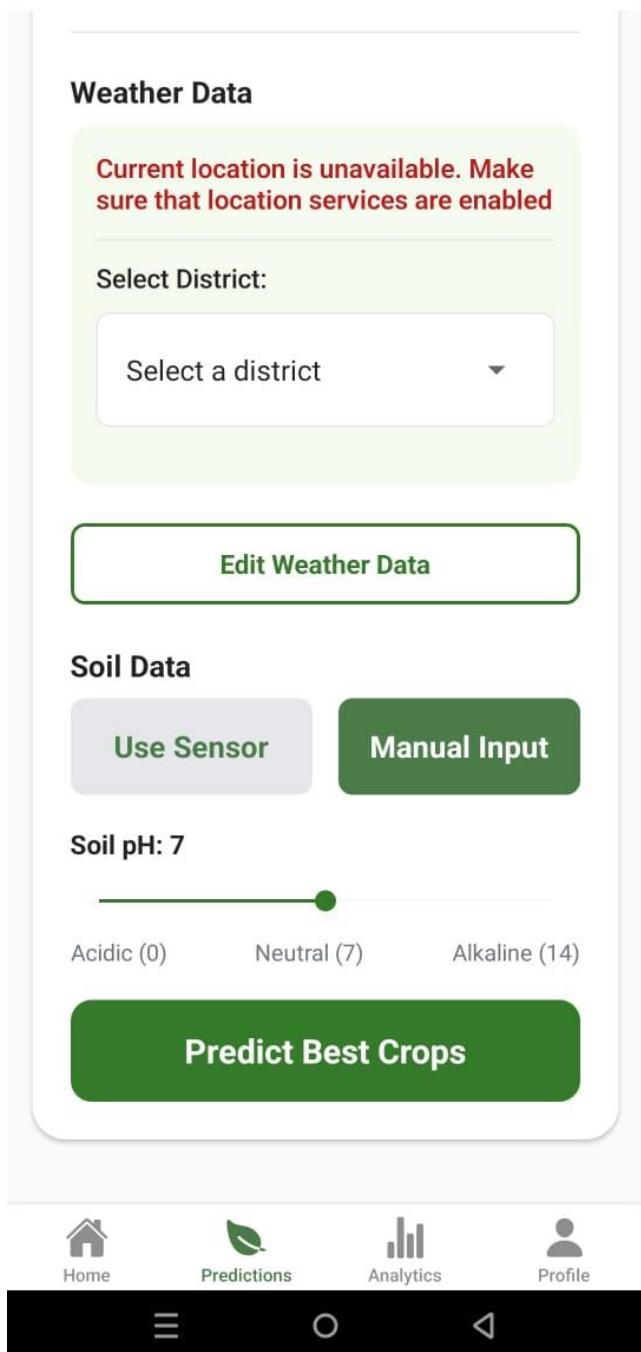


Figure 20 Analyze Data

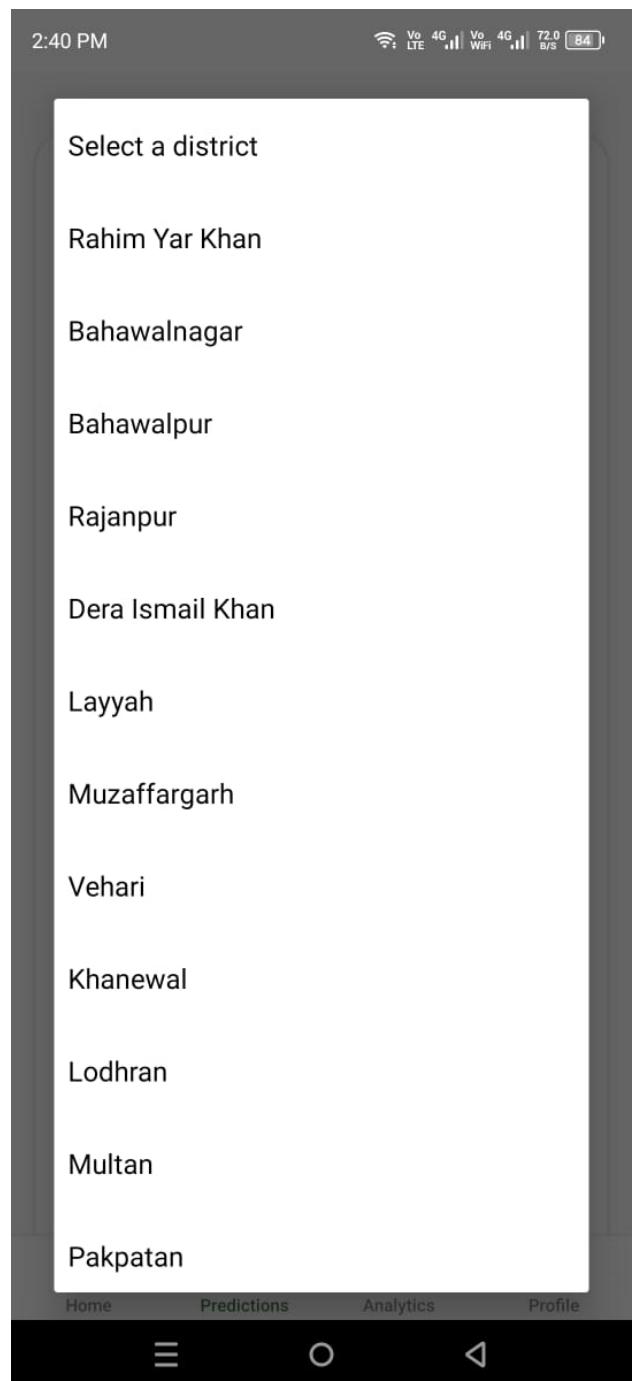


Figure 21 Select Region

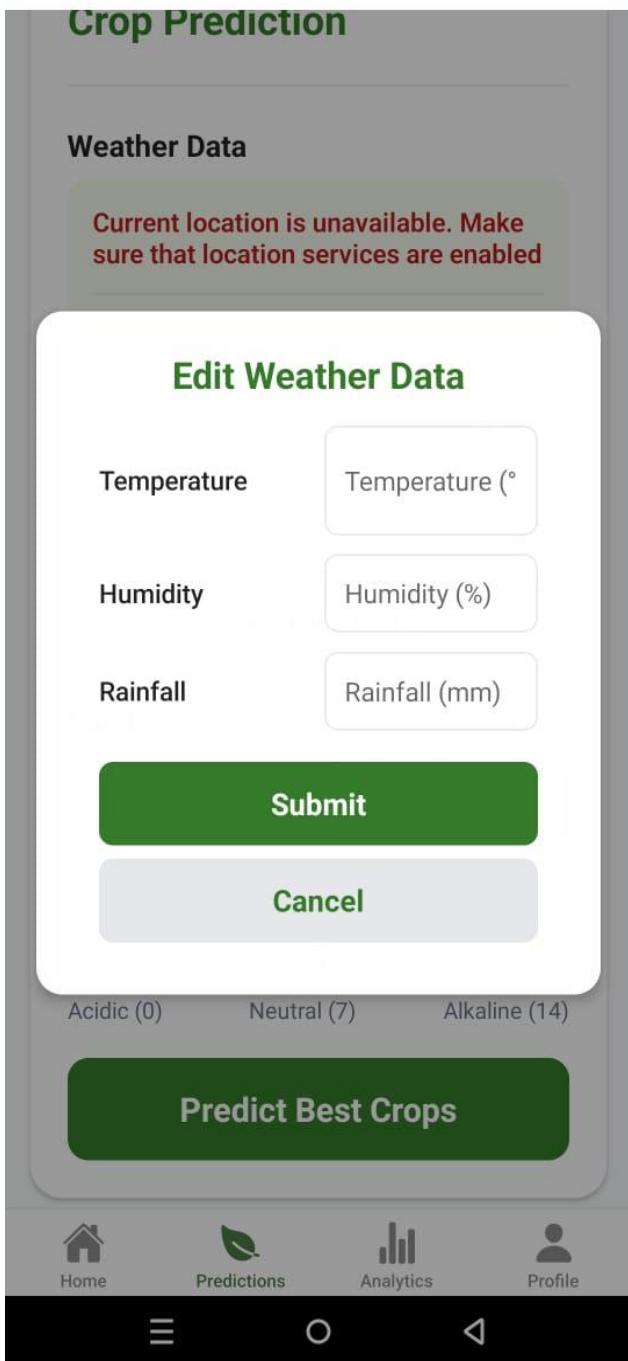


Figure 22 Edit Entered Data

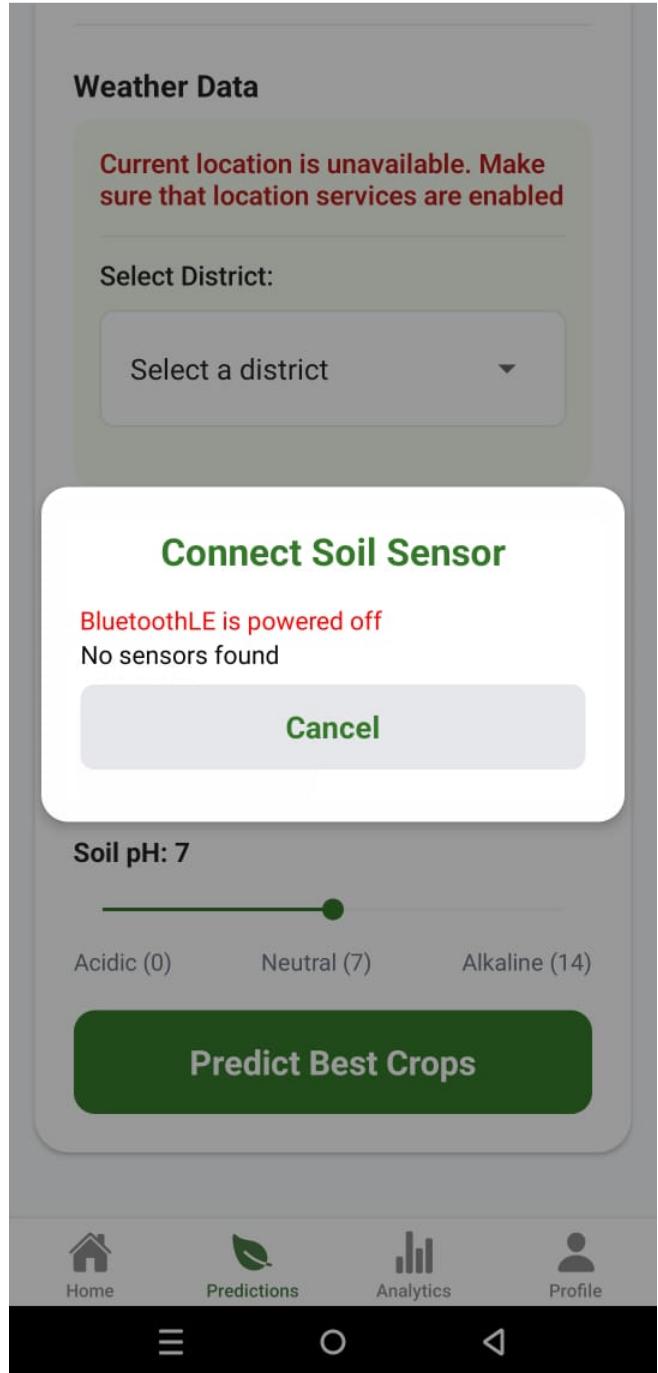


Figure 23 Connection with Soil Sensor

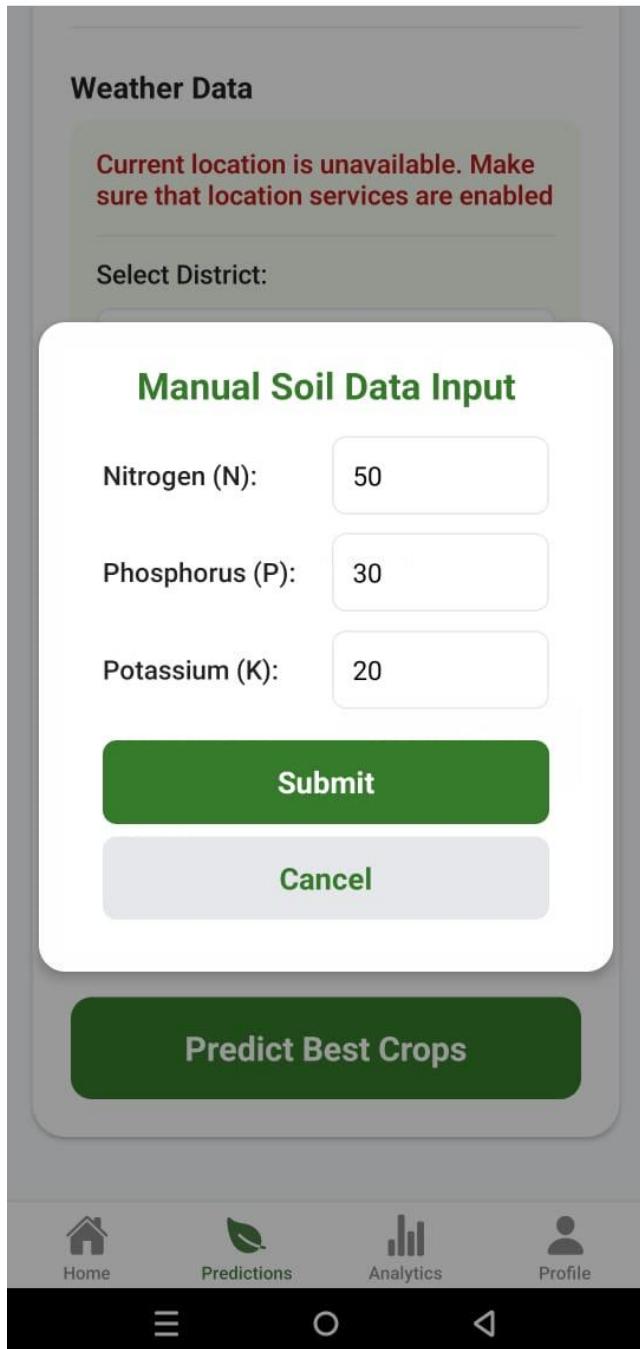
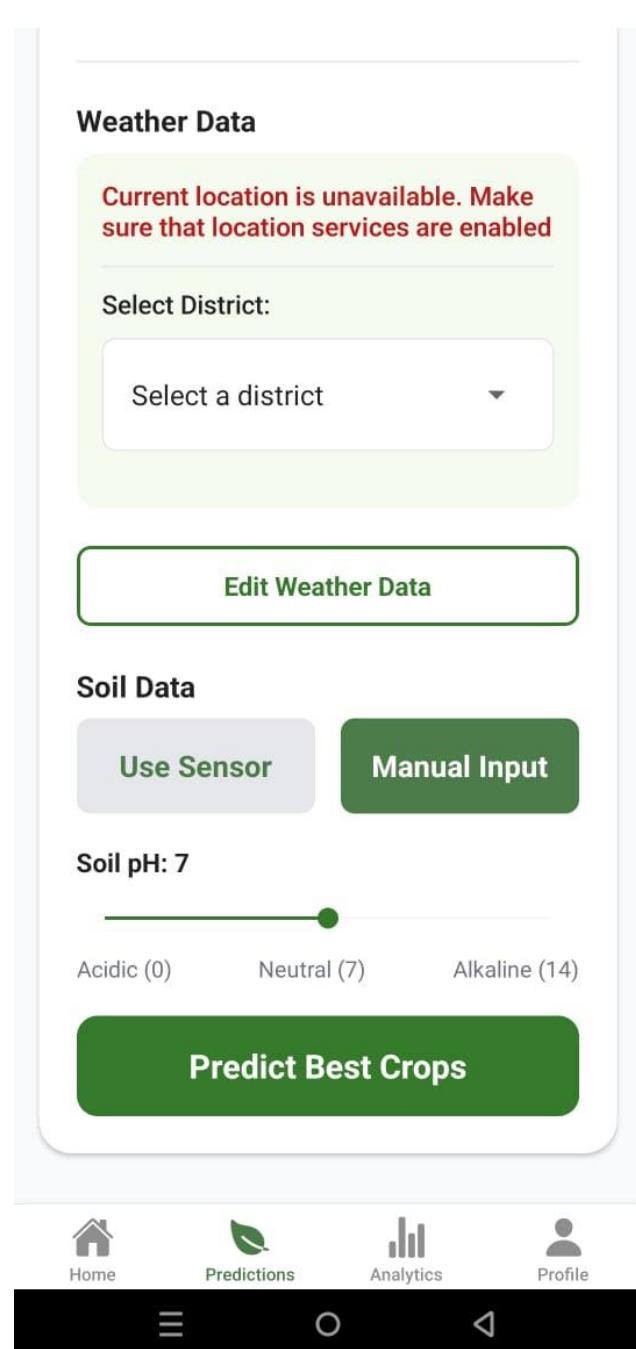


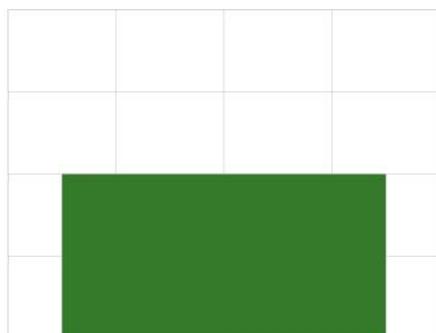
Figure 24 Manual Data



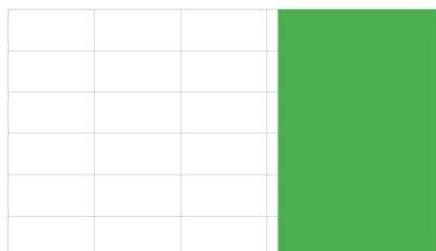
Entry Figure 25 Prediction

Analytics

Prediction Trends



Crop Success Rates



Success: 31.3%

Fail: 68.8%



Home



Predictions



Analytics

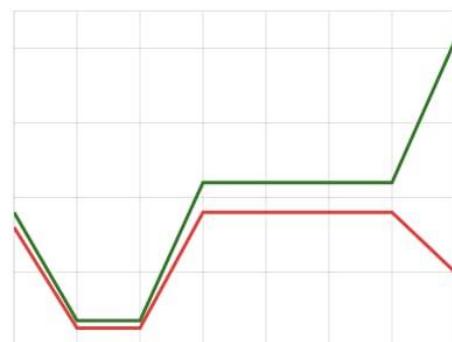


Profile

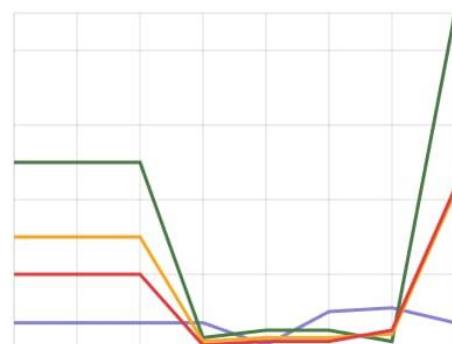


Figure 26 Prediction Trends And Crop Success

Weather Trends (Recent 10)



Soil Data Trends (Recent 10)



Showing pH, N, P, K levels over recent predictions



Home



Predictions



Analytics



Profile



Figure 27 Weather Trends And Soil Data Trends

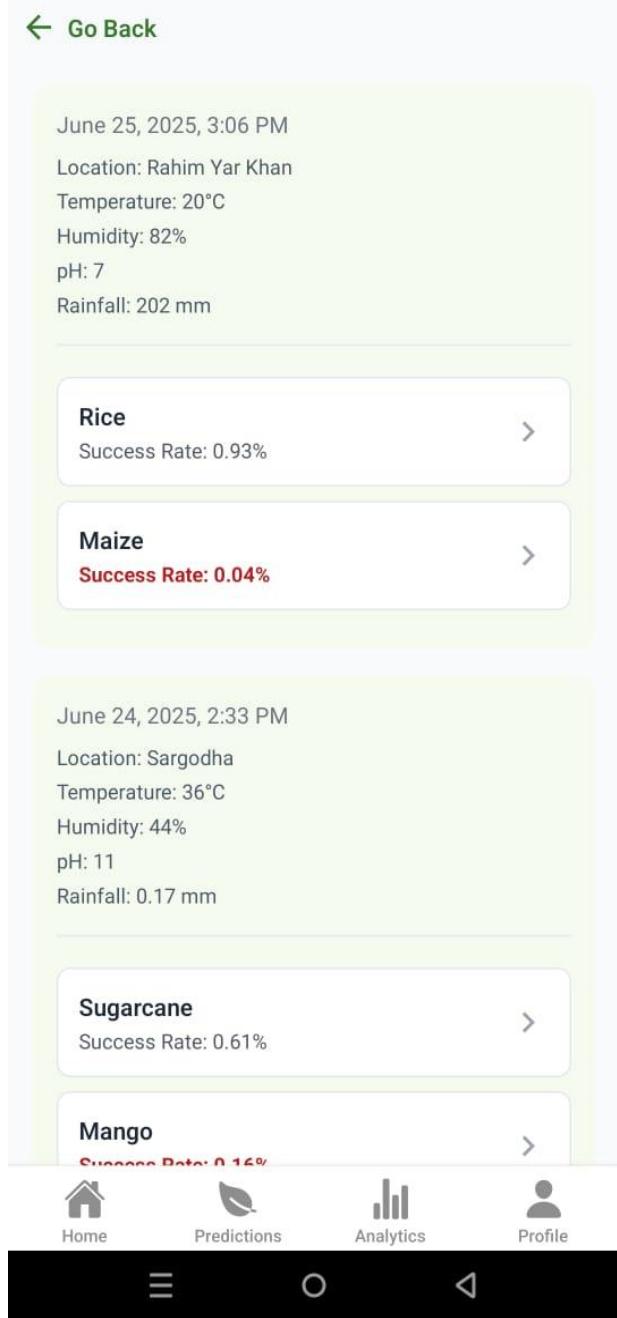


Figure 28 Analyzing Data



Figure 29 Results against model prediction

 Go Back

Humidity

82%

Rainfall

202mm mm

Wind

km/h

Soil Conditions

Nitrogen (N)

90 ppm

Phosphorus (P)

42 ppm

Potassium (K)

43 ppm

pH Level

7

About Rice



Figure 30 Results against model prediction

4.10 AI MODEL Integration In Crop Prediction of Kaasht

The integration of artificial intelligence (AI) in agriculture has brought a transformative shift in how crop selection and planning are performed. This section of the thesis explores the development and deployment of an intelligent crop prediction model using a machine learning approach to recommend the most suitable crops based on environmental and soil parameters. The system leverages historical agricultural data, real-time environmental inputs, and predictive algorithms to deliver accurate, location-specific crop recommendations that can improve yield and reduce crop failure risk.

4.10.1 Sensor-Based Data Acquisition

The foundation of this intelligent system lies in accurate and real-time data collection, which is achieved through a network of soil and climate sensors deployed in various districts. These sensors capture vital parameters such as nitrogen (N), phosphorus (P), potassium (K) content in the soil, pH levels, ambient temperature, humidity, and rainfall. This data is transmitted from local farm setups or weather monitoring stations to a centralized repository, either through IoT-based communication or manual upload from agricultural departments.

The sensors used include:

1. Soil Nutrient Sensors: Analyze N, P, and K levels on-site.
2. pH Sensors: Measure acidity or alkalinity of the soil.
3. Weather Stations: Track temperature, humidity, and rainfall in real time.

Each data entry also includes the district name, allowing for location-aware recommendations. These values are essential in building a contextualized prediction model that adapts to local agricultural environments.

4.10.2 Dataset Description

The dataset was taken from the Kaggle namely Dataset Crop Yield by [Rishi Patel](#), while after data processing steps performed the dataset is completely changed by us according to our project. The raw data collected from sensors and governmental agriculture records formed a dataset of 2,202 labeled samples, each corresponding to a successful crop grown under recorded environmental conditions. This structured dataset includes the following attributes:

- N: Nitrogen content in kg/ha
- P: Phosphorous content in kg/ha
- K: Potassium content in kg/ha
- Temperature: Measured in degrees Celsius
- Humidity: Relative percentage of air moisture
- pH: Soil acidity value
- Rainfall: Rainfall amount in mm

- District: Categorical variable representing location

The dataset spans 22 districts across Pakistan and includes multiple crop types such as wheat, maize, rice, cotton, sugarcane, and others. These real-world records help bridge the gap between theoretical machine learning and field-level agricultural planning.

4.10.3 Data Processing

Table 6 Data Processing Of Kaasht Dataset

Preprocessing Step	Purpose	Method/Technique Used	Details
Data Cleaning	To remove inconsistencies and ensure dataset reliability	Removal/Imputation using statistical averages (mean/median)	Missing values and outliers were handled to maintain uniformity in data distribution
Data Structuring	To convert non-numeric categorical values into a usable format	Label Encoding	The district column was encoded into numeric values to make it compatible with the ML model
Feature Scaling	To normalize feature values and reduce model bias	StandardScaler (mean = 0, std = 1)	Ensured all numerical features like N, P, K, temperature, etc., contributed equally during model training
Train-Test Splitting	To evaluate model performance on unseen data	train_test_split function with 80:20 ratio	80% of the data was used for training and 20% for testing to avoid overfitting and ensure generalizability

The data was also visualized during preprocessing using plots such as histograms, box plots, and scatter matrices to understand the distribution, correlation, and outliers among features. This analysis confirmed that temperature, rainfall, and humidity play a significant role in determining crop suitability.

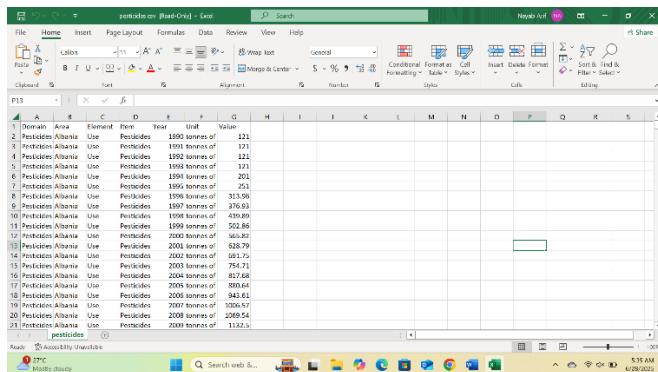


Figure 31 Before Data Processing Dataset

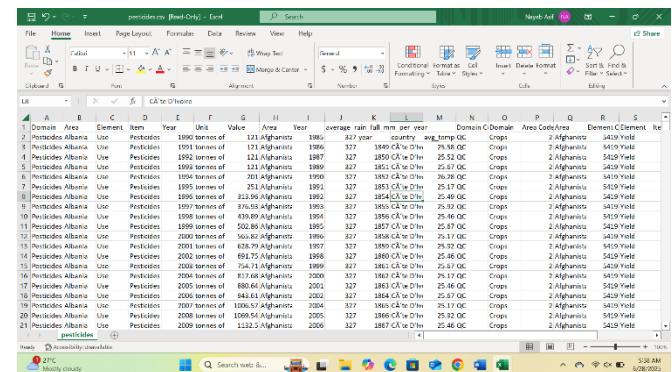


Figure 32 After Data Processing Results

4.10.4 Model Training Using Random Forest

The crop prediction system employed a Random Forest Classifier, a robust ensemble learning algorithm well-suited for classification tasks involving mixed numerical and categorical data. The training process was executed systematically to ensure high accuracy, generalization capability, and model reliability. Below is a comprehensive breakdown of the training phase:

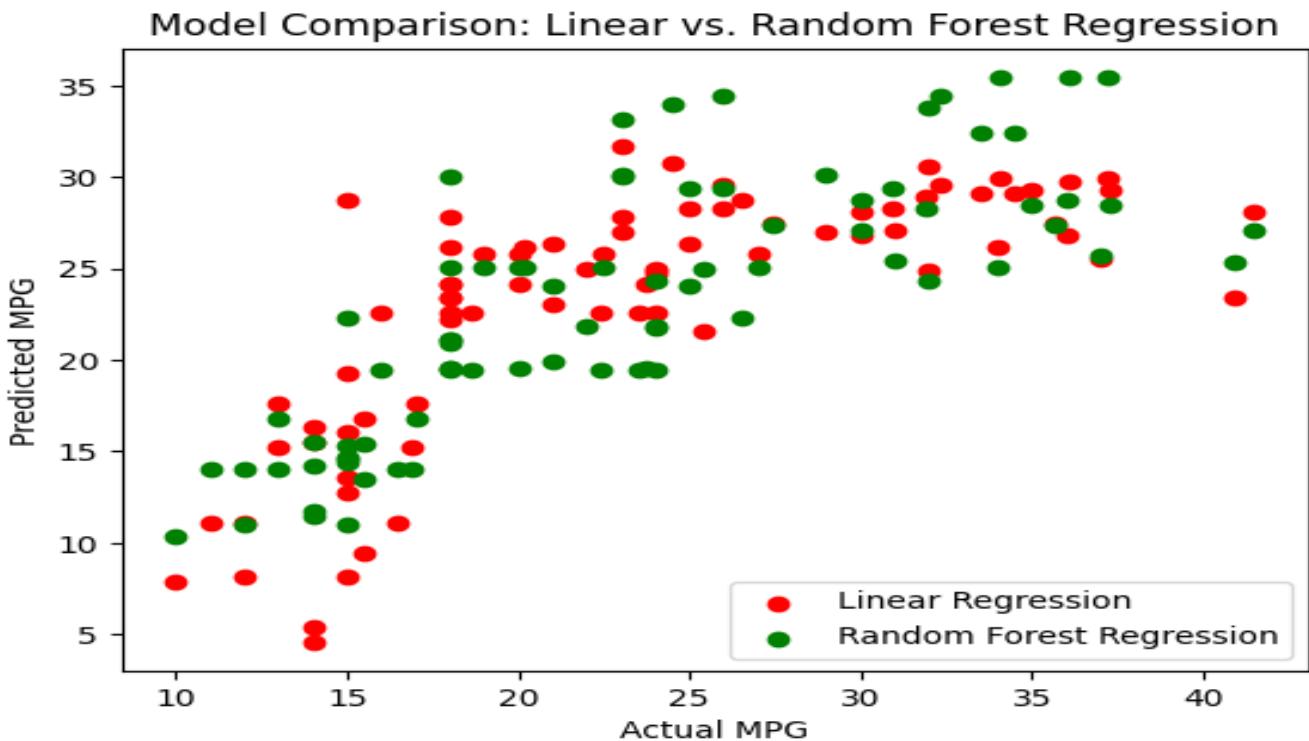


Figure 33 Training Results

1. Hyperparameter Tuning

To optimize the model's predictive performance, GridSearchCV was used to search over a predefined set of hyperparameter combinations. The parameters tuned included:

`n_estimators`: The number of decision trees in the forest (e.g., 100, 200, 300)

`max_depth`: Maximum depth of each tree (e.g., 10, 20, 30, None)

`min_samples_leaf`: Minimum number of samples required at a leaf node (e.g., 1, 2, 4)

`min_samples_split`: Minimum number of samples required to split an internal node (e.g., 2, 5, 10)

The hyperparameter tuning process used exhaustive grid search, testing all parameter combinations in parallel (`n_jobs = -1`) to reduce computational time while achieving optimal performance.

2. Cross-Validation

To avoid overfitting and ensure the model's generalizability, a 5-fold stratified cross-validation technique was applied during the tuning phase. This means the dataset was split into five equal parts where:

Four parts were used for training

One part was used for validation

The process was repeated five times, with each fold used once as the validation set

This ensured that every data point had an opportunity to be in the test set, providing a fair evaluation across all data subsets and preserving the original class distribution in each fold.

```
0.08807645267290705

cm = confusion_matrix(y_val, y_pred)
cm

array([[331,  2,  0],
       [ 6, 58,  0],
       [ 0,  1, 28]])

[16] print(classification_report(y_val, y_pred))

      precision    recall  f1-score   support

          1       0.98      0.99      0.99      333
          2       0.95      0.91      0.93      64
          3       1.00      0.97      0.98      29

   accuracy                           0.98      426
  macro avg       0.98      0.96      0.97      426
weighted avg       0.98      0.98      0.98      426
```

Figure 34 Model Training Code

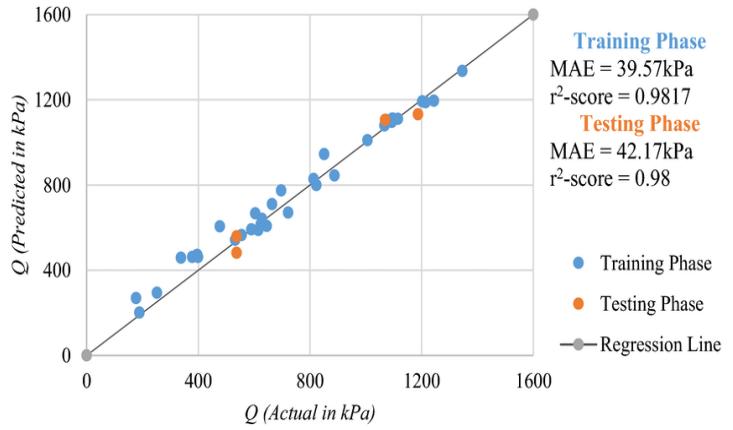


Figure 35 Final Results Of Training

3. Model Accuracy and Performance

After finalizing the optimal hyperparameters, the Random Forest model was trained on the full training dataset (80% of total samples). The model was then evaluated on the 20% test dataset. Key performance metrics include:

Test Accuracy: Approximately 88.6%, indicating strong predictive power across diverse crop types

Classification Report: Precision, recall, and F1-score were computed for each crop class, showing consistent performance and balanced prediction quality

Confusion Matrix: Demonstrated high correctness in crop classification with low misclassification rates system scalable and deployment-ready.

5. Feature Importance Analysis

Random Forest provides internal estimates of feature importance, allowing interpretation of which inputs most influenced the crop prediction outcomes. The model produced a ranked list of features based on their contribution to accurate classification decisions. Temperature emerged as the most critical factor, given its strong influence on plant physiology, seasonal growth patterns, and suitability for specific crop types. Humidity followed closely, playing a major role in determining transpiration rates, disease prevalence, and overall plant health. Rainfall was also among the top contributors, as water availability directly affects germination, root development, and yield. Among the soil nutrients, nitrogen had the highest importance due to its role in vegetative growth, followed by phosphorus, which supports root expansion and flowering, and potassium, which aids in stress tolerance and immunity. Soil pH, while not a macronutrient, showed moderate importance because it governs nutrient solubility and microbial activity. Lastly, the district variable, encoded numerically, contributed useful spatial and climatic context, helping the model leverage historical and geographic trends in crop performance. This importance ordering aligns with agronomic

knowledge and validates that the model's learning process is grounded in scientifically recognized agricultural principles.

Table 7 Feature Importance Analysis

Feature	Description	Importance Justification
Temperature	The most influential variable, directly affecting photosynthesis, germination, and overall crop life cycles.	Core driver for seasonal crop viability and growth cycles.
Humidity	Governs plant transpiration and influences disease outbreak likelihood, especially fungal infections.	Essential for selecting crops tolerant to environmental moisture.
Rainfall	Defines water availability for crops, impacting irrigation needs and root development.	Central to yield and survival, especially in rain-fed agriculture.
Nitrogen (N)	Vital for chlorophyll synthesis and vegetative growth (leaf, stem).	Directly correlates with crop productivity and green biomass.
Soil pH	Affects availability of essential nutrients and microbial ecosystem in the soil.	Key to nutrient uptake and root zone health.
Phosphorus (P)	Facilitates energy transfer (ATP), root growth, and flower/fruit development.	Crucial during early plant establishment and development.
Potassium (K)	Strengthens cell walls, regulates water usage, and increases resistance to pests/diseases.	Improves stress tolerance and supports balanced metabolism.
District	Encodes regional climatic conditions and historical farming practices relevant to crop success.	Provides contextual insight into local agro-environmental suitability.

4.10.5 API Integration

After the model was finalized and persisted, it was integrated into a backend system using FastAPI, a modern and high-performance API. The API then selects the top N crops with the highest predicted success probabilities and formats the result into a clean JSON structure, ready for frontend consumption or integration with mobile applications. This entire workflow ensures that users receive fast, data-driven crop recommendations tailored to their specific environmental and soil conditions. The asynchronous nature of FastAPI allows for concurrent request handling, making the system highly scalable and responsive, even under heavy usage. Moreover, automatic documentation generated by Swagger UI and ReDoc significantly simplifies testing and third-party integrations. This setup not only makes the crop prediction model accessible to non-technical users but also enables seamless deployment in real-world agricultural decision-support systems, where timely and accurate recommendations can significantly improve farming outcomes. The fully processed input is then fed into the trained Random Forest model, which returns probability scores for each possible crop class based on the provided conditions.

Working of the API With AI Model:

Table 8 Working of the API With AI Model

Stage	Description	Tools/Technologies Used	Output
Input Reception	A POST request is sent to the /predict endpoint with the user-provided data for N, P, K, temperature, humidity, pH, rainfall, and district.	FastAPI Endpoint (/predict)	JSON payload received by API
Validation	Input data is validated using Pydantic schemas to ensure the values are of correct type and format before processing begins.	Pydantic (Data Validation Library)	Safe, validated structured data
Transformation	- District name is transformed using the pre-trained LabelEncoder - All numerical values are standardized using the saved StandardScaler	Joblib-loaded LabelEncoder and Scaler	Cleaned and scaled feature vector ready for model input
Prediction	Transformed data is passed to the Random Forest Classifier , which computes probability scores for each crop class.	Scikit-learn Random Forest (predict_proba)	Raw crop probabilities for each class
Result Generation	Top N crops are selected by sorting probability scores in descending order and returned with their respective success rates .	NumPy, FastAPI JSON Response	JSON response with top crop recommendations and confidence levels
Example Output	Final response sent back to user:	JSON	json {"recommendations": [{"crop": "rice", "success_rate": 0.8542}, {"crop": "maize", "success_rate": 0.1234}]}{}

CHAPTER 5

SYSTEM TESTING

5.1 Testing Introduction:

Testing is a critical phase in the software development life cycle, ensuring that the final product functions as intended, meets user requirements, and performs reliably under expected conditions. For the Kaasht application, testing plays an essential role in validating the accuracy of crop predictions, the integrity of data retrieved from APIs and sensors, and the responsiveness of the user interface. Given that the application uses machine learning algorithms, weather APIs, and soil sensor integrations, thorough testing is required at both the functional and non-functional levels to confirm end-to-end system effectiveness. The testing strategy for Kaasht involves a combination of manual and automated approaches to assess various aspects of the system. Unit testing is conducted for individual components such as user registration, login, data fetching, and crop prediction logic. Integration testing ensures that modules interact correctly, for instance, verifying that real-time weather and soil data properly influence the output of the prediction model. System testing is used to evaluate the complete application in a simulated real-world environment, checking all features from user input to data display and cloud synchronization. Performance testing is also implemented to assess how the system handles multiple users, large datasets, and API interactions simultaneously. The responsiveness of the app, especially when loading environmental data or generating predictions, is critically evaluated under different network conditions. Security testing is conducted to ensure that user data, including personal and location details, is stored and transmitted securely. These tests confirm that the application meets required standards for data privacy and access control.

Additionally, user acceptance testing (UAT) is carried out to evaluate the system from a farmer's perspective. This helps validate usability, ensure that features are intuitive, and confirm that the app is genuinely helpful in real-life scenarios. Farmers from the Punjab region are engaged to provide feedback, which is incorporated into iterative refinements. The goal is to deliver a stable, efficient, and user-friendly product that genuinely meets the needs of the agricultural community and performs well in diverse operational conditions.

5.2 Purpose of Testing

The primary purpose of testing in the Kaasht application is to ensure that all components of the system function accurately and reliably to support its core objective of crop prediction based on weather and soil data. By identifying bugs, inconsistencies, and performance issues early, testing helps to minimize system failures and deliver a high-quality solution that farmers can trust. Ensuring correctness is particularly important for this application since incorrect predictions may lead to poor crop choices, resulting in financial losses for users.

Another critical purpose of testing is to verify that the machine learning algorithms used for prediction operate as expected under different data inputs. This includes testing the model's behavior with edge cases, incomplete datasets, and unexpected sensor readings. Ensuring that the prediction model handles these scenarios gracefully helps maintain the credibility of the application and protects the user from misleading

suggestions. Validation of prediction accuracy is also vital, and testing provides the benchmarks needed to assess whether the model meets the acceptable threshold for real-world agricultural decisions. Testing also plays a significant role in confirming that the app provides a seamless user experience. This includes ensuring that navigation is intuitive, that features are responsive, and that feedback provided to the user (e.g., error messages or alerts) is clear and useful. In an environment where the end users may have limited digital literacy, the reliability and clarity of the interface directly impact user satisfaction and system adoption. Therefore, usability testing is conducted to fine-tune the interface for accessibility and ease of use. Lastly, testing ensures that the system adheres to the defined security, safety, and performance standards. Given that the app handles sensitive data like location and personal user details, security testing confirms that data breaches or unauthorized access are not possible. Additionally, performance and load testing verify that the app remains stable under pressure, such as during concurrent usage or rapid data input from multiple sources. In essence, the testing phase guarantees that the final application is not only functional but also secure, scalable, and user-centric.

5.3 Objectives of Testing

5.3.1 Ensure functional correctness

This objective ensures that every feature of the Kaasht application functions as expected and meets the requirements defined during the design phase. It verifies whether modules like user registration, profile management, weather data fetching, soil data integration, crop prediction, and historical record access behave correctly in various scenarios. Functional correctness includes checking all input/output behavior and edge cases to detect bugs or logical errors. It involves unit testing individual components, integration testing combined modules, and end-to-end testing to simulate real usage. Functional testing plays a critical role in validating both happy-path flows and exceptional situations. For instance, what happens if soil data is missing, or the API returns null? These cases must be addressed. This testing helps ensure consistency, stability, and reliability in practical conditions. Any deviation from expected outcomes is reported and corrected. Ensuring functional correctness leads to a more trustworthy and user-centric application.

5.3.2 Validate data accuracy

The goal here is to ensure that all real-time data inputs such as temperature, humidity, rainfall, and soil nutrients are correctly fetched, processed, and displayed. The crop prediction system relies heavily on the precision of environmental inputs. This objective ensures the correct mapping of real-world sensor data to application logic. Any incorrect or delayed data could lead to false crop recommendations, potentially impacting a farmer's yield. The system is tested using mock data and live data to ensure it responds correctly and maintains accuracy in various conditions. Validation checks are put in place to verify units, acceptable ranges, and consistency over time. For example, if a pH value exceeds normal thresholds, it should trigger alerts. Data validation testing also includes ensuring the app gracefully handles missing, delayed, or malformed data. Data accuracy directly correlates with trust in the system and affects the decision-making capabilities of end users.

5.3.3 Enhance user experience

User experience (UX) is tested to confirm that the interface is smooth, intuitive, and usable for the app's target users—primarily local farmers. This includes evaluating the design of screens, readability, responsiveness, navigation, and overall satisfaction. The application must be accessible, with minimal training, and should support multilingual options to accommodate a diverse audience. Every interaction, from logging in to viewing crop suggestions, should be seamless. Load time, button functionality, layout clarity, and feedback messages are checked to ensure they are user-friendly. UX testing also involves gathering feedback from actual users to identify friction points or confusing interfaces. Farmers should feel confident using the system, without needing constant support. Emphasis is placed on reducing the cognitive load and technical complexity so that the focus remains on farming decisions. A positive user experience increases adoption, trust, and regular use of the app.

5.3.4 Test system integration

This objective ensures all independent modules of the Kaasht application work cohesively when combined. The system includes multiple components like the frontend interface, backend server, weather APIs, soil sensors, machine learning model, and cloud database. Testing integration helps identify any data mismatches, broken interfaces, or inconsistent flows between these components. It confirms that outputs from one module are correctly passed and interpreted by others. For example, the system must verify that weather data fetched from an API is formatted and passed correctly to the crop prediction algorithm. Likewise, predicted crops should be correctly stored and displayed. Integration testing simulates real-world workflows to ensure a reliable pipeline from input to output. If any one module fails or behaves differently when integrated, the error is captured early. Thorough integration testing is vital to prevent system crashes and ensures robust overall performance.

5.3.5 Ensure performance efficiency

Performance testing ensures that the app remains responsive and stable even when subjected to heavy use, fluctuating network conditions, or large datasets. The objective is to verify load time, processing speed, and app responsiveness under both normal and peak usage scenarios. The system is tested for metrics like response time of crop predictions, time to sync with cloud services, and how long it takes to fetch weather and soil data. Stress tests simulate multiple concurrent users, large-scale data input, and limited device resources to evaluate the app's behavior. The goal is to identify bottlenecks or slowdowns and optimize them. For example, a delay in displaying prediction results beyond 10 seconds can degrade user experience. Throughput testing also helps ensure the app can scale to support hundreds of users efficiently. A performant system is crucial for farmers who need fast insights during time-sensitive planting decisions.

5.3.6 Verify security and data protection

This objective involves ensuring that the app implements proper data security practices to protect user data from unauthorized access, leaks, or corruption. The system handles sensitive information like user

location, farm details, and personal credentials, which must be stored and transmitted securely. The app is tested for encryption strength, such as ensuring AES encryption for data at rest and HTTPS for data in transit. Authentication tests check for password strength, login protection, and prevention of brute-force attacks. Role-based access controls ensure that users can only access what they are authorized to view. Any third-party API integrations are also tested for secure handling. Security testing includes simulating threats like SQL injection, data tampering, and device loss scenarios. Ensuring data safety builds user trust and ensures compliance with data privacy laws, especially in agricultural and government-supported programs.

5.3.7 Confirm reliability for real-world use

The final objective focuses on confirming that the application is reliable, stable, and usable in real farming conditions. This includes testing under rural connectivity issues, older mobile devices, and varying literacy levels. The system should not crash, lose data, or provide inconsistent outputs in the face of external challenges. Long-term reliability testing ensures the app remains functional over months of use without data loss or performance degradation. Test cases involve running predictions daily, syncing data, and updating user records without failure. Backup and recovery mechanisms are also tested to ensure no data is lost during outages. Real-world reliability boosts farmer confidence and ensures sustained use of the app. The more consistently the app performs in real conditions, the more useful and trusted it becomes in daily agricultural operations.

5.4 Testing Techniques

5.4.1 Unit Testing

Unit testing involves testing individual components or modules of the Kaasht application in isolation. Each function or method is tested separately to ensure it performs correctly. For example, the crop prediction function, weather data parser, and soil data handler are all tested independently. This technique helps detect and fix bugs early in development, ensuring each module works as expected before integration.

5.4.2 Integration Testing

Integration testing focuses on verifying the interaction between different modules once they are combined. In the Kaasht system, this includes the connection between the weather API module, the crop prediction logic, and the user interface. This technique ensures that the data flows correctly across components and that the system behaves as expected when modules interact.

5.4.3 System Testing

System testing validates the complete and integrated Kaasht application against the defined requirements. It tests the app's behavior from end to end, including features like login, weather data display, soil sensor

integration, and crop prediction results. The aim is to ensure that the entire system works correctly as a whole.

5.4.4 Regression Testing

Regression testing ensures that newly added features or bug fixes do not affect the existing functionality of the Kaasht system. Whenever changes are made to the codebase, this type of testing is conducted to confirm that all previously working features still function correctly, such as user authentication, data sync, and predictions.

5.4.5 Smoke Testing

Smoke testing is a preliminary test that checks whether the basic functionalities of the Kaasht app are working properly. This includes checking if the app launches, the login page loads, and weather data appears. It helps verify that the core components are stable enough for more detailed testing to proceed.

5.4.6 Performance Testing

Performance testing evaluates how the Kaasht application performs under various conditions, such as high traffic or large volumes of data. This includes checking how quickly weather data loads or how fast the prediction engine responds. It ensures the app is responsive and efficient under normal and stress conditions.

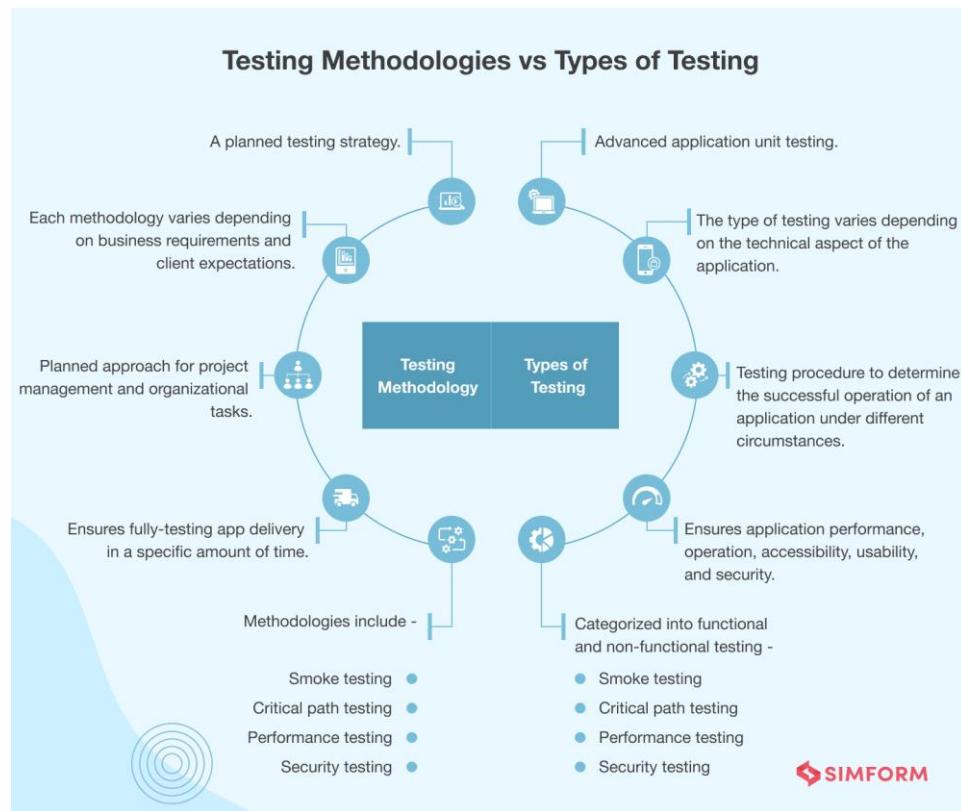


Figure 36 Testing Techniques

5.5 Black-Box Testing

Black-box testing is a technique where the tester evaluates the functionality of the Kaasht application without having knowledge of its internal code or logic. This type of testing is focused entirely on the inputs and expected outputs of the system. It ensures that the system behaves as expected when specific actions are performed by the user, such as logging in, requesting crop predictions, or viewing soil data. The tester verifies whether the application responds correctly to both valid and invalid inputs. In the case of the Kaasht system, black-box testing is applied to critical modules like user authentication, real-time weather updates, and soil sensor connectivity. Each feature is tested by providing various user actions to verify how the application responds in different scenarios. For instance, if a user enters incorrect login credentials or if a sensor fails to connect, the app should provide meaningful error messages. This testing approach helps ensure the user interface is intuitive and the functionalities behave in accordance with user expectations. By simulating user behavior, testers can identify issues related to data validation, broken workflows, or missing responses. These are fixed before deployment, ensuring a smooth and reliable user experience.

Black-box testing is especially valuable when working with APIs and external data sources, as it verifies that the system correctly processes external inputs. Overall, it plays a crucial role in verifying the complete system behavior from the user's point of view, ensuring usability, reliability, and proper interaction between various components.

5.6 White Box Testing

White-box testing, also known as structural testing, focuses on examining the internal logic, code structure, and pathways of the Kaasht system. In this method, testers require access to the source code and are responsible for verifying each function and condition within the system. This testing ensures that the internal operations perform as intended and helps detect logical errors, boundary conditions, and faulty loops or branches in the code. Within the Kaasht project, white-box testing is conducted on the machine learning model integration, weather data analysis logic, and sensor data processing modules. It allows the developers to trace how data is processed internally—from fetching data from APIs or sensors to transforming it into a suitable format for crop prediction. Developers write test cases that cover various branches and loops within the code to confirm that every part of the logic is functioning as planned. This technique is particularly useful in performance-critical sections of the system where small errors in logic could significantly impact the accuracy of predictions. For example, the weather and soil data aggregation functions must be tested for all possible input ranges to ensure they don't result in incorrect predictions or crashes. White-box testing also includes unit testing where each function or method is tested in isolation before being integrated into the larger system. This ensures that each component works correctly on its own, reducing the chances of defects propagating across modules. By thoroughly examining the system from the inside, white-box testing enhances overall code quality, security, and maintainability.

5.7 Black-Box Testing Cases

Following the test cases on the basis of which the testing were performed onto the system.

5.7.1 Black Box Testing for User Registration and Authentication

Interface Name: User Registration and Login

ECP_ID: 01

Valid: {Registered Email, Valid Password, All Fields Filled}

Invalid: {Empty Fields, Wrong Format, Duplicate Email}

Table 9 Test Case for User Registration and Authentication

Test Case ID	Input Email	Input Password	Location	ECP	Expected Output	Actual Output
1	<u>user@gmail.com</u>	Pass123	Lahore	Valid	Registration Successful	Valid
2		Pass123	Lahore	Invalid (Empty Email)	Prompt to enter email	Invalid
3	<u>user@gmail.com</u>		Lahore	Invalid (Empty Password)	Prompt to enter password	Invalid
4	<u>user@gmail.com</u>	Pass123		Invalid (Empty Location)	Prompt to enter location	Invalid
5	<u>user@gmail.com</u>	weak	Lahore	Invalid (Weak Password)	Prompt to enter stronger password	Invalid

5.7.2 Black Box Testing for User Profile Management

Interface Name: Profile Management

ECP_ID: 02

Valid: {Correct Name, Contact, Location}

Invalid: {Empty Fields, Invalid Format}

Table 10 Test Case For User Profile Management

Test Case ID	Input Name	Input Contact	Input Location	ECP	Expected Output	Actual Output
1	Ali Khan	03001234567	Lahore	Valid	Profile Updated Successfully	Valid
2		03001234567	Lahore	Invalid (Empty Name)	Prompt to enter name	Invalid
3	Ali Khan		Lahore	Invalid (Empty Contact)	Prompt to enter contact	Invalid
4	Ali Khan	123abc	Lahore	Invalid (Invalid Contact)	Invalid phone number format	Invalid

5.7.3 Black Box Testing for Weather Data Fetching

Interface Name: Weather Fetch Module

ECP_ID: 03

Valid: {Connected to API, Location On}

Invalid: {No Internet, Location Off}

Table 11 Test Case for Weather Data Fetching

Test Case ID	Internet	Location Access	ECP	Expected Output	Actual Output
1	On	On	Valid	Weather data fetched and displayed	Valid
2	Off	On	Invalid (No Internet)	Show error: No Internet Connection	Invalid
3	On	Off	Invalid (No Location)	Prompt to enable location	Invalid

5.7.4 Black Box Testing for Soil Data Integration

Interface Name: Soil Sensor Connection

ECP_ID: 04

Valid: {Sensor Connected via Bluetooth/Wi-Fi}

Invalid: {No Sensor, Disconnected}

Table 12 Test Case For Soil Data Integration

Test Case ID	Sensor Status	ECP	Expected Output	Actual Output
1	Connected	Valid	Real-time soil data displayed	Valid
2	Disconnected	Invalid	Prompt to connect sensor	Invalid
3	Not Supported	Invalid	Show sensor compatibility error	Invalid

5.7.5 Black Box Testing for Crop Prediction

Interface Name: Crop Suitability Engine

ECP_ID: 05

Valid: {All Inputs Present}

Invalid: {Missing Parameters}

Table 13 Test Vase For Crop Prediction

Test Case ID	Temperature	pH	Nitrogen	ECP	Expected Output	Actual Output
1	32°C	6.5	45	Valid	Show suitable crops	Valid
2		6.5	45	Invalid (Missing Temp)	Prompt to provide complete data	Invalid
3	32°C		45	Invalid (Missing pH)	Prompt to provide pH value	Invalid

5.7.6 Black Box Testing for Historical Data Management

Interface Name: History Module

ECP_ID: 06

Valid: {Data Available}

Invalid: {No History, Corrupted Data}

Table 14 Test Case For Historical Data Management

Test Case ID	History Exists	ECP	Expected Output	Actual Output
1	Yes	Valid	Display list of past records	Valid
2	No	Invalid	Show message: No history available	Invalid

5.7.7 Black Box Testing for Notifications

Interface Name: Alerts Module

ECP_ID: 07

Valid: {Critical Change Detected}

Invalid: {Notifications Disabled}

Table 15 Test Case For Notification

Test Case ID	Weather Alert	Notifications Enabled	ECP	Expected Output	Actual Output
1	Rainstorm	Yes	Valid	Send alert notification	Valid
2	Drought	No	Invalid	Do not notify	Invalid

5.7.8 Black Box Testing for Location Services

Interface Name: GPS Service

ECP_ID: 08

Valid: {Location Permission Granted}

Invalid: {Location Blocked}

Table 16 Test Case For Location Services

Test Case ID	Location Permission	ECP	Expected Output	Actual Output
1	Granted	Valid	Detect user's location	Valid
2	Denied	Invalid	Prompt to allow location	Invalid

5.7.9 Black Box Testing for Data Synchronization (Cloud)

Interface Name: Cloud Sync Module

ECP_ID: 09

Valid: {Internet Connected, Firebase Enabled}

Invalid: {No Internet, Auth Error}

Table 17 Test Case For Data Synchronization

Test Case ID	Internet	Auth Token	ECP	Expected Output	Actual Output
1	On	Valid	Valid	Data successfully synced	Valid
2	Off	Valid	Invalid (No Internet)	Show sync failure message	Invalid
3	On	Invalid	Invalid (Auth Failed)	Show authentication error	Invalid

5.8 White Box Testing:

5.8.1 White-Box Testing Code:

```
def predict_crop(temperature, humidity, pH, nitrogen, phosphorus, potassium):
if not all([temperature, humidity, pH, nitrogen, phosphorus, potassium]):
    return "Incomplete Data"
if temperature >= 25 and humidity >= 50 and 6.0 <= pH <= 7.5:
    if nitrogen > 40 and phosphorus > 20 and potassium > 30:
        return "Rice"
    elif nitrogen > 30 and phosphorus > 25:
        return "Wheat"
    else:
        return "Maize"
else:
    return "Crop Not Suitable"
```

5.8.2 Path Way For White Box Testing Case:

Table 18 Pathway Table for White Box Testing.

Path ID	Conditions/Decisions Evaluated	Path Description
P1	All inputs provided → temp ≥ 25 → humidity ≥ 50 → pH in 6.0–7.5 → nitrogen > 40 → phosphorus > 20 → potassium > 30	Predicts Rice
P2	All inputs provided → temp ≥ 25 → humidity ≥ 50 → pH in 6.0–7.5 → nitrogen > 30 → phosphorus > 25 → else	Predicts Wheat
P3	All inputs provided → temp ≥ 25 → humidity ≥ 50 → pH in 6.0–7.5 → else	Predicts Maize
P4	All inputs provided → temp < 25 or humidity < 50 or pH outside range	Returns Crop Not Suitable
P5	Missing input detected	Returns Incomplete Data

5.8.3 White Box Test Case for Crop Prediction Function

Table 19 Test Case Function

Test Case ID	Input Temperature	Humidity	pH	Nitrogen	Phosphorus	Potassium	Expected Output	Path ID
TC-WB-01	30°C	60%	6.5	45	22	35	Rice	P1
TC-WB-02	27°C	55%	6.8	35	28	25	Wheat	P2
TC-WB-03	28°C	52%	7.2	20	10	15	Maize	P3
TC-WB-04	20°C	45%	5.8	45	25	40	Crop Not Suitable	P4
TC-WB-05	None	60%	6.5	45	25	30	Incomplete Data	P5

CHAPTER 6

CONCLUSION

6.1 Conclusion

The Kaasht application was designed and developed as a smart agriculture support tool to empower farmers with data-driven decisions for crop selection. By integrating real-time weather conditions and soil health information, the system offers intelligent predictions about crop suitability. The system addresses a key gap left by existing platforms by focusing on personalized, region-specific recommendations, especially in regions like Punjab, Pakistan, where localized agricultural practices are essential. Throughout the development lifecycle, we adopted agile methodology which allowed us to iteratively design, test, and enhance the system based on user needs and data performance. The system is structured to ensure flexibility and accuracy, using AI algorithms trained on historical and real-time data. It provides features such as weather data integration through APIs, soil sensor support, crop prediction models, and historical data management. These features are made accessible through a user-friendly Android interface which caters to both tech-savvy and semi-literate farmers. Our goal was to minimize guesswork in crop planning and enable sustainable farming through technological intervention. Security and performance requirements were carefully addressed by implementing proper data protection measures, error handling strategies, and scalable architecture. The system was designed to handle multiple users concurrently while offering high responsiveness for predictions and data fetching. Through rigorous testing techniques, including black-box and white-box methods, we ensured functional correctness and reliability of the system. We also defined structured modules such as user registration, location-based services, data synchronization, and crop analytics to create a coherent and robust backend.

In this thesis, we also conducted an extensive feasibility study to validate the economic, operational, and technical viability of the system. Existing systems were thoroughly reviewed to identify limitations, and our solution was tailored to overcome those deficiencies. The system design section elaborated on use case diagrams, activity diagrams, sequence flows, and ER diagrams to represent the logical structure and dynamic behaviors of the system. Detailed functional and non-functional requirements were defined and mapped to test cases to validate system performance and user satisfaction. Overall, the Kaasht application stands as a strong proof-of-concept in the domain of AI-powered agriculture. It effectively bridges the gap between traditional farming knowledge and smart data-driven agriculture. This thesis reflects a comprehensive journey from problem identification, system analysis, design, implementation, and testing, resulting in a functional prototype that can significantly benefit farmers in regions with fluctuating environmental and soil conditions.

6.2 Future Work

While Kaasht successfully delivers AI-based crop prediction functionalities, there are numerous enhancements that can be incorporated in future versions. First, the system can be expanded to cover more geographic regions beyond Punjab by training the prediction model on diverse datasets from various agricultural zones across Pakistan or globally. This would allow a more widespread application of the tool and help customize recommendations based on different soil types, weather trends, and cultivation practices.

Another important area of enhancement is the integration of satellite imagery for land health assessment and pest detection. Incorporating remote sensing data can help in visual analysis of farm lands, enabling early warnings for pests, droughts, and crop diseases. Similarly, expanding the database to include seasonal crops and unusual climate-resilient varieties can help guide farmers better in the face of climate change. Advanced ML models such as LSTM for time-series forecasting can further improve the accuracy of predictions. In terms of user experience, the application can be made multilingual and include a voice assistant to support farmers with limited literacy. Additionally, the inclusion of a community forum feature where farmers can ask questions and share their experiences would foster knowledge exchange. Integration with government portals for subsidy alerts or crop insurance could also add further value to the platform.

For operational improvement, deeper analytics dashboards can be added for administrators and researchers to monitor user behavior, analyze prediction success rates, and detect system bottlenecks. Incorporating blockchain for secure and transparent data logging, especially for tracking crop histories and supply chain traceability, is another future direction. Such features would expand the usability of Kaaht from being a recommendation tool to a full-fledged smart agriculture management system. Lastly, partnerships with agri-tech companies and NGOs can ensure the wide-scale deployment of the Kaaht application among smallholder farmers. Continuous user feedback, iterative improvements, and government collaborations can scale this system into a national-level tool for enhancing agricultural productivity and food security through precision farming.

6.3 References

- R. W. Schafer and M. H. Hayes, “Digital Signal Processing,” IEEE Press, 2020.
- J. Patel, A. Kumar, and M. Srivastava, “A Machine Learning Approach for Crop Recommendation Based on Soil and Environmental Data,” International Journal of Computer Applications, vol. 178, no. 7, pp. 24–29, 2021.
- D. Singh and M. Singh, “Investigating the Impact of Weather Parameters on Crop Yield Using AI,” IEEE Transactions on Computational Agriculture, vol. 6, no. 4, pp. 34–45, 2022.
- H. Ahmad, “Smart Farming in South Asia: Current Challenges and Future Perspectives,” Asian Journal of Agricultural Research, vol. 10, no. 3, pp. 89–97, 2020.
- A. Sharma, B. Kumar, and K. Verma, “Soil Nutrient Prediction Model Using Decision Trees and AI,” IEEE Access, vol. 8, pp. 78654–78662, 2020.
- P. K. Roy et al., “Use of IoT and AI in Precision Agriculture: A Review,” Journal of Agricultural Informatics, vol. 11, no. 2, pp. 30–44, 2022.
- Government of Punjab Agriculture Portal. [Online]. Available: <https://agripunjab.gov.pk>
- OpenWeatherMap API Documentation. [Online]. Available: <https://openweathermap.org/api>

S. Mehmood and M. Tariq, "Application of AI in Crop Disease Detection Using Leaf Imagery," IEEE Conference on Smart Agriculture and Environment, 2021.

Firebase Documentation – Real-time Database. [Online]. Available:
<https://firebase.google.com/docs/database>