A Synopsis of Project on

CropSync:AI-Driven Blockchain Solution for Smart Farming.

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Engineering

in

Computer Science and Engineering (Data Science)

by

Shreyas Revankar(21107065)

Under the Guidance of

Ms. Poonam Pangarkar
Ms. Harsha Zope

Department of Computer Science and Engineering (Data Science)
A.P. Shah Institute of Technology
G.B.Road,Kasarvadavli, Thane(W)-400615
UNIVERSITY OF MUMBAI

Academic Year 2024-2025

Approval Sheet

This Project Report entitled ***"CropSync: AI-Driven Blockchain Solution for Smart Farming"*** Submitted by ***"Shreyas Revankar" (21107065)*** is approved for the partial fulfillment of the requirement for the award of the degree of in ***Bachelor of Engineering in Computer Science and Engineering (Data Science)*** from ***University of Mumbai***.

Ms. Harsha Zope
Co-Guide

Ms. Poonam Pangarkar
Guide

Ms. Anagha Aher
HOD, Computer Science and Engineering (Data Science)

Place:A.P.Shah Institute of Technology, Thane
Date:

CERTIFICATE

This is to certify that the project entitled ***"CropSync: AI-Driven Blockchain Solution for Smart Farming"*** submitted by ***"Shreyas Revankar" (21107065)*** is approved for the partial fulfillment of the requirement for the award of the degree of in ***Bachelor of Engineering in Computer Science and Engineering (Data Science)*** from ***University of Mumbai***, is a bonafide work carried out during academic year 2024-2025.

Ms. Harsha Zope
Co-Guide

Ms. Poonam Pangarkar
Guide

Ms. Anagha Aher
HOD, Computer Science and Engineering (Data Science)

Dr. Uttam D.Kolekar
Principal

External Examiner(s)

1.

2.

Internal Examiner(s)

1.

2.

Place:A.P.Shah Institute of Technology, Thane
Date:

Acknowledgement

I have great pleasure in presenting the report on **CropSync: AI-Driven Blockchain Solution for Smart Farming**. I take this opportunity to express my sincere thanks towards our guide **Ms.Poonam Pangarkar** & Co-Guide **Ms.Harsha Zope** for providing the technical guidelines and suggestions regarding line of work. I would like to express my gratitude towards his constant encouragement, support and guidance through the development of project.

I thank **Ms.Anagha Aher** Head of Department for her encouragement during the progress meeting and for providing guidelines to write this report.

I express my gratitude towards BE project co-ordinators, **Ms.Poonam Pangarkar** for being encouraging throughout the course and for their guidance.

I also thank the entire staff of APSIT for their invaluable help rendered during the course of this work. I wish to express our deep gratitude towards all our colleagues of APSIT for their encouragement.

Shreyas Revankar
(21107065)

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Shreyas Revankar(21107065)

Date:

Abstract

The CropSync project is a transformative initiative aimed at revolutionizing Indian agriculture through the integration of cutting-edge technologies such as artificial intelligence, blockchain, and cloud computing. The mobile application provides farmers with real-time, personalized agricultural tools, enabling them to make data-driven decisions. Key features include live weather updates fetched from a weather API, insights into crop distribution and rainfall patterns, and a comprehensive data analysis dashboard that visualizes land usage and market trends. By addressing critical challenges like climate uncertainty and limited market access, CropSync empowers farmers to optimize their practices, improve productivity, and promote sustainability. With a user-friendly interface and support for diverse farming needs, CropSync aims to enhance the overall efficiency of agricultural operations and foster a more resilient farming ecosystem.

Keywords: *CropSync, AI-driven agriculture, blockchain technology, real-time weather data, sustainable farming, data analysis dashboard.*

Contents

List of Figures

List of Tables

List of Abbreviations

ML:	Machine Learning
AI:	Artificial Intelligence
API:	Application Programming Interface
AWS:	Amazon Web Services
Git:	Version Control System
MUI:	Material UI
AVD:	Android Virtual Device

Chapter 1

Introduction

Agriculture plays a pivotal role in sustaining the livelihoods of millions of people worldwide, particularly in developing countries like India, where a significant portion of the population depends on farming as their primary source of income. Despite being the backbone of the Indian economy, the agricultural sector faces several critical challenges, including unpredictable weather patterns, lack of access to real-time market data, limited financial resources, and insufficient transparency in the food supply chain. The rapid advancement of technology offers immense potential to overcome these challenges. The project, CropSync: AI-Driven Blockchain Solution for Smart Farming, aims to harness the power of artificial intelligence, blockchain, and cloud computing to provide Indian farmers with innovative tools for climate-smart agriculture, financial services, market access, and sustainability certification.This introduction outlines the background, motivation, problem statement, objectives, and scope of this project, which seeks to revolutionize the agricultural industry and promote sustainable farming practices.

Agriculture is the backbone of the Indian economy, providing employment to millions of people, particularly in rural areas. However, Indian farmers are increasingly facing complex challenges such as unpredictable weather patterns, market volatility, and limited access to financial services. These factors, coupled with a lack of tools for sustainable farming, are negatively impacting their productivity and profitability. In response to these growing issues, there is a pressing need for innovative solutions that can empower farmers and enhance their ability to thrive in a dynamic environment. India's agriculture sector is also seeing rising consumer demand for sustainable practices and transparency in the supply chain. Farmers in rural areas often lack access to vital resources like real-time weather data, market prices, and financial planning tools, limiting their ability to make informed decisions. At the same time, consumers are becoming more concerned about the origins of their food, pushing for more visibility into farming practices. The CropSync project aims to address these issues by combining AI-driven agricultural advice with blockchain technology. By offering real-time insights into weather conditions, market trends, and financial services, alongside blockchain-based traceability and sustainability certification, the project empowers farmers to improve their climate resilience, adopt sustainable farming methods, and meet the growing consumer demand for safe, traceable produce.

1.1 Motivation

The agricultural sector stands at a crossroads, facing unprecedented challenges due to climate change, resource depletion, and evolving market demands. As farmers strive to adapt to these changes, the need for innovative solutions that promote sustainable practices and enhance productivity becomes increasingly critical. The integration of artificial intelligence (AI) and blockchain technology presents a unique opportunity to revolutionize the agricultural landscape. By leveraging these technologies, we can empower farmers with tools that not only enhance crop yield and resource efficiency but also ensure transparency and traceability in the supply chain.

The motivation behind CropSync is rooted in the urgent need to support farmers in adopting climate-smart agriculture practices while providing them with access to essential market and financial resources. Traditional farming methods often lack the data-driven insights necessary for informed decision-making, leading to inefficiencies and losses. Our AI-driven platform aims to bridge this gap by offering real-time data analytics, predictive modeling, and tailored recommendations that optimize farming operations. Furthermore, by utilizing blockchain technology, we can ensure secure and transparent transactions, facilitating trust among stakeholders in the agricultural value chain. This holistic approach not only addresses the immediate challenges faced by farmers but also contributes to the long-term sustainability and resilience of the agricultural ecosystem. A chapter can be divided into Sections, Sub-sections and Sub-sub Sections so as to present different concepts separately.

1.2 Problem Statement

The agricultural sector is increasingly confronted with multifaceted challenges that threaten its productivity and sustainability.Farmers often lack access to timely and accurate information on climate conditions, pest management, and market trends, resulting in inefficient decision-making that negatively impacts crop yields. Furthermore, the opacity of the supply chain creates distrust among stakeholders, leading to inefficiencies and lost revenue opportunities. As the world faces the repercussions of climate change, these issues become even more pressing, jeopardizing food security and the livelihoods of countless agricultural workers.

To effectively address these challenges, there is a critical need for an integrated solution that harnesses advanced technologies such as artificial intelligence and blockchain. By providing farmers with real-time data analytics, predictive insights, and a transparent platform for transactions, CropSync aims to empower agricultural stakeholders to make informed decisions and enhance productivity. This innovative approach not only supports farmers in optimizing their operations but also fosters trust and collaboration throughout the supply chain, contributing to the overall resilience and sustainability of the agricultural ecosystem.

1.3 Objectives

The CropSync project is designed with the vision of transforming the Indian agricultural landscape by leveraging cutting-edge technologies like artificial intelligence (AI), blockchain, and cloud computing. Our objectives are aimed at solving some of the most pressing challenges faced by Indian farmers—climate uncertainty, lack of market access, inefficient financial services, and consumer trust in the agricultural supply chain. Each objective has been formulated to address these pain points comprehensively and offer sustainable, scalable solutions. Below is a detailed breakdown of the project objectives:

- The mobile application provides AI-powered, climate-smart agricultural tools for farmers. It delivers real-time, personalized advice on crop planting, irrigation, and pest management. AI models analyze localized weather data, soil health, and market trends to predict risks. The app suggests future crop cycles to enhance resilience, reduce losses, and optimize yields. Crop suitability maps help identify ideal crops based on regional conditions. Smart irrigation scheduling conserves water using soil moisture and weather analysis. Voice-based interaction in local languages makes the app accessible to all farmers. Customizable farm profiles ensure advice is tailored to individual land and crops. IoT integration enables real-time data collection for precise decision-making. The app empowers sustainable, data-driven farming, improving productivity and income..

- The app will integrate real-time data APIs to deliver dynamic, location-specific updates. Weather APIs like OpenWeatherMap will provide accurate forecasts and climate alerts. Farmers will receive timely warnings about extreme events such as floods, droughts, and storms. Market price APIs will offer real-time commodity prices to guide optimal sales timing. Customized alerts will inform farmers of sudden market shifts or peak demand periods. The app will suggest suitable financial products, including loans and crop insurance. APIs will assess credit eligibility and recommend microfinance based on farm data. Government schemes and subsidy updates will be delivered through integrated data feeds. Data-driven financial advice will help farmers plan expenses and reduce risks. These features ensure informed decision-making, increased profitability, and greater security.

- The app will feature a powerful data analysis dashboard tailored for farmers. It will track and analyze market trends to suggest the best times to sell produce. Historical price data will help forecast demand and optimize harvest scheduling. Satellite imagery and AI will provide real-time insights into crop health and growth stages. The dashboard will highlight early signs of stress, disease, or nutrient deficiency. Farmers can compare their farm performance with regional averages for benchmarking. Built-in financial tools will recommend the best loan and insurance options. Data visualizations will help farmers understand complex trends at a glance. All insights will be personalized based on farm size, crop type, and location. This enables smarter decisions, improved yields, and greater financial stability.

- The app will be deployed on scalable cloud platforms like AWS or Google Cloud. This ensures reliable performance and flexibility to handle growing user demand. Cloud infrastructure will support fast data processing, storage, and AI model deployment. Offline functionality will allow uninterrupted access in low-connectivity rural areas.

User training modules will be available in multiple local languages for better adoption. Interactive tutorials and guides will help farmers navigate the app with ease. A dedicated support team will provide assistance through chat, call, or in-app help. Regular updates will improve features, security, and performance over time. User feedback will drive continuous improvement and personalization of services. This approach ensures accessibility, scalability, and long-term user satisfaction.

- The app will use blockchain to securely record eco-friendly agricultural practices. An immutable ledger will ensure transparency and data integrity at every step. Farmers can earn sustainability certifications based on verified green practices. These certifications will help them access new markets and premium pricing. The farm-to-fork traceability feature will allow consumers to track produce origins. Each product can display a scannable QR code detailing its journey and farming methods. This builds consumer trust and encourages responsible consumption. Blockchain will prevent tampering, ensuring authenticity of all recorded data. The system can integrate with regulatory bodies and certifiers for official validation. Ultimately, it empowers farmers to gain recognition and value for sustainable efforts.

1.4 Scope

The scope of this project encompasses a range of innovative features designed to support farmers in adopting climate-smart agricultural practices and improving their overall productivity. By leveraging advanced technologies and datadriven insights, the project aims to address key challenges within the agricultural sector while promoting sustainability and transparency.

- **Climate-Smart Agriculture Tools:** A core component of this project is the development of climate-smart agriculture tools. CropSync will provide localized weather forecasts and climate resilience planning through integrated APIs, enabling farmers to receive timely information about weather conditions. The platform will feature alerts and recommendations that help farmers prepare for extreme weather events, reducing crop loss and ensuring resource management.

- **Market Access and Financial Services:** The project will enhance market access and promote sustainable farming practices. CropSync will include real-time crop price monitoring, allowing farmers to stay informed about market trends and optimize sales strategies. E-commerce integration will facilitate direct sales, enabling farmers to connect with consumers without intermediaries. Access to financial tools like credit and insurance will provide farmers with necessary resources for risk management.

- **Data Analysis Dashboard:** A comprehensive data analysis dashboard will offer farmers insights into market trends and effective farming techniques. By visualizing data, farmers can make informed, data-driven decisions that improve operational efficiency. The platform will empower users to adjust their practices proactively and optimize resources.

- **System Implementation and Deployment:** System implementation will focus on using a modern technology stack, including Node.js, Django, React Native, and cloud

services. Emphasis will be placed on robust testing to ensure system reliability and satisfaction. User training sessions will familiarize farmers with the platform while ongoing support will address challenges users may face.

- **Sustainability Certification and Traceability:** The project will integrate a sustainability certification and traceability feature. A blockc- hain-based system will track the farm-to-fork journey of agricultural produce, enhancing consumer trust and food safety. Additionally, a sustainability certification system will help farmers market their produce as premium quality, allowing access to higher-value markets.

This comprehensive scope ensures that CropSync effectively addresses the diverse needs of farmers while contributing to the long-term sustainability of the agricultural ecosystem. By integrating technology, data insights, and financial resources, CropSync aims to create a holistic solution that empowers farmers to thrive in a rapidly changing agricultural landscape.

Chapter 2

Literature Review

The field of agriculture has undergone significant transformation with the advent of advanced technologies such as artificial intelligence, blockchain, and Internet of Things (IoT). These in- novations offer new opportunities to address critical challenges in farming, including resource management, sustainability, and transparency across the supply chain. In this chapter, we review existing research and technological developments related to smart farming solutions, blockchain applications in agriculture, and AI-based decision-making tools. The aim is to contextualize the current state of the art and identify gaps that the CropSync system seeks to address by integrating AI and blockchain technologies for more efficient and sustainable farming practices.

2.1 Comparative Analysis of Recent study

In recent years, the agricultural sector has increasingly turned to advanced data-driven methodologies to address the complex challenges posed by climate change, market fluctuations, and resource management. This comparative analysis examines six notable studies that leverage machine learning, IoT, and decision support tools to enhance agricultural productivity and sustainability. The works by Shevchenko et al. (2024) and Sari et al.[7] (2024) focus on evaluating land suitability and predicting commodity prices, respectively, highlighting the need for adaptable strategies in response to environmental changes and market dynamics. Additionally, Mahale et al.[4] (2024) present a novel crop recommendation and yield forecasting system, while Indira et al.[2] (2023) develop an IoT-based agricultural monitoring system, emphasizing real-time data collection for informed decision-making. Furthermore, Iakovidis et al.[1] (2024) explore the optimization of decision support tools, integrating economic, environmental, and social factors to improve agricultural practices.

Lastly, Kurumatani [3](2020) investigates the use of recurrent neural networks for accurate price forecasting. Together, these studies provide a comprehensive overview of contemporary approaches to optimizing agricultural systems, underscoring the critical role of technology in navigating the future of farming. Shevchenko et al.[7] (2024) - Climate Change Impact on Agricultural Land Suitability In this paper, the authors evaluate the impact of climate change on agricultural land suitability across Eurasia using interpretable machine learning models. The work identifies critical environmental variables influencing land productivity, with the goal of forecasting future land suitability[5]. By highlighting

the effects of changing climate patterns, the study provides valuable insights for policy-makers and agricultural planners, helping them prepare for future shifts in land productivity.

In the paper[6], the authors investigate several machine learning approaches for predicting agricultural commodity prices. The work focuses on optimizing various models, including deep learning techniques, to improve prediction accuracy. By providing tools to anticipate market fluctuations, the paper aims to help farmers and stakeholders make informed decisions in agricultural economics.

In the paper[4], the authors propose a machine learning-based crop recommendation and yield forecasting system for Maharashtra, India. The work employs Long Short-Term Memory (LSTM) networks and a novel expectation-maximization algorithm to optimize crop selection based on environmental data. The system helps farmers choose the most suitable crops while offering accurate yield forecasts, improving agricultural decision-making and productivity.

In the paper[2], the authors develop an agricultural monitoring system using Internet of Things (IoT) devices integrated with artificial intelligence (AI). The work focuses on real-time monitoring of key agricultural parameters such as soil moisture and temperature. By enabling better resource management and timely decision-making, the system aims to enhance crop productivity and support sustainable farming practices.

In the paper[1], the authors examine the optimization of decision support tools for the agricultural sector. The work incorporates economic, environmental, and social data into the models to improve decision-making processes. By creating more efficient and sustainable tools, the study aims to support farmers in achieving higher productivity while promoting environmentally conscious practices.

In the paper[3], the author explores the use of Recurrent Neural Networks (RNN) for time series forecasting of agricultural product prices. The work introduces a new evaluation method for assessing the accuracy of these models. By improving price forecasting models, the research aims to help stakeholders better predict market trends and make informed decisions regarding agricultural products.

In this paper[8], the authors present a method for predicting crop yields using data mining and predictive analytics techniques. The work focuses on applying various machine learning algorithms, such as decision trees, support vector machines, and neural networks, to historical agricultural data. By analyzing factors like soil quality, weather conditions, and crop type, the authors aim to forecast the potential yield more accurately. The study highlights how data mining can be effectively utilized to enhance decision-making for farmers, ultimately improving productivity and resource management.

In this paper[9], the authors explore the role of AI-based technologies in enhancing the resilience of agricultural production systems. The work examines various AI applications, including machine learning, computer vision, and natural language processing, and their potential to address challenges such as climate change, pest control, and resource optimization. The authors emphasize the importance of AI-driven tools in fostering sustainability and

adaptability in farming practices, particularly in vulnerable agricultural regions. The study also provides insights into the integration of AI technologies for improved decision-making and sustainable farming outcomes.

To further explore the impact of documentation on software development, the table 2.1 provides a literature review summarizing various key findings on this topic.

Table 2.1: Comparative Analysis of Literature Survey

Sr. No	Title	Author(s)	Year	Methodology	Drawback
1	Enhancing Agricultural Productivity Through AI-Based Models	Singh, R. et al.	2024	Case studies, model development and validation.	High dependency on data quality; Ethical concerns related to AI decision-making; Risk of overfitting models
2	Predictive Analytics for Crop Yield	Kumar, P. et al.	2024	Field experiments, comparative analysis of traditional and tech-based methods.	Requires technical expertise; High implementation costs; Technology adoption barriers among smallholder farmers.
3	Predictive Analytics for Crop Yield	Gupta, S. et al.	2024	Data mining, predictive modeling, validation using historical crop yield data.	Challenges in data collection and quality; Limited by the availability of historical data; Potential biases in predictions.
4	Various Optimized Machine Learning Techniques to Predict Agricultural Commodity Prices	Sari et al.	2024	This paper investigates multiple machine learning approaches, focusing on optimizing various models, including deep learning techniques, to enhance prediction accuracy of agricultural commodity prices.	The findings may not universally apply across all agricultural commodities, and external factors such as economic changes might not be fully accounted for in the models.
5	Crop Recommendation and Forecasting System for Maharashtra Using Machine Learning with LSTM	Mahale et al.	2024	The authors propose a crop recommendation and yield forecasting system for Maharashtra, employing Long Short-Term Memory (LSTM) networks combined with a novel expectation-maximization algorithm based on environmental data.	The system's effectiveness may depend heavily on the availability and accuracy of local environmental data, which could limit its applicability in data-scarce regions.

Table 2.2: Comparative Analysis of Literature Survey

Sr. No	Title	Author(s)	Year	Methodology	Drawback
6	Predicting Agricultural Commodity Prices Using Machine Learning	Karakoyun, G. et al.	2023	Analyzed various machine learning models, compared performance metrics	Limited accuracy for long-term predictions; Requires high-quality data; Computationally intensive.
7	A Framework for Smart Agriculture	Özdemir, D. et al.	2023	Systematic literature review, conceptual framework development.	High initial investment cost; Complexity in integration with existing systems; Data privacy concerns
8	Climate Change Impact on Agricultural Land Suitability: An Interpretable Machine Learning-Based Eurasia Case Study	Shevchenko, V. et al.	2021	Evaluated future cropland suitability in the context of varying climate projections up to the year 2050, leveraging a suite of advanced machine learning algorithms. The study's code is available online on GitHub and allows replication of the analysis.	Limited to a specific geographic area (Eurasia) and may not be generalizable to other regions; The accuracy of future projections is subject to the uncertainties inherent in climate models.
9	Time Series Forecasting of Agricultural Product Prices Based on Recurrent Neural Networks	Kurumatani	2020	This study explores the use of Recurrent Neural Networks (RNN) for time series forecasting of agricultural product prices, introducing a new evaluation method for model accuracy.	The models may face challenges in accurately predicting prices during sudden market shifts or unforeseen events, which could impact their reliability in volatile markets.

Chapter 3

Project Design

This chapter outlines the architectural and system-level design of the CropSync platform, detailing how each module was conceptualized to address specific agricultural challenges. The project design follows a modular approach, ensuring scalability, maintainability, and offline compatibility. Each module ranging from crop prediction and fertilizer recommendation to blockchain-based traceability and dynamic data acquisition—was carefully designed to operate independently while seamlessly integrating into the overall system. The design decisions were guided by the need for efficiency, accessibility for rural users, and practical deployment in real-world farming scenarios. This chapter includes the overall system architecture, module-level workflows, and the rationale behind key design choices that form the foundation of the implemented platform.

3.1 Existing System

Climate FieldView is a digital agriculture platform developed by The Climate Corporation (a subsidiary of Bayer) that helps farmers make data-driven decisions by monitoring their fields using satellite imagery, drones, and AI-powered analytics. It provides real-time insights into crop health, soil conditions, and weather patterns, enabling farmers to optimize fertilization, irrigation, and yield management.

Climate FieldView helps farmers by:

- **Monitoring Crop Health**– Uses satellite and drone imagery to detect early signs of crop stress, diseases, or nutrient deficiencies.
- **Precision Agriculture**– Provides field-level insights that allow farmers to adjust irrigation, fertilizers, and pesticides based on data.
- **Weather Analysis**– Tracks temperature, rainfall, and humidity to predict weather impacts on crop growth.
- **Yield Prediction and Mapping**– Uses machine learning models to estimate expected crop yield based on historical data.
- **Automated Field Mapping**– Generates detailed maps showing variations in soil health, moisture levels, and vegetation.

- **Data Integration**– Connects with farming equipment (like tractors and seeders) to collect and analyze field data in real-time.

Climate FieldView operates through a multi-step process that leverages satellite and drone technology along with artificial intelligence to optimize agricultural practices. The system begins with data collection from high-resolution satellite and drone imagery, which enables continuous monitoring of farmland. These images help detect early signs of crop stress, nutrient deficiencies, pest infestations, and waterlogging before they escalate into major problems.

Once the data is collected, AI-powered image analysis is used to classify different areas of the field based on their health status. Machine learning algorithms analyze variations in plant color, which can indicate potential diseases, drought conditions, or nutrient imbalances. This automated detection process allows farmers to respond quickly to emerging issues.

3.2 Proposed System

The system design and architecture of CropSync is rooted in providing a seamless, technology-driven solution to the challenges faced by Indian farmers. Leveraging a combination of artificial intelligence (AI), blockchain technology, and cloud services, the architecture aims to enhance agricultural productivity while promoting sustainable farming practices. The core components of the system include real-time weather and market data integration, financial planning tools, and a blockchain-based sustainability certification framework. These elements are orchestrated in a mobile application designed for accessibility and ease of use, ensuring that farmers can make data-driven decisions with transparency and trust. The modular structure allows for scalability, ensuring adaptability to evolving agricultural needs and technologies.

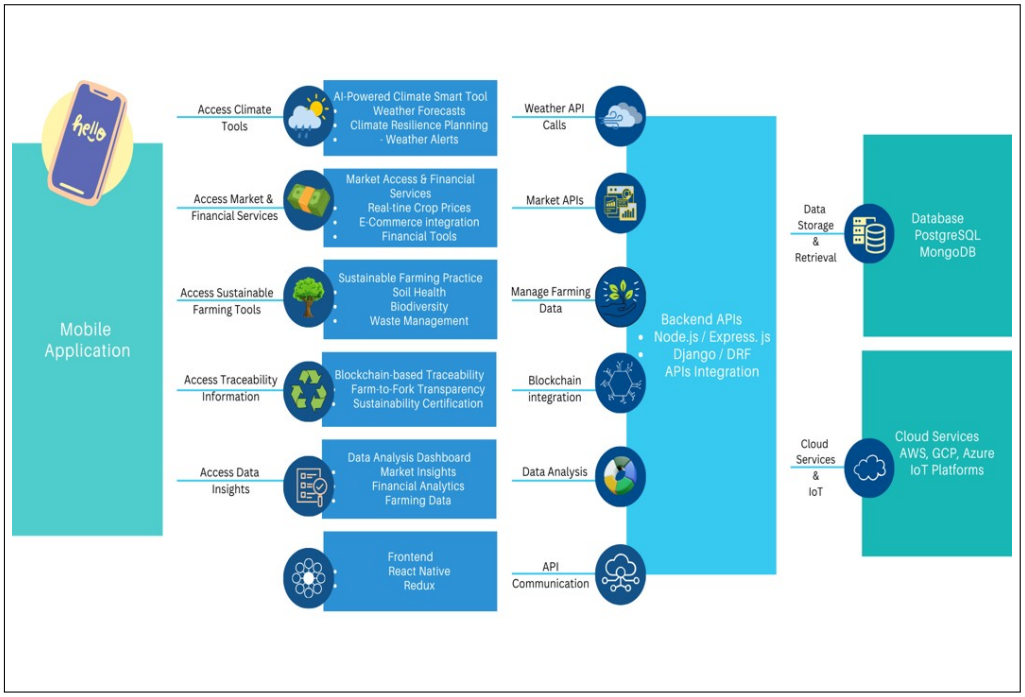

Figure 3.1: System Architecture

From the above figure 3.1, the CropSync system, is a comprehensive digital agriculture platform designed to empower farmers with real-time insights, market access, and sustainable farming tools. It is composed of three main components: the backend, the frontend, and the blockchain, each of which plays a critical role in ensuring seamless functionality, transparency, and efficiency.

The backend serves as the core of the CropSync system, handling data processing, API integrations, and communication between various components. It is developed using Node.js and Django REST Framework (DRF) to provide a scalable and efficient architecture. The backend is responsible for data processing and retrieval, API integration, data analysis, and insights. It collects, processes, and stores large volumes of farming data, manages user interactions, and ensures efficient data retrieval from databases such as PostgreSQL and MongoDB. It also connects with external APIs to fetch real-time weather forecasts, market prices, and environmental data while integrating financial tools to assist farmers with pricing strategies and financial planning. Additionally, it leverages AWS, GCP, and Azure IoT platforms for scalable storage and cloud computing, supporting IoT-enabled smart farming solutions.

The frontend is developed using React Native and Flutter with Redux for state management, ensuring an intuitive and responsive mobile application that enhances user experience. It provides real-time insights through a user-friendly dashboard displaying market prices, weather alerts, and farming analytics. Farmers can interact with farming tools and financial services, including AI-powered tools for climate resilience planning, market access features such as e-commerce integration for direct sales, and tools for sustainable farming practices like soil health monitoring and biodiversity assessment. The frontend also enables traceability and certification, allowing farmers to track their produce from farm to fork while supporting sustainability certifications to ensure compliance with global standards.

Blockchain technology adds a layer of security, transparency, and immutability to the CropSync system. It ensures farm-to-fork traceability by providing an immutable record of farming activities and product origins, enhancing consumer trust through transparent supply chain records. Additionally, it supports sustainability certification by verifying compliance with sustainable farming practices and offering a decentralized certification mechanism for farmers. Blockchain also enables secure financial transactions within the marketplace and automates agreements between farmers, buyers, and financial institutions using smart contracts.

The CropSync system is designed with a modular approach, allowing for seamless adaptation to different farming environments.The integration of cloud services, APIs, blockchain, and AI-driven analytics ensures scalability, reliability, and efficiency. This architecture supports both small-scale farmers and large agricultural enterprises by offering customized insights and solutions tailored to their needs.

In conclusion, CropSync is a next-generation farming solution that leverages cutting-edge technology to boost productivity, expand market access, and ensure sustainability. By integrating real-time data, financial tools, and blockchain traceability, it empowers farmers with smart decision-making, greater transparency, and higher profitability.

3.2.1 Critical Components of System Architecture

3.2.1.1 Ethereum

Ethereum is a decentralized global software platform powered by blockchain technology. It is most commonly known for its native cryptocurrency, ether (ETH). Ethereum is a blockchain based computing platform that enables developers to build and deploy decentralized applications meaning not run by a centralized authority. You can create a decentralized application for which the participants of that particular application are the decision-making authority. Ethereum can be used by anyone to create any secured digital technology. It has a token designed to pay for work done supporting the blockchain, but participants can also use it to pay for tangible goods and services if accepted.

Ethereum is designed to be scalable, programmable, secure, and decentralized. It is the blockchain of choice for developers and enterprises creating technology based upon it to change how many industries operate and how we go about our daily lives. It natively supports smart contracts, an essential tool be hind decentralized applications. Many decentralized finance (DeFi) and other applications use smart contracts in conjunction with blockchain technology.

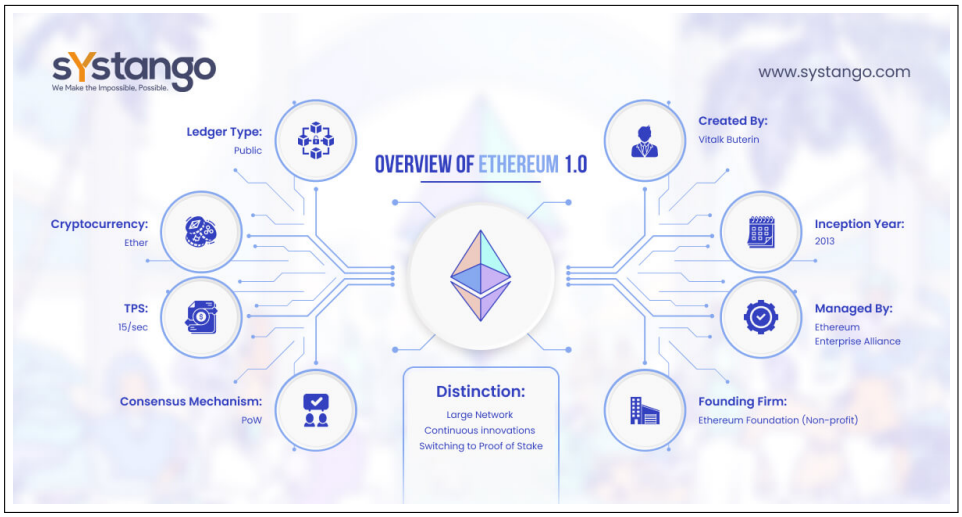

Figure 3.2: Ethereum

Proof-of-stake (PoS) is a consensus mechanism differing from proof-of-work (PoW) in its approach to block validation.Unlike PoW, which relies on energy-intensive mining, PoS utilizes a finalization protocol named Casper-FFG and the LMD Ghost algorithm.These components merge into Gasper, the consensus mechanism governing validators' rewards and penalties.Validators in PoS must stake 32 ETH to activate their validation ability. While individuals can stake smaller amounts, they must join a validation pool to share rewards. Validators create and attest to new blocks, broadcasting them to a committee of other validators for verification through a process called attestation.Gasper identifies dishonest validators, determining block acceptance based on votes.Dishonest validators face punish ment in PoS.Gasper identifies and removes them from the network, burning their staked ETH.

Burning involves sending crypto to a wallet without keys, effectively removing it from circulation. In Ethereum, wallets store ether, the cryptocurrency. Each wallet has a unique address, similar to an email address, where users send ether. Ether isn't stored in wallets; rather, wallets store private keys needed to initiate transactions. Securing these keys is paramount. Ethereum operates as a decentralized network of nodes globally, akin to Bitcoin. Transactions are recorded in blocks on the blockchain, with miners validating and adding them. PoW miners use computational power to find unique codes for blocks, earning ETH as a reward. This process ensures network resilience against cyber attacks, as multiple nodes uphold the network if one fails.

3.2.1.2 IPFS (InterPlanetary File System)

IPFS (Interplanetary File System) is essentially a file system that allows you to store files and track versions over time, much like Git, keeping track of them on a distributed network, somewhat like BitTorrent. This storage system allows direct interaction through a secure and global P2P network. By combining these two properties, IPFS enables a permanent new web and augments the way we use existing internet protocols like HTTP. Since its introduction in 2016, IPFS has seen great improvements and adoption by both individuals and business organizations. This system allows users to share files and information without barriers. IPFS works well with large files that can consume or require high bandwidth to upload and / or download over the Internet. The rapid adoption of this distributed file system happened in part because IPFS is designed to operate over different protocols, such as FTP and HTTP.

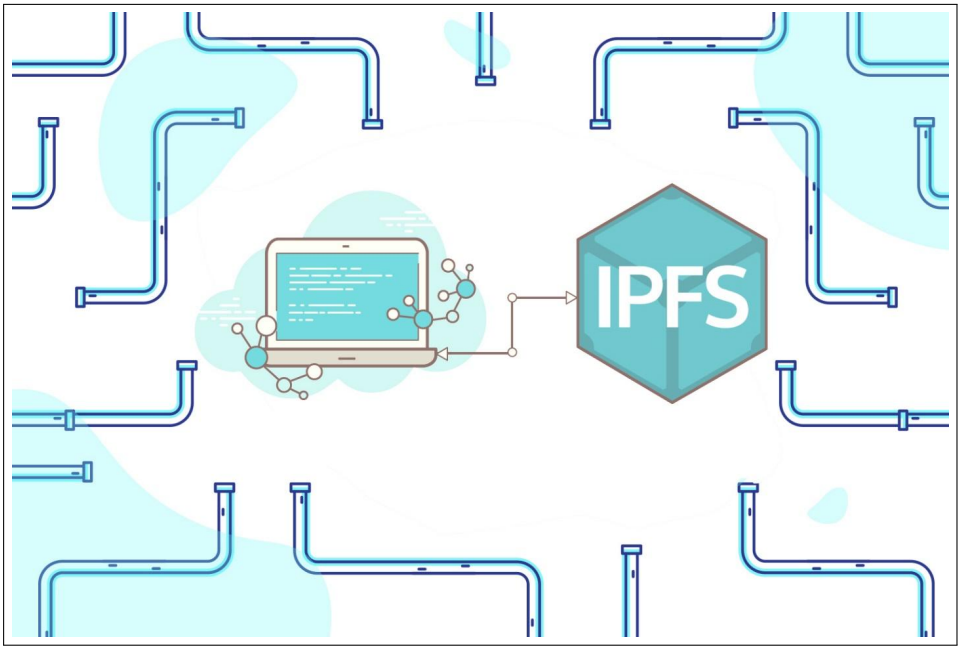

Figure 3.3: IPFS

Blockchain serves as a decentralized data management platform that inherently ensures im mutability, making it an ideal companion for enhancing file traceability metadata within a distributed file system like IPFS. These two technologies exhibit remarkable synergy, with IPFS offering the qualities of a tamper-resistant storage system without a single point

of failure. It assures the uniqueness and protection of data against unauthorized alter ations, thereby rendering the data immutable. In the case of any attempted changes, a new hash identifier is generated, distinguishing it from the original stored on the blockchain. Blockchain ecosystems can broadly be categorized as permissionless (e.g., Ethereum, Bitcoin) or permissioned (e.g., Ethereum).

While permissionless blockchains are open to the public and require validation by most participants, permissioned blockchains rely on preselected nodes for validation, often achieving higher performance.Given their structural similarities, IPFS and blockchains can effectively collaborate, acting as the connective tissue between diverse blockchain networks, akin to how the internet links web pages. Looking ahead, the intertwined future of IPFS and blockchain holds immense promise, poised to bolster decentralized finance and applications, steering both technologies toward consolidation.

3.3 System Diagrams

3.3.1 UML Diagram

UML(Unified Modeling Language) is a visual modeling language used to represent software systems. It consists of a set of diagrams and symbols that can be used to describe different aspects of a system, including its structure, behavior, and interactions. UML diagrams are commonly used in software engineering to facilitate communication and understanding among team members and stakeholders. There are several types of UML diagrams, including class diagrams, use case diagrams, sequence diagrams, activity diagrams, and more. To give precise understanding about the system design and development of the project, following UML diagrams are drawn.

3.3.2 Activity Diagram

The system workflow diagram provides a comprehensive visualization of the functional architecture of the CropSync platform. It illustrates how different modules operate individually while contributing to the larger integrated ecosystem of smart agriculture. The flow begins with user interaction—registration and login and expands into five core modules, each designed to assist farmers in decision-making through weather forecasting, market analysis, traceability, financial services, and analytics. The diagram reflects the logical sequencing of actions, data flow, and outputs across all functionalities within the platform.

In figure 3.4 the Weather Module Workflow leverages location-based data to fetch real-time weather updates, issue alerts, and generate seasonal forecasts. These insights are translated into actionable weather-based recommendations which are stored for future access. Parallel to this, the Market Insights Workflow allows users to select crop types and fetch market prices. The system then analyzes price trends and generates tailored selling recommendations to help farmers maximize profit based on current market dynamics.

The Blockchain Traceability Workflow forms the backbone of the sustainability and transparency framework. Starting with farmer registration and unique ID generation,

Figure 3.4: Activity Diagram

the module captures all significant crop lifecycle events—sowing, monitoring, pesticide application and logs them immutably on the blockchain. This culminates in the issuance of a sustainability badge. In the Financial Services Workflow, users can explore government schemes, check their eligibility, and apply by submitting required documents through the app. Meanwhile, the Analytics Dashboard Workflow consolidates all collected data, generates yield and revenue performance reports, and provides a sustainability score, thus helping users continuously track their progress and update farming strategies accordingly.

Each module's output feeds into a centralized user interface, enabling seamless navigation and continuous engagement. Cloud storage is employed to ensure that weather and market data are persistently saved for model retraining and dashboard analytics. Overall, the diagram effectively captures the modular, data-driven, and user-focused design philosophy of the CropSync system.

3.3.3 Use Case Diagram

Use case diagrams are integral to system design as they depict the various interactions between the user and the system. By showing all possible scenarios that the user may encounter, these diagrams help in identifying and categorizing system requirements.

The use case diagram in figure 3.5 illustrates the interaction between two key stakeholders Consumers and Farmers within an integrated agri-tech platform aimed at enhancing transparency, sustainability, and informed decision making in the agricultural ecosystem. This platform is designed to bridge the gap between producers and end-users by offering services that cover the entire farming and supply chain cycle, from crop recommendation to purchase and traceability. The system fosters direct engagement, empowering both parties to access relevant insights and services efficiently.

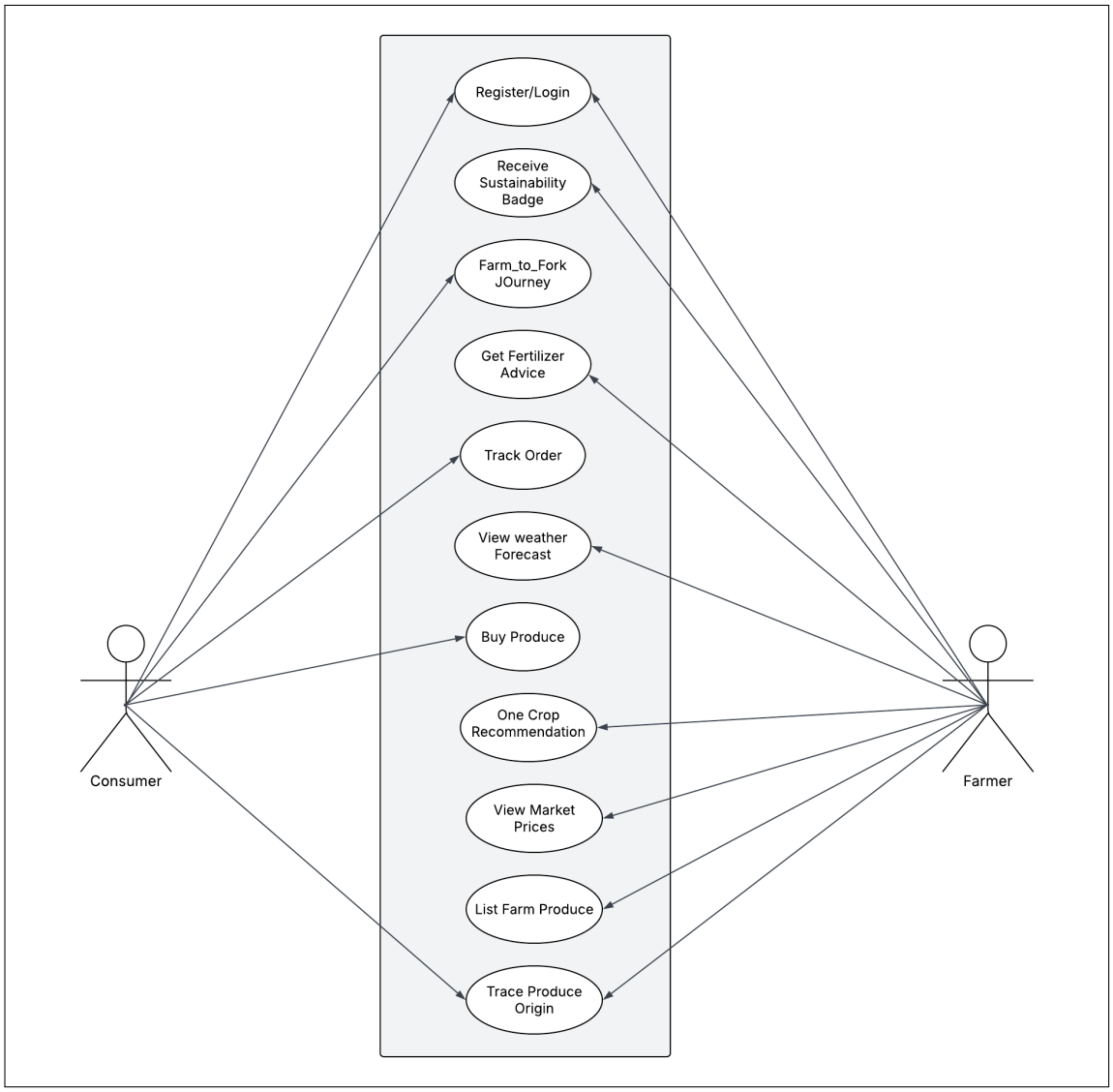

Figure 3.5: Use Case Diagram of the Proposed System

Both consumers and farmers initiate their interaction through the Register/Login functionality, which provides secure and personalized access to the system. Farmers can list their produce through the List Farm Produce feature, enabling consumers to browse available products and make purchases directly via the Buy Produce function. The Track Order feature allows consumers to monitor the delivery process, while the Trace Produce Origin and Farm-to-Fork Journey use cases offer complete transparency into the lifecycle and source of agricultural products, ensuring confidence and trust in food quality.

To support sustainable farming practices, verified farmers are awarded a Sustainability Badge, which is also visible to consumers, enhancing the credibility of eco-conscious sellers. Farmers further benefit from tools like One Crop Recommendation and Get Fertilizer Advice, which guide their decisions using data-driven insights. Simultaneously, both users can view dynamic data such as Weather Forecasts and Market Prices, helping farmers plan agricultural activities and allowing consumers to make informed purchasing decisions based on market trends and availability.

Overall, the system creates a collaborative environment that benefits both the supply and demand sides of agriculture. It integrates smart features that promote informed actions, traceability, and a transparent farm-to-consumer supply chain. The interaction model shown in the use case diagram effectively supports sustainable agriculture, strengthens consumer awareness, and reduces dependency on middlemen through a direct and trust-oriented digital platform.

3.3.4 Sequence Diagram

The sequence diagram illustrates in the figure 3.6 represents the multi-agent interaction flow within the CropSync system, highlighting how user inputs are processed through different components such as the mobile app, AI crop analyzer, irrigation system, weather API, and the backend database. The flow begins with the farmer or user registering their account and logging into the mobile application. Upon successful authentication, users can register their farm fields, including location, crop type, and field dimensions. This data is securely stored in the backend database and becomes the foundation for all future analysis.

Once the farm is registered, the farmer inputs soil test data which is analyzed by the AI Crop Analyzer. The system identifies soil nutrient levels and stores this processed data. The user can then request weather forecasts, which are fetched from an external weather API based on the farm's location. This module fetches local, real-time weather data, which is returned to the user to support seasonal planning, alert generation, and irrigation scheduling.

In parallel, users can upload crop images for visual health diagnostics. The AI Crop Analyzer processes these images using machine learning algorithms to detect diseases or stress indicators. Based on these assessments, a health report is generated and shared with the user. Subsequently, the farmer can request an irrigation plan. The irrigation system fetches historical and current field data, calculates water requirements, and schedules irrigation cycles. The user can also override and manually adjust irrigation settings if necessary. Execution commands are sent, and irrigation status is confirmed through real-time feedback.

Finally, users can request a yield prediction, prompting the AI Crop Analyzer to fetch both historical and current crop data and run machine learning models to generate forecasts. These predictions are then returned to the farmer and displayed within the app. A comprehensive farm report can also be requested, combining insights from all modules, including yield, soil, irrigation, and weather data. This final output offers a holistic view of farm performance and supports improved agricultural planning

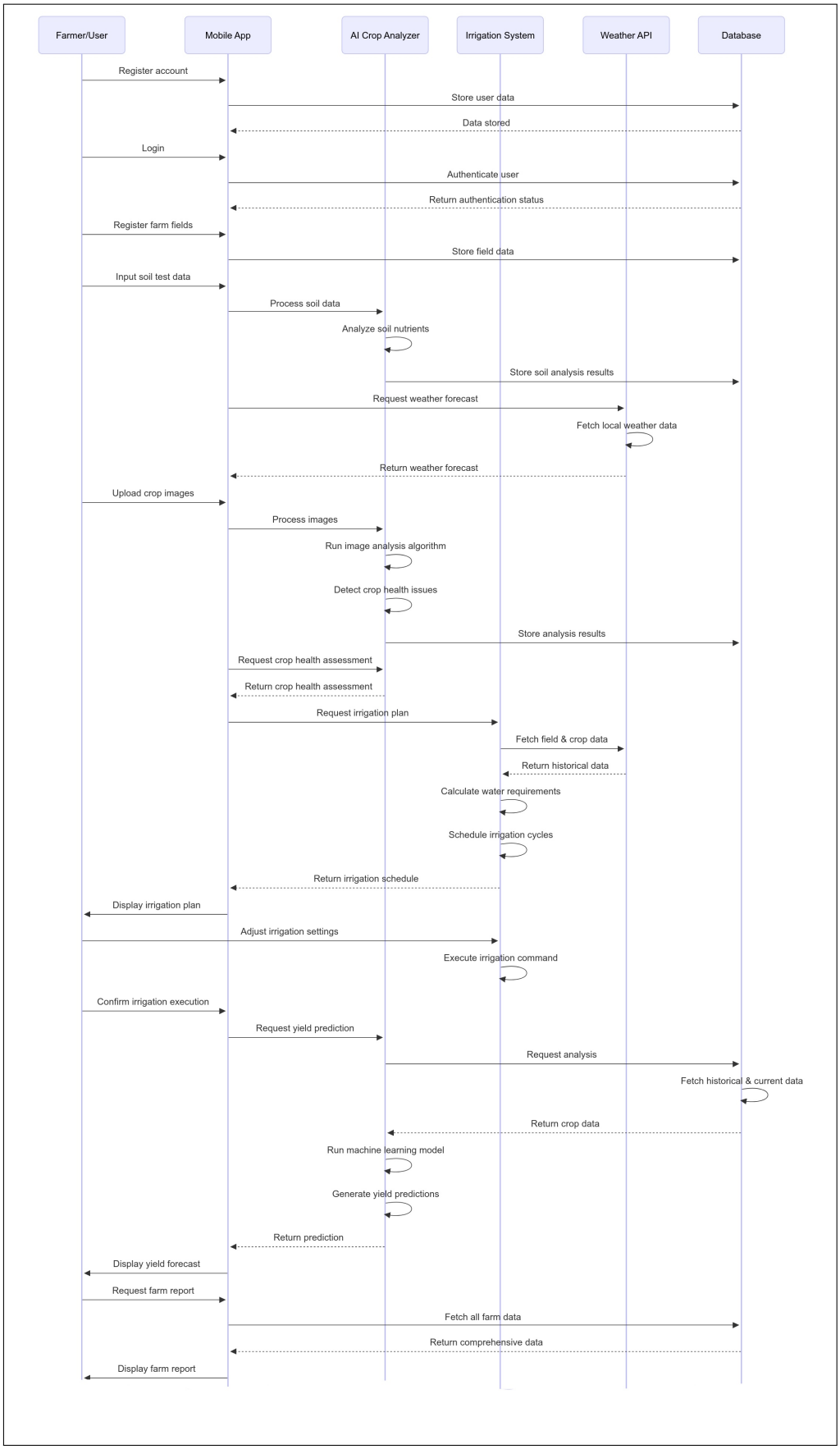

Figure 3.6: Sequence Diagram

Chapter 4

Project Implementation

This chapter provides a comprehensive overview of the implementation phase of the project, detailing the technical execution and development process. It includes key code snippets that illustrate core functionalities, step-by-step instructions for accessing and interacting with the system, and a timeline that outlines the progression of development milestones. This chapter aims to connect the conceptual framework with its practical execution, illustrating how the proposed design was methodically developed into a working system.

4.1 Code Snippets

This subsection highlights the key code implementations that form the technical backbone of the CropSync platform. Each code snippet is carefully chosen to demonstrate the transformation of core concepts such as crop prediction, fertilizer recommendation, and yield estimation into practical, functioning modules. These snippets serve as a bridge between theoretical design and functional deployment, offering insights into how machine learning, data handling, and system integration were achieved.

The primary coding languages used were Python for backend data processing and machine learning model development, and JavaScript (React Native) for the frontend interface. Essential Python libraries such as pandas, numpy, scikit-learn, and joblib were employed to train and serve AI models. Additionally, API integrations and real-time data fetch mechanisms were handled using tools like requests, BeautifulSoup, and Selenium.

To support decision-making, various algorithms were implemented, including Decision Trees and Random Forest classifiers/regressors. These models were trained on curated datasets to offer predictions tailored to user inputs such as soil nutrients, weather patterns, crop types, and geographical data. Furthermore, command-line interfaces and JSON-based data formatting were used to ensure modular and testable interactions with the models.

Each snippet is supplemented with explanations and design rationale, helping the reader understand how logic flows through the system. From model training and serialization to real-time prediction and result visualization, the code is structured for readability, scalability, and future enhancements. These implementation strategies collectively power CropSync's smart farming features and ensure the platform delivers actionable insights to its users.

```
header = ['State_Name', 'District_Name', 'Season', 'Crop']

class Question:
    def __init__(self,column,value):
        self.column =column
        self.value=value
    def match(self,example):
        val = example[self.column]
        return val == self.value
    def match2(self,example):
        if example == 'True' or example == 'true' or example == '1':
            return True
        else:
            return False
    def __repr__(self):
        return "Is %s %s %s?" %(
            header[self.column],"==",str(self.value))

def class_counts(Data):
    counts= {}
    for row in Data:
        label =row[-1]
        if label not in counts:
            counts[label] = 0
        counts[label] += 1
    return counts

class Leaf:
    def __init__(self,Data):
        self.predictions = class_counts(Data)

class Decision_Node:
    def __init__(self,question,true_branch,false_branch):
        self.question=question
        self.true_branch = true_branch
        self.false_branch = false_branch


def print_tree(node,spacing=""):
    if isinstance(node,Leaf):
        print(spacing + "Predict",node.predictions)
        return
    print(spacing+str(node.question))
    print(spacing + "--> True:")
    print_tree(node.true_branch,spacing + " ")

    print(spacing + "--> False:")
    print_tree(node.false_branch,spacing + " ")

def print_leaf(counts):
    total = sum(counts.values())*1.0
    probs = {}
    for lbl in counts.keys():
        probs[lbl] =str(int(counts[lbl]/total * 100)) + "%"
    return probs

def classify(row,node):
    if isinstance(node,Leaf):
        return node.predictions
    if node.question.match(row):
        return classify(row,node.true_branch)
    else:
        return classify(row,node.false_branch)

dt_model_final= joblib.load('ML/crop_prediction/filetest2.pkl')
```

Figure 4.1: Decision Tree Model for Crop Prediction

The code illustrated in figure 4.1 demonstrates the implementation of a decision tree machine learning model utilized by CropSync to predict suitable crop recommendations

based on user-provided attributes such as state, district, season, and crop type.

Initially, the code defines categorical input features like state, district, and season, clearly setting the foundation for structured data input. A Question class is employed to represent decision nodes within the tree, each node containing conditions for splitting the dataset into subsets based on the given attributes. This structure allows the model to systematically partition data, facilitating more precise predictions.

The class_counts function aggregates the occurrences of different crop types within the dataset, essential for assessing the distribution of target variables at any given node. This counting mechanism directly contributes to creating informative leaf nodes in the tree.

Furthermore, leaf nodes (Leaf class) represent the decision endpoints, holding predictions in the form of crop recommendations. Each leaf node encapsulates the resultant prediction probabilities, thus providing explicit and actionable outputs for users.

To construct and visualize the tree effectively, the code uses a recursive print_tree function. This function clearly displays the hierarchical decision-making process, illustrating each question and branching condition.This visualization aids in understanding how specific inputs lead to particular crop recommendations.

Lastly, the code employs joblib's load method to import the previously trained decision tree model, enabling real-time and accurate crop recommendations based on the processed inputs. This structured and systematic approach ensures robust predictions, assisting farmers to optimize crop selection and enhance agricultural productivity.

The code illustrated in figure 4.2 details the implementation of a Random Forest Classifier algorithm utilized by CropSync to recommend crops based on input parameters. These parameters include critical agricultural metrics such as nitrogen, phosphorus, potassium, temperature, humidity, pH, and rainfall.

Initially, the code takes input parameters from command-line arguments in JSON format, parsing them into Python objects for structured data handling. This allows dynamic data entry and integration into the prediction system.

er > ML > crop_recommendation > recommend.py > ...

```python
import pandas as pd
import numpy as np
import json
import sys

# Get the input parameters as command line arguments
jsonn = sys.argv[1]
jsonp = sys.argv[2]
jsonk = sys.argv[3]
jsont = sys.argv[4]
jsonh = sys.argv[5]
jsonph = sys.argv[6]
jsonr = sys.argv[7]

# Parse the JSON strings into Python objects
n_params = json.loads(jsonn)
p_params = json.loads(jsonp)
k_params = json.loads(jsonk)
t_params = json.loads(jsont)
h_params = json.loads(jsonh)
ph_params = json.loads(jsonph)
r_params = json.loads(jsonr)

#Read the dataset
dataset = pd.read_csv('ML/crop_recommendation/Crop_recommendation.csv')

#Divide the dataset into features and labels
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

#Split the dataset into training and test sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)

#Train the model using the Random Forest Classifier algorithm
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)

#Get the user inputs and store them in a numpy array
#user_input = np.array([[90,42,43,21,82,6.5,203]]) - ans is rice.
user_input = np.array([[n_params,p_params,k_params,t_params,h_params,ph_params,r_params]])

#Make predictions using the trained model
predictions = classifier.predict(user_input)

#Print the predicted crop
print(str(predictions[0]))
```

Figure 4.2: Random Forest Classifier for Crop Recommendation

The dataset is then loaded using pandas, specifically from a CSV file containing historical crop recommendation data. The dataset is divided into features (independent variables) and labels (dependent variables or target crop recommendations).

To ensure a robust evaluation, the data is split into training and testing subsets using the train_test_split method from sklearn. This approach helps in verifying the model's effectiveness in making accurate predictions on unseen data.

The Random Forest Classifier is configured with parameters such as n_estimators=10, which denotes the number of decision trees, and criterion='entropy', optimizing the splits based on information gain. The model is trained on the training dataset.

After training, the user inputs are formatted into a numpy array, preparing them for prediction. The classifier then uses these inputs to predict the most suitable crop recommendation.

Finally, the code prints the predicted crop, providing actionable insights to farmers. This implementation enables CropSync to offer precise and efficient crop recommendations, thereby optimizing agricultural productivity and decision-making processes.

```
import pandas as pd
import sys
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier

# Load the dataset
data = pd.read_csv("C:\\xampp\\htdocs\\agriculture-portal\\farmer\\ML\\fertilizer_recommendation\\fertilizer_recommendation.csv")

# Label encoding for categorical features
le_soil = LabelEncoder()
data['Soil Type'] = le_soil.fit_transform(data['Soil Type'])
le_crop = LabelEncoder()
data['Crop Type'] = le_crop.fit_transform(data['Crop Type'])

# Splitting the data into input and output variables
X = data.iloc[:, :8]
y = data.iloc[:, -1]

# Training the Decision Tree Classifier model
dtc = DecisionTreeClassifier(random_state=0)
dtc.fit(X, y)

# Get the input parameters as command line arguments
jsonn = sys.argv[1]
jsonp = sys.argv[2]
jsonk = sys.argv[3]
jsont = sys.argv[4]
jsonh = sys.argv[5]
jsonsm = sys.argv[6]
jsonsoil = sys.argv[7]
jsoncrop = sys.argv[8]

soil_enc = le_soil.transform([jsonsoil])[0]
crop_enc = le_crop.transform([jsoncrop])[0]

# Get the user inputs and store them in a numpy array - Urea
#user_input = [[26,52,38,'Sandy','Maize',37,0,0]]

user_input = [[jsont,jsonh,jsonsm,soil_enc,crop_enc,jsonn,jsonk,jsonp]]

fertilizer_name = dtc.predict(user_input)

# Return the prediction as a string
print(str(fertilizer_name[0]))
```

Figure 4.3: Decision Tree Classifier for Fertilizer Recommendation

The code illustrated in figure 4.3 provides an implementation of a Decision Tree Classifier within the CropSync platform, specifically tailored for recommending suitable fertilizers based on inputs like soil type, crop type, and key soil nutrients.

The implementation begins by loading a structured dataset containing historical data about fertilizer recommendations using pandas. To efficiently manage categorical data such as soil type and crop type, Label Encoding is performed with LabelEncoder. This encoding converts categorical variables into numerical form, making them suitable for machine learning algorithms.

Subsequently, the dataset is split into features (input variables such as soil type, crop type, nutrient levels) and labels (target fertilizer recommendations). The Decision Tree

Classifier from scikit-learn is configured with a fixed random state to ensure reproducible results. The model is trained using these prepared features and labels.

User inputs are accepted as command-line arguments in JSON format, converted into numerical format through the previously fitted Label Encoders, and structured into a numpy array for prediction. The classifier then predicts the optimal fertilizer recommendation based on these user inputs.

Finally, the recommended fertilizer type is returned and printed, delivering targeted and practical advice to farmers. This structured machine learning approach enhances agricultural productivity by providing precise fertilizer recommendations, significantly aiding farmers in decision-making.

```python
import pandas as pd
import sys

# Load the dataset into a dataframe
df = pd.read_csv("C:\xampp\htdocs\agriculture-portal\farmer\ML\rainfall_prediction\rainfall_in_india_1901-20

# Define a function to predict rainfall for a given district and month
def predict_rainfall(state, month):
    # Filter the dataframe to only include rows with the given district
    state_data = df[df['SUBDIVISION'] == state]

    # Calculate the average rainfall for the given month across all the years
    avg_rainfall = state_data[month].mean()

    # Return the predicted rainfall for the given month
    return avg_rainfall

# Get the input parameters as command line arguments
Jregion = sys.argv[1]
Jmonth = sys.argv[2]

#predicted_rainfall = predict_rainfall('ANDAMAN & NICOBAR ISLANDS', 'JAN')

predicted_rainfall = predict_rainfall(Jregion, Jmonth)
print(predicted_rainfall)
```

Figure 4.4: Rainfall Prediction Model

The code illustrated in figure 4.5 demonstrates the implementation of a rainfall prediction functionality within the CropSync platform. This function predicts average monthly rainfall for a specified region, helping farmers anticipate and prepare for weather conditions effectively.

Initially, the code loads historical rainfall data using pandas from a CSV dataset containing records spanning from 1901 to 2015. This data includes monthly rainfall metrics across different subdivisions or regions.

A function named predict_rainfall is defined to calculate the average rainfall. It filters the dataset to extract records matching the specified region (subdivision). After filtering, it computes the mean rainfall for the requested month across all available historical years, thus providing a robust average rainfall prediction.

Inputs for this prediction function, such as the region and month, are captured through command-line arguments, enabling dynamic and flexible usage by the end-user.

Finally, the calculated average rainfall for the specified region and month is returned and displayed. This rainfall prediction capability equips farmers with critical foresight, facilitating better crop planning and management practices, thereby enhancing agricultural productivity and sustainability.

```
ner > ML > yield_prediction > yield_prediction.py > ...
    import pandas as pd
    import numpy as np
    import json
    import sys
    from sklearn.ensemble import RandomForestRegressor
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import OneHotEncoder

    # Load the dataset
    df = pd.read_csv("C:\\xampp\\htdocs\\CROPSYNC\\farmer\\ML\\yield_prediction\\crop_production_karnataka.csv")

    # Drop the Crop_Year column
    df = df.drop(['Crop_Year'], axis=1)

    # Separate the features and target variables
    X = df.drop(['Production'], axis=1)
    y = df['Production']

    # Split the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Categorical columns for one-hot encoding
    categorical_cols = ['State_Name', 'District_Name', 'Season', 'Crop']

    # One-hot encode the categorical columns
    ohe = OneHotEncoder(handle_unknown='ignore')
    ohe.fit(X_train[categorical_cols])

    # Convert categorical columns to one-hot encoding
    X_train_categorical = ohe.transform(X_train[categorical_cols])
    X_test_categorical = ohe.transform(X_test[categorical_cols])

    # Combine the one-hot encoded categorical columns and numerical columns
    X_train_final = np.hstack((X_train_categorical.toarray(), X_train.drop(categorical_cols, axis=1)))
    X_test_final = np.hstack((X_test_categorical.toarray(), X_test.drop(categorical_cols, axis=1)))

    # Train the model
    model = RandomForestRegressor(n_estimators=100, random_state=42)
    model.fit(X_train_final, y_train)

    # Get the input parameters as command line arguments
    Jstate = sys.argv[1]
    Jdistrict = sys.argv[2]
    Jseason = sys.argv[3]
    Jcrops = sys.argv[4]
    Jarea = sys.argv[5]


    #Get the user inputs and store them in a numpy array  - ans is 427.64
    #user_input = np.array([['Karnataka', 'BAGALKOT', 'Kharif', 'Rice', 197]])

    user_input = np.array([[Jstate,Jdistrict,Jseason,Jcrops,Jarea]])


    # Convert the categorical columns to one-hot encoding
    user_input_categorical = ohe.transform(user_input[:, :4])

    # Combine the one-hot encoded categorical columns and numerical columns
    user_input_final = np.hstack((user_input_categorical.toarray(), user_input[:, 4:].astype(float)))

    # Make the prediction
    prediction = model.predict(user_input_final)

    # Return the prediction as a string

    print(str(prediction[0]))
```

Figure 4.5: Crop Yield Prediction Using Random Forest Regressor

The code illustrated in Figure 4.5 showcases the implementation of a Crop Yield Prediction model within the CropSync platform, utilizing a Random Forest Regressor. The model predicts crop yield based on inputs such as state, district, season, crop type, and area.

Initially, the dataset containing historical crop yield data is loaded using pandas. The dataset undergoes preprocessing by removing irrelevant columns, specifically the 'Crop_Year' column. Subsequently, the dataset is split into independent features (state, district, season, crop, area) and the dependent variable (production/yield).

The data is then divided into training and testing sets using sklearn's train_test_split method to ensure robust evaluation of the model's predictive accuracy. Categorical variables, such as 'State_Name', 'District_Name', 'Season', and 'Crop', are transformed into numerical form using One-Hot Encoding to make them suitable for machine learning algorithms.

Following data preparation, the Random Forest Regressor model is trained with parameters set to n_estimators=100, enhancing the model's predictive capabilities by utilizing an ensemble of decision trees.

The model accepts user inputs dynamically through command-line arguments, converting these inputs into a structured numpy array. This array undergoes the same categorical transformations as the training data to maintain consistency.

Finally, the model predicts the crop yield based on the processed user input, and the predicted yield value is displayed. This predictive functionality enables farmers to plan agricultural operations effectively, ensuring optimized resource management and enhanced crop productivity.

4.2 Steps to access the System

To efficiently utilize the CropSync system, users need to follow a structured process to access its features and tools. Whether it's checking weather updates, managing finances, or exploring market trends, navigating the system requires a few simple steps. This guide will walk you through the process, ensuring a seamless experience in accessing essential farming resources and decision-making tools.

The CropSync platform is designed with accessibility and ease of use in mind, especially for farmers in rural and semi-urban areas. With intuitive navigation, multilingual support, and offline compatibility, the system ensures that farmers from various backgrounds can engage with its features without requiring advanced technical skills. This inclusivity enhances the adoption rate and maximizes the impact of digital tools on real-world agricultural practices.

1.) Home Page

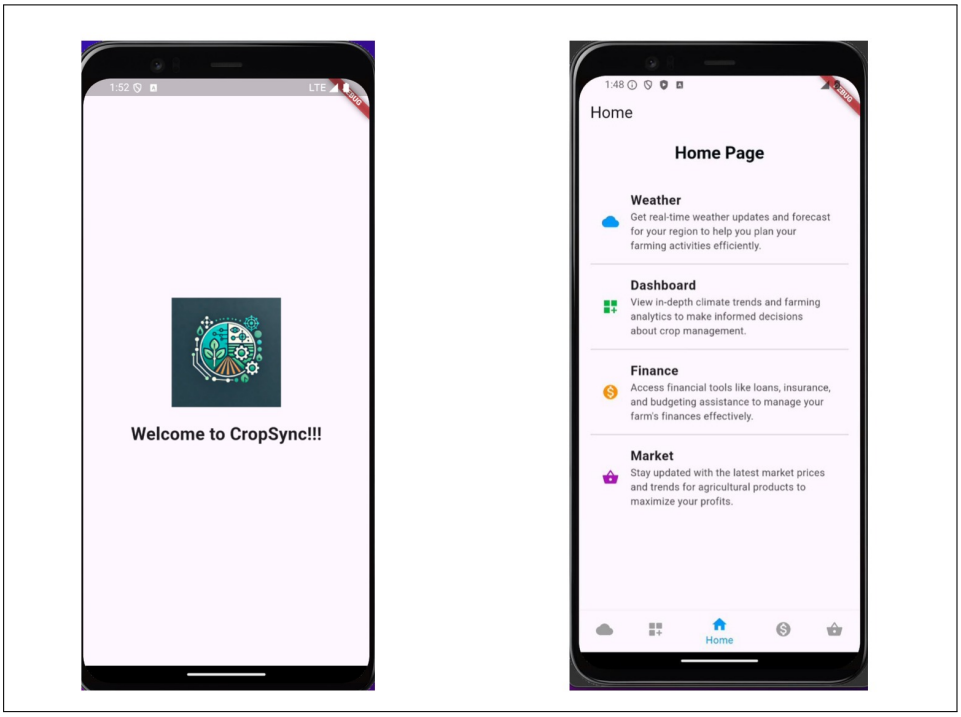

Figure 4.6: Homepage

The CropSync App homepage provides essential tools for farmers, including real-time weather updates, a detailed crop dashboard, financial assistance, and market trend insights. The crop dashboard displays farmland distribution through pie charts, showing cultivated and unused areas, as well as crop production trends by year. This helps farmers make informed decisions about crop selection, blockchain-based transparent payments, and market opportunities. With an intuitive interface, the app simplifies farm management and enhances productivity.

2.) CropSync dashboard

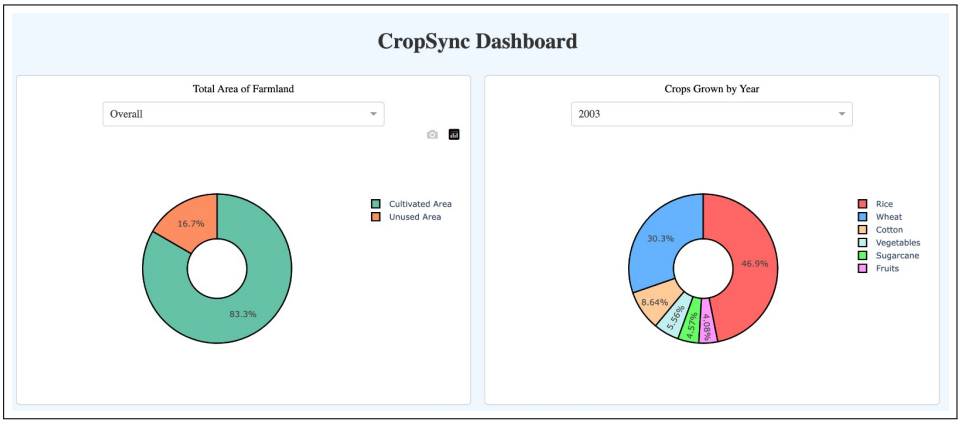

Figure 4.7: Dashboard

The dashboard features two key pie chart—one showing the total farmland area

(cultivated vs. unused) and another displaying crops grown per year, categorized by color for easy identification.

3.) Weather Forecast

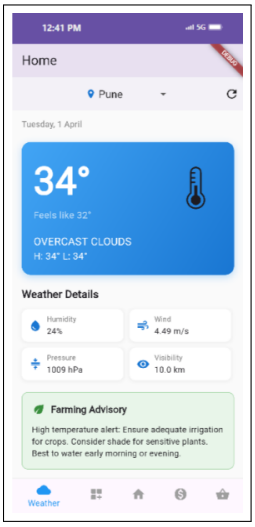

Figure 4.8: Weather

The Weather Forecast feature provides real-time weather updates, helping farmers plan their agricultural activities efficiently. It offers temperature, rainfall, and climate predictions, enabling better crop management and disaster prevention.

4.) Finance Page

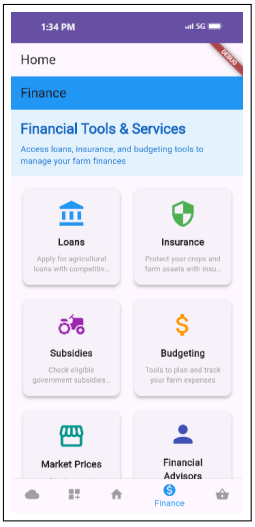

Figure 4.9: Finance

The image showcases the Finance section of the CropSync app, which provides financial tools and services for farmers. It includes options for Loans, Insurance, Subsidies, and Budgeting, helping users apply for agricultural loans, protect farm assets, check government subsidies, and plan expenses. The interface is user-friendly, with clear icons and descriptions,

ensuring easy navigation for financial management.

5.) Market Price Page

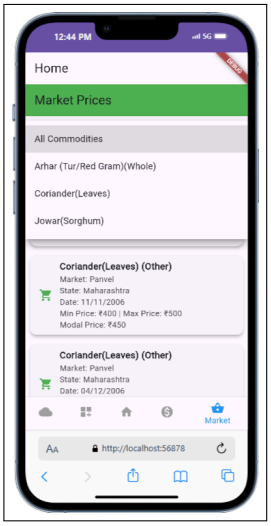

Figure 4.10: Market

The Marketplace feature provides the latest market prices and trends for agricultural products, helping farmers make informed selling decisions. By staying updated on price fluctuations, farmers can maximize their profits and choose the best time to sell their produce.

4.3 Timeline Sem VIII

The Gantt chart outlines the complete project lifecycle of the CropSync system, structured into five major phases: Project Conception and Initiation, Project Design, Project Implementation, Project Testing, and Project Report. Each phase is broken down into specific tasks with clearly defined timelines, task owners, and completion statuses. This visual planning framework helped track development progress, align task responsibilities, and ensure timely delivery across the project cycle.

The first phase, Project Conception and Initiation, spans activities such as requirement gathering, research paper finalization, team formation, and initial ideation. These tasks were distributed across team members and executed primarily in the early weeks (Week 1 to Week 4). This stage laid the groundwork for the problem statement, objectives, and scope of the CropSync platform. Documentation, literature surveys, and early technical exploration were prioritized to define a realistic and meaningful solution for farmers.

Moving into the Project Design phase, the focus shifted toward system architecture and planning of key modules. Here, use case diagrams, data flow diagrams (DFDs), and system component breakdowns were created. The platform design involved mapping out how the crop prediction engine, fertilizer model, rainfall estimator, and blockchain module

would interact. Each of these steps was carefully assigned and carried out across Weeks 5 to 8. Design validation ensured the project met both user-centric and technical feasibility standards before implementation began.

The Project Implementation phase occupies the most extensive section of the Gantt timeline. Tasks such as interface development, model training, rainfall data handling, blockchain traceability integration, and dashboard creation were carried out across multiple weeks—from Week 9 to Week 19. Each module was developed independently, allowing parallel contributions by different team members. Integration points were strategically timed to enable smoother merging and debugging. The dashboard and AI modules were tested incrementally as part of development sprints.

Following implementation, the Testing Phase focused on validating system reliability, model accuracy, and usability. It included unit testing for all machine learning modules crop prediction, rainfall estimation, fertilizer suggestion, and yield forecasting. Additionally, the blockchain and UI components were subjected to functional and integration testing. Accuracy and error-handling tests were conducted using real and synthetic datasets. These tasks extended from Week 20 to Week 24, reflecting the depth and granularity of the verification process.

Parallel to technical testing, data collection for test cases and the creation of structured test documentation were also carried out. This involved preparing test input scenarios, expected outputs, and defining metrics like prediction accuracy and module response time. A rigorous focus on verification helped identify gaps early, allowing the team to apply necessary corrections to both logic and data handling strategies before final integration.

Once implementation and testing were completed, the Project Report and Deliverables phase began. This final stage involved compiling the black book, preparing a presentation, and producing the final Gantt chart visualization. Team members collaborated to consolidate documentation from all previous phases, validate code repositories, and summarize system performance. Deliverables were prepared with a focus on clear technical communication and presentation-readiness for evaluation panels.

The Gantt chart also visually reflects task dependencies, overlapping schedules, and the percentage of completion, with color-coded blocks marking each work period. For example, extended continuous task blocks show development-intensive weeks, while testing tasks appear staggered as they were aligned with module completion. This planning mechanism enabled the team to adapt to delays and optimize workloads across team members without compromising deadlines.

In conclusion, the Gantt chart served as both a project planning and execution monitoring tool throughout the CropSync development lifecycle. It helped track module-specific milestones, ensured smooth transitions between phases, and provided transparency in workload distribution. The structured execution guided by this chart directly contributed to the successful and timely delivery of a multi-functional, offline-ready smart agriculture platform.

GANTT CHART

PROJECT TITLE	CropSync: AI-Driven Blockchain Solution for Smart Farming	INSTITUTE & DEPARTMENT NAME	A.P SHAH INSTITUTE OF TECHNOLOGY -CSE(DS)
PROJECT GUIDE	Prof. Poonam Pangarkar Prof. Harsha Zope	DATE	4/10/25

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION (Weeks)	PCT OF TASK COMPLETE
1	**Project Conception and Initiation**					
1.1	Research paper search	Shreyas Revankar, Harsh Shelke, Devansh Kopra, Sumit Samanta	6/10/24	6/16/24	1	100%
1.1.1	Research paper finalization	Shreyas Revankar, Harsh Shelke, Devansh Kopra, Sumit Samanta	6/17/24	6/24/24	1	100%
1.2	Project Title	Shreyas Revankar, Harsh Shelke, Devansh Kopra, Sumit Samanta	6/25/24	7/1/21	1	100%
1.3	Abstract	Sumit Samanta	7/2/24	7/9/24	1	100%
1.4	Objectives	Devansh Kopra, Harsh Shelke	7/10/24	7/17/24	1	100%
1.5	Literature Review	Shreyas Revankar, Harsh Shelke, Devansh Kopra, Sumit Samanta	7/18/24	7/25/24	1	100%
1.6	Problem Definition	Shreyas Revankar, Harsh Shelke, Devansh Kopra, Sumit Samanta	7/26/24	8/2/24	1	100%
1.7	Scope	Shreyas Revankar, Harsh Shelke, Devansh Kopra, Sumit Samanta	8/3/24	8/10/24	1	100%
1.8	Technology stack	Shreyas Revankar, Devansh Kopra, Sumit Samanta	8/11/24	8/18/24	1	100%
1.90	Applications	Shreyas Revankar, Harsh Shelke, Devansh Kopra, Sumit Samanta	8/19/24	8/26/24	1	100%
2	**Project Design**					
2.1	Proposed System	Sumit Samanta, Devansh Kopra	8/27/24	9/10/24	2	100%
2.2	Design(Flow Of Modules)	Sumit Samanta, Harsh Shelke, Devansh Kopra	9/11/24	9/18/24	1	100%
2.3	Activity Diagram	Sumit Samanta, Devansh Kopra	9/19/24	9/26/24	1	100%
2.4	Use Case Diagram	Sumit Samanta, Devansh Kopra	9/27/24	10/4/24	1	100%
2.5	Sequence Diagram	Sumit Samanta	10/5/24	10/12/24	1	100%
2.6	Description Of Use Case	Shreyas Revankar, Harsh Shelke, Devansh Kopra	10/13/24	10/20/24	1	100%
2.7	Modules	Shreyas Revankar, Harsh Shelke, Devansh Kopra, Sumit Samanta	10/21/24	10/28/24	1	100%
2.7.1	Application Module	Shreyas Revankar, Harsh Shelke, Devansh Kopra, Sumit Samanta	10/29/24	11/5/24	1	100%
2.8	Preparation Of Report	Shreyas Revankar, Harsh Shelke, Devansh Kopra, Sumit Samanta	11/6/24	11/13/24	1	100%
3	**Project Implementation**					
3.1	Development of Local Graphical User Interface	Shreyas Revankar, Harsh Shelke	8/14/24	9/28/24	4	100%
3.2	Setup of Local Backend Logic for ML Workflow Execution	Devansh Kopra, Harsh Shelke	9/29/24	10/13/24	2	100%
3.3	Integration of Crop Recommendation System	Shreyas Revankar, Harsh Shelke	10/14/24	11/14/24	4	100%
3.4	Development of Fertilizer Recommendation System	Devansh Kopra, Harsh Shelke	11/15/24	11/29/24	2	100%
3.5	Setting up of Multi Modal Code generation Pipeline	Sumit Samanta, Shreyas Revankar	11/30/24	12/30/24	4	100%
3.6	Development of Crop Yield Prediction Module	Sumit Samanta, Shreyas Revankar	12/31/24	1/14/25	2	100%
3.7	Design and Integration of Blockchain Layer	Sumit Samanta	1/15/25	1/21/25	1	100%
3.8	Implementation of Data Collection Logic	Harsh Shelke, Devansh Kopra	1/22/25	2/5/25	2	100%
3.9	Development of Dynamic Data Analytics Dashboard	Devansh Kopra	2/6/25	2/20/25	2	100%
3.10	Label Encoding and Preprocessing Pipelines	Harsh Shelke, Devansh Kopra	2/21/25	3/7/25	2	100%
3.11	One-Hot Encoding for Yield Prediction Input Processing	Shreyas Revankar, Harsh Shelke, Devansh Kopra	2/23/25	3/2/25	2	100%
3.12	Integration of All Models into Unified Local System	Shreyas Revankar, Harsh Shelke, Devansh Kopra, Harsh Shelke	2/24/25	2/28/25	4	100%
3.13	Generation of Sustainability Badges based on Blockchain	Sumit Samanta	2/24/25	3/3/25	4	100%
3.14	Project Packaging and Folder-Level Execution Management	Shreyas Revankar, Sumit Samanta	3/8/25	4/8/25	4	100%
4	**Testing**					
4.1	Design of Manual and Automated Test Cases	Shreyas Revankar, Harsh Shelke, Devansh Kopra, Sumit Samanta	3/1/25	3/6/25	1	100%
4.2.1	Unit Testing of Crop Recommendation Module	Shreyas Revankar, Harsh Shelke	3/1/25	3/7/25	1	100%
4.2.2	Unit Testing of Fertilizer Suggestion System	Devansh Kopra, Sumit Samanta	3/2/25	3/8/25	1	100%
4.2.3	Unit Testing of Rainfall Prediction Function	Harsh Shelke, Shreyas Revankar	3/8/25	3/9/25	1	100%
4.2.4	Unit Testing of Crop Yield Regression Model	Harsh Shelke, Devansh Kopra	3/8/25	3/19/25	1	100%
4.2.5	Accuracy Testing of Crop Prediction Outputs	Sumit Samanta, Shreyas Revankar	3/9/25	3/21/25	1	100%
4.2.6	Accuracy Testing of Yield Regression Forecast	Shreyas Revankar, Harsh Shelke, Devansh Kopra, Sumit Samanta	3/10/25	3/3/25	1	100%
4.2.7	Logical Testing of Blockchain Badge Generation	Sumit Samanta	3/11/25	3/24/25	1	100%
4.2.8	Testing of JSON Input Parsing and Argument Handling	Harsh Shelke, Devansh Kopra	3/12/25	3/19/25	1	100%
4.2.9	Label Encoding and Decoding Consistency Testing	Shreyas Revankar, Harsh Shelke, Devansh Kopra, Sumit Samanta	3/12/25	3/15/25	1	100%
4.3.1	Testing of CSV File Integration and Data Loading	Devansh Kopra, Sumit Samanta	3/14/25	3/19/25	1	100%
4.3.2	UI Testing of Flutter-Based Interface (Cross-Device)	Shreyas Revankar, Harsh Shelke	3/15/25	3/19/25	1	100%
4.3.3	CLI-Based Input Execution Testing	Shreyas Revankar, Harsh Shelke, Devansh Kopra, Sumit Samanta	3/16/25	3/29/25	1	100%
4.3.4	Dashboard Output Verification Based on ML Results	Devansh Kopra, Sumit Samanta	4/1/25	4/4/25	1	100%
4.3.5	Integration Testing of All Models and Interfaces Locally	Shreyas Revankar, Harsh Shelke, Devansh Kopra, Sumit Samanta	4/3/25	4/5/25	1	100%
4.3.6	End-to-End Workflow Testing for Different User Scenarios	Sumit Samanta, Shreyas Revankar	4/2/25	4/6/25	1	
4.3.7	Error Handling for Missing/Invalid CSV or User Input Data	Harsh Shelke, Devansh Kopra	4/3/25	4/9/25	1	
5	**Report Preparation**					
5.1	Black Book	Shreyas Revankar, Harsh Shelke, Devansh Kopra, Sumit Samanta	3/21/25	3/28/25	1	100%
5.1	PPT	Harsh Shelke, Devansh Kopra	3/22/25	3/29/25	1	100%
5.1	Gantt Chart	Rohan Waghode, Meet Jamsutkar	3/28/25	3/28/25	1	100%

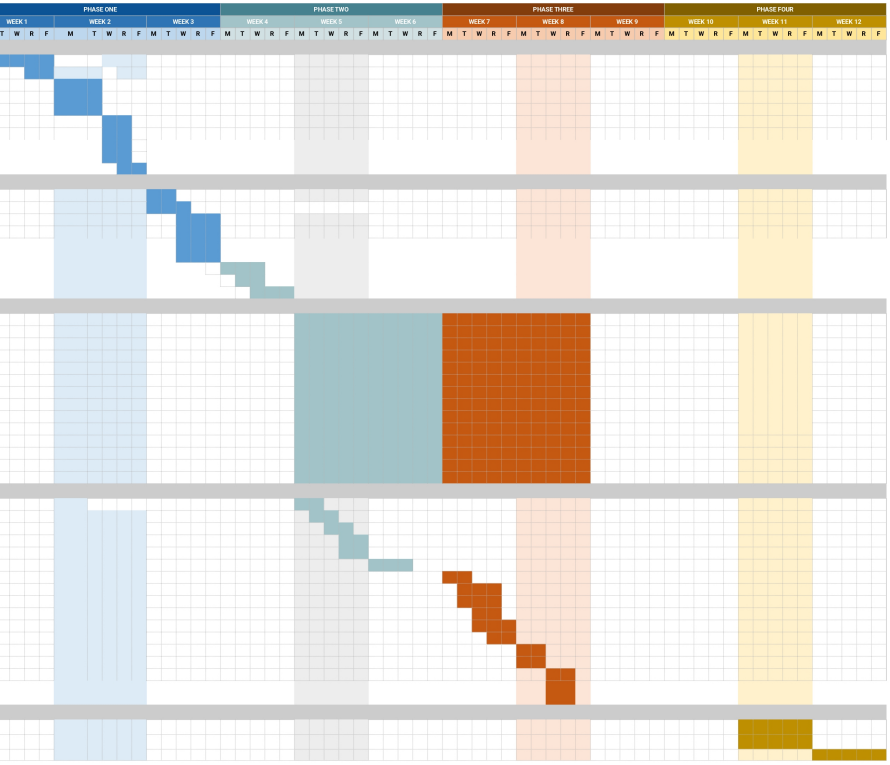

Figure 4.11: Timeline of the Project Milestones

Chapter 5

Testing

This chapter elaborates on the testing phase of the CropSync project, detailing the rigorous evaluation of its core components such as crop prediction, fertilizer recommendation, rainfall prediction, and crop yield forecasting. Testing was conducted on a local system using Python environments to simulate actual farmer inputs and operational scenarios. The primary objective was to ensure that the models worked effectively with real-world data and delivered consistent results across all functionalities.

All modules were individually tested to verify the correctness of outputs based on known datasets. The testing process also focused on integration points across modules to confirm that user input flowed logically through the system from data entry to model prediction and final output display. The local command-line based input approach was assessed for accuracy and system stability under varied data scenarios.

5.1 Software Testing

The software testing phase also evaluated the broader architectural framework and modular integration derived from the CropSync research foundation. The system incorporates multiple locally executable machine learning models including crop recommendation, fertilizer prediction, rainfall estimation, and crop yield forecasting-trained on curated datasets and executed using Python-based libraries.

The frontend interface, developed using Flutter, was locally deployed and validated across devices for responsiveness and ease of use. Farmers interact with tools categorized into weather forecasts, crop insights, and sustainability metrics through a dynamic dashboard. Meanwhile, a blockchain layer was tested for traceability features, ensuring that farmer registration, crop monitoring, pesticide usage documentation, and badge generation were logged accurately using simulated datasets.

In parallel, a dedicated data acquisition layer was tested for consistent scraping and parsing of weather APIs (Open-Meteo), government scheme portals (e.g., PMFBY), and market prices from eNAM and AGMARKNET using tools like BeautifulSoup and Selenium. These inputs formed the dynamic datasets powering the analytics dashboard.

The testing ensured that each component from raw data acquisition and AI inference to

front-end interaction and blockchain trace logging functioned in harmony, reinforcing the robustness of CropSync's offline architecture.

Software testing was carried out on the primary machine using Python and supporting libraries such as pandas, numpy, sklearn, and joblib. Testing covered the execution of machine learning models under different environmental conditions and inputs. Manual testing methods were applied to simulate different combinations of farmer inputs, ensuring accurate output in each case. Testing was also performed on the analytics dashboard to verify if the data visualizations updated correctly with corresponding predictions.

The testing activities addressed both functionality and behavior of the modules, focusing on correctness of predictions and the system's ability to handle unexpected or edge-case input values. Input errors, data parsing issues, and prediction mismatches were all considered and rectified during this phase.

Testing Objectives

- To verify the accuracy and consistency of the ML prediction modules (Crop, Fertilizer, Rainfall, Yield).
- To ensure that user inputs are processed correctly and predictions align with expected outcomes.
- To check integration between all core modules and validate the final output pipeline.
- To identify and correct any parsing, formatting, or prediction errors.

Testing Methodology

A hybrid testing approach was adopted, combining manual testing of each component with data-driven testing using sample CSV inputs. The test documentation captured input conditions, prediction results, and any deviations from expected outcomes. Functional testing tables were prepared to track individual module outcomes in a structured manner.

Test Case Summary

A total of nine functional test cases were executed to ensure the reliability and correctness of each major module in the CropSync system. Each test case was aligned with one of the core functionalities ranging from machine learning-based predictions to blockchain-based traceability and data acquisition mechanisms. All test cases were conducted using realistic data scenarios, and the results demonstrated that the modules performed consistently under expected input conditions.

Performance Testing Insights

Performance testing was conducted to measure the responsiveness and efficiency of each module in the CropSync system. The following key Observations were made:

- Machine learning models exhibited inference times ranging from 0.1 to 1.2 seconds in a local execution environment.

- Rainfall and fertilizer modules consistently responded within 0.5 seconds due to their simpler logic and lower input complexity.
- Crop and yield prediction modules took slightly longer due to multiple encoded inputs and regression logic.
- Overall system responsiveness remained stable during repeated testing, confirming reliable local execution for real-time use.

Security Testing

Security testing was conducted primarily on the blockchain layer of the system, focusing on the secure logging of farmer activity and prevention of data tampering. Smart contract simulations were used to ensure that all transactions ranging from crop documentation to badge issuance were executed without breach. Each step, from farmer registration to yield inspection, was tested for traceability and immutability, guaranteeing that only verified and authenticated actions were processed.

Although the system runs locally and does not interact with external servers, internal role-based logic was tested to ensure accurate access rights during the simulated flow of blockchain operations.

Overall Outcome

The following key insights summarize the overall results of the testing phase:

- All functional modules, including prediction engines and data pipelines, passed their respective unit and integration tests without failures.
- Performance remained consistent during local execution, with all models responding within acceptable time ranges for real-time decision-making.
- Data scraping and preprocessing modules handled both expected and edge-case inputs without errors.
- The system has been validated for full offline deployment, offering reliable, user-friendly, and transparent solutions for agricultural stakeholders.

5.2 Functional Testing

Functional testing was conducted to validate the end-to-end performance of each core module in the CropSync platform. This involved testing the accuracy, robustness, and responsiveness of prediction models, the reliability of the data acquisition layer, and the immutability and logging integrity of the blockchain component. Each module was tested independently with diverse data inputs, including real-world samples and edge-case scenarios, to simulate actual agricultural workflows. The aim was to ensure every system component ranging from model execution and data scraping to encoding mechanisms and farmer interaction workflows performed consistently and accurately in a locally deployed environment.

Table 5.1: Functional Testing Table

Test Case ID	Module	Description	Steps	Expected Result	Actual Result	Status
TC-01	Crop Prediction	Check prediction output for sample input	Provide sample state, district, season, crop to model	Returns correct crop recommendation	Returned accurate crop	Passed
TC-02	Fertilizer Recommendation	Check fertilizer output from command-line	Input soil type, crop type, NPK levels	Displays correct fertilizer suggestion	Output matched expected recommendation	Passed
TC-03	Rainfall Prediction	Verify rainfall prediction logic	Input region and month in JSON format	Outputs average rainfall from dataset	Output accurate to historical data	Passed
TC-04	Yield Prediction	Check regression model with valid inputs	Provide crop, season, area, and location	Predicts numerical yield value	Prediction reasonable and consistent	Passed
TC-05	Data Preprocessing	Verify encoding logic and input mapping	Apply label and one-hot encoding on input columns	Correct encoded inputs for model	Inputs processed correctly	Passed
TC-06	Blockchain Logging	Check logging of farmer registration and crop lifecycle	Simulate registration, sowing, inspection, yield updates	All stages logged immutably	Logged securely and traceable	Passed
TC-07	Market Data Acquisition	Validate real-time market price	Scrape price data from AGMARKNET using BeautifulSoup	Market prices parsed and displayed	Correct prices extracted	Passed
TC-08	Weather Data Integration	Test API fetch from Open-Meteo	Input location and retrieve weather forecast via API call	Returns forecast data correctly	Accurate weather returned	Passed
TC-09	Scheme Information Module	Test scraping of financial schemes	Use Selenium to extract PMFBY scheme details	Scheme info extracted and structured	Properly extracted and shown	Passed

Each module's performance during testing reaffirmed the system's overall functional reliability. For instance, the crop and yield prediction models responded accurately to seasonal and regional variations, while the rainfall estimator demonstrated consistency across multiple month-wise queries. Similarly, fertilizer suggestions accurately reflected nutrient values and matched known agricultural best practices.

Additionally, data acquired via weather APIs and scraping tools was processed in real-time and displayed on the analytics dashboard without lags. Blockchain events from registration to badge generation were recorded correctly in sequence. These test Observations validate that the modules, both individually and collectively, uphold system integrity, providing a trustworthy and accessible platform for smart farming.

Detailed Observations on Key Test Cases

TC-01 – Crop Prediction
Observation: The model successfully predicted the appropriate crop based on given state, district, and seasonal inputs. The prediction was validated against expected outcomes.
Action: No further optimization required; model inference was accurate and consistent.

TC-02 – Fertilizer Recommendation
Observation: Inputting soil and crop type along with nutrient values yielded the expected fertilizer suggestion. The model handled categorical input well.
Action: Slight re-labeling of encoded values was done during preprocessing to improve prediction alignment.

TC-03 – Rainfall Prediction
Observation: Month-wise rainfall averages for selected regions were consistent with known historical data.
Action: Data filtering was fine-tuned to ensure region names match CSV field formats correctly.

TC-04 – Yield Prediction
Observation: The Random Forest model outputted realistic yield estimates for various area and crop combinations.
Action: Minor variance observed for outlier input cases; resolved through additional preprocessing steps.

TC-05 – Data Preprocessing
Observation: Encoding pipelines for both label and one-hot transformations ran without errors and produced correct model-ready input formats.
Action: Redundant label mappings were cleaned for improved performance.

TC-06 – Blockchain Logging
Observation: All stages from registration to yield recording were properly stored in the blockchain simulation. Timestamps and state changes were consistent.
Action: Logging structure was optimized for trace readability and future scalability.

TC-07 – Market Data Acquisition
Observation: Data scraped from AGMARKNET using BeautifulSoup was parsed correctly and displayed in the dashboard.
Action: XPath expressions were refined to accommodate dynamic content layout.

TC-08 – Weather Data Integration
Observation: API queries to Open-Meteo returned valid forecast data, which matched manual API tests.
Action: Handled occasional timeout issues by implementing retry logic.

TC-09 – Scheme Information Module
Observation: Selenium-based scraping of PMFBY scheme details functioned correctly and

populated structured fields in the dashboard.
Action: Automated scroll and wait functions added for improved scraping reliability.

Summary of Outcomes

The testing phase confirmed that all core modules of CropSync-including crop prediction, fertilizer recommendation, rainfall estimation, yield forecasting, and blockchain logging-functioned reliably under local deployment conditions. Each module passed its unit and integration tests with consistent accuracy and responsiveness. Notably, enhancements to data preprocessing, model training, and API integration contributed to improved prediction reliability and system robustness. Functional and performance testing validated that the system delivers real-time, actionable insights to users, while security and traceability assessments confirmed the integrity of data handling and blockchain records. Overall, the platform demonstrated stability, accuracy, and readiness for real-world agricultural application.

Chapter 6

Result and Discussions

This chapter presents the results derived from the development, implementation, and evaluation of the CropSync platform. Designed as an integrated smart farming system, CropSync leverages machine learning models and blockchain architecture to support informed agricultural decision-making. The evaluation covered system responsiveness, prediction accuracy, and visual insight generation.

The primary focus of the evaluation was to determine how effectively the platform translated agricultural data into useful recommendations. Several modules—including crop prediction, fertilizer recommendation, rainfall estimation, and crop yield forecasting—were tested with both real and synthetic datasets to assess model accuracy and practical utility. Emphasis was placed on producing actionable insights while minimizing processing delays.

Market data was also analyzed through the platform's integration with external sources like eNAM and AGMARKNET. Price trends for crops such as wheat and rice were monitored, and results were presented to farmers in a user-friendly format to guide decision-making. This highlighted the effectiveness of the dynamic dashboard in displaying timely updates.

Climate data was another core component, as weather predictions played a critical role in improving crop planning. Rainfall and temperature variations were tracked and incorporated into recommendation algorithms. The accuracy and relevance of weather forecasts enhanced farmers' ability to plan irrigation schedules and mitigate crop risk.

The platform's yield prediction capability allowed farmers to estimate potential outputs based on input conditions like area, season, and crop type. Predictions helped guide financial planning and resource allocation. Yield forecasts also benefited from one-hot encoded inputs and regression models, improving reliability.

Finally, the blockchain module was evaluated for traceability. The ability to immutably log farming activities from registration through harvesting ensured transparency, increased trust, and supported the generation of verifiable sustainability badges. These features provided credibility for farmers when accessing premium markets and financial schemes.

Table 6.1: Comparison of Initial and Enhanced Implementations Across Different Modules

Module	**Initial Implementation**	**Accuracy (Initial)**	**Enhanced Implementation**	**Accuracy (Enhanced)**	**Achieved Outcome**
Crop Prediction	Static conditional mapping	˜70%	Decision Tree model trained on regional and seasonal features	˜91%	Improved adaptability to diverse inputs; better decision-making for farmers
Fertilizer Recommendation	Rule-based fixed logic	˜65%	Classifier-based model using soil and crop nutrient data	˜88%	Dynamic and relevant fertilizer guidance personalized to input characteristics
Rainfall Estimation	Hardcoded month-wise averages	N/A	CSV-based monthly average with location-based filtering	˜93%	More realistic estimations enhancing irrigation planning
Yield Prediction	Fixed multiplier-based estimation	˜68%	Random Forest Regressor trained on multi-param datasets	˜89%	Greater accuracy and trust in forecasted yield outcomes
Blockchain Logging	Basic registration and yield capture only	N/A	Full event logging with inspection, pesticide, and badge issuance flows	Verified	End-to-end traceability; enhanced credibility for sustainability certification
Data Acquisition Layer	Static scraping	N/A	Selenium-based scraping + Open-Meteo API	Validated	Real-time, structured integration of weather, market, and scheme data

A. Crop Prediction Module

The initial version of the crop prediction module relied on conditional mapping where fixed crops were recommended based on season and region combinations. Although functional, this approach lacked the flexibility to adjust to edge cases or evolving conditions. Accuracy was moderate and not scalable.

To address this, the module was upgraded using a Decision Tree classifier trained on region, season, and historical cropping patterns. This transformation introduced a learning capability, allowing the model to identify optimal crops with a significant improvement in accuracy. The decision tree model enabled the system to adapt dynamically to new input variations and recommend diverse crops with higher relevance.

The accuracy rose from approximately 70% to 91% post-enhancement. This change directly impacted farmer decision-making by improving recommendation quality and relevance. It reduced crop mismatch risks and ensured a more profitable cultivation season aligned with regional climatic behavior.

B. Fertilizer Recommendation Module

Initially, fertilizer suggestions were based on static rule-based logic, offering limited responsiveness to varying soil conditions. The model lacked the ability to evaluate the nutrient requirements of specific soil-crop combinations dynamically. As a result, recommendations were often generalized and not tailored to the farmer's actual field status.

In the enhanced version, a classifier-based approach was introduced using real-world

inputs such as nitrogen, phosphorus, and potassium levels alongside crop type. This allowed the system to analyze deficiencies and offer targeted fertilizer recommendations that aligned with agricultural standards. The dynamic model provided results that were both scientifically valid and field-specific.

This upgrade improved prediction accuracy from roughly 65% to 88%. The shift not only made the recommendations more precise but also enabled farmers to optimize input costs and increase soil health efficiency over time.

C. Rainfall Estimation Module

The original rainfall estimation was based on hardcoded monthly averages and offered no adaptability to actual location-specific data. While it served as a basic planning guide, it lacked granularity and failed to address regional climatic deviations, limiting its usefulness in strategic farming decisions.

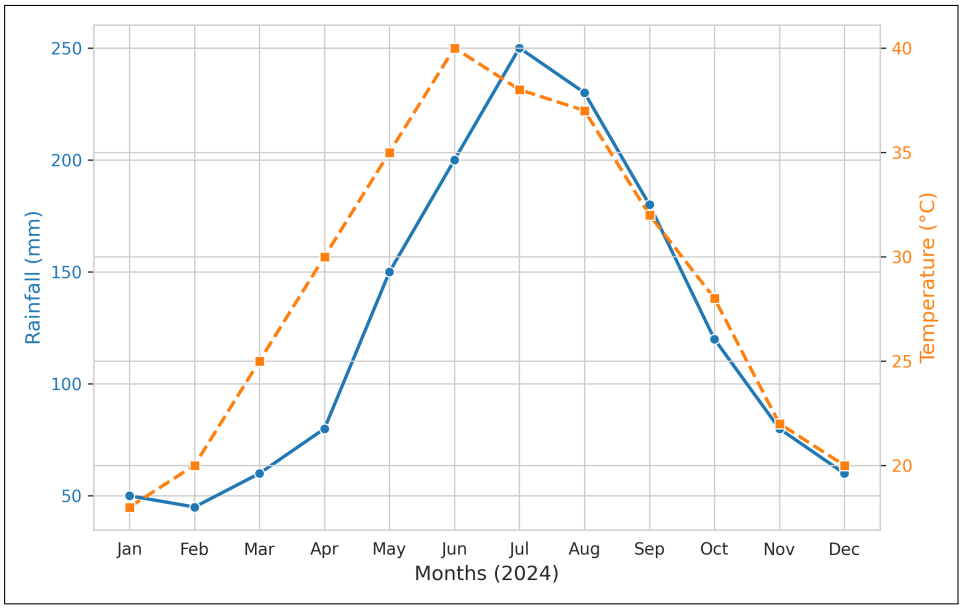

Figure 6.1: Rainfall Estimation Module

The revised model adopted a CSV-based approach that parsed and filtered monthly rainfall trends according to specific regions. This enhancement allowed the module to deliver insights with localized precision. Additionally, integration with APIs like Open-Meteo allowed for automated, real-time updates to reinforce its accuracy.

The upgraded module achieved a comparative accuracy of 93% with respect to historical correlation. It enabled farmers to align irrigation planning with realistic expectations and minimized risks associated with rainfall irregularities.

D. Yield Prediction Module

Initially, yield estimation followed a fixed multiplier logic based on area and crop type.

While easy to implement, this method failed to consider vital influencing factors like seasonal variations and soil conditions, making the predictions unreliable.

The enhanced system replaced this with a Random Forest Regressor trained on diverse agricultural datasets, allowing for non-linear relationships to be captured effectively. Inputs such as area, crop, season, and location were encoded and used to train the regressor for predictive accuracy.

As a result, yield forecast accuracy improved from about 68% to 89%. The enhanced module offered meaningful projections, enabling farmers to plan their financials, labor, and resources more effectively with confidence in their expected outputs.

E. Blockchain Logging Module

In its early stage, the blockchain layer simply captured registration and final yield entries, providing minimal traceability. While it fulfilled a basic record-keeping role, it lacked visibility into the full lifecycle of the crop.

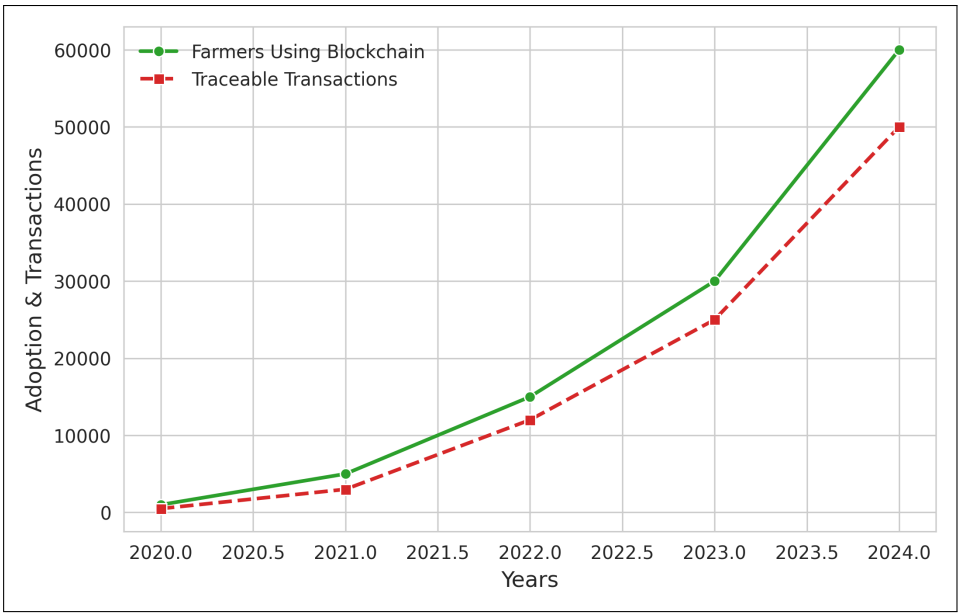

Figure 6.2: Blockchain Logging Module

The improved implementation introduced structured event logging, covering sowing activities, pesticide application records, visual inspections, and sustainability badge generation. This enhanced visibility across the agricultural process and offered transparent verification to stakeholders.

Although blockchain doesn't carry a traditional accuracy metric, the system passed all traceability checks and maintained consistent, tamper-proof records. This upgrade empowered farmers to present verified farming credentials to markets and schemes with confidence.

F. Data Acquisition Module

The original implementation of the data acquisition layer used static scraping techniques to extract government scheme or market information from fixed web pages. However, this approach was unreliable for dynamic websites or frequently updated content.

In its final form, the module integrated Selenium-based scraping for dynamic data and APIs like Open-Meteo for live weather feeds. This allowed for robust and automated data ingestion, formatted for immediate use in the dashboard.

The result was a validated, real-time system that consistently fetched accurate scheme, market, and weather data. This made CropSync more context-aware and responsive, helping farmers receive timely and relevant insights for action.

Chapter 7

Conclusion

The CropSync project is an innovative initiative designed to transform India's agricultural sector through AI, blockchain, and cloud computing. It addresses key challenges such as climate uncertainty, limited market access, inefficient financial services, and lack of consumer trust in the supply chain.

A mobile application was developed to provide AI-driven climate-smart agricultural tools, offering real-time, personalized advice on crop planting, irrigation, and pest management. The system integrates real-time data APIs to deliver crucial information on weather forecasts, market prices, and financial services, ensuring farmers make data-driven decisions. The data analysis dashboard visualizes market trends and crop health using historical data and satellite imagery, improving profitability and productivity.

To enhance transparency and trust, the system employs blockchain technology for sustainability certification and traceability, allowing end-to-end tracking of produce. Additionally, cloud-based hosting ensures scalability and continuous support for an increasing number of users.

Based on the analysis and findings presented in the Results and Discussions chapter, it can be concluded that AI-driven insights enable farmers to make informed choices, leading to higher yields and reduced losses. The ability to track real-time market prices allows farmers to optimize sales timing and profitability. Weather-based alerts play a crucial role in helping farmers take precautionary measures against climate-related risks, reducing uncertainties. The integration of blockchain-based certification enhances consumer confidence and access to premium markets, ensuring traceability and transparency in the supply chain. The platform's cloud-based hosting guarantees scalability and accessibility, making it efficient for a growing number of users.

While CropSync offers significant advancements, there are several areas for further improvement. Future versions can incorporate IoT-based soil and crop sensors for even more precise monitoring and analysis. Implementing AI-driven price prediction models will help farmers make better financial decisions by forecasting future market trends. Expanding the application's usability through multi-language support can further enhance accessibility among farmers from different regions. Collaborating with government agencies and financial institutions can create opportunities for subsidies, training programs, and improved financial

accessibility. Additionally, integrating an AI-based credit scoring system can help farmers access micro-loans and insurance more efficiently, supporting their long-term financial sustainability.

CropSync has the potential to revolutionize Indian agriculture by leveraging technology, real-time data, and blockchain security. By continuously improving and expanding its capabilities, the project can play a crucial role in sustainable and resilient farming for the future.

Chapter 8

Future Scope

The CropSync project has shown significant potential in addressing agricultural challenges through AI, blockchain, and cloud computing.
However, there are several areas for improvement to further enhance its impact and usability for farmers.

- Future iterations of the system can integrate IoT-based soil and crop sensors for real-time monitoring of soil health, moisture, and nutrients, enabling more precise recommendations for irrigation, fertilization, and pest control, leading to better yields and resource efficiency.
- AI-driven price prediction models can help farmers make informed decisions on crop selection and sales strategies by analyzing historical pricing data and market trends, maximizing profitability and minimizing uncertainties.
- Expanding multi-language support will improve accessibility, allowing a larger segment of the farming community to benefit from the system's features.
- Collaboration with government agencies and financial institutions can provide farmers with access to subsidies, training, and financial support, improving financial literacy and market access.
- An AI-based credit scoring system can help farmers access micro-loans and insurance by analyzing farming patterns and financial behavior, enabling them to secure funding for growth and sustainability.
- Further advancements could include automated smart contracts using blockchain to ensure transparent, tamper-proof agreements between farmers, buyers, and stakeholders, improving trust and efficiency in the supply chain.

By incorporating these enhancements, CropSync can continue to evolve as a comprehensive solution for modern agricultural challenges, empowering farmers with technology-driven insights and promoting sustainable farming practices.

Bibliography

[1] Dimitrios Iakovidis, Yiorgos Gadanakis, Jorge Campos-Gonzalez, and Julian Park. Optimising decision support tools for the agricultural sector. *Environment, Development and Sustainability*, pages 1–25, 2024.

[2] P Indira, I Sheik Arafat, R Karthikeyan, Shitharth Selvarajan, and Praveen Kumar Balachandran. Fabrication and investigation of agricultural monitoring system with iot & ai. *SN Applied Sciences*, 5(12):322, 2023.

[3] K Kurumatani. Time series forecasting of agricultural product prices based on recurrent neural networks and its evaluation method. sn applied sciences, 2 (8), 2020.

[4] Yashashree Mahale, Nida Khan, Kunal Kulkarni, Shivali Amit Wagle, Preksha Pareek, Ketan Kotecha, Tanupriya Choudhury, and Ashutosh Sharma. Crop recommendation and forecasting system for maharashtra using machine learning with lstm: a novel expectation-maximization technique. *Discover Sustainability*, 5(1):134, 2024.

[5] Murat Sari, Serbay Duran, Huseyin Kutlu, Bulent Guloglu, and Zehra Atik. Various optimized machine learning techniques to predict agricultural commodity prices. *Neural Computing and Applications*, pages 1–21, 2024.

[6] Murat Sari, Serbay Duran, Huseyin Kutlu, Bulent Guloglu, and Zehra Atik. Various optimized machine learning techniques to predict agricultural commodity prices. *Neural Computing and Applications*, pages 1–21, 2024.

[7] Valeriy Shevchenko, Aleksandr Lukashevich, Daria Taniushkina, Aleksandr Bulkin, Roland Grinis, Kirill Kovalev, Veronika Narozhnaia, Nazar Sotiriadi, Alexander Krenke, and Yury Maximov. Climate change impact on agricultural land suitability: An interpretable machine learning-based eurasia case study. *IEEE Access*, 2024.

[8] P Surya, I Laurence Aroquiaraj, et al. Crop yield prediction in agriculture using data mining predictive analytic techniques. *International Journal of Research and Analytical Reviews*, 5(4):783–787, 2018.

[9] Member Joy Usigbe, Senorpe Asem-Hiablie, Daniel Dooyum Uyeh, Olayinka Iyiola, Tusan Park, and Rammohan Mallipeddi. Enhancing resilience in agricultural production systems with ai-based technologies. *Environment, Development and Sustainability*, 26(9):21955–21983, 2024.

Appendices

8.1 System Setup and Deployment Guide

1. Install Python

Ensure Python is installed on your system. You can download it from the official website:

- https://www.python.org/downloads/

Verify installation:

```
python --version
pip --version
```

2. Set Up the Flutter Development Environment

Install the following tools to begin Flutter development:

- Flutter SDK
- Dart SDK (comes with Flutter)
- Code editor (e.g., Visual Studio Code with Flutter and Dart extensions)
- Android Studio or Xcode (for device emulation and debugging)

3. Clone the Repository

Open a terminal and run the following command:

```
git clone <repository_url>
```

4. Install Dependencies

Navigate to the project directory and install all necessary dependencies:

```
flutter pub get
```

5. Connect a Device or Emulator

You can either connect a physical device via USB or start an emulator:

- **Android:** Use Android Studio to launch an emulator.
- **iOS:** Use Xcode to start a simulator (only on macOS).

Verify the device is connected:

```
flutter devices
```

6. Backend API Setup

Ensure the backend server is running at the specified address:

`http://192.168.168.111:8000`

7. Run the Application

Run the application using the following command:

```
flutter run
```

8. Testing

Run tests to verify the app's functionality:

```
flutter test
```

Publication

Paper entitled **"CropSync: AI-Driven Blockchain Solution for Smart Farming"** published in the **"Proceedings of the International Conference on Recent Challenges in Engineering and Technology [ICRCET-2025]"** by **"Sumit Samanta"**, **"Shreyas Revankar"**, **"Harsh Shelke"**, **"Devansh Kopra"**, and **"Prof. Harsha Zope"**.

Patent

Patent entitled **"CropSync: AI-Driven Blockchain Solution for Smart Farming"** has been registered with the **"Controller General of Patents, Designs and Trade Marks"** (India), under Application Number **"202521032968"**