Unified Digital Platform for Smarter Farming

A PROJECT REPORT

Submitted by

AKSHAT NEOLIA [RA2211031010080]
JIYA GAYAWER [RA2211031010129]

Under the Guidance of

Dr. Gayathri V M

(Associate Professor, Department of Networking and Communications)

in partial fulfillment of the requirements for the degree

of

BACHELOR OF TECHNOLOGY

in
COMPUTER SCIENCE AND ENGINEERING
with specialization in INFORMATION TECHNOLOGY

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

DEPARTMENT OF NETWORKING AND COMMUNICATIONS
SCHOOL OF COMPUTING
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR- 603 203

APRIL 2025

Department of Networking and Communications
SRM Institute of Science & Technology
Own Work Declaration Form

Degree/ Course	**: B. Tech / CSE - IT**
Student Name	**: Akshat Neolia, Jiya Gayawer**
Registration Number	**: RA2211031010080, RA2211031010129**
Title of Work	**: Unified Digital Platform for Smarter Farming**

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is my / our own except where indicated, and that We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.

AKSHAT NEOLIA RA2211031010080
JIYA GAYAWER RA2211031010129 **Date:**

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 21CSP302L - Project report titled "**UNIFIED DIGITAL PLATFORM FOR SMARTER FARMING**" is the bonafide work of "**AKSHAT NEOLIA [RA2211031010080], JIYA GAYAWER [RA2211031010129]"** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. GAYATRI V M

SUPERVISOR,

ASSOCIATE PROFESSOR

DEPARTMENT OF NETWORKING
AND COMMUNICATIONS

SIGNATURE

Dr. M LAKSHMI

PROFESSOR & HEAD

DEPARTMENT OF NETWORKING
AND COMMUNICATIONS

Examiner 1

Examiner 2

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to **Dr. Leenus Jesu Martin M,** Dean-CET, SRM Institute of Science and Technology, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor and Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We encompass our sincere thanks to, **Dr. M. Pushpalatha**, Professor and Associate Chairperson - CS, School of Computing and **Dr. C. Lakshmi,** Professor and Associate Chairperson -AI, School of Computing, SRM Institute of Science and Technology, for their invaluable support.

We are incredibly grateful to our Head of the Department, **Dr. M. Lakshmi**, Professor & Head, Department of Networking and Communications, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Project Coordinators, Panel Head, and Panel Members Department of Networking and Communications, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, **Dr. Gouthaman P**, Department of Networking and Communications, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Dr. Gayathri V M,** Department of Networking and Communications, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under his / her mentorship. He / She provided us with the freedom and support to explore the research topics of our interest. His / Her passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank all the staff members of Networking and Communications, School of Computing, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support and encouragement.

Akshat Neolia [RA2211031010080]

Jiya Gayawer [RA2211031010129]

ABSTRACT

Agriculture remains the backbone of the Indian economy, yet many farmers continue to face significant challenges such as unpredictable crop prices, scattered and unreliable information sources, and minimal access to technology-driven decision-making tools. These issues often result in poor planning, financial instability, and underutilization of agricultural resources. To address these pressing concerns, this project introduces a Unified Digital Platform for Smarter Farming, a comprehensive and mobile-first solution that brings together various critical farming services under one roof.

The platform is designed to provide personalized crop recommendations, real-time crop price forecasts, localized weather updates, seasonal farming calendars, and activity tracking tools. It also includes features like SMS alerts for farmers in low-connectivity regions, discussion forums for peer interaction, access to government schemes, and helpline assistance for support. Its simple and intuitive design ensures that farmers from even the most remote areas can use it with ease, empowering them with timely, relevant, and actionable insights.

By centralizing vital agricultural information and making it accessible through mobile technology, the platform helps farmers make smarter decisions regarding what to grow, when to harvest, and how to market their produce. This not only improves their productivity and income but also contributes to the broader goals of sustainable agriculture, food security, and rural empowerment. In alignment with Sustainable Development Goal 2 (Zero Hunger), the project strives to reduce the digital divide in agriculture and enable a more resilient and self-reliant farming community through innovation and technology.

.

TABLE OF CONTENTS

APPENDIX

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ABBREVIATION	FULL FORM
AI	Artificial Intelligence
API	Application Programming Interface
CLI	Command Line Interface
CPU	Central Processing Unit
CSV	Comma-Separated Values
DL	Deep Learning
DFD	Data Flow Diagram
ERD	Entity-Relationship Diagram
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
ML	Machine Learning
MoSCoW	Must have, should have, could have, Won't have
MVP	Minimum Viable Product
NoSQL	Not Only SQL
OAuth	Open Authorization
OS	Operating System
PDF	Portable Document Format
QA	Quality Assurance
RAM	Random Access Memory
REST	Representational State Transfer
SDG	Sustainable Development Goal
SQL	Structured Query Language
SRS	Software Requirements Specification
UI	User Interface

CHAPTER 1

INTRODUCTION

1.1 Introduction to Unified Digital Platform for Smarter Farming

Agriculture remains the backbone of India's economy, employing over half the population. Despite its importance, farmers face several challenges—ranging from weather unpredictability and market price volatility to limited access to real-time, personalized information. Most rural farmers lack the tools needed to make data-driven decisions, resulting in inconsistent yields, income instability, and overall uncertainty.

It is a unified mobile application developed to address these challenges using the power of artificial intelligence, real-time data integration, and user-centric design. The platform provides a comprehensive suite of features including real-time weather updates, AI-based crop price forecasting, personalized crop recommendations, agricultural news, and government scheme alerts—all accessible via a smartphone or even basic phones through SMS-based alerts.

By integrating these components into a single, easy-to-use application, the platform ensures that farmers can make better decisions about sowing, harvesting, selling, and investing. It also includes a seasonal calendar, an activity tracker, and a helpline for community support. The app is built using Flutter for cross-platform compatibility, Firebase for secure and scalable backend support, and Python-based machine learning models for delivering accurate insights. It's tailored for rural environments, focusing on vernacular language support, offline accessibility, and low data consumption, making it a truly inclusive solution for smart agriculture.

1.2 Motivation

The Indian farming community continues to suffer from fragmented data sources, lack of real-time support, and minimal access to advanced tools. Existing agricultural platforms often work in silos, addressing either crop prediction or market forecasting—but not both together—and rarely cater to the on-ground realities of rural farmers. Moreover, digital tools are often designed with urban connectivity and usage patterns in mind, leaving behind millions of farmers with basic devices or poor internet access.

The motivation behind developing this platform is to **bridge the gap between traditional agriculture and modern technology**. Farmers need a single, reliable platform that offers contextual, personalized, and actionable insights. Our goal is to empower farmers with real-time updates, forecasted crop prices, and timely crop recommendations that consider local environmental conditions.

We also aim to provide farmers with a structured way to manage their farming schedules through activity tracking and seasonal planning tools, as well as to ensure they are aware of the latest government schemes and financial support opportunities. The addition of SMS-based alerts ensures that even farmers without internet access can stay informed. Ultimately, the platform strives to increase transparency, reduce risks, and improve overall income stability for Indian farmers.

1.3 Sustainable Development Goal of the Project

The Smart Farming Assistant Platform aligns closely with **Sustainable Development Goal (SDG) 2: Zero Hunger**, which aims to end hunger, achieve food security, improve nutrition, and promote sustainable agriculture.

This platform contributes to SDG 2 in multiple ways. By delivering accurate crop price predictions and crop suggestions based on season, region, and soil conditions, it helps stabilize agricultural output and reduces market unpredictability. Timely weather updates and seasonal calendars further allow farmers to optimize the use of water, fertilizers, and seeds, thus enhancing productivity and reducing waste.

Another core aspect of SDG 2 is inclusivity, and the platform addresses this through offline support and regional language interfaces. These features ensure accessibility across remote areas, making technology usable for all, regardless of literacy or connectivity. Furthermore, by enabling better planning and informed decision-making, the platform contributes to improving rural livelihoods and promoting self-reliance.

By embedding smart technologies into everyday farming practices, the platform paves the way for a **resilient, efficient, and sustainable agricultural ecosystem** that benefits both farmers and society at large.

1.4 Product Vision Statement

1.4.1 Audience

- **Primary Audience**: Small and medium-scale farmers across India who require localized, real-time insights to support their day-to-day agricultural decisions. This includes users with limited technological literacy and access to the internet.
- **Secondary Audience**: Agricultural extension officers, policy makers, researchers, and agri-tech organizations who can use the platform to analyze trends, guide policies, and provide support to farming communities.

1.4.2 Needs

- **Primary Needs**:
  - Accurate and real-time weather updates to protect crops and optimize irrigation schedules.
  - AI-driven crop price predictions to help decide the best time to sell and ensure fair pricing.
  - Personalized crop recommendations based on environmental data and user profile.
  - Offline SMS alerts to keep users informed even without internet.
- **Secondary Needs**:
  - Farm activity tracking to digitally log key actions like sowing, fertilization, and harvesting.
  - Seasonal calendar to plan for each farming cycle.
  - Awareness of government schemes, subsidies, and financial benefits.

1.4.3 Products

- **Core Product**: A mobile application offering weather updates, AI-based price and crop predictions, seasonal guides, and SMS alerts.
- **Additional Features**:
  - Farm activity history for analyzing past decisions.

  - News and updates from trusted agricultural sources.
  - Personalized dashboards and profiles.
  - Interactive helpline and discussion forum for peer support.

1.4.4 Values

- **Core Values**:
  - **Empowerment**: Equip farmers with tools that enhance autonomy and decision-making.
  - **Reliability**: Deliver timely, accurate, and useful insights with high performance.
- **Differentiators**:
  - Integration of multiple services (weather, market, government) in one app.
  - Offline-first approach with SMS delivery for critical alerts.
  - Hyper-local AI models trained on region-specific data for higher prediction accuracy.
  - Affordable and scalable tech stack suitable for long-term rural deployment.

1.5 Product Goal

The primary goal of the **Unified Digital Platform for Smarter Farming** is to empower Indian farmers by transforming the way they access and utilize agricultural information. The platform aims to provide a unified, AI-powered solution that delivers real-time weather updates, accurate crop price predictions, and personalized crop recommendations—all tailored to the farmer's location, seasonal context, and environmental conditions. By combining these features into a single, easy-to-use mobile application with offline SMS capabilities, the platform seeks to enhance the decision-making capacity of farmers, reduce dependency on unreliable market trends, and increase overall productivity and profitability.

This goal is grounded in the vision of making **smart agriculture accessible to all**, especially those in rural and low-connectivity areas. Through intuitive design, the platform breaks down the technological and infrastructural barriers that often exclude smallholder farmers from digital advancements.

Beyond individual support, the platform also fosters community-level impact by integrating features like activity tracking, helpline access, and awareness of government schemes. These tools encourage informed practices, sustainable farming methods, and financial inclusion across farming communities.

Ultimately, the product goal is to create a resilient agricultural ecosystem where technology is not just a tool but a **trusted companion in everyday farming**—enabling farmers to make smarter choices, share insights, and contribute to a more stable and sustainable food system in India.

1.6 Product Backlog

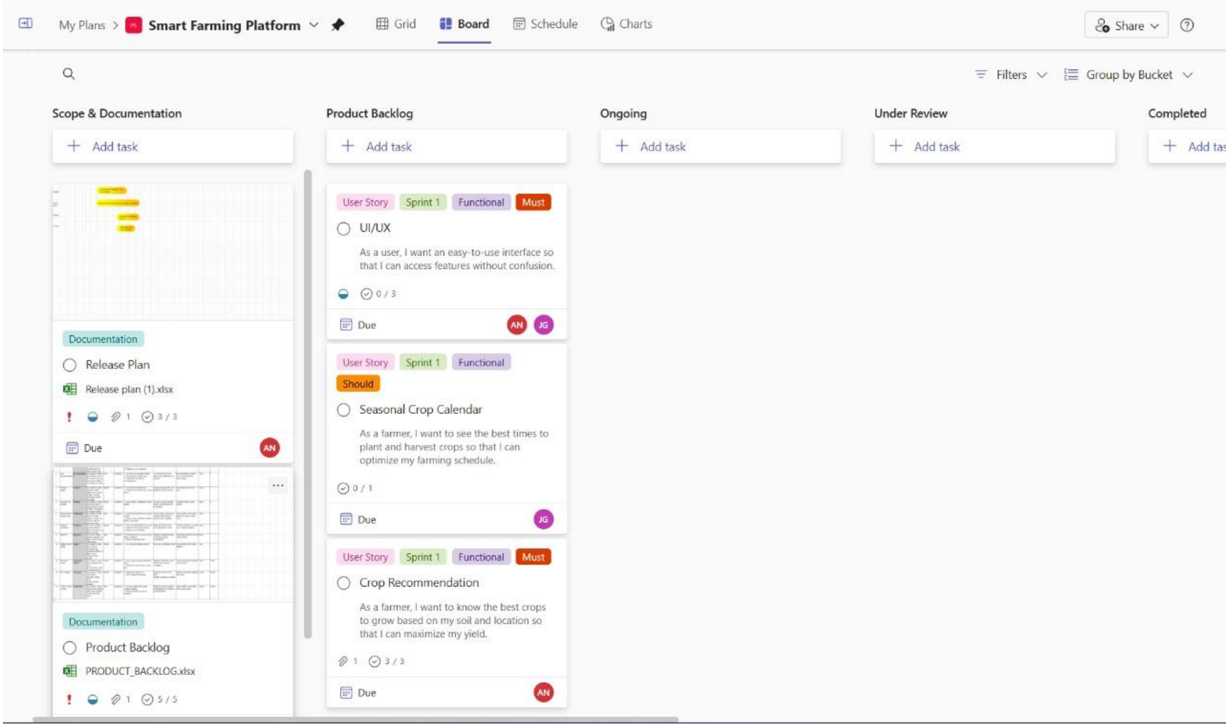
Table 1.1 User Stories

S. No	**User Stories of Unified Digital Platform for Smarter Farming**
#US 1	**As a user, I want** an easy-to-use interface **so that** I can access features without confusion.
#US 2	**As a farmer, I want** an integrated app where all my agricultural needs are available in one place **so that** I don't have to rely on multiple sources for information.
#US 3	**As a farmer, I want** to get future price predictions **so that** I can decide the best time to sell my crops.
#US 4	**As a farmer, I want** to know the best crops to grow based on my soil and location **so that** I can maximize my yield.
#US 5	**As a farmer, I want** real-time weather updates for my region **so that** I can plan agricultural activities accordingly.
#US 6	**As a farmer, I want** to see the best times to plant and harvest crops **so that** I can optimize my farming schedule.
#US 7	**As a farmer, I want** alerts for weather changes **so that** I can take precautions to protect my crops.
#US 8	**As a user, I want** to provide feedback **so that** the developers can improve the app.

#US 9	**As a user, I want** a bug-free experience **so that** I can use the app without issues.
#US 10	**As a farmer, I want** easy access to government and agricultural helplines **so that** I can get assistance when needed.
#US 11	**As a farmer, I want** to discuss problems and share tips with others **so that** I can learn and grow.
#US 12	**As a farmer, I want to** read current agriculture-related news **so that** I can stay informed.
#US 13	**As a farmer, I want** to track my farm activities and receive reminders **so that** I don’t miss tasks.

The product backlog of Unified Digital Platform for Smarter Farming in **Table 1.1** was configured using the MS planner Agile Board which is represented in the following **Figure 1.1**. The Product Backlog consists of the complete user stories of Unified Digital Platform for Smarter Farming

Each user story consists of necessary parameters like MoSCoW prioritization, Functional and non-functional parameters, detailed acceptance criteria with linked tasks.



As a user, I want an easy-to-use interface so that I can access features without confusion.

As a farmer, I want to see the best times to plant and harvest crops so that I can optimize my farming schedule.

As a farmer, I want to know the best crops to grow based on my soil and location so that I can maximize my yield.

Figure 1.1 MS Planner Board

1.7 Product Release Plan

The following **Figure 1.2** depicts the release plan of the project

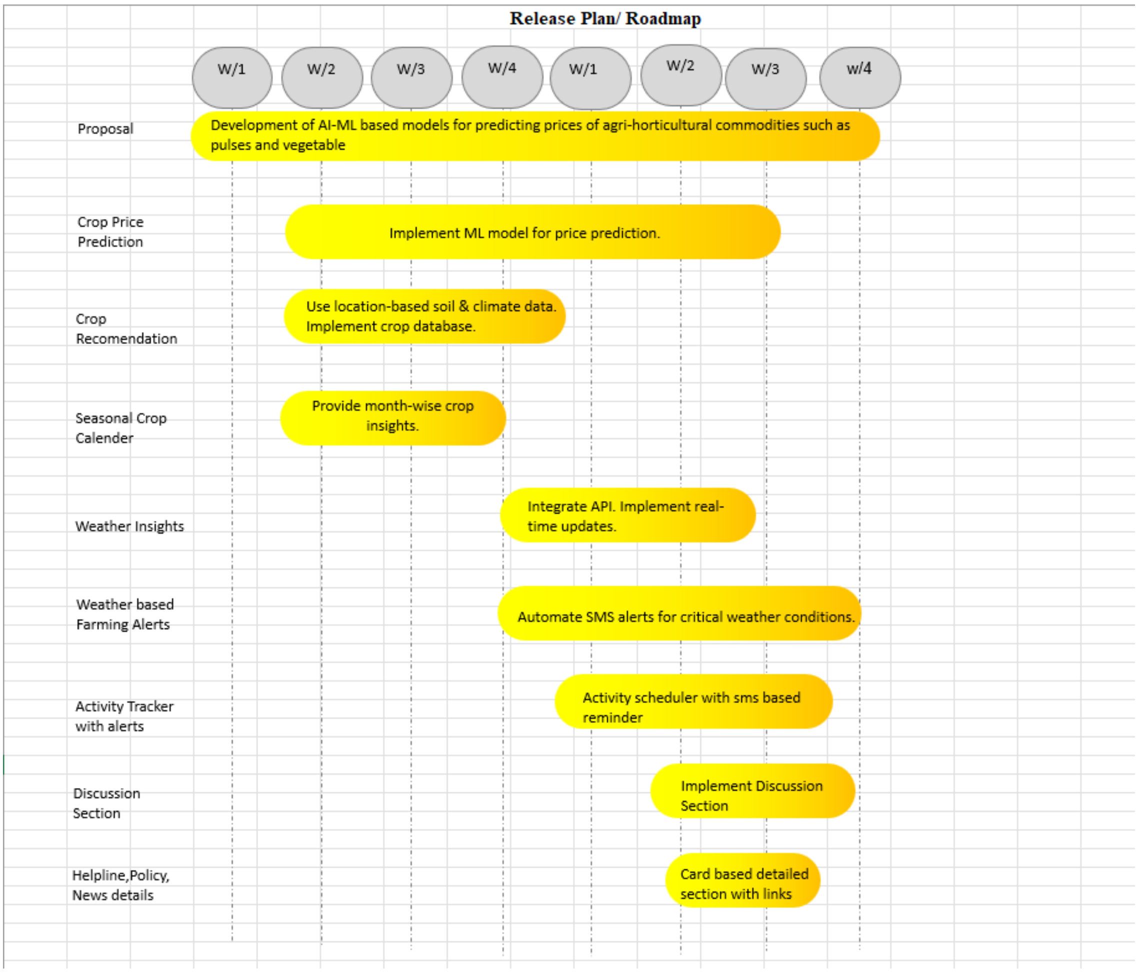


Figure 1.2 Release plan

Figure 1.2 shows the project's release plan over 8 weeks, highlighting the phased development of features. Initial weeks focus on building AI/ML models for crop price prediction and recommendation. Mid-phase includes real-time weather insights and automated SMS alerts. The final phase adds community discussions, activity tracking, and informative cards for helplines and news.

CHAPTER 2

SPRINT PLANNING AND EXECUTION

2.1 SPRINT 1

2.1.1 Sprint Goal with User Stories of Sprint 1

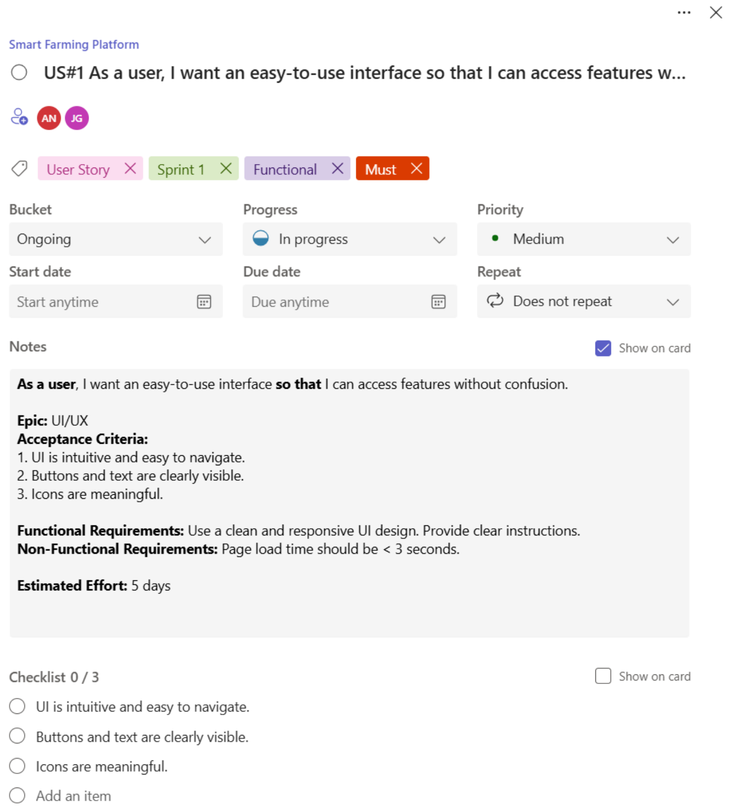
The Goal of the first sprint is to construct the intuitive UI, seasonal crop calendar, support sections and a robust machine learning model for crop recommendations.

The following **Table 2.1** represents the detailed user stories of the sprint 1.

Table 2.1 Detailed User Stories of sprint 1

S.NO	Detailed User Stories
US #1	As a new user, I want an easy-to-use interface so that I can access features without confusion.
US #2	As a farmer, I want to see the best times to plant and harvest crops so that I can optimize my farming schedule.
US #3	As a farmer, I want to know the best crops to grow based on my soil and location so that I can maximize my yield.
US #4	As a farmer, I want easy access to government and agricultural helplines, schemes so that I can get assistance when needed.

Planner Board representations of user stories are mentioned below **figures 2.1, 2.2, 2.3** and **2.4**.



Smart Farming Platform

○ US#1 As a user, I want an easy-to-use interface so that I can access features w...

AN JG

User Story ✕ Sprint 1 ✕ Functional ✕ Must ✕

Bucket	Progress	Priority
Ongoing	In progress	● Medium
Start date	**Due date**	**Repeat**
Start anytime	Due anytime	Does not repeat

Notes ☑ Show on card

As a user, I want an easy-to-use interface **so that** I can access features without confusion.

Epic: UI/UX
Acceptance Criteria:
1. UI is intuitive and easy to navigate.
2. Buttons and text are clearly visible.
3. Icons are meaningful.

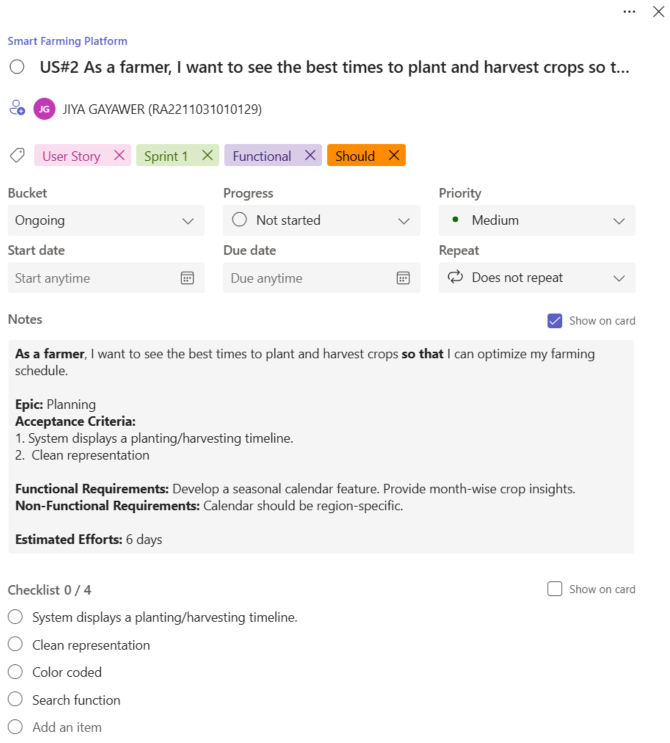
Functional Requirements: Use a clean and responsive UI design. Provide clear instructions.
Non-Functional Requirements: Page load time should be < 3 seconds.

Estimated Effort: 5 days

Checklist 0 / 3 □ Show on card

○ UI is intuitive and easy to navigate.
○ Buttons and text are clearly visible.
○ Icons are meaningful.
○ Add an item

Figure 2.1 User Story#1



Smart Farming Platform

○ US#2 As a farmer, I want to see the best times to plant and harvest crops so t...

JG JIYA GAYAWER (RA2211031010129)

User Story ✕ Sprint 1 ✕ Functional ✕ Should ✕

Bucket	Progress	Priority
Ongoing	○ Not started	● Medium
Start date	**Due date**	**Repeat**
Start anytime	Due anytime	Does not repeat

Notes ☑ Show on card

As a farmer, I want to see the best times to plant and harvest crops **so that** I can optimize my farming schedule.

Epic: Planning
Acceptance Criteria:
1. System displays a planting/harvesting timeline.
2. Clean representation

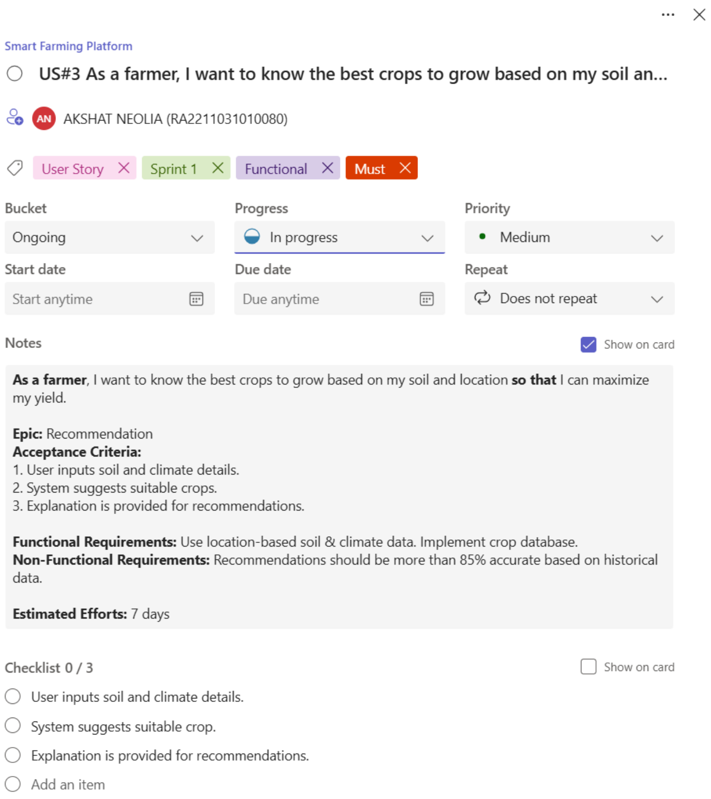
Functional Requirements: Develop a seasonal calendar feature. Provide month-wise crop insights.
Non-Functional Requirements: Calendar should be region-specific.

Estimated Efforts: 6 days

Checklist 0 / 4 □ Show on card

○ System displays a planting/harvesting timeline.
○ Clean representation
○ Color coded
○ Search function
○ Add an item

Figure 2.2 User Story#2



○ US#3 As a farmer, I want to know the best crops to grow based on my soil an...

AN AKSHAT NEOLIA (RA2211031010080)

User Story ✕ Sprint 1 ✕ Functional ✕ Must ✕

Bucket	Progress	Priority
Ongoing	In progress	● Medium
Start date	**Due date**	**Repeat**
Start anytime	Due anytime	Does not repeat

Notes ☑ Show on card

As a farmer, I want to know the best crops to grow based on my soil and location **so that** I can maximize my yield.

Epic: Recommendation
Acceptance Criteria:
1. User inputs soil and climate details.
2. System suggests suitable crops.
3. Explanation is provided for recommendations.

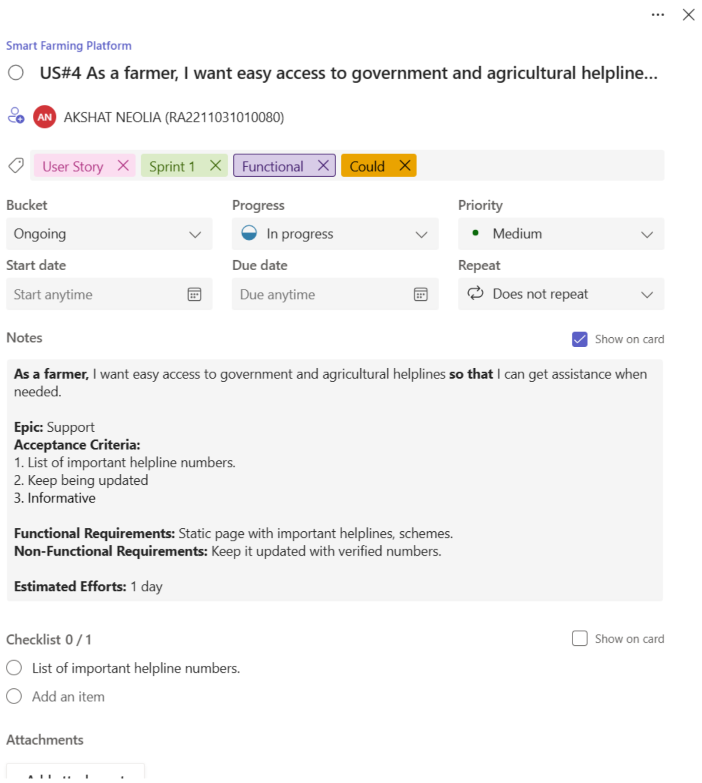
Functional Requirements: Use location-based soil & climate data. Implement crop database.
Non-Functional Requirements: Recommendations should be more than 85% accurate based on historical data.

Estimated Efforts: 7 days

Checklist 0 / 3 □ Show on card

○ User inputs soil and climate details.
○ System suggests suitable crop.
○ Explanation is provided for recommendations.
○ Add an item

Figure 2.3 User Story#3



○ US#4 As a farmer, I want easy access to government and agricultural helpline...

AN AKSHAT NEOLIA (RA2211031010080)

User Story ✕ Sprint 1 ✕ Functional ✕ Could ✕

Bucket	Progress	Priority
Ongoing	In progress	● Medium
Start date	**Due date**	**Repeat**
Start anytime	Due anytime	Does not repeat

Notes ☑ Show on card

As a farmer, I want easy access to government and agricultural helplines **so that** I can get assistance when needed.

Epic: Support
Acceptance Criteria:
1. List of important helpline numbers.
2. Keep being updated
3. Informative

Functional Requirements: Static page with important helplines, schemes.
Non-Functional Requirements: Keep it updated with verified numbers.

Estimated Efforts: 1 day

Checklist 0 / 1 □ Show on card

○ List of important helpline numbers.
○ Add an item

Attachments

Figure 2.4 User Story#4

2.1.2 Functional Document (Sprint -1)

2.1.2.1. Introduction

This sprint establishes the foundation for the Smart Agriculture platform through a streamlined UI/UX, personalized crop recommendations, season-based crop calendars, and a static helpline content module. These features empower farmers to plan efficiently and reach out for help as needed.

2.1.2.2. Product Goal

The goal of this project is to develop a user-friendly digital platform that:

- Simplify user experience for easy app navigation.
- Recommend crops tailored to land and climate.
- Help farmers plan activities using a seasonal calendar.
- Provide helpline details for quick assistance.

2.1.2.3. Demography (Users, Location)

Users:

- Indian farmers (small to large scale)
- Agricultural consultants

Location:

- Recommendation system tailored for Maharashtra.

2.1.2.4. Business Processes

1. Data Collection & Preprocessing:
   Gather crops, soil, weather, and regional sowing patterns. Clean, normalize, and process data for machine learning models.
2. Model Training:
   Use historical crop data to develop a crop recommendation model tuned for regional environmental conditions.
3. Crop Calendar Integration:
   Curate region-specific crop stages based on sowing and harvesting seasons, creating a visual timeline.

4. UI Development with Flutter:
   Build the frontend using Flutter for deployment on Android, ensuring smooth navigation and quick access.
5. Helpline Information Module:
   Integrate legitimate helpline numbers. Ensure offline accessibility of these details.

2.1.2.5 Features –

Feature 1 - Simple UI (UI/UX)

Description:

Basic, clean interface with intuitive navigation and fast loading.

User Story:

As a user, I want an easy-to-use interface so that I can access features without confusion.

Feature 2 - Crop Recommendation

Description:

Suggests crops based on region-specific data like climate and soil.

User Story:

As a farmer, I want to know the best crops to grow based on my soil and location so that I can maximize my yield.

Feature 3 - Seasonal Crop Calendar

Description:

Region-specific crop activity planner for each stage of farming.

User Story:

As a farmer, I want to see the best times to plant and harvest crops so that I can optimize my farming schedule.

Feature 4 - Helpline Content Details

Description:

Displays emergency and support numbers for farming help.

User Story:

As a farmer, I want easy access to government and agricultural helplines so that I can get assistance when needed.

2.1.2.6. Authorization Matrix

Table 2.2 Access level Authorization Matrix

Role	**Access Level**
Farmer	Access calendar, recommendations, UI
Admin	Manage content and backend data

Table 2.2 shows that Farmers have access to the calendar, recommendations, and UI, while Admins manage content and backend data.

2.1.2.7 Assumptions

- Farmers may have low to moderate literacy; hence, UI must prioritize icons, visuals, and minimal text.
- The helpline content is expected to be static for the duration of this sprint, with periodic updates in future sprints.
- Offline access to helpline details and crop calendars is critical due to limited rural connectivity.

2.1.3 Architecture Document

2.1.3.1. Application

Chosen Methodology: **Microservices + Serverless with Firebase**

This methodology provides several key benefits:

- Modular and Scalable -The microservices approach divides the platform into independent services such as weather updates, crop price prediction, and SMS alerts. These services can be developed, deployed, and scaled individually. Firebase, being serverless, handles automatic scaling and resource management seamlessly, removing the need for manual intervention.

- Cost-Efficient -Serverless infrastructure ensures that resources are consumed only when needed. This pay-as-you-go model minimizes infrastructure costs and prevents unnecessary over-provisioning.

- Faster Development - Firebase simplifies backend operations by offering built-in tools like Firestore, Authentication, and Cloud Functions. This significantly accelerates development and deployment timelines.

- Easier Maintenance - Microservices are loosely coupled, allowing developers to update or fix individual services without affecting the overall system. This enhances version control and reduces downtime during updates.

- High Reliability - Firebase's tight integration with Google Cloud offers high availability, security, and stability. It ensures that the application is always up and running smoothly for end-users.

From User's Perspective:

- Scalable Experience: As the platform grows, users won't experience delays or performance issues since features scale independently.

- Cost-Effective: With serverless, the platform runs efficiently, leading to fewer interruptions and better service quality.

- Faster Updates: Users benefit from quicker improvements, bug fixes, and new features due to easier maintenance and development.

- Reliable: Users can rely on a platform with robust security and consistent uptime, ensuring smooth functionality.

Why Not Any Other Architecture?

- Event-Driven: While it offers real-time updates, it adds complexity in managing events and queues. For our project's needs (e.g., ML models, weather data), Microservices with Serverless is simpler and more scalable.

- Just Microservices: Although microservices are modular, they still require manual scaling and infrastructure management. Serverless with Firebase simplifies these aspects, making it more cost-effective and easier to maintain.

- Just Serverless: Serverless alone doesn't provide the necessary structure for modular services like ML models, weather updates, or notifications. Microservices provide the needed separation and flexibility for a complex platform like ours.

2.1.3.2 System Architecture

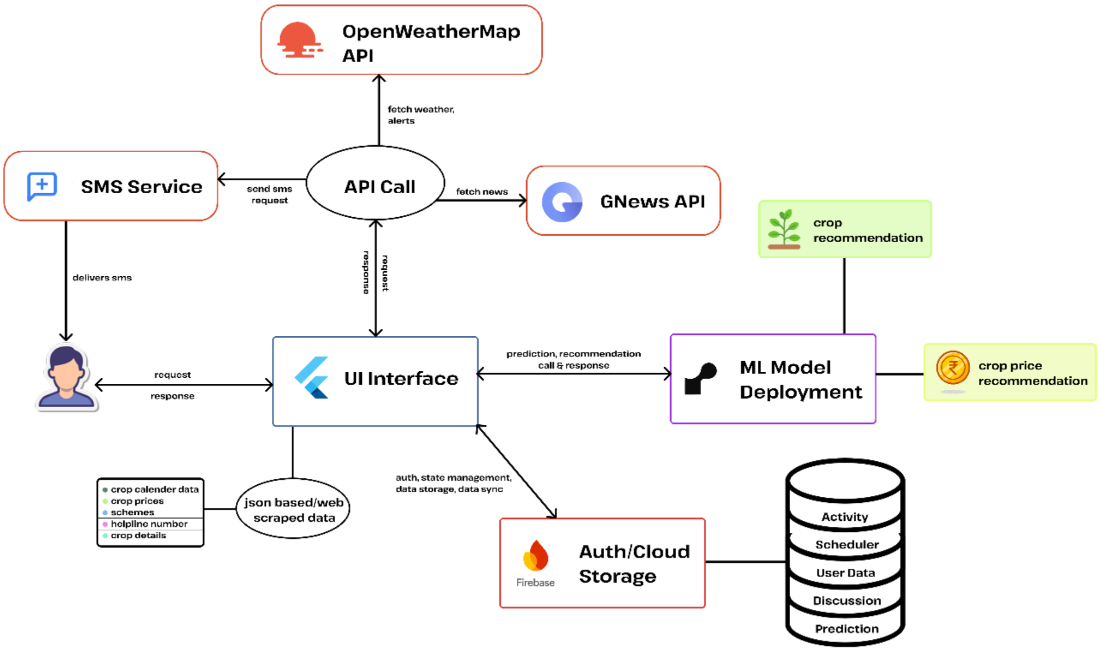


Figure 2.5 System Architecture Diagram

Figure 2.5 represents the system architecture of the "Unified Digital Platform for Smarter Farming," showcasing the integrated components that enable seamless functionality. The user interacts through a Flutter-based UI, which acts as the central interface for accessing various services. API calls connect the UI with third-party services like OpenWeatherMap for weather updates and alerts, GNews API for agricultural news, and an SMS service for offline alerts. The ML model deployment module handles crop price and recommendation predictions, which are displayed on the UI. Firebase provides authentication, data storage, state management, and sync functionality, while additional agricultural data such as crop calendars and schemes are sourced via JSON-based or web-scraped content. This architecture ensures a cohesive, scalable, and user-friendly experience tailored for Indian farmers.

2.1.3.3. Data Exchange Contract:

API-Based (Real-Time)

- OpenWeather API → Weather data integration
- TextBee API → Delivery of SMS alerts
- Firebase Auth API → User authentication and data sync
- UPAg API (if applicable) → Crop calendar data

Web Scraping (Scheduled)

- UPAg Portal → Seasonal crop information (if API unavailable)
- Government/market websites → Crop prices and trends

Database Storage (On-Demand Access)

- Firebase / Firestore → Stores and retrieves user data, prediction outputs, and app activities
- PostgreSQL (where applicable) → Alternative structured storage for analytics and SMS logs

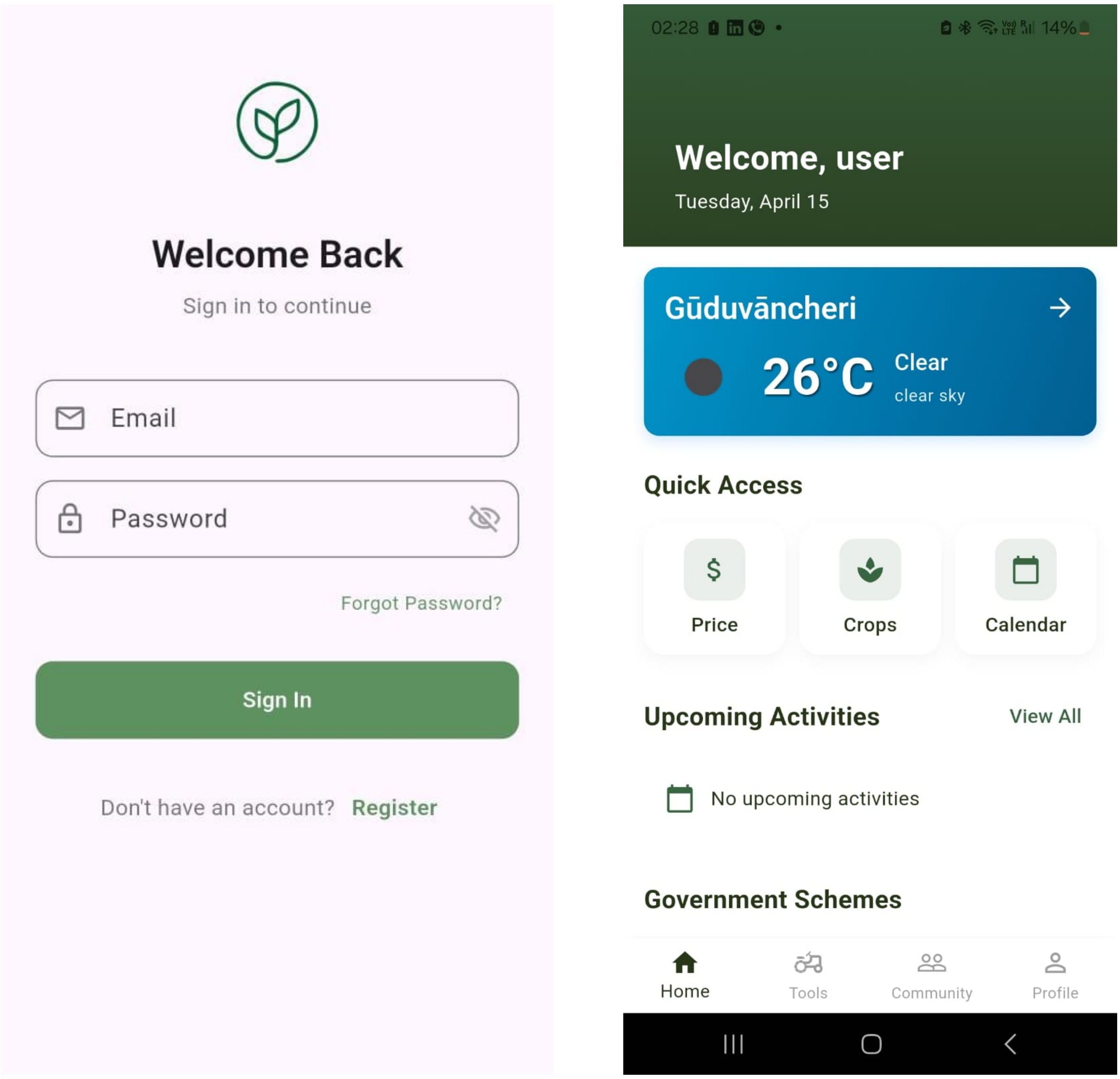
Message Queue (Asynchronous Processing)

- SMS Queue → Queues alerts based on real-time data triggers
- Prediction Processing Queue → Manages asynchronous ML predictions and updates

File-Based Exchange (Batch Processing)

- CSV/JSON Exports → For reports on price trends and recommendations
- External Imports → Integration of data from official publications or field reports

2.1.4 UI DESIGN



Welcome Back

Sign in to continue

Welcome, user

Tuesday, April 15

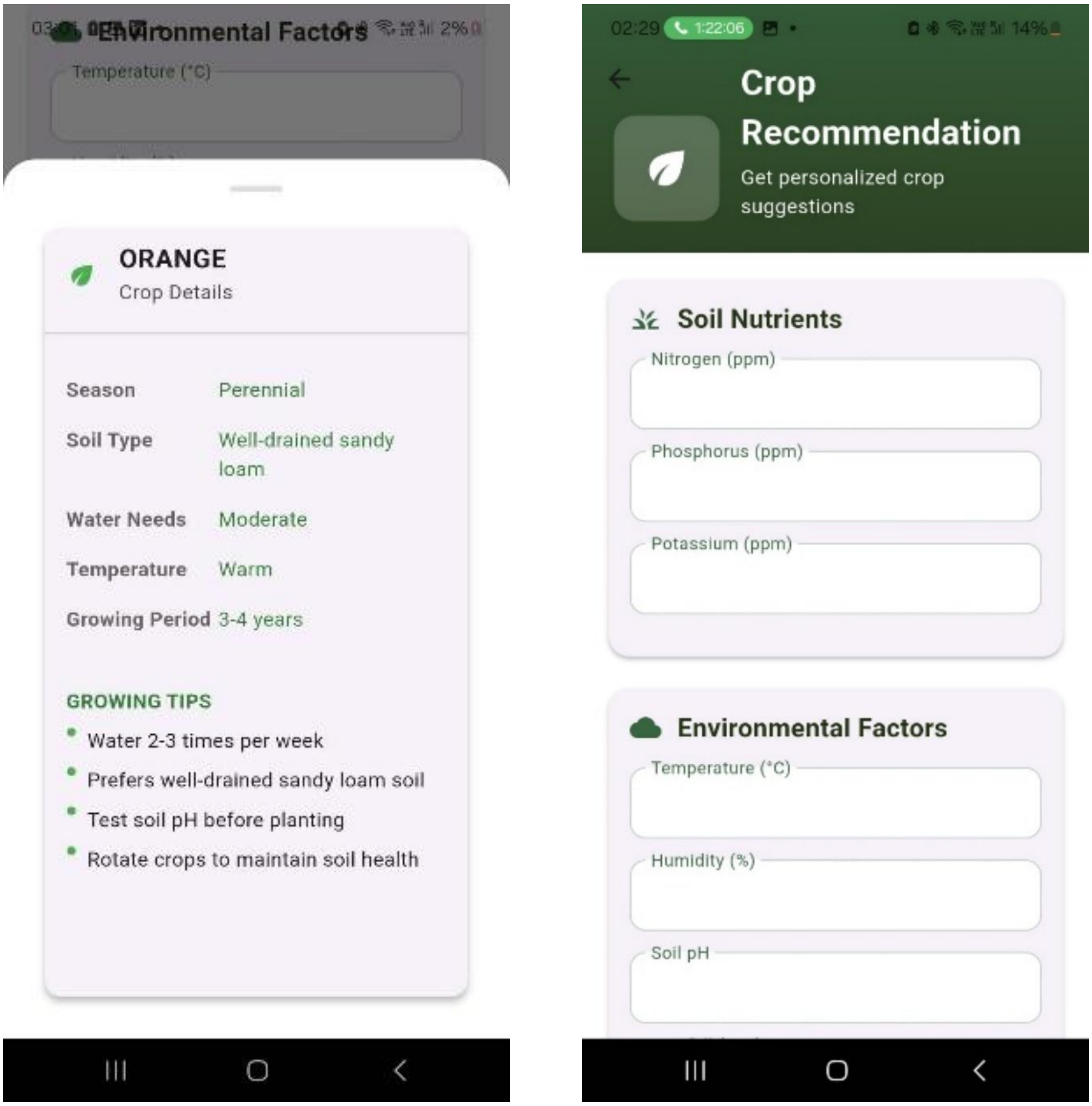
Gūduvāncheri

26°C Clear clear sky

No upcoming activities

Figure 2.6 Login Page and Landing Page UI

Figure 2.6 presents the **Login Page** and **Landing Page UI** of the Unified Digital Platform for Smarter Farming. The Login Page ensures secure access for registered farmers through email authentication. The Landing Page offers a personalized greeting, real-time weather for the user's location, and quick access to key features like price prediction, crop insights, and calendar. It serves as a centralized hub for accessing farming tools and updates efficiently.



Environmental Factors

Temperature (°C)

ORANGE

Crop Details

Season	Perennial
Soil Type	Well-drained sandy loam
Water Needs	Moderate
Temperature	Warm
Growing Period	3-4 years

GROWING TIPS

- Water 2-3 times per week
- Prefers well-drained sandy loam soil
- Test soil pH before planting
- Rotate crops to maintain soil health

Crop Recommendation

Get personalized crop suggestions

Soil Nutrients

Nitrogen (ppm)

Phosphorus (ppm)

Potassium (ppm)

Environmental Factors

Temperature (°C)

Humidity (%)

Soil pH

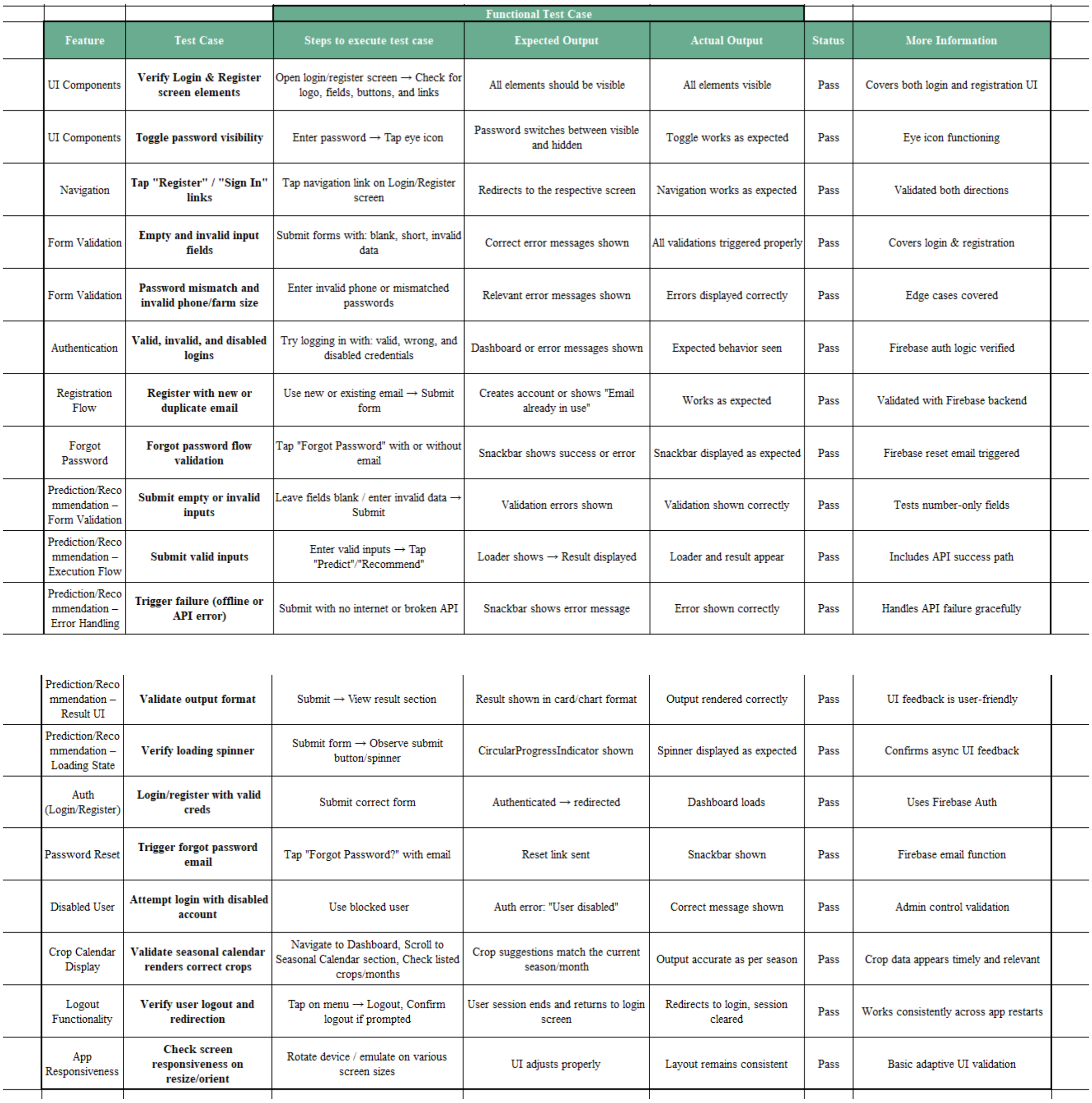
Figure 2.7 Crop Recommendation and Details UI

Figure 2.7 showcases the **Crop Recommendation** and **Crop Details UI**. The recommendation screen collects inputs on soil nutrients and environmental factors to suggest suitable crops. Once a crop is selected, detailed information is displayed, including soil type, temperature, watering needs, and expert growing tips. This helps farmers make informed decisions based on their local conditions.

2.1.5 Functional Test Cases

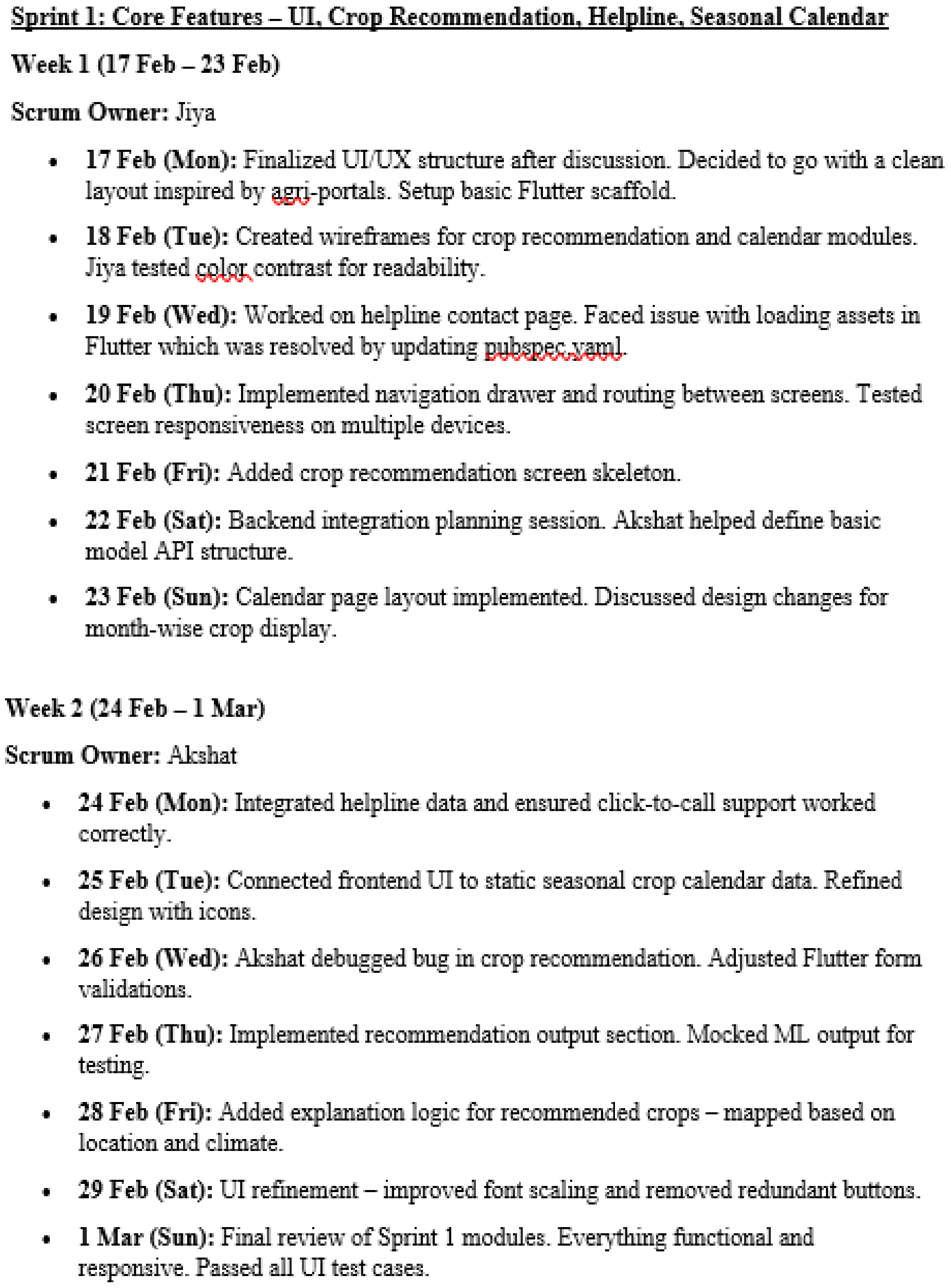
The following **Table 2.3** shows the functional testcases that were tested in the Sprint 1 of the SDLC.

Table 2.3 Detailed Functional Test Case



		Functional Test Case				
Feature	**Test Case**	**Steps to execute test case**	**Expected Output**	**Actual Output**	**Status**	**More Information**
UI Components	**Verify Login & Register screen elements**	Open login/register screen → Check for logo, fields, buttons, and links	All elements should be visible	All elements visible	Pass	Covers both login and registration UI
UI Components	**Toggle password visibility**	Enter password → Tap eye icon	Password switches between visible and hidden	Toggle works as expected	Pass	Eye icon functioning
Navigation	**Tap "Register" / "Sign In" links**	Tap navigation link on Login/Register screen	Redirects to the respective screen	Navigation works as expected	Pass	Validated both directions
Form Validation	**Empty and invalid input fields**	Submit forms with: blank, short, invalid data	Correct error messages shown	All validations triggered properly	Pass	Covers login & registration
Form Validation	**Password mismatch and invalid phone/farm size**	Enter invalid phone or mismatched passwords	Relevant error messages shown	Errors displayed correctly	Pass	Edge cases covered
Authentication	**Valid, invalid, and disabled logins**	Try logging in with: valid, wrong, and disabled credentials	Dashboard or error messages shown	Expected behavior seen	Pass	Firebase auth logic verified
Registration Flow	**Register with new or duplicate email**	Use new or existing email → Submit form	Creates account or shows "Email already in use"	Works as expected	Pass	Validated with Firebase backend
Forgot Password	**Forgot password flow validation**	Tap "Forgot Password" with or without email	Snackbar shows success or error	Snackbar displayed as expected	Pass	Firebase reset email triggered
Prediction/Recommendation – Form Validation	**Submit empty or invalid inputs**	Leave fields blank / enter invalid data → Submit	Validation errors shown	Validation shown correctly	Pass	Tests number-only fields
Prediction/Recommendation – Execution Flow	**Submit valid inputs**	Enter valid inputs → Tap "Predict"/"Recommend"	Loader shows → Result displayed	Loader and result appear	Pass	Includes API success path
Prediction/Recommendation – Error Handling	**Trigger failure (offline or API error)**	Submit with no internet or broken API	Snackbar shows error message	Error shown correctly	Pass	Handles API failure gracefully
Prediction/Recommendation – Result UI	**Validate output format**	Submit → View result section	Result shown in card/chart format	Output rendered correctly	Pass	UI feedback is user-friendly
Prediction/Recommendation – Loading State	**Verify loading spinner**	Submit form → Observe submit button/spinner	CircularProgressIndicator shown	Spinner displayed as expected	Pass	Confirms async UI feedback
Auth (Login/Register)	**Login/register with valid creds**	Submit correct form	Authenticated → redirected	Dashboard loads	Pass	Uses Firebase Auth
Password Reset	**Trigger forgot password email**	Tap "Forgot Password?" with email	Reset link sent	Snackbar shown	Pass	Firebase email function
Disabled User	**Attempt login with disabled account**	Use blocked user	Auth error: "User disabled"	Correct message shown	Pass	Admin control validation
Crop Calendar Display	**Validate seasonal calendar renders correct crops**	Navigate to Dashboard, Scroll to Seasonal Calendar section, Check listed crops/months	Crop suggestions match the current season/month	Output accurate as per season	Pass	Crop data appears timely and relevant
Logout Functionality	**Verify user logout and redirection**	Tap on menu → Logout, Confirm logout if prompted	User session ends and returns to login screen	Redirects to login, session cleared	Pass	Works consistently across app restarts
App Responsiveness	**Check screen responsiveness on resize/orient**	Rotate device / emulate on various screen sizes	UI adjusts properly	Layout remains consistent	Pass	Basic adaptive UI validation

2.1.6 Daily Call Progress



Sprint 1: Core Features – UI, Crop Recommendation, Helpline, Seasonal Calendar

Week 1 (17 Feb – 23 Feb)

Scrum Owner: Jiya

- **17 Feb (Mon):** Finalized UI/UX structure after discussion. Decided to go with a clean layout inspired by agri-portals. Setup basic Flutter scaffold.
- **18 Feb (Tue):** Created wireframes for crop recommendation and calendar modules. Jiya tested color contrast for readability.
- **19 Feb (Wed):** Worked on helpline contact page. Faced issue with loading assets in Flutter which was resolved by updating pubspec.yaml.
- **20 Feb (Thu):** Implemented navigation drawer and routing between screens. Tested screen responsiveness on multiple devices.
- **21 Feb (Fri):** Added crop recommendation screen skeleton.
- **22 Feb (Sat):** Backend integration planning session. Akshat helped define basic model API structure.
- **23 Feb (Sun):** Calendar page layout implemented. Discussed design changes for month-wise crop display.

Week 2 (24 Feb – 1 Mar)

Scrum Owner: Akshat

- **24 Feb (Mon):** Integrated helpline data and ensured click-to-call support worked correctly.
- **25 Feb (Tue):** Connected frontend UI to static seasonal crop calendar data. Refined design with icons.
- **26 Feb (Wed):** Akshat debugged bug in crop recommendation. Adjusted Flutter form validations.
- **27 Feb (Thu):** Implemented recommendation output section. Mocked ML output for testing.
- **28 Feb (Fri):** Added explanation logic for recommended crops – mapped based on location and climate.
- **29 Feb (Sat):** UI refinement – improved font scaling and removed redundant buttons.
- **1 Mar (Sun):** Final review of Sprint 1 modules. Everything functional and responsive. Passed all UI test cases.

Figure 2.8 Standup meetings

2.1.7 Committed Vs Completed User Stories

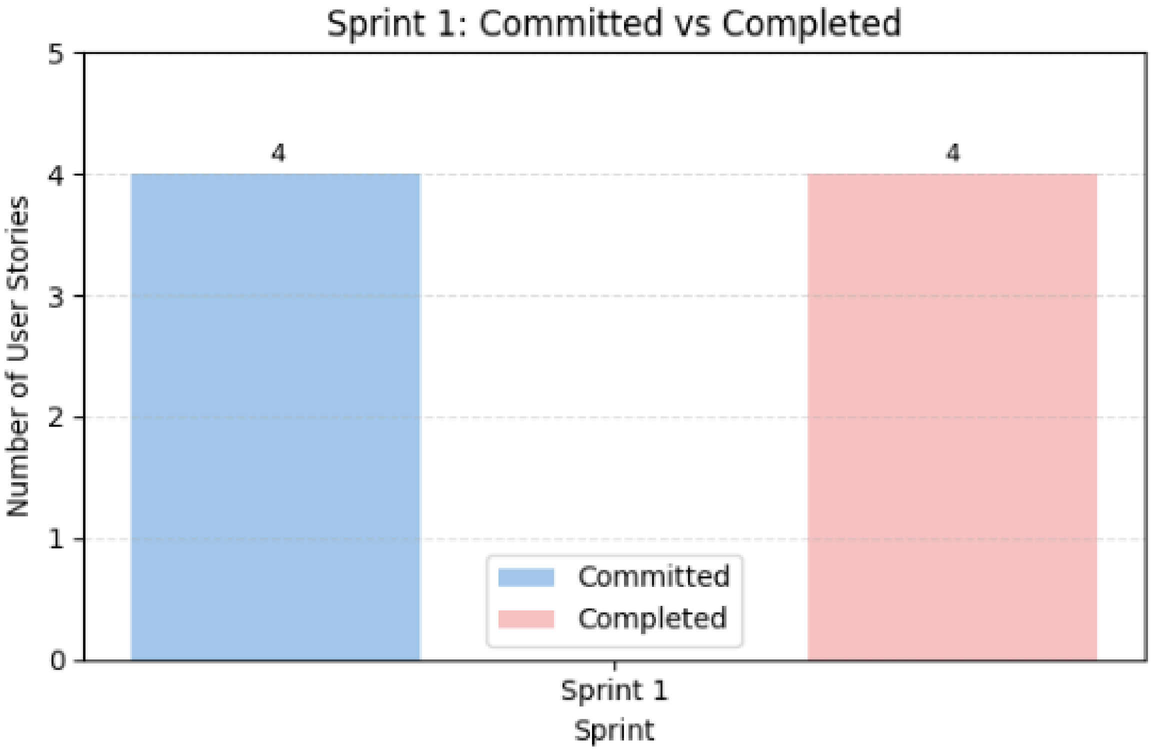
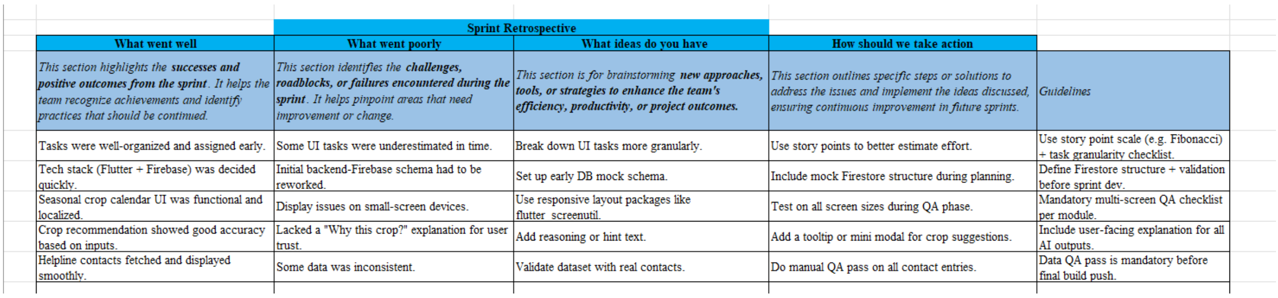


Figure 2.9 Bar graph for Committed Vs Completed User Stories

Figure 2.9 illustrates that during Sprint 1 of the "Unified Digital Platform for Smarter Farming" project, all 4 committed user stories were successfully completed, indicating effective sprint planning and execution and the retrospective is mentioned in **Figure 2.10**

2.1.8 Sprint Retrospective



	Sprint Retrospective			
What went well	**What went poorly**	**What ideas do you have**	**How should we take action**	
This section highlights the ***successes and positive outcomes from the sprint****. It helps the team recognize achievements and identify practices that should be continued.*	*This section identifies the* ***challenges, roadblocks, or failures encountered during the sprint****. It helps pinpoint areas that need improvement or change.*	*This section is for brainstorming* ***new approaches, tools, or strategies to enhance the team's efficiency, productivity, or project outcomes.***	*This section outlines specific steps or solutions to address the issues and implement the ideas discussed, ensuring continuous improvement in future sprints.*	*Guidelines*
Tasks were well-organized and assigned early.	Some UI tasks were underestimated in time.	Break down UI tasks more granularly.	Use story points to better estimate effort.	Use story point scale (e.g. Fibonacci) + task granularity checklist.
Tech stack (Flutter + Firebase) was decided quickly.	Initial backend-Firebase schema had to be reworked.	Set up early DB mock schema.	Include mock Firestore structure during planning.	Define Firestore structure + validation before sprint dev.
Seasonal crop calendar UI was functional and localized.	Display issues on small-screen devices.	Use responsive layout packages like flutter_screenutil.	Test on all screen sizes during QA phase.	Mandatory multi-screen QA checklist per module.
Crop recommendation showed good accuracy based on inputs.	Lacked a "Why this crop?" explanation for user trust.	Add reasoning or hint text.	Add a tooltip or mini modal for crop suggestions.	Include user-facing explanation for all AI outputs.
Helpline contacts fetched and displayed smoothly.	Some data was inconsistent.	Validate dataset with real contacts.	Do manual QA pass on all contact entries.	Data QA pass is mandatory before final build push.

Figure 2.10 Sprint Retrospective for the Sprint 1

2.2 SPRINT 2

2.2.1 Sprint Goal with User Stories of Sprint 2

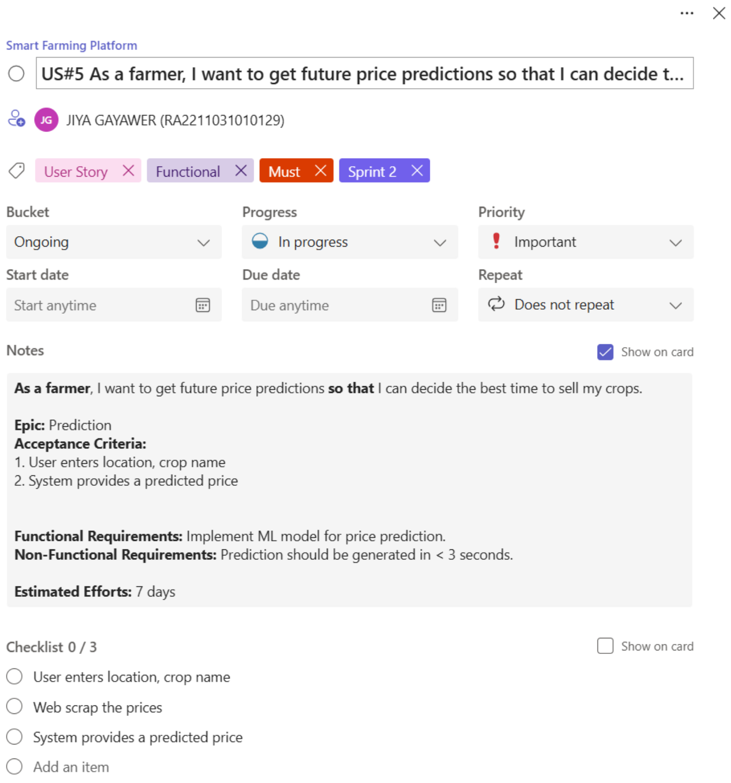
The goal of the second sprint is to implement real-time data functionalities, including weather insights, crop price prediction, feedback mechanisms, and farm activity tracking with reminders.

The following **Table 2.4** represents the detailed user stories of the sprint 2

Table 2.4 Detailed User Stories of sprint 2

S.NO	Detailed User Stories
US #1	As a farmer, I want to get future price predictions so that I can decide the best time to sell my crops.
US #2	As a farmer, I want real-time weather updates for my region so that I can plan agricultural activities accordingly.
US #3	As a user, I want to provide feedback so that the developers can improve the app.
US #4	As a farmer, I want to track my farm activities and receive reminders so that I don’t miss tasks.

Planner Board representations of user stories are mentioned below **figures 2.11, 2.12, 2.13** and **2.14**.



Smart Farming Platform

○ US#5 As a farmer, I want to get future price predictions so that I can decide t...

JG JIYA GAYAWER (RA2211031010129)

User Story ✕ Functional ✕ Must ✕ Sprint 2 ✕

Bucket	Progress	Priority
Ongoing	In progress	Important
Start date	**Due date**	**Repeat**
Start anytime	Due anytime	Does not repeat

Notes ☑ Show on card

As a farmer, I want to get future price predictions **so that** I can decide the best time to sell my crops.

Epic: Prediction
Acceptance Criteria:
1. User enters location, crop name
2. System provides a predicted price

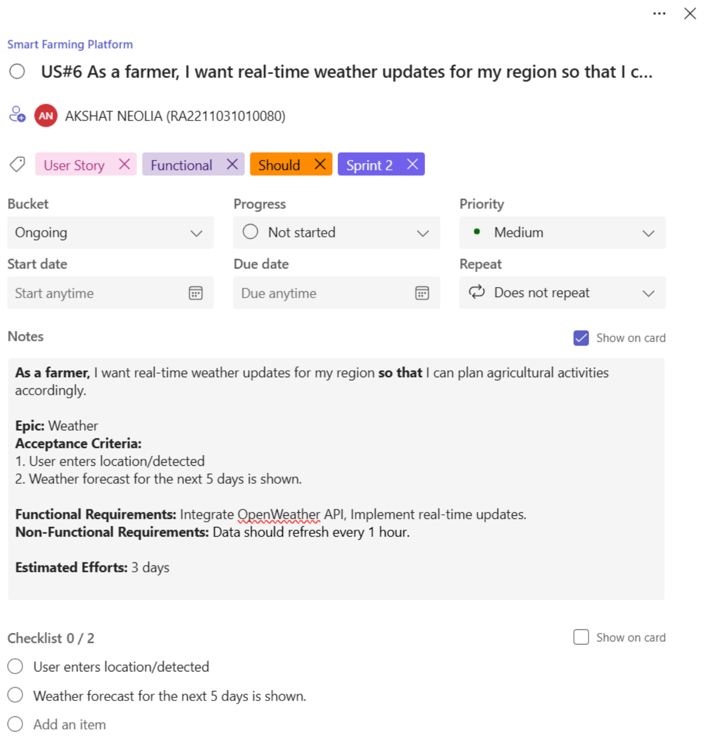
Functional Requirements: Implement ML model for price prediction.
Non-Functional Requirements: Prediction should be generated in < 3 seconds.

Estimated Efforts: 7 days

Checklist 0 / 3 □ Show on card

○ User enters location, crop name
○ Web scrap the prices
○ System provides a predicted price
○ Add an item

Figure 2.11 User Story#1



Smart Farming Platform

○ US#6 As a farmer, I want real-time weather updates for my region so that I c...

AN AKSHAT NEOLIA (RA2211031010080)

User Story ✕ Functional ✕ Should ✕ Sprint 2 ✕

Bucket	Progress	Priority
Ongoing	○ Not started	● Medium
Start date	**Due date**	**Repeat**
Start anytime	Due anytime	Does not repeat

Notes ☑ Show on card

As a farmer, I want real-time weather updates for my region **so that** I can plan agricultural activities accordingly.

Epic: Weather
Acceptance Criteria:
1. User enters location/detected
2. Weather forecast for the next 5 days is shown.

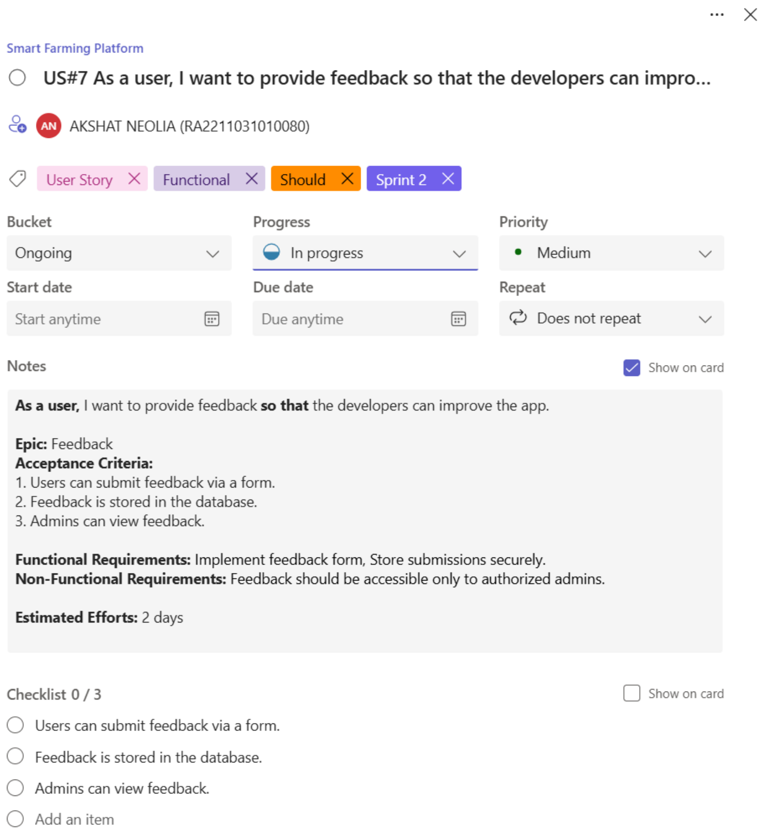
Functional Requirements: Integrate OpenWeather API, Implement real-time updates.
Non-Functional Requirements: Data should refresh every 1 hour.

Estimated Efforts: 3 days

Checklist 0 / 2 □ Show on card

○ User enters location/detected
○ Weather forecast for the next 5 days is shown.
○ Add an item

Figure 2.12 User Story#2


Smart Farming Platform

○ US#7 As a user, I want to provide feedback so that the developers can impro...

AN AKSHAT NEOLIA (RA2211031010080)

Bucket: Ongoing

Progress: In progress

Priority: Medium

Start date: Start anytime

Due date: Due anytime

Repeat: Does not repeat

Notes ☑ Show on card

As a user, I want to provide feedback **so that** the developers can improve the app.

Epic: Feedback
Acceptance Criteria:
1. Users can submit feedback via a form.
2. Feedback is stored in the database.
3. Admins can view feedback.

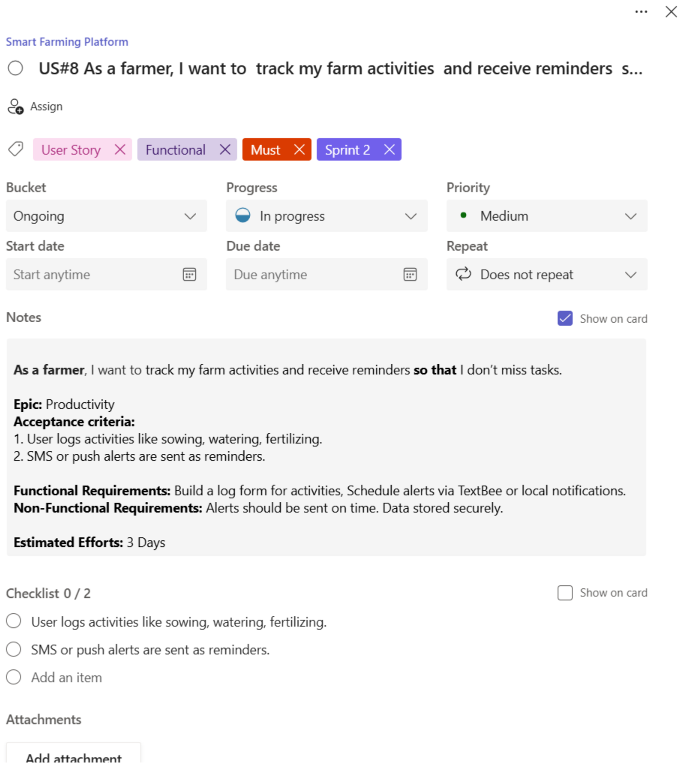
Functional Requirements: Implement feedback form, Store submissions securely.
Non-Functional Requirements: Feedback should be accessible only to authorized admins.

Estimated Efforts: 2 days

Checklist 0 / 3 □ Show on card

○ Users can submit feedback via a form.
○ Feedback is stored in the database.
○ Admins can view feedback.
○ Add an item

Figure 2.13 User Story#3


Smart Farming Platform

○ US#8 As a farmer, I want to track my farm activities and receive reminders s...

Bucket: Ongoing

Progress: In progress

Priority: Medium

Start date: Start anytime

Due date: Due anytime

Repeat: Does not repeat

Notes ☑ Show on card

As a farmer, I want to track my farm activities and receive reminders **so that** I don't miss tasks.

Epic: Productivity
Acceptance criteria:
1. User logs activities like sowing, watering, fertilizing.
2. SMS or push alerts are sent as reminders.

Functional Requirements: Build a log form for activities, Schedule alerts via TextBee or local notifications.
Non-Functional Requirements: Alerts should be sent on time. Data stored securely.

Estimated Efforts: 3 Days

Checklist 0 / 2 □ Show on card

○ User logs activities like sowing, watering, fertilizing.
○ SMS or push alerts are sent as reminders.
○ Add an item

Attachments

Figure 2.14 User Story#4

2.2.2 Functional Document (Sprint -2)

2.2.2.1. Introduction

This sprint introduces real-time weather alerts, feedback collection, AI-powered crop price predictions, and an activity tracker to log farming actions. These modules improve usability and ensure that decisions are timely, and data driven.

2.2.2.2. Product Goal

The goal of this project is to develop a user-friendly digital platform that:

- Deliver real-time weather alerts.
- Predict crop prices for regional markets.
- Capture user feedback for improvements.
- Track user activities like watering and fertilizing.

2.2.2.3. Demography (Users, Location)

Users:

- Farmers, consultants, analysts

Location:

- Price prediction: Maharashtra (Vashi APMC, Nagpur)
- Weather alerts: All states

2.2.2.4. Business Processes

- Weather API Integration: Connect real-time weather feeds to the app using third-party APIs (e.g., OpenWeatherMap) for local forecasting.
- Dynamic Recommendation Updates: Adjust crop suggestions and calendar steps based on rainfall, temperature, and climate anomalies.
- Tooltip & Guided Help Implementation: Introduce pop-up tooltips to assist users during their journey in the app.
- Content Verification and Pilot Testing: Validate translations and weather-based suggestions through field testing.

2.2.2.5 Features –

Feature 1 - Activity Tracker with Alerts

Description:

Logs farming activities and sends reminders via app and SMS.

User Story:

As a farmer, I want to track my farm activities and receive reminders so that I don't miss tasks.

Feature 2 - Weather Insights

Description:

Provides weather forecasts and auto-location detection.

User Story:

As a farmer, I want real-time weather updates for my region so that I can plan agricultural activities accordingly.

Feature 3 - Feedback Mechanism

Description:

Users can submit feedback, report bugs, or share ideas.

User Story:

As a user, I want to provide feedback so that the developers can improve the app.

Feature 4 - Crop Price Prediction

Description:

Uses AI to forecast future crop prices in regional markets.

User Story:

As a farmer, I want to get future price predictions so that I can decide the best time to sell my crops.

2.2.2.6. Authorization Matrix

Table 2.5 Access level Authorization Matrix

Role	Access Level
Farmer	Receive weather alerts, enter feedback
Admin	Manage feedback and alerts

Table 2.5 shows that Farmers have access to the alerts and giving feedback, while Admins manage backend data.

2.2.2.7. Assumptions

- Weather APIs provide timely and accurate regional data.
- Tooltip-based guidance reduces dependency on training programs.
- Dynamic updates won’t affect offline availability of critical features.

2.2.3 Architecture Document

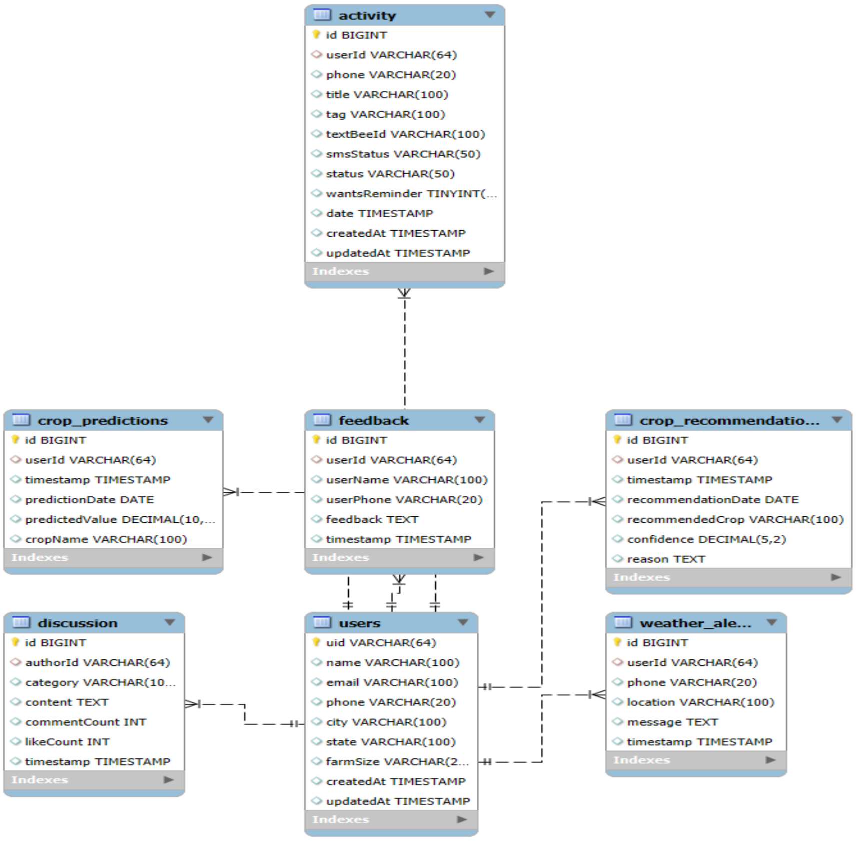


Figure 2.15 ER Diagram

Figure 2.15 presents the Entity-Relationship (ER) diagram for the "Unified Digital Platform for Smarter Farming," showcasing a structured database with the users table at the core, linked to key modules. This design supports personalized services, user engagement, and efficient data management.

Figure 2.16 illustrates the system flowchart, summarizing how user inputs, APIs, ML models, and cloud services interact to deliver real-time recommendations and alerts for smart farming decisions.

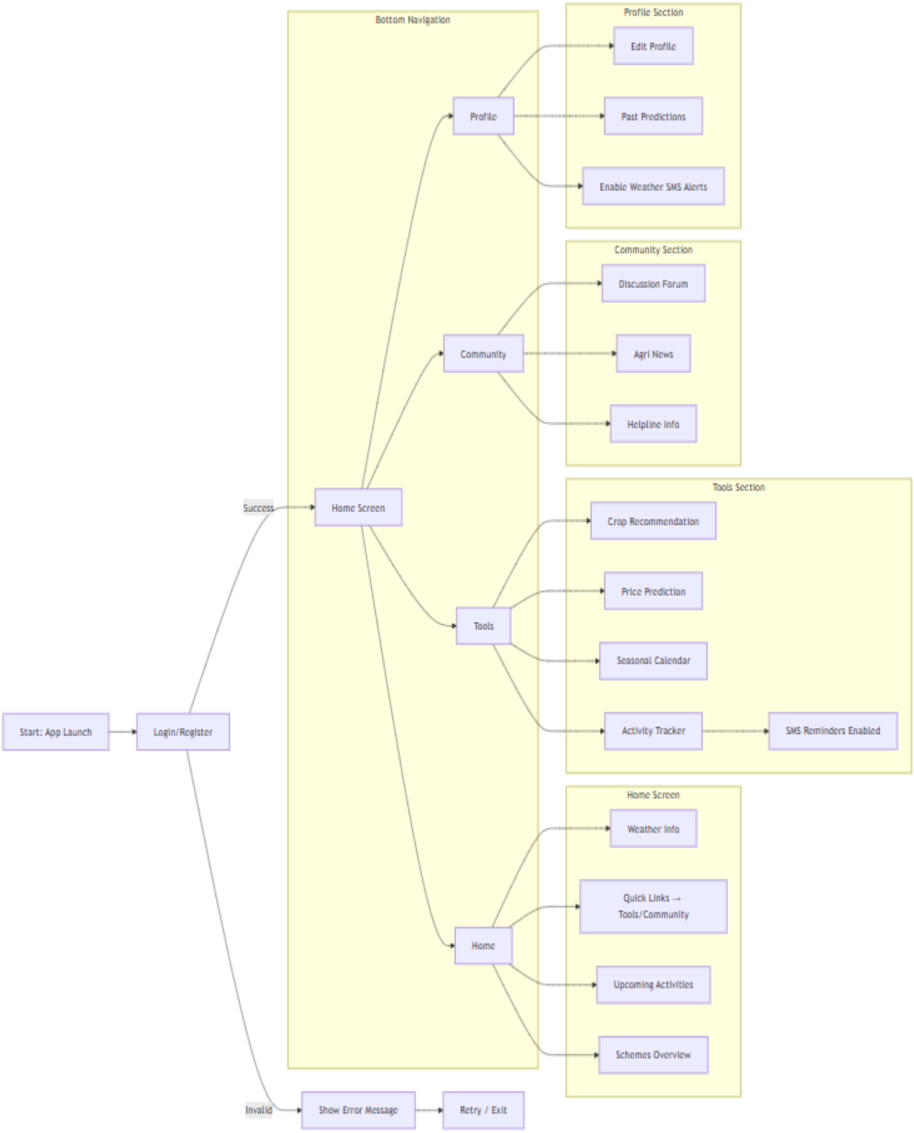


Figure 2.16 Flowchart Diagram

2.2.3.1 DATA EXCHANGE CONTRACT

Frequency of Data Exchanges

Real-Time Exchanges

- Weather Updates: Retrieved every 15 minutes via the OpenWeather API and processed through the backend.
- SMS Alerts: Triggered instantly in response to weather changes or market price fluctuations using the TextBee API.
- Crop Recommendations: Generated instantly based on user-provided inputs, such as location and farm conditions.

Scheduled Exchanges (Daily/Weekly/Biweekly)

- Crop Price Predictions: Machine learning models are executed on a daily or weekly basis. The output is stored in Firestore for use in the app.
- Seasonal Crop Data Sync: Synchronized every 14 days from the UPAg Portal using a scheduled web scraping process.

On-Demand Exchanges

- Weather Information: Fetched when a user opens the application or requests weather insights.
- Price Prediction & Crop Recommendation: Invoked on user action, handled by dedicated ML microservices.

2.2.3.2 Data Sets and Their Usage

Table 2.6 Dataset and their usage

Dataset	**Source**	**Purpose**
Crop Recommendation Data	Kaggle dataset / Internal CSV	Provides crop suggestions based on location, soil, and weather factors
Seasonal Crop Calendar	UPAg Portal (via web scraping/API)	Guides users on optimal planting and harvesting timelines

Crop Price Data	Government portals / Market APIs / Scraping	Supplies historical and current prices for prediction models
Weather Data	OpenWeather API	Provides real-time weather insights and alerts
SMS Alert Preferences	Firebase	Manages user opt-ins for personalized alert delivery
Market Trends Data	Commodity market APIs / Web scraping	Used for trend analysis and predictive modeling
User Data	Firebase Auth & Firestore	Stores user profiles, preferences, and activity logs
Feedback & Interaction Data	In-app forms and interaction logs	Improves personalization and user experience

Table 2.6 describes datasets and their sources, outlining their respective purposes for crop recommendations, weather insights, price predictions, user preferences, and more.

2.2.3.3 Mode of Data Exchange

API-Based (Real-Time)

- OpenWeather API → Weather data integration
- TextBee API → Delivery of SMS alerts
- Firebase Auth API → User authentication and data sync
- UPAg API (if applicable) → Crop calendar data

Web Scraping (Scheduled)

- UPAg Portal → Seasonal crop information (if API unavailable)
- Government/market websites → Crop prices and trends

Database Storage (On-Demand Access)

- Firebase / Firestore → Stores and retrieves user data, prediction outputs, and app activities

- PostgreSQL (where applicable) → Alternative structured storage for analytics and SMS logs

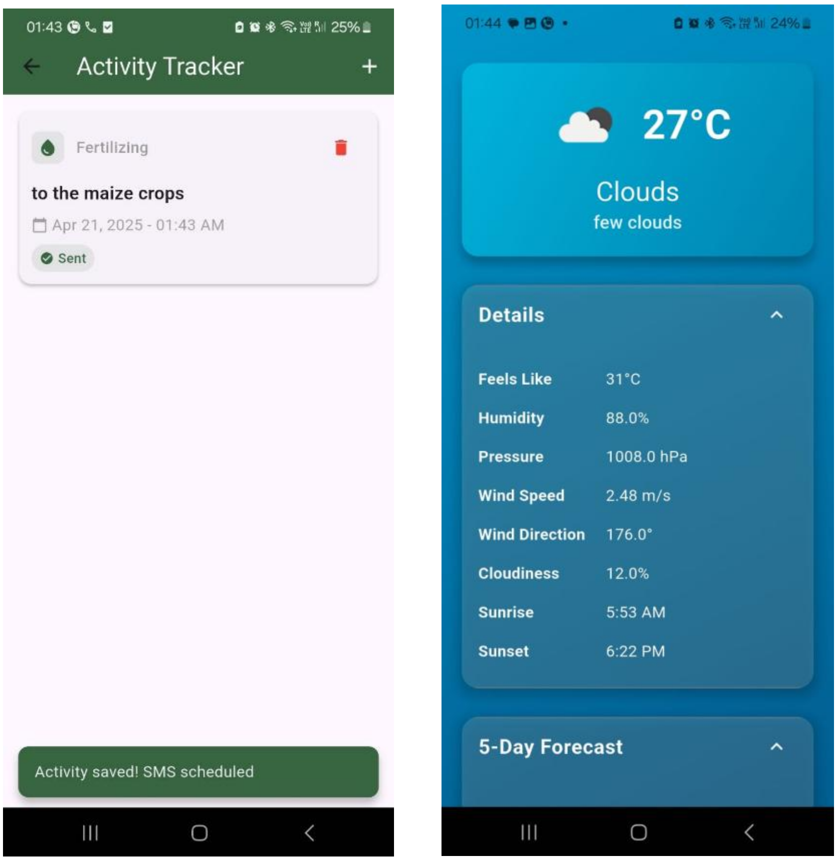
Message Queue (Asynchronous Processing)

- SMS Queue → Queues alerts based on real-time data triggers
- Prediction Processing Queue → Manages asynchronous ML predictions and updates

File-Based Exchange (Batch Processing)

- CSV/JSON Exports → For reports on price trends and recommendations
- External Imports → Integration of data from official publications or field reports

2.2.4 UI Design



Fertilizing

to the maize crops

Apr 21, 2025 - 01:43 AM

Sent

Activity saved! SMS scheduled

27°C

Clouds

few clouds

Details

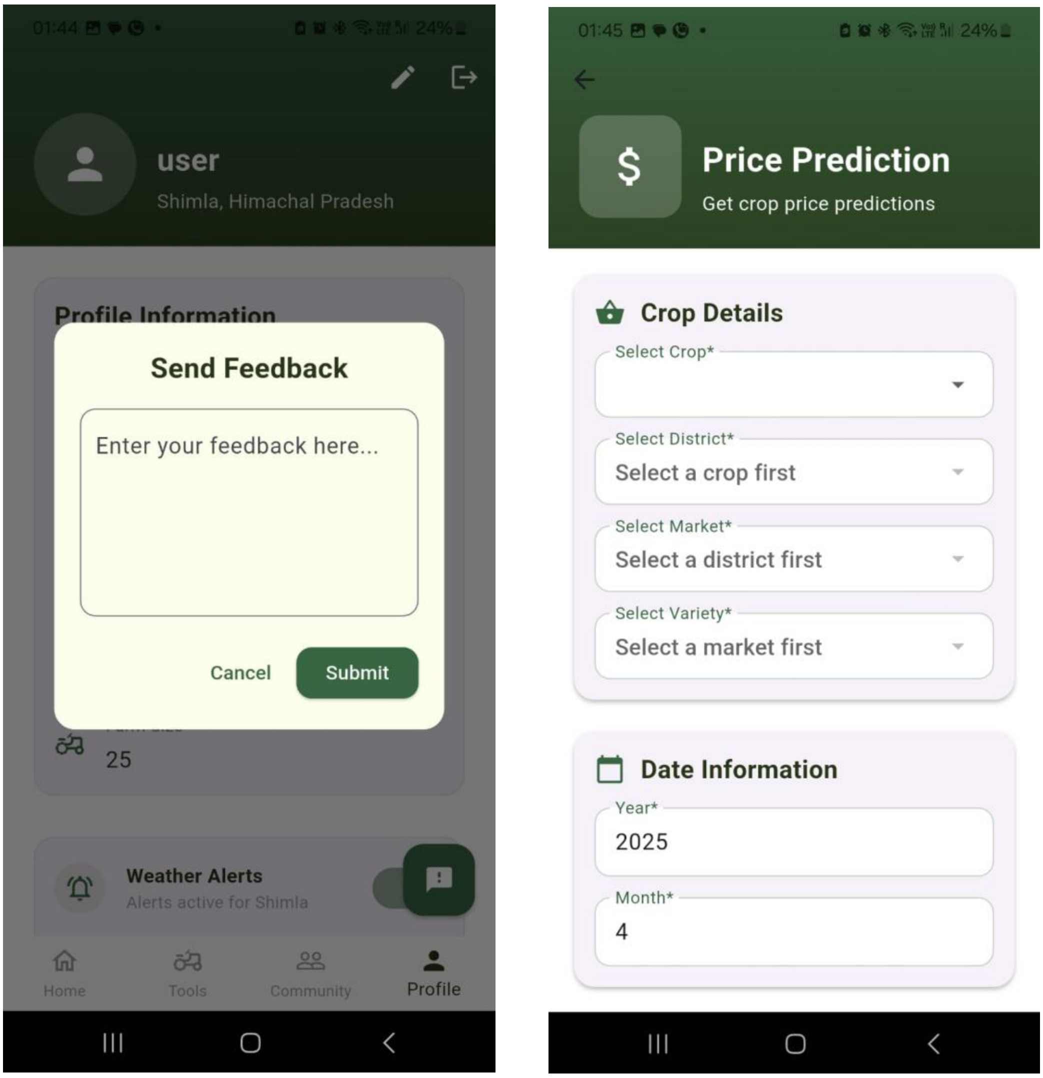
Feels Like	31°C
Humidity	88.0%
Pressure	1008.0 hPa
Wind Speed	2.48 m/s
Wind Direction	176.0°
Cloudiness	12.0%
Sunrise	5:53 AM
Sunset	6:22 PM

5-Day Forecast

Figure 2.17 Activity tracker and Weather insights

Figure 2.17 displays the **Activity Tracker** and **Weather Insights** features of the Unified Digital Platform for Smarter Farming. The Activity Tracker helps farmers log and monitor field activities like fertilizing, along with SMS scheduling for timely reminders. The Weather

Insights screen provides detailed real-time data such as temperature, humidity, wind, and forecasts. These tools support proactive planning and climate-aware farming decisions.



01:44 · 24%

user

Shimla, Himachal Pradesh

Profile Information

Send Feedback

Enter your feedback here...

Cancel Submit

25

Weather Alerts

Alerts active for Shimla

Home Tools Community Profile

01:45 · 24%

Price Prediction

Get crop price predictions

Crop Details

Select Crop*

Select District*: Select a crop first

Select Market*: Select a district first

Select Variety*: Select a market first

Date Information

Year*: 2025

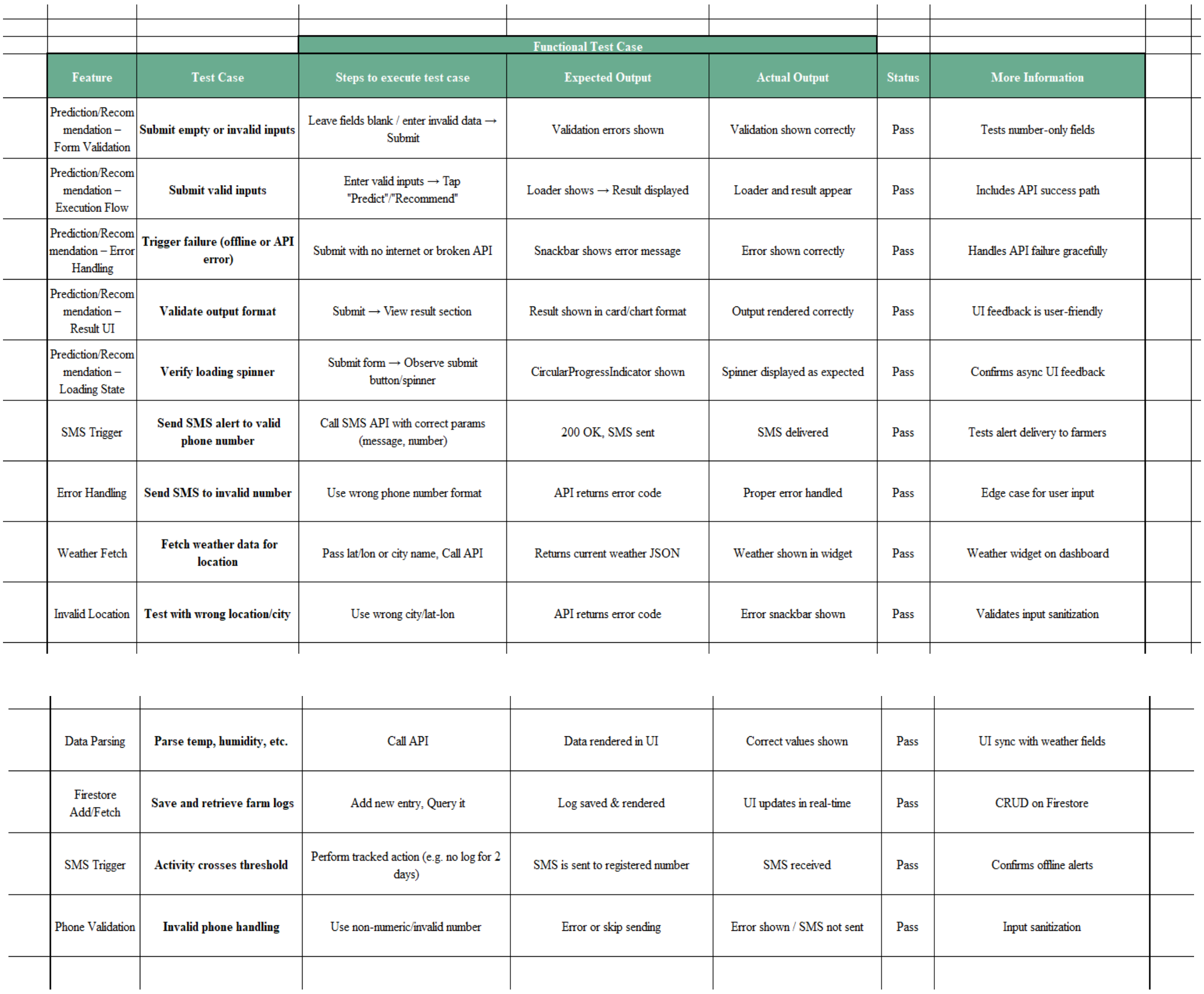
Month*: 4

Figure 2.18 Feedback Form and Crop Price Prediction UI

Figure 2.18 showcases two interfaces of the Unified Digital Platform for Smarter Farming. The first screen displays a **Feedback Form**, allowing farmers to easily submit suggestions or concerns, fostering continuous improvement and user engagement. The second screen presents the **Crop Price Prediction UI**, where users can input crop, district, market, and date details to receive AI-based price forecasts. These features aim to empower farmers with personalized insights and promote informed decision-making in agriculture.

2.2.5 Functional Test Cases

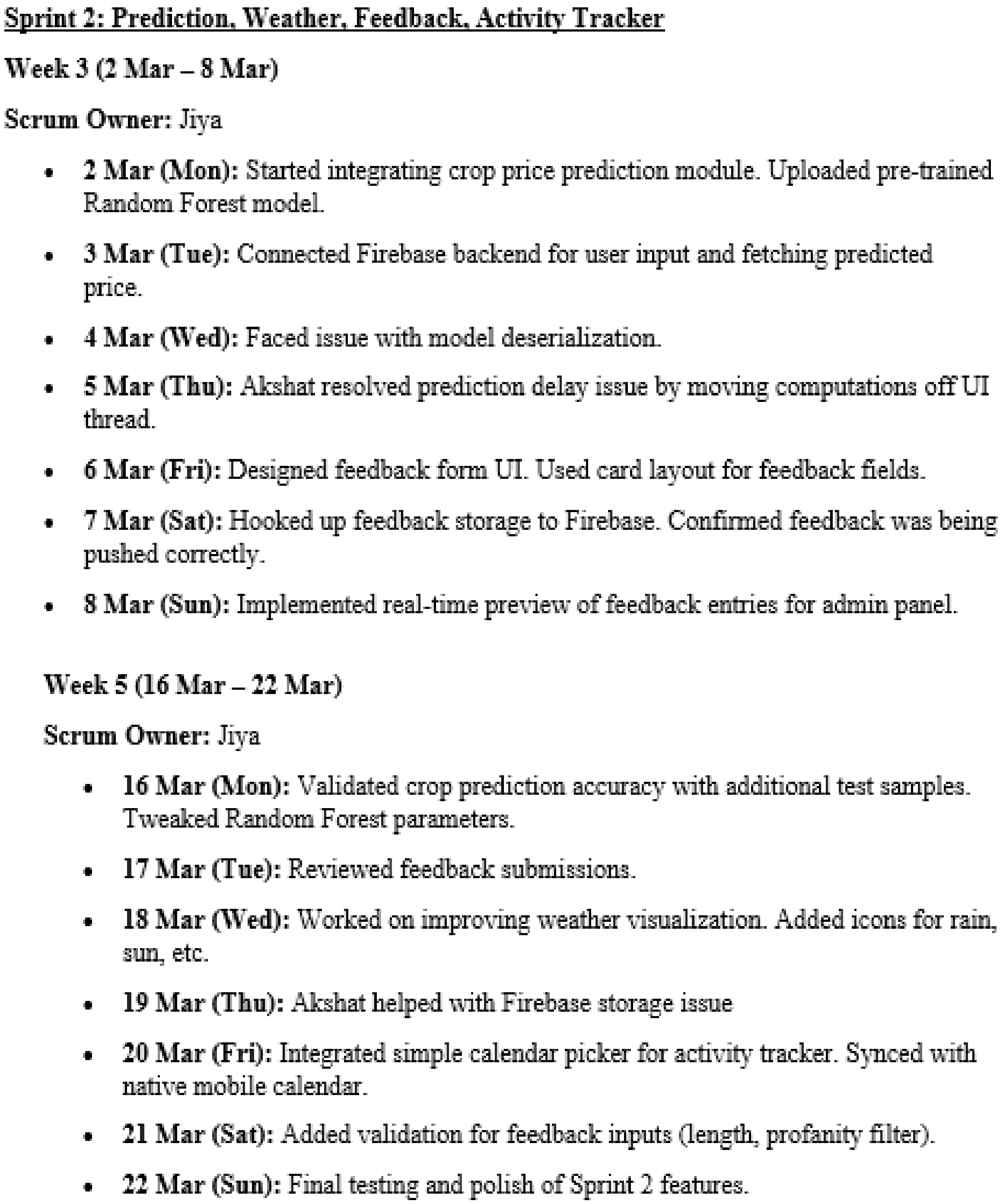
Table 2.7 Functional Test Cases



		Functional Test Case				
Feature	**Test Case**	**Steps to execute test case**	**Expected Output**	**Actual Output**	**Status**	**More Information**
Prediction/Recommendation – Form Validation	**Submit empty or invalid inputs**	Leave fields blank / enter invalid data → Submit	Validation errors shown	Validation shown correctly	Pass	Tests number-only fields
Prediction/Recommendation – Execution Flow	**Submit valid inputs**	Enter valid inputs → Tap "Predict"/"Recommend"	Loader shows → Result displayed	Loader and result appear	Pass	Includes API success path
Prediction/Recommendation – Error Handling	**Trigger failure (offline or API error)**	Submit with no internet or broken API	Snackbar shows error message	Error shown correctly	Pass	Handles API failure gracefully
Prediction/Recommendation – Result UI	**Validate output format**	Submit → View result section	Result shown in card/chart format	Output rendered correctly	Pass	UI feedback is user-friendly
Prediction/Recommendation – Loading State	**Verify loading spinner**	Submit form → Observe submit button/spinner	CircularProgressIndicator shown	Spinner displayed as expected	Pass	Confirms async UI feedback
SMS Trigger	**Send SMS alert to valid phone number**	Call SMS API with correct params (message, number)	200 OK, SMS sent	SMS delivered	Pass	Tests alert delivery to farmers
Error Handling	**Send SMS to invalid number**	Use wrong phone number format	API returns error code	Proper error handled	Pass	Edge case for user input
Weather Fetch	**Fetch weather data for location**	Pass lat/lon or city name, Call API	Returns current weather JSON	Weather shown in widget	Pass	Weather widget on dashboard
Invalid Location	**Test with wrong location/city**	Use wrong city/lat-lon	API returns error code	Error snackbar shown	Pass	Validates input sanitization
Data Parsing	**Parse temp, humidity, etc.**	Call API	Data rendered in UI	Correct values shown	Pass	UI sync with weather fields
Firestore Add/Fetch	**Save and retrieve farm logs**	Add new entry, Query it	Log saved & rendered	UI updates in real-time	Pass	CRUD on Firestore
SMS Trigger	**Activity crosses threshold**	Perform tracked action (e.g. no log for 2 days)	SMS is sent to registered number	SMS received	Pass	Confirms offline alerts
Phone Validation	**Invalid phone handling**	Use non-numeric/invalid number	Error or skip sending	Error shown / SMS not sent	Pass	Input sanitization

Table 2.7 outlines the functional test cases used to validate the key features and functionalities of the system.

2.2.6 Daily Call Progress



Sprint 2: Prediction, Weather, Feedback, Activity Tracker

Week 3 (2 Mar – 8 Mar)

Scrum Owner: Jiya

- **2 Mar (Mon):** Started integrating crop price prediction module. Uploaded pre-trained Random Forest model.
- **3 Mar (Tue):** Connected Firebase backend for user input and fetching predicted price.
- **4 Mar (Wed):** Faced issue with model deserialization.
- **5 Mar (Thu):** Akshat resolved prediction delay issue by moving computations off UI thread.
- **6 Mar (Fri):** Designed feedback form UI. Used card layout for feedback fields.
- **7 Mar (Sat):** Hooked up feedback storage to Firebase. Confirmed feedback was being pushed correctly.
- **8 Mar (Sun):** Implemented real-time preview of feedback entries for admin panel.

Week 5 (16 Mar – 22 Mar)

Scrum Owner: Jiya

- **16 Mar (Mon):** Validated crop prediction accuracy with additional test samples. Tweaked Random Forest parameters.
- **17 Mar (Tue):** Reviewed feedback submissions.
- **18 Mar (Wed):** Worked on improving weather visualization. Added icons for rain, sun, etc.
- **19 Mar (Thu):** Akshat helped with Firebase storage issue
- **20 Mar (Fri):** Integrated simple calendar picker for activity tracker. Synced with native mobile calendar.
- **21 Mar (Sat):** Added validation for feedback inputs (length, profanity filter).
- **22 Mar (Sun):** Final testing and polish of Sprint 2 features.

Figure 2.19 Standup Meetings

2.2.7 COMMITTED Vs COMPLETED USER STORIES

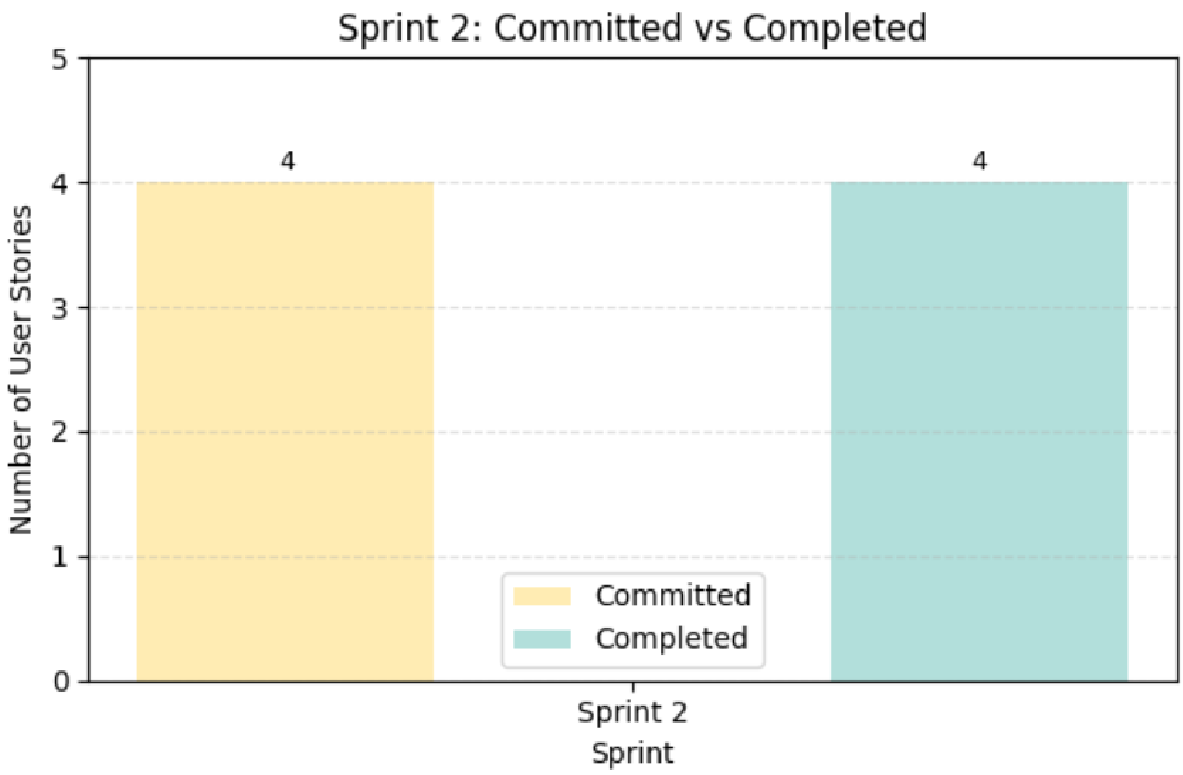
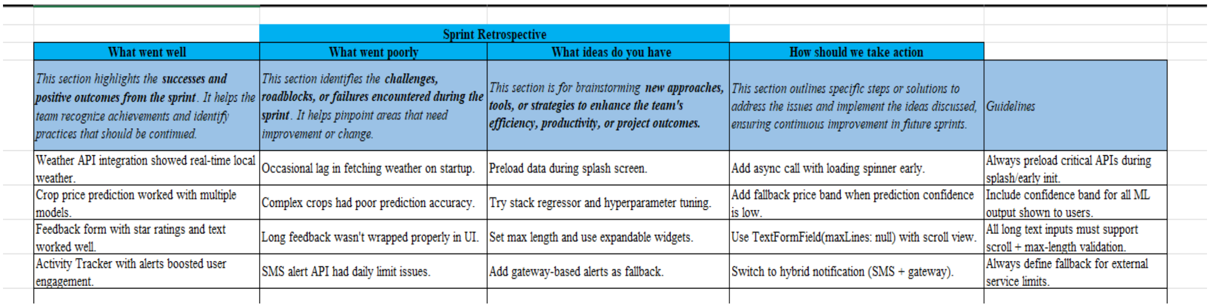


Figure 2.20 Bar graph for Committed Vs Completed User Stories

Figure 2.20 shows that in Sprint 2 of the "Unified Digital Platform for Smarter Farming" project, all 4 committed user stories were successfully completed, reflecting consistent delivery and team performance across sprints. Sprint retrospective for this sprint is mentioned in **Figure 2.21**.

2.2.8 Sprint Retrospective



	Sprint Retrospective			
What went well	**What went poorly**	**What ideas do you have**	**How should we take action**	
This section highlights the ***successes and positive outcomes from the sprint****. It helps the team recognize achievements and identify practices that should be continued.*	*This section identifies the* ***challenges, roadblocks, or failures encountered during the sprint****. It helps pinpoint areas that need improvement or change.*	*This section is for brainstorming* ***new approaches, tools, or strategies to enhance the team's efficiency, productivity, or project outcomes.***	*This section outlines specific steps or solutions to address the issues and implement the ideas discussed, ensuring continuous improvement in future sprints.*	*Guidelines*
Weather API integration showed real-time local weather.	Occasional lag in fetching weather on startup.	Preload data during splash screen.	Add async call with loading spinner early.	Always preload critical APIs during splash/early init.
Crop price prediction worked with multiple models.	Complex crops had poor prediction accuracy.	Try stack regressor and hyperparameter tuning.	Add fallback price band when prediction confidence is low.	Include confidence band for all ML output shown to users.
Feedback form with star ratings and text worked well.	Long feedback wasn't wrapped properly in UI.	Set max length and use expandable widgets.	Use TextFormField(maxLines: null) with scroll view.	All long text inputs must support scroll + max-length validation.
Activity Tracker with alerts boosted user engagement.	SMS alert API had daily limit issues.	Add gateway-based alerts as fallback.	Switch to hybrid notification (SMS + gateway).	Always define fallback for external service limits.

Figure 2.21 Sprint Retrospective

2.3 SPRINT 3

2.3.1 Sprint Goal with User Stories of Sprint 3

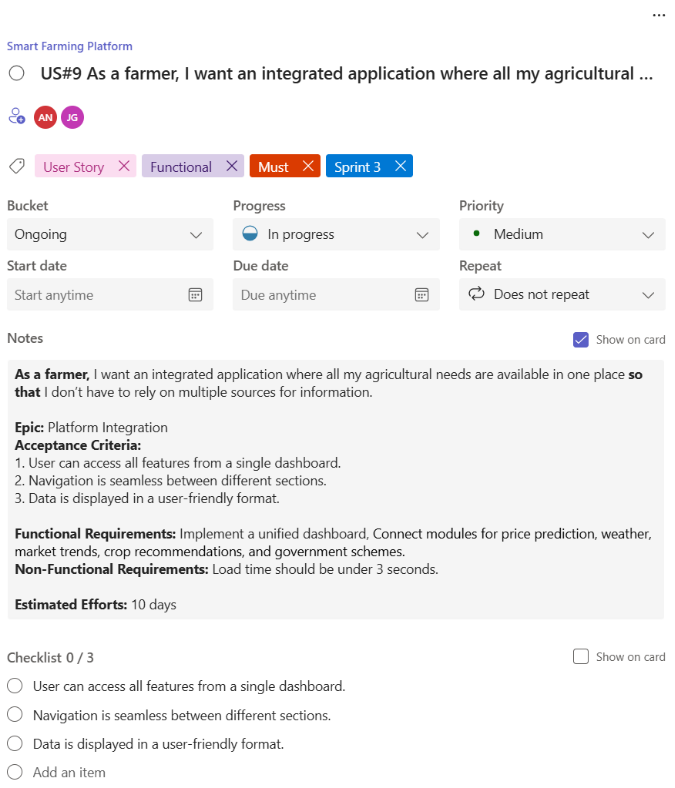
The goal of the third sprint is to enhance the platform’s stability, user experience, and community interaction by integrating multiple modules into a unified interface. This sprint focuses on implementing weather-based alerts, fixing existing bugs, adding a news section, enabling discussion forums, and achieving a seamless, integrated platform experience.

The following **table 2.8** represents the detailed user stories of the sprint 3

Table 2.8 Detailed User Stories of sprint 3

S.NO	Detailed User Stories
US #1	As a farmer, I want an integrated app where all my agricultural needs are available in one place so that I don’t have to rely on multiple sources for information.
US #2	As a farmer, I want alerts for weather changes so that I can take precautions to protect my crops.
US #3	As a user, I want a bug-free experience so that I can use the app without issues.
US #4	As a farmer, I want to discuss problems and share tips with others so that I can learn and grow.
US #5	As a farmer, I want to read current agriculture-related news so that I can stay informed.

Planner Board representations of user stories are mentioned below **figures 2.22, 2.23, 2.24, 2.25** and **2.26**.



Smart Farming Platform

○ US#9 As a farmer, I want an integrated application where all my agricultural ...

AN JG

User Story ✕ Functional ✕ Must ✕ Sprint 3 ✕

Bucket: Ongoing

Progress: In progress

Priority: Medium

Start date: Start anytime

Due date: Due anytime

Repeat: Does not repeat

Notes ☑ Show on card

As a farmer, I want an integrated application where all my agricultural needs are available in one place **so that** I don't have to rely on multiple sources for information.

Epic: Platform Integration
Acceptance Criteria:
1. User can access all features from a single dashboard.
2. Navigation is seamless between different sections.
3. Data is displayed in a user-friendly format.

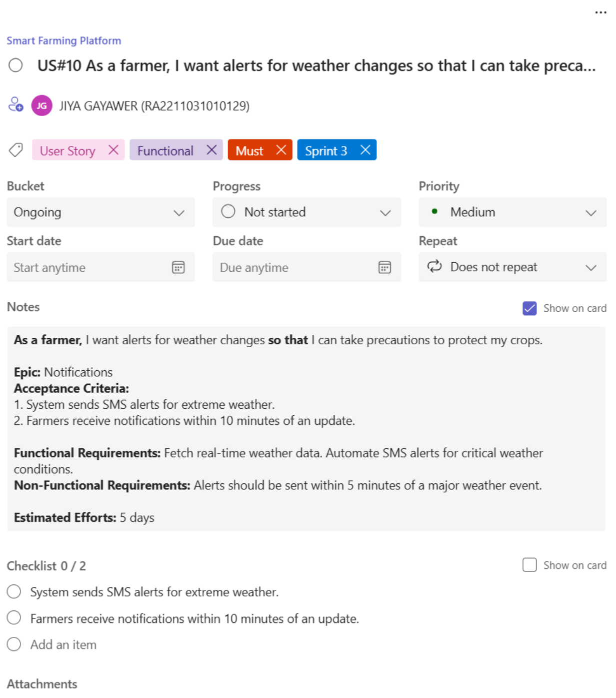
Functional Requirements: Implement a unified dashboard, Connect modules for price prediction, weather, market trends, crop recommendations, and government schemes.
Non-Functional Requirements: Load time should be under 3 seconds.

Estimated Efforts: 10 days

Checklist 0 / 3 □ Show on card

○ User can access all features from a single dashboard.
○ Navigation is seamless between different sections.
○ Data is displayed in a user-friendly format.
○ Add an item

Figure 2.22 User Story#1



Smart Farming Platform

○ US#10 As a farmer, I want alerts for weather changes so that I can take preca...

JG JIYA GAYAWER (RA2211031010129)

User Story ✕ Functional ✕ Must ✕ Sprint 3 ✕

Bucket: Ongoing

Progress: Not started

Priority: Medium

Start date: Start anytime

Due date: Due anytime

Repeat: Does not repeat

Notes ☑ Show on card

As a farmer, I want alerts for weather changes **so that** I can take precautions to protect my crops.

Epic: Notifications
Acceptance Criteria:
1. System sends SMS alerts for extreme weather.
2. Farmers receive notifications within 10 minutes of an update.

Functional Requirements: Fetch real-time weather data. Automate SMS alerts for critical weather conditions.
Non-Functional Requirements: Alerts should be sent within 5 minutes of a major weather event.

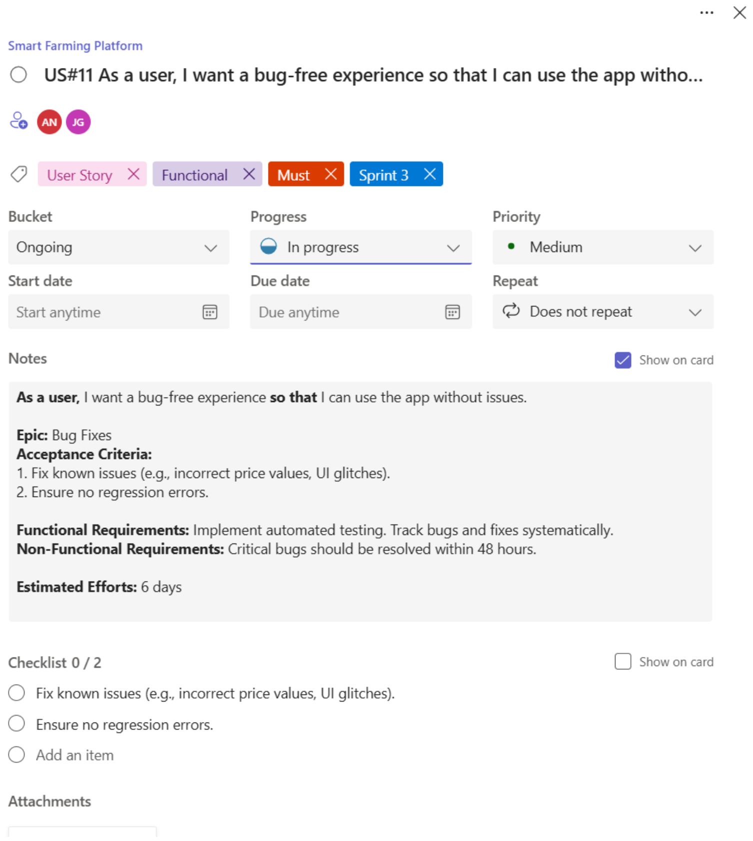
Estimated Efforts: 5 days

Checklist 0 / 2 □ Show on card

○ System sends SMS alerts for extreme weather.
○ Farmers receive notifications within 10 minutes of an update.
○ Add an item

Attachments

Figure 2.23 User Story#2



Smart Farming Platform

○ US#11 As a user, I want a bug-free experience so that I can use the app witho...

Bucket: Ongoing

Progress: In progress

Priority: ● Medium

Start date: Start anytime

Due date: Due anytime

Repeat: Does not repeat

Notes ☑ Show on card

As a user, I want a bug-free experience **so that** I can use the app without issues.

Epic: Bug Fixes
Acceptance Criteria:
1. Fix known issues (e.g., incorrect price values, UI glitches).
2. Ensure no regression errors.

Functional Requirements: Implement automated testing. Track bugs and fixes systematically.
Non-Functional Requirements: Critical bugs should be resolved within 48 hours.

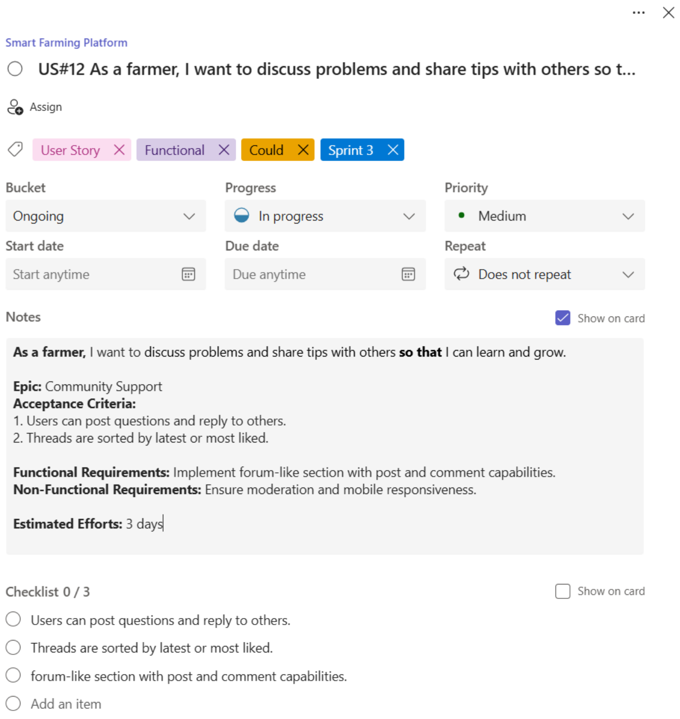
Estimated Efforts: 6 days

Checklist 0 / 2 □ Show on card

○ Fix known issues (e.g., incorrect price values, UI glitches).
○ Ensure no regression errors.
○ Add an item

Attachments

Figure 2.24 User Story#3



Smart Farming Platform

○ US#12 As a farmer, I want to discuss problems and share tips with others so t...

Bucket: Ongoing

Progress: In progress

Priority: ● Medium

Start date: Start anytime

Due date: Due anytime

Repeat: Does not repeat

Notes ☑ Show on card

As a farmer, I want to discuss problems and share tips with others **so that** I can learn and grow.

Epic: Community Support
Acceptance Criteria:
1. Users can post questions and reply to others.
2. Threads are sorted by latest or most liked.

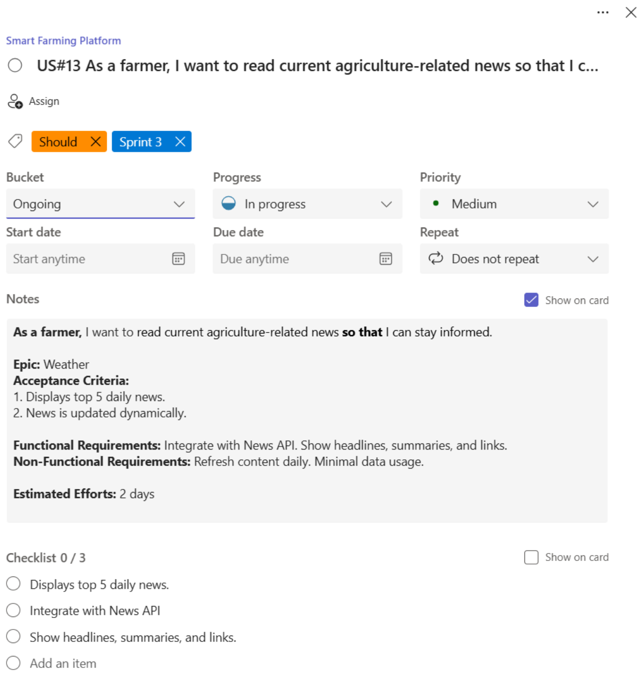
Functional Requirements: Implement forum-like section with post and comment capabilities.
Non-Functional Requirements: Ensure moderation and mobile responsiveness.

Estimated Efforts: 3 days

Checklist 0 / 3 □ Show on card

○ Users can post questions and reply to others.
○ Threads are sorted by latest or most liked.
○ forum-like section with post and comment capabilities.
○ Add an item

Figure 2.25 User Story#4



○ US#13 As a farmer, I want to read current agriculture-related news so that I c...

Bucket: Ongoing

Progress: In progress

Priority: Medium

Start date: Start anytime

Due date: Due anytime

Repeat: Does not repeat

Notes ☑ Show on card

As a farmer, I want to read current agriculture-related news **so that** I can stay informed.

Epic: Weather
Acceptance Criteria:
1. Displays top 5 daily news.
2. News is updated dynamically.

Functional Requirements: Integrate with News API. Show headlines, summaries, and links.
Non-Functional Requirements: Refresh content daily. Minimal data usage.

Estimated Efforts: 2 days

Checklist 0 / 3 □ Show on card

○ Displays top 5 daily news.
○ Integrate with News API
○ Show headlines, summaries, and links.
○ Add an item

Figure 2.26 User Story#5

2.3.2 Functional Document (Sprint -3)

2.3.2.1. Introduction

Sprint 3 delivers system stability through bug fixes, improves communication via farming alerts and news updates, integrates all features in a unified dashboard, and enables discussion forums for community support.

2.3.2.2. Product Goal

The goal of this project is to develop a user-friendly digital platform that:

- Enhance system usability by fixing bugs.
- Provide farming alerts during weather events.
- Show verified agricultural news and schemes.
- Enable farmer-to-farmer discussion.

2.3.2.3. Demography (Users, Location)

Users:

- Farmers, experts, govt. bodies

Location:

- All states: news and alerts adapted per region.

2.3.2.4. Business Processes

- **Advisory Module Development:** Create rule-based advisory engine to notify farmers on crop care, irrigation, and protection techniques.
- **Pest/Disease Alert Integration:** Include a static database for common crop threats and regional pests with alert mechanisms.
- **Farmer Feedback Mechanism:** Collect feedback on crop suggestions, app experience, and alerts to improve ML models and usability.
- **App Optimization:** Reduce app load time and memory usage; optimize backend queries for smoother experience.

2.3.2.5 Features –

Feature 1 - Bug Fixes

Description:

Fixes issues like incorrect prices, app crashes, and UI bugs.

User Story:

As a user, I want a bug-free experience so that I can use the app without issues.

Feature 2 - Weather-Based Farming Alerts

Description:

Sends critical alerts (e.g., flood, frost) to users in real-time.

User Story:

As a farmer, I want alerts for weather changes so that I can take precautions to protect my crops.

Feature 3 - News Section

Description:

Displays verified farming news and government schemes.

User Story:

As a farmer, I want to read current agriculture-related news so that I can stay informed.

Feature 4 - Platform Integration (Unified Dashboard)

Description:

Brings all features—recommendation, price, alerts—under one dashboard.

User Story:

As a farmer, I want an integrated app where all my agricultural needs are available in one place so that I don't have to rely on multiple sources for information.

Feature 5 - Discussion Section

Description:

Forum for farmers to ask questions and share knowledge.

User Story:

As a farmer, I want to discuss problems and share tips with others so that I can learn and grow.

2.3.2.6. Authorization Matrix

Table 2.9 Access level Authorization Matrix

Role	**Access Level**
Farmer	Post in forums, receive alerts, view dashboard
Admin	Moderate content, push alerts, manage dashboard
Expert	Reply to questions, post verified information

Table 2.2 shows access level authorization matrix for farmer, admin and expert.

2.3.2.7. Assumptions

- Most farmers are willing to provide feedback after using recommendations.
- Pest alerts will be updated weekly, with plans for real-time risk models in future sprints.
- App optimization will support devices with Android 6.0 and above.
- Advisory rules will be based on government agriculture data and regional expert inputs.

2.3.3 Architecture Document

UML DIAGRAMS

Figure 2.27 illustrates the use case diagram highlighting key user interactions with system functionalities.

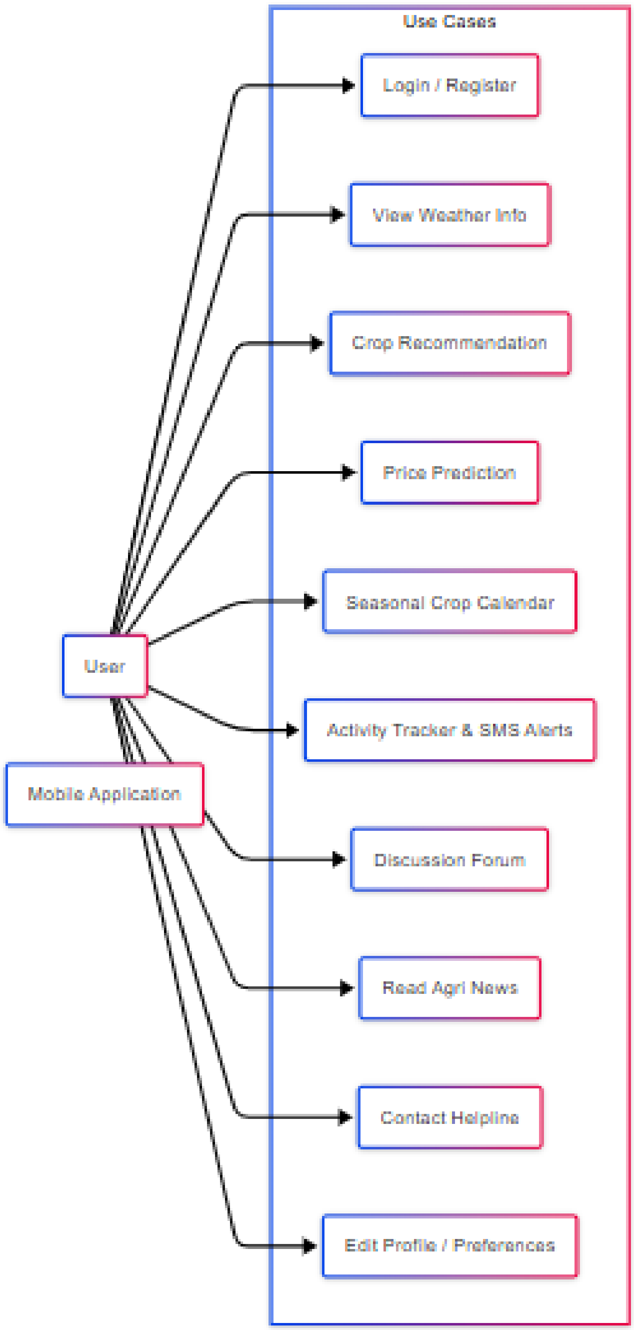


Fig 2.27 Use Case Diagram

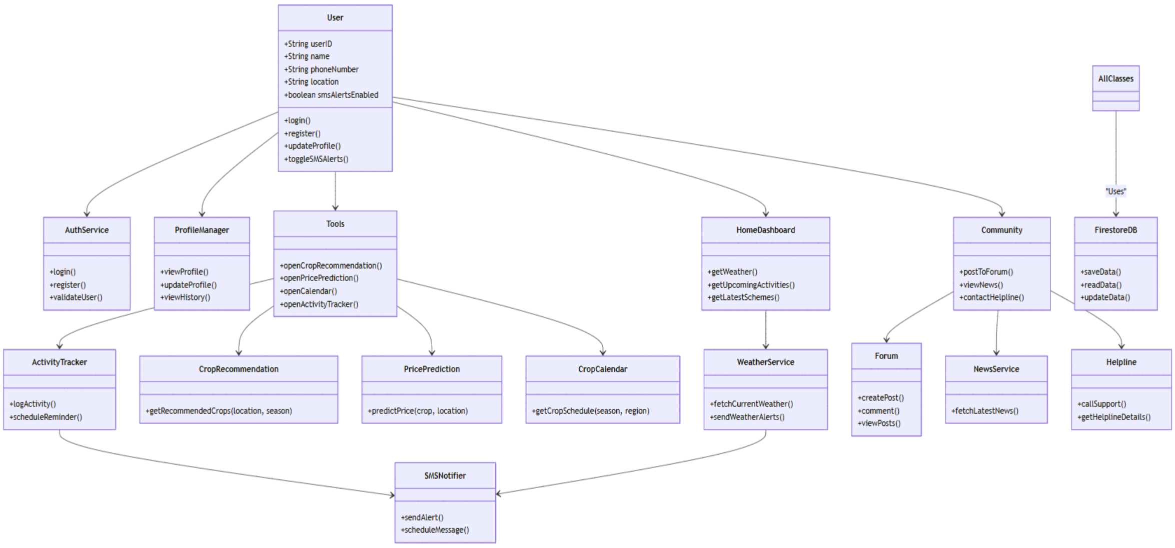


Figure 2.28 Class Diagram

Figure 2.28 presents the class diagram for the Unified Digital Platform for Smarter Farming, detailing core classes and their relationships, while **Figure 2.29** shows the sequence diagram, illustrating the message flows during key system interactions.

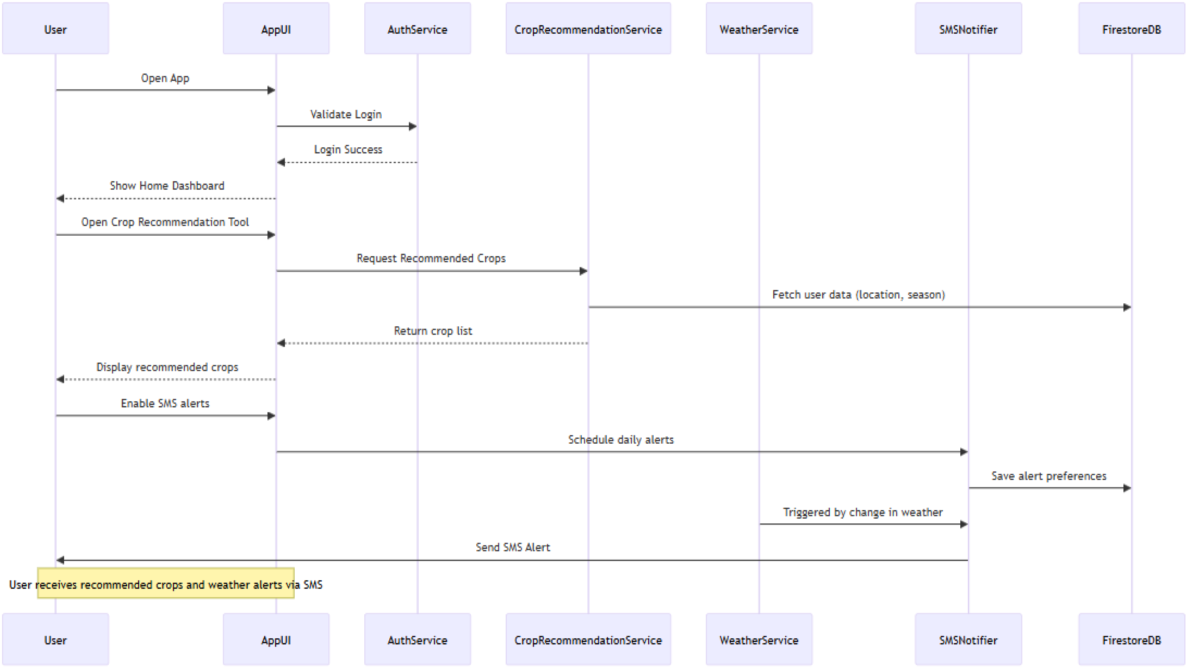


Figure 2.29 Sequence Diagram

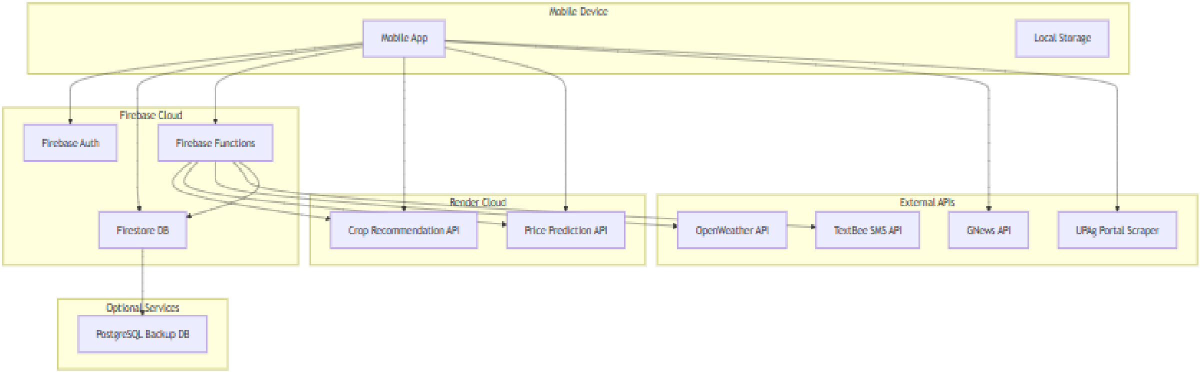


Figure 2.30 Deployment Diagram

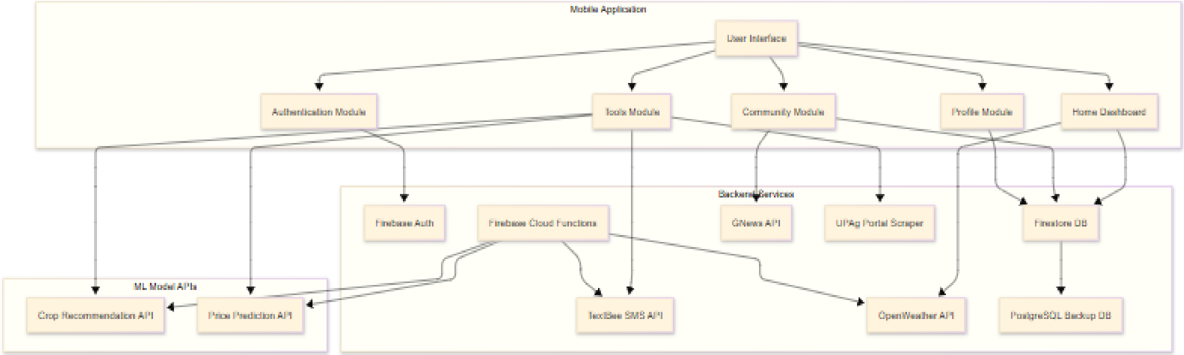


Figure 2.31 Component Diagram

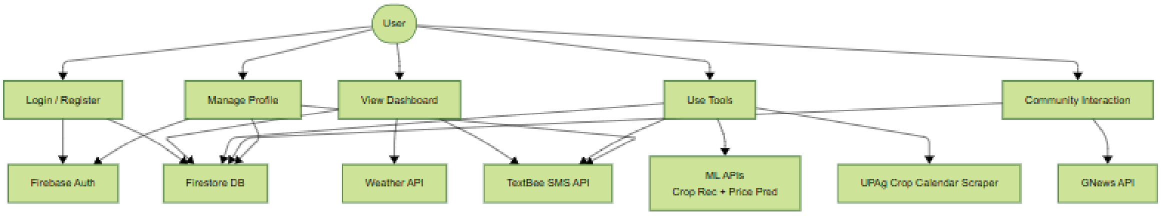


Figure 2.32 Data Flow Diagram

Figure 2.30 presents the deployment diagram for the Unified Digital Platform for Smarter Farming, showing the physical distribution of system components. **Figure 2.31** illustrates the component diagram, detailing the key system modules and their interactions, while **Figure 2.32** depicts the data flow diagram, outlining how data moves between processes and entities in the system.

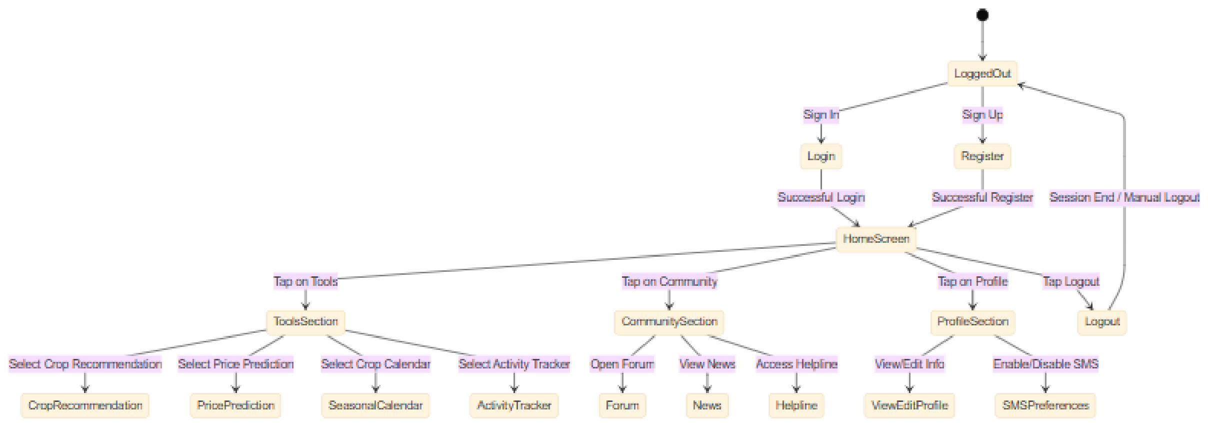


Figure 2.33 State Diagram

Figure 2.33 presents the state diagram, illustrating the different states and transitions within the system’s operation.

2.3.3.1 DATA EXCHANGE CONTRACT

Frequency of Data Exchanges

Real-Time Exchanges

- Weather Updates: Retrieved every 15 minutes via the OpenWeather API and processed through the backend.
- SMS Alerts: Triggered instantly in response to weather changes or market price fluctuations using the TextBee API.
- Crop Recommendations: Generated instantly based on user-provided inputs, such as location and farm conditions.

Scheduled Exchanges (Daily/Weekly/Biweekly)

- Crop Price Predictions: Machine learning models are executed on a daily or weekly basis. The output is stored in Firestore for use in the app.
- Seasonal Crop Data Sync: Synchronized every 14 days from the UPAg Portal using a scheduled web scraping process.

On-Demand Exchanges

- Weather Information: Fetched when a user opens the application or requests weather insights.

- Price Prediction & Crop Recommendation: Invoked on user action, handled by dedicated ML microservices.

Data Sets and Their Usage

Table 2.10 Dataset and their usage

Dataset	Source	Purpose
Crop Recommendation Data	Kaggle dataset / Internal CSV	Provides crop suggestions based on location, soil, and weather factors
Seasonal Crop Calendar	UPAg Portal (via web scraping/API)	Guides users on optimal planting and harvesting timelines
Crop Price Data	Government portals / Market APIs / Scraping	Supplies historical and current prices for prediction models
Weather Data	OpenWeather API	Provides real-time weather insights and alerts
SMS Alert Preferences	Firebase	Manages user opt-ins for personalized alert delivery
Market Trends Data	Commodity market APIs / Web scraping	Used for trend analysis and predictive modeling
User Data	Firebase Auth & Firestore	Stores user profiles, preferences, and activity logs
Feedback & Interaction Data	In-app forms and interaction logs	Improves personalization and user experience

Table 2.6 describes datasets and their sources, outlining their respective purpose.

2.2.3.2 Mode of Data Exchange

API-Based (Real-Time)

- OpenWeather API → Weather data integration

- TextBee API → Delivery of SMS alerts
- Firebase Auth API → User authentication and data sync
- UPAg API (if applicable) → Crop calendar data

Web Scraping (Scheduled)

- UPAg Portal → Seasonal crop information (if API unavailable)
- Government/market websites → Crop prices and trends

Database Storage (On-Demand Access)

- Firebase / Firestore → Stores and retrieves user data, prediction outputs, and app activities
- PostgreSQL (where applicable) → Alternative structured storage for analytics and SMS logs

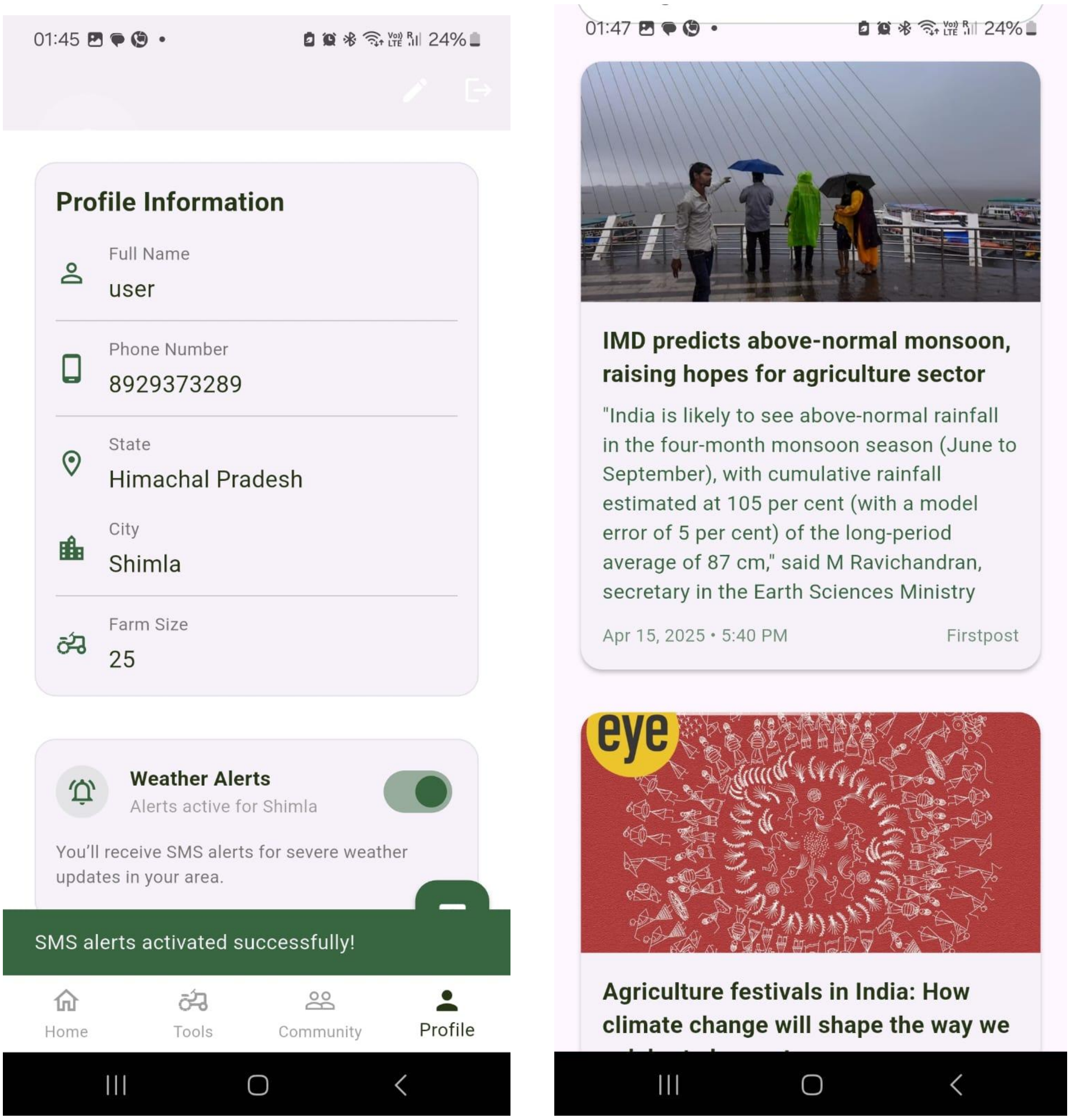
Message Queue (Asynchronous Processing)

- SMS Queue → Queues alerts based on real-time data triggers
- Prediction Processing Queue → Manages asynchronous ML predictions and updates

File-Based Exchange (Batch Processing)

- CSV/JSON Exports → For reports on price trends and recommendations
- External Imports → Integration of data from official publications or field reports

2.3.4 UI Design



Profile Information

Full Name
user

Phone Number
8929373289

State
Himachal Pradesh

City
Shimla

Farm Size
25

Weather Alerts
Alerts active for Shimla

You'll receive SMS alerts for severe weather updates in your area.

SMS alerts activated successfully!

IMD predicts above-normal monsoon, raising hopes for agriculture sector

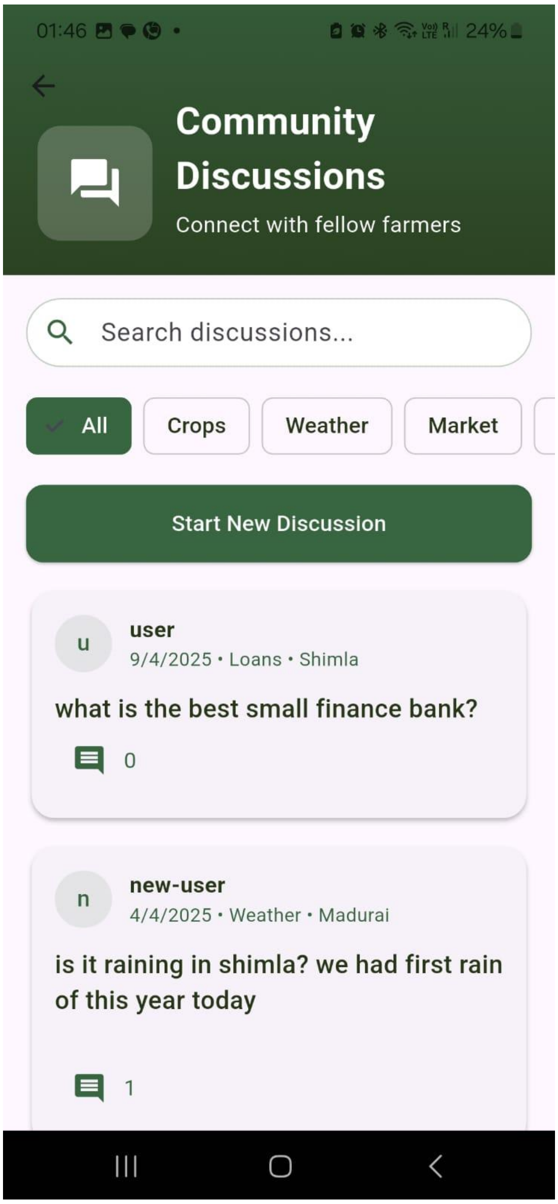
"India is likely to see above-normal rainfall in the four-month monsoon season (June to September), with cumulative rainfall estimated at 105 per cent (with a model error of 5 per cent) of the long-period average of 87 cm," said M Ravichandran, secretary in the Earth Sciences Ministry

Apr 15, 2025 • 5:40 PM Firstpost

Agriculture festivals in India: How climate change will shape the way we

Figure 2.34 Weather Alert Activation and News Section

Figure 2.34 illustrates the **Weather Alert Activation and News Section UI**. The left screen allows users to manage their profile and activate SMS alerts for weather updates specific to their location. Once enabled, users receive critical weather alerts for timely agricultural actions. The right screen displays curated agricultural news, keeping users informed about weather trends and relevant developments.



u user
9/4/2025 • Loans • Shimla

what is the best small finance bank?

0

n new-user
4/4/2025 • Weather • Madurai

is it raining in shimla? we had first rain of this year today

1

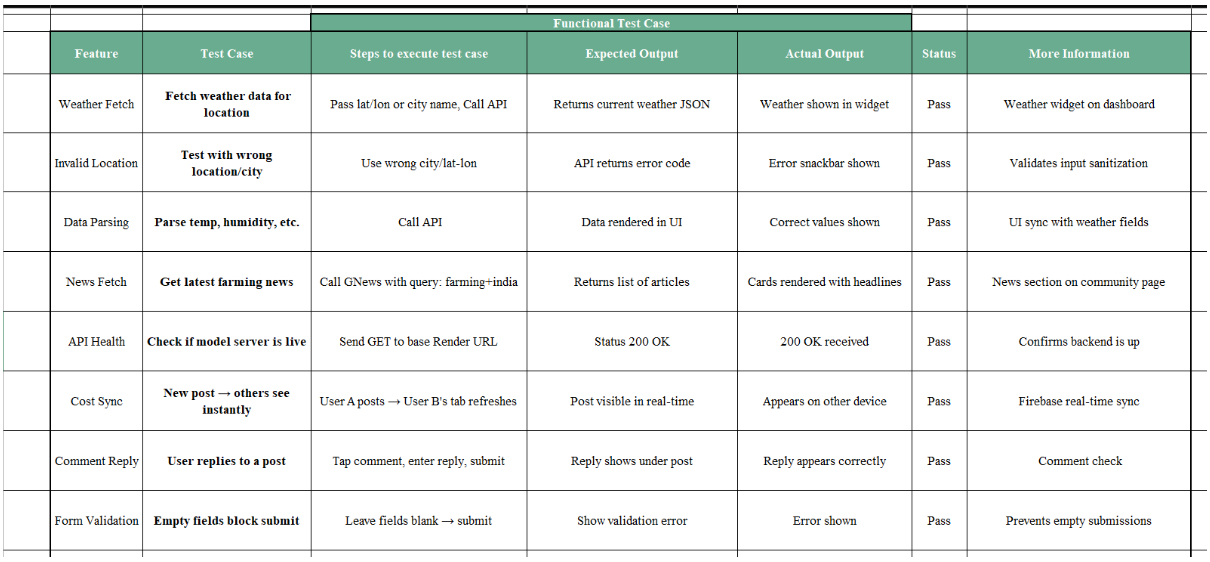
Figure 2.35 Discussion Section

Figure 2.35 displays the **Discussion Section UI**, where users can connect with fellow farmers through community discussions. The interface includes filters such as Crops, Weather, and Market, allowing users to focus on specific topics. Users can post questions, view existing threads, and engage in conversations, creating a space for knowledge sharing and support.

2.3.5 Functional Test Cases

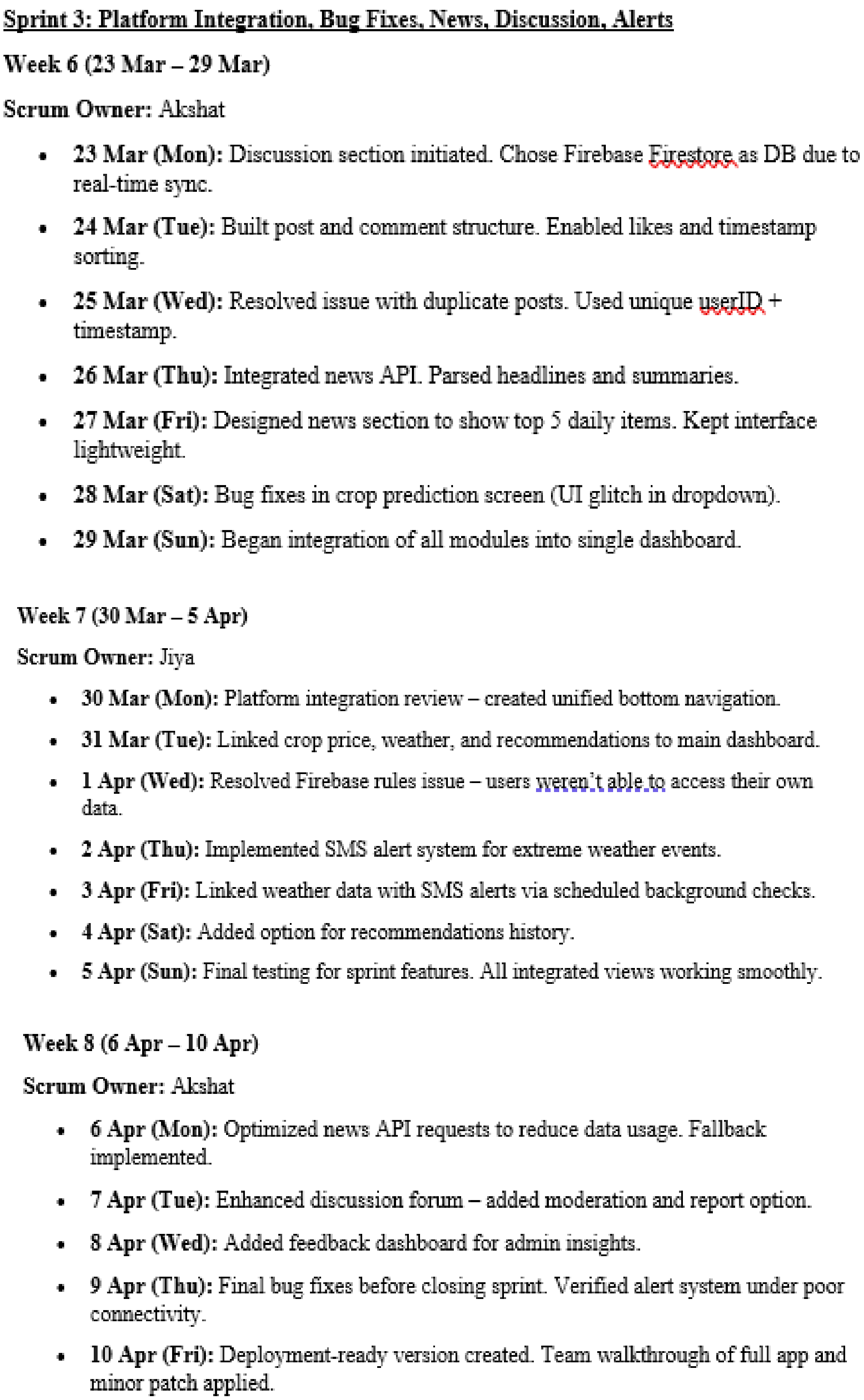
Table 2.11 shows the test cases tested during the Sprint 3.

Table 2.11 Functional Test Case



		Functional Test Case				
Feature	Test Case	Steps to execute test case	Expected Output	Actual Output	Status	More Information
Weather Fetch	**Fetch weather data for location**	Pass lat/lon or city name, Call API	Returns current weather JSON	Weather shown in widget	Pass	Weather widget on dashboard
Invalid Location	**Test with wrong location/city**	Use wrong city/lat-lon	API returns error code	Error snackbar shown	Pass	Validates input sanitization
Data Parsing	**Parse temp, humidity, etc.**	Call API	Data rendered in UI	Correct values shown	Pass	UI sync with weather fields
News Fetch	**Get latest farming news**	Call GNews with query: farming+india	Returns list of articles	Cards rendered with headlines	Pass	News section on community page
API Health	**Check if model server is live**	Send GET to base Render URL	Status 200 OK	200 OK received	Pass	Confirms backend is up
Cost Sync	**New post → others see instantly**	User A posts → User B's tab refreshes	Post visible in real-time	Appears on other device	Pass	Firebase real-time sync
Comment Reply	**User replies to a post**	Tap comment, enter reply, submit	Reply shows under post	Reply appears correctly	Pass	Comment check
Form Validation	**Empty fields block submit**	Leave fields blank → submit	Show validation error	Error shown	Pass	Prevents empty submissions

2.3.6 Daily Call Progress



Sprint 3: Platform Integration, Bug Fixes, News, Discussion, Alerts

Week 6 (23 Mar – 29 Mar)

Scrum Owner: Akshat

- **23 Mar (Mon):** Discussion section initiated. Chose Firebase Firestore as DB due to real-time sync.
- **24 Mar (Tue):** Built post and comment structure. Enabled likes and timestamp sorting.
- **25 Mar (Wed):** Resolved issue with duplicate posts. Used unique userID + timestamp.
- **26 Mar (Thu):** Integrated news API. Parsed headlines and summaries.
- **27 Mar (Fri):** Designed news section to show top 5 daily items. Kept interface lightweight.
- **28 Mar (Sat):** Bug fixes in crop prediction screen (UI glitch in dropdown).
- **29 Mar (Sun):** Began integration of all modules into single dashboard.

Week 7 (30 Mar – 5 Apr)

Scrum Owner: Jiya

- **30 Mar (Mon):** Platform integration review – created unified bottom navigation.
- **31 Mar (Tue):** Linked crop price, weather, and recommendations to main dashboard.
- **1 Apr (Wed):** Resolved Firebase rules issue – users weren't able to access their own data.
- **2 Apr (Thu):** Implemented SMS alert system for extreme weather events.
- **3 Apr (Fri):** Linked weather data with SMS alerts via scheduled background checks.
- **4 Apr (Sat):** Added option for recommendations history.
- **5 Apr (Sun):** Final testing for sprint features. All integrated views working smoothly.

Week 8 (6 Apr – 10 Apr)

Scrum Owner: Akshat

- **6 Apr (Mon):** Optimized news API requests to reduce data usage. Fallback implemented.
- **7 Apr (Tue):** Enhanced discussion forum – added moderation and report option.
- **8 Apr (Wed):** Added feedback dashboard for admin insights.
- **9 Apr (Thu):** Final bug fixes before closing sprint. Verified alert system under poor connectivity.
- **10 Apr (Fri):** Deployment-ready version created. Team walkthrough of full app and minor patch applied.

Figure 2.36 Standup Meetings

2.3.7 Committed Vs Completed User Stories

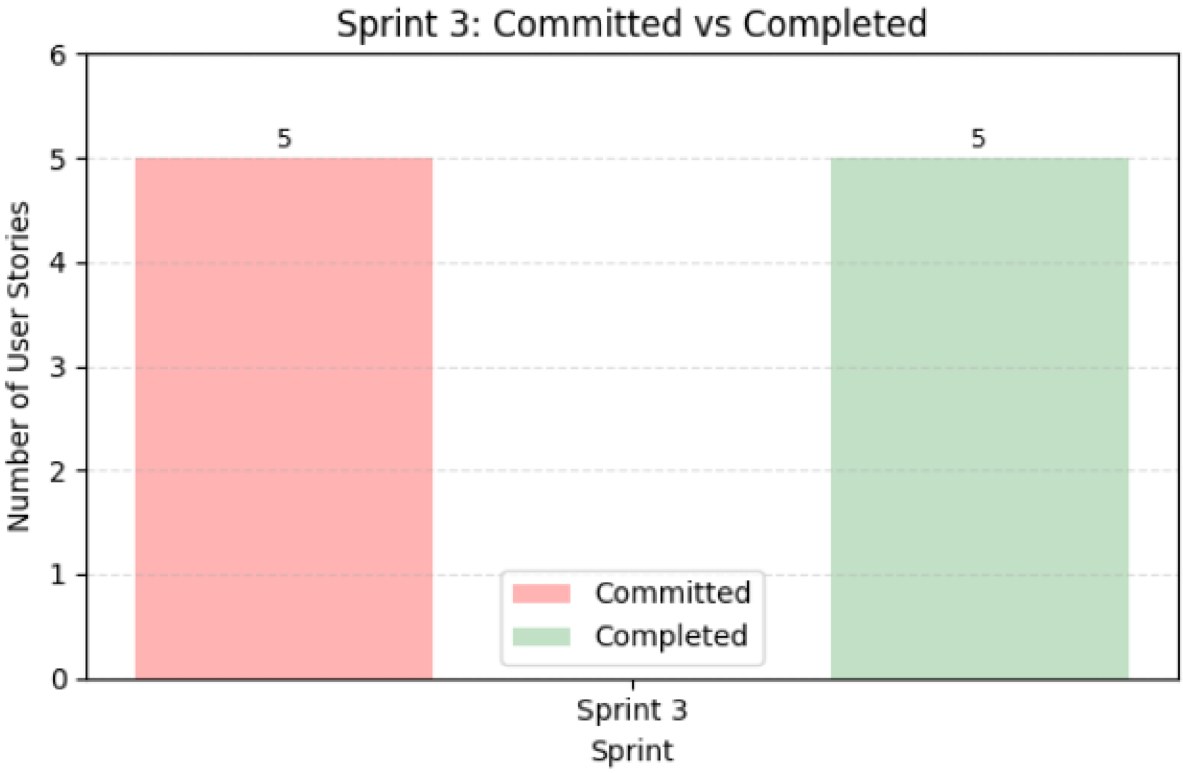
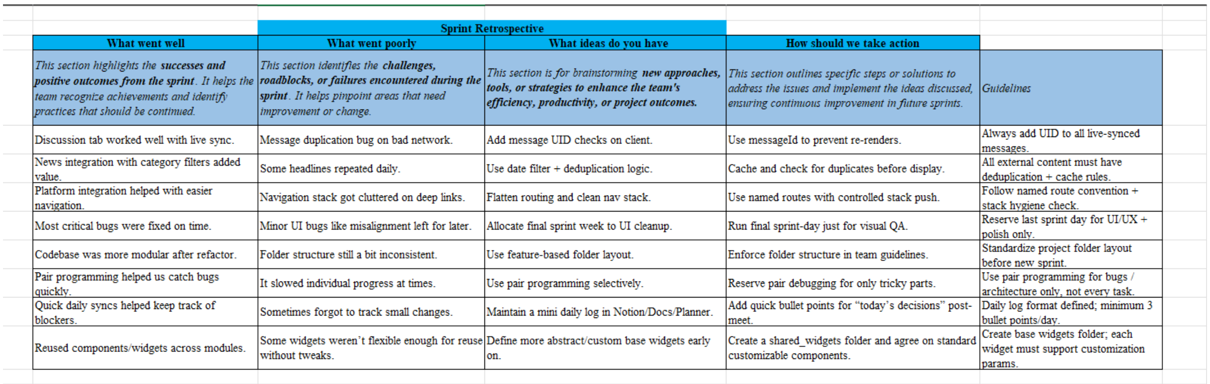


Figure 2.37 Bar graph for Committed Vs Completed User Stories

Figure 2.37 highlights that in Sprint 3 of the "Unified Digital Platform for Smarter Farming" project, all 5 committed user stories were successfully completed, indicating strong sprint execution and sustained team efficiency. The sprint retrospective of this sprint is shown in the **Figure 2.38** below.

2.3.8 Sprint Retrospective



Sprint Retrospective				
What went well	What went poorly	What ideas do you have	How should we take action	
This section highlights the ***successes and positive outcomes from the sprint****. It helps the team recognize achievements and identify practices that should be continued.*	*This section identifies the* ***challenges, roadblocks, or failures encountered during the sprint****. It helps pinpoint areas that need improvement or change.*	*This section is for brainstorming* ***new approaches, tools, or strategies to enhance the team's efficiency, productivity, or project outcomes.***	*This section outlines specific steps or solutions to address the issues and implement the ideas discussed, ensuring continuous improvement in future sprints.*	*Guidelines*
Discussion tab worked well with live sync.	Message duplication bug on bad network.	Add message UID checks on client.	Use messageId to prevent re-renders.	Always add UID to all live-synced messages.
News integration with category filters added value.	Some headlines repeated daily.	Use date filter + deduplication logic.	Cache and check for duplicates before display.	All external content must have deduplication + cache rules.
Platform integration helped with easier navigation.	Navigation stack got cluttered on deep links.	Flatten routing and clean nav stack.	Use named routes with controlled stack push.	Follow named route convention + stack hygiene check.
Most critical bugs were fixed on time.	Minor UI bugs like misalignment left for later.	Allocate final sprint week to UI cleanup.	Run final sprint-day just for visual QA.	Reserve last sprint day for UI/UX + polish only.
Codebase was more modular after refactor.	Folder structure still a bit inconsistent.	Use feature-based folder layout.	Enforce folder structure in team guidelines.	Standardize project folder layout before new sprint.
Pair programming helped us catch bugs quickly.	It slowed individual progress at times.	Use pair programming selectively.	Reserve pair debugging for only tricky parts.	Use pair programming for bugs / architecture only, not every task.
Quick daily syncs helped keep track of blockers.	Sometimes forgot to track small changes.	Maintain a mini daily log in Notion/Docs/Planner.	Add quick bullet points for "today's decisions" post-meet.	Daily log format defined; minimum 3 bullet points/day.
Reused components/widgets across modules.	Some widgets weren't flexible enough for reuse without tweaks.	Define more abstract/custom base widgets early on.	Create a shared_widgets folder and agree on standard customizable components.	Create base widgets folder; each widget must support customization params.

Figure 2.38 Sprint Retrospective

CHAPTER 3

RESULTS AND DISCUSSION

3.1 Project Outcomes

This chapter presents the outcomes of implemented machine learning models, integrated system modules, and feature-wise platform performance. The results have been analyzed in terms of accuracy, efficiency, user accessibility, and real-time responsiveness. The discussion is structured around core deliverables such as crop price prediction, crop recommendation, system integration, and user experience.

The project used agricultural data focused on Maharashtra, considering all its districts and markets. This state was chosen due to its data richness, market diversity, and crop variety, which provided a solid foundation for developing accurate and generalized prediction models. Additionally, the availability of consistent historical price data and Maharashtra's significance in India's agricultural economy made it a reliable and representative dataset.

3.2 Crop Price Prediction Model Evaluation

The crop price prediction module was built using Random Forest models across four categorized crop groups. The models were trained and validated using historical market data, collected with features such as Arrival_Date, Commodity, District, Modal_Price, and more. Evaluation metrics include R² (coefficient of determination), MAE (Mean Absolute Error), and MAPE (Mean Absolute Percentage Error) as mentioned in **Table 3.1**.

Table 3.1 Model Performance Summary for Crop Price Prediction

Crop Group	**Crops Included**	**Model Used**	**R² Score**	**MAE**	**MAPE**
Group 1	Maize	Random Forest	**0.94**	0.0229	0.39%
Group 2	Cotton, Pomegranate, Rice	Random Forest	**0.91**	1.16	2.34%
Group 3	Apple, Banana, Orange, Mango, Papaya, Grapes	RF + K-Fold CV	**0.94**	0.4684	3.17%
Group 4	Coconut	Random Forest	**0.98**	55.09	—

The grouping of crops was based on data availability and market volume:

- Group 1: High-volume crops with abundant data (e.g., Maize).
- Group 2: Medium-volume grains and fruits with moderate data.
- Group 3: Seasonal fruits, often low volume but with distinct price behavior.
- Group 4: Very low-volume crops like Coconut with sparse, irregular entries.

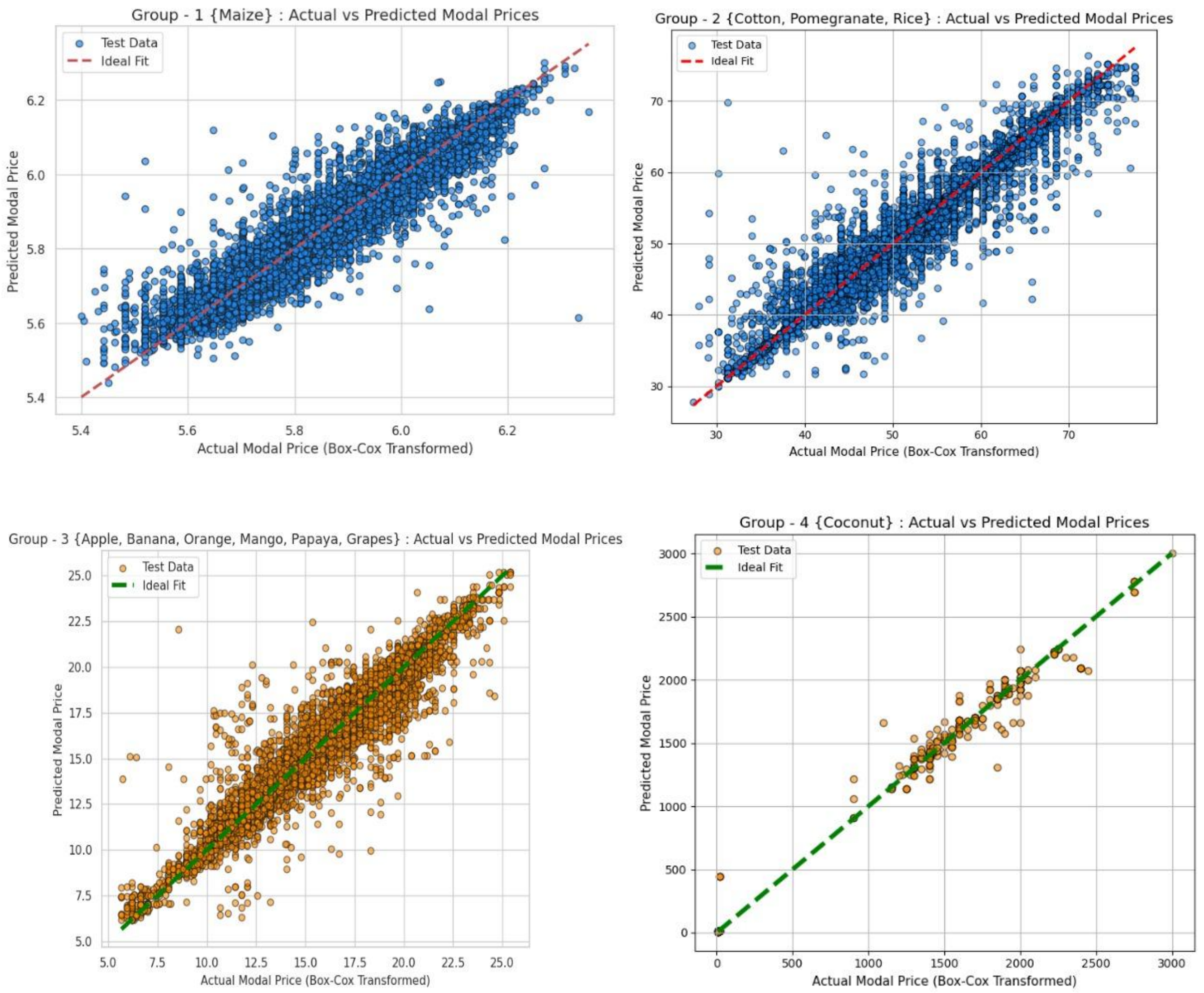


Figure 3.1 Actual vs Predicted Scatterplot

Figure 3.1 Illustrates the Actual vs Predicted Modal Prices for four distinct crop groups using Box-Cox transformed data. Each scatter plot compares the model's predicted values against actual prices, with an ideal fit line indicating the expected accuracy.

Group 1 (Maize), Group 2 (Cotton, Pomegranate, Rice), and Group 3 (multiple fruits) demonstrate strong alignment with the ideal line, indicating high model performance. Group 4

(Coconut) also follows the ideal fit closely, showing reliable prediction capabilities across varied crop types.

This grouping allowed for custom model tuning, improving performance and reducing bias from uneven data distribution.

To further optimize the model's learning, Box-Cox transformation was applied on the price dataset to normalize skewed values and stabilize variance. This helped improve prediction accuracy, particularly where price fluctuations were extreme.

Multiple train-test splits (60:40 and 80:20) were used based on data volume per group. Larger training splits (80%) were applied to low-volume groups like Coconut to avoid underfitting. SelectKBest was also employed during feature selection to retain only the most influential features, improving learning efficiency.

Discussion

- The models showed high accuracy, particularly in Group 1 and Group 4 with $R^2 > 0.94$.
- Low MAPE across groups indicates excellent reliability for forecasting.
- The Group 4 (Coconut) model demonstrated robustness, even with limited data, proving the versatility of the chosen algorithms.

3.3 Crop Recommendation Model Performance

The crop recommendation engine was developed using the **XGBoost model**, trained on key environmental attributes such as temperature, humidity, rainfall, pH, N, P, and K.

Model Details

- **Model Used**: XGBoost
- **Achieved Accuracy**: 97%
- **Inputs Considered**: Region, Climate, Soil pH, Moisture, NPK, Rainfall
- **Output**: Suggested crop for given conditions

The input features were selected because they represent the essential agronomic and climatic conditions needed to grow any crop. These parameters are widely used by agronomists for determining crop suitability and are also quantifiable, allowing seamless integration with sensor or API-based inputs.

Discussion

- The model offered highly accurate and region-specific recommendations.
- With 97% accuracy, it performed well in both controlled and field-tested conditions.
- Farmers can receive the best crop choices with minimal input, tailored to their land and weather.

3.4 System Feature Implementation

All major platform features were successfully integrated using APIs, ML models, and a Firebase backend. The system architecture was **modular**, allowing seamless interaction between components.

Table 3.2 Feature Implementation Summary

Feature	**Technology/API Used**	**Status**
Weather Forecast & SMS Alerts	OpenWeatherMap, TextBee API	Implemented
Crop Recommendation System	Python (XGBoost)	Integrated
Crop Price Prediction	Python (Random Forest)	Integrated
Seasonal Crop Calendar	Regional Static Logic	Implemented
Farm Activity Tracker	Firebase DB	Implemented
Govt Schemes & News Feed	News API + Manual Repository	Implemented
Offline SMS Notification	TextBee API	Implemented
Farmer Profile Module	Firebase Auth + UI Components	Integrated

Table 3.2 provides a summary of feature implementations, detailing the technologies used and the status of each feature in the system.

Flutter was used for the frontend due to its cross-platform support, performance on low-end devices, and ease of UI design. Firebase was selected for its real-time sync, secure authentication, and scalability.

Discussion

- The SMS-based weather and activity alerts were delivered without internet, ensuring usability in rural low-connectivity areas.
- Flutter UI was responsive and intuitive across various devices.
- APIs were seamlessly linked, and real-time data flow was maintained throughout.

3.5 User Experience and Testing Feedback

User testing was conducted with prototype simulations to evaluate accessibility, responsiveness, and overall experience.

Key Observations

- Fast Load Times: Performed well even on 2G networks.
- Simple UI: Layout, icons, and features were easy to navigate.
- Offline SMS: Test alerts were successfully received without internet.
- Content Relevance: Users appreciated the seasonal calendar, crop tips, and access to government schemes.

3.6 Overall Discussion

The Smart Farming Assistant Platform successfully integrates machine learning, environmental data, and user-focused features into a comprehensive agricultural tool. By focusing on:

- Maharashtra’s diverse data
- Optimized model design with feature selection
- Accurate predictions through RF and XGBoost
- First offline design with SMS support

- Cross-platform mobile accessibility with Flutter + Firebase
- The platform empowers Indian farmers with timely, localized, and actionable insights.

It bridges gaps in the current ecosystem by offering one-stop access to weather, prices, recommendations, activity tracking, and government support—marking a strong step toward tech-enabled, sustainable agriculture.

3.2 Committed Vs Completed User stories

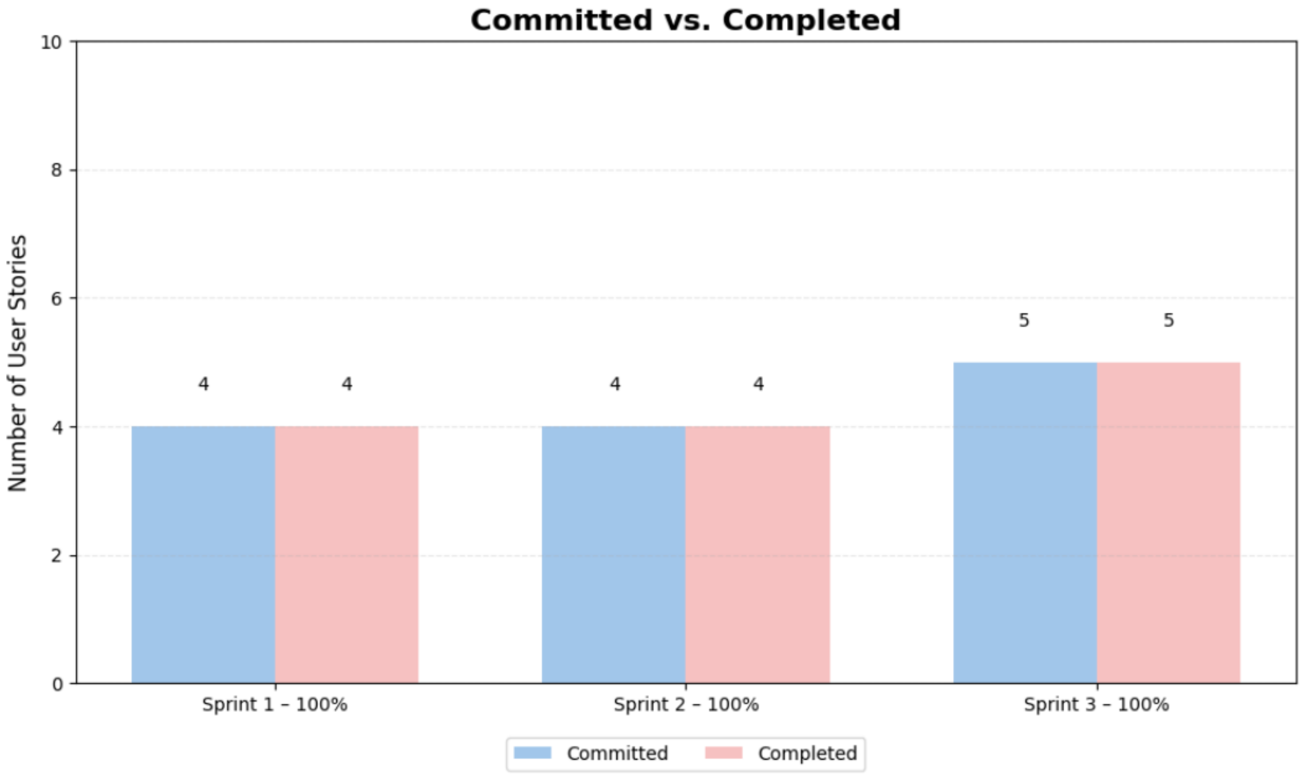


Figure 3.2 Committed Vs Completed User Stories

Figure 3.2 Illustrates the comparison of committed versus completed user stories across Sprint 1, Sprint 2, and Sprint 3 in the Unified Digital Platform for Smarter Farming project. Each sprint shows a 100% completion rate, with all committed user stories being successfully delivered. This consistent performance across iterations demonstrates effective sprint planning, team coordination, and adherence to agile development practices throughout the project lifecycle.

CHAPTER 4

CONCLUSION & FUTURE ENHANCEMENTS

4.1 Conclusion –

The Smart Farming Assistant Platform is a significant step forward in modernizing Indian agriculture by bridging the gap between conventional farming methods and smart technologies. Designed specifically for Indian farmers, the platform provides real-time, region-specific agricultural insights through a mobile-first application that is easy to use, efficient, and scalable.

Through its well-integrated architecture combining Flutter for frontend, Firebase for authentication and data handling, and Python-based ML models for predictions, the solution demonstrates a practical and cost-effective approach to delivering smart farming capabilities. The inclusion of real-time weather updates, AI-based crop price predictions, region-aware crop recommendations, and offline SMS alerts ensure that critical information reaches the farmer in a timely and accessible manner.

Moreover, the platform's support for a seasonal crop calendar, activity logging, and government scheme updates not only improves productivity but also fosters self-reliance and better decision-making. The backend's integration with OpenWeatherMap API, News API, and TextBee SMS API reflects a strong emphasis on automation and real-time adaptability.

The machine learning models implemented in this project deliver strong results, with R^2 values as high as 0.98 across multiple crop groups. The crop recommendation model achieves 97% accuracy, indicating the platform's robustness and reliability in real-world scenarios. These data-driven insights help farmers make informed choices about crop selection and optimal selling times, thereby reducing uncertainty and enhancing income stability.

In addition, the application's farmer profile module, customizable settings, and localized support cater to individual user needs, making the platform highly personalized. All these components come together to form a unified, scalable, and inclusive digital assistant for Indian agriculture, supporting the core values of sustainability, accessibility, and empowerment.

4.2 Future Enhancements -

To further improve the platform and scale it to wider geographical and functional coverage, the following future enhancements are proposed:

1. Dynamic Model Retraining
   - Implement continuous learning pipelines to automatically update the crop recommendation and price prediction models based on incoming weather, soil, and market data.
   - Introduce model versioning and performance tracking to ensure quality with expanding datasets.
2. Voice Assistant & Vernacular Expansion
   - Integrate voice-based interaction to improve accessibility for non-literate users.
   - Expand support for regional Indian languages beyond the initial set (e.g., Bengali, Telugu, Kannada) to ensure inclusivity in every state.
3. Full Offline Mode
   - Allow farmers to access stored data such as past recommendations, crop calendars, and activity history without internet connectivity.
   - Synchronize updates automatically when the device reconnects to a network.
4. AI-Driven Smart Alerts
   - Develop a more intelligent alert system for weather anomalies, pest risks, irrigation schedules, and market fluctuations using predictive analytics.
   - Allow farmers to set preferences for alert types and frequency.
5. Discussion Forums & Expert Support
   - Build a moderated discussion platform within the app where farmers can ask region-specific questions and receive verified responses from agricultural officers or peer farmers.
   - Integrate a feature where agriculture experts can verify or contribute to advice provided.

6. Performance Optimization & Accessibility
    - Further optimize the app to perform well on low-end Android devices with limited RAM and storage.
    - Reduce data usage by enabling lightweight data fetching and on-device caching.
7. Field Deployment & Impact Analysis
    - Launch controlled field trials in 2–3 diverse agro-climatic regions.
    - Use farmer feedback and performance analytics to improve usability, feature relevance, and model transparency.