

# Phytora

---

By Team 3 SALAAR

SPRINT 1

# Agenda

- Agenda
- Team Member Roles and Responsibilities
- Project Description
- Improvements Made from Professor Feedback
- Team Working Agreement
- Personas
- Minimal Viable Product (MVP)

- Technologies
- Algorithms
- Diagrams
- Product Backlog
- Sprint 1 Backlog
- Sprint 1 Test Cases
- Stories Completed (Sprint 1)
- Stories Not Completed (Sprint 1 Carry Over)
- Metrics

- Retrospective
- Sprint 2 Planning
- Project Demo - Sprint 1
- GitHub Link & Live Application Demo



# Team Member Roles and Responsibilities

01

# Team Members



**Machine Learning  
Engineer**

Manoj Kumar Reddy Mule



**Backend  
Developer**

Krishna Kishore Varma  
Kalidindi



**Backend  
Developer**

Gopi Krishna Bhookya



**Frontend  
Developer**

Saipriya Rampudi

# Team Members



**Machine Learning  
Engineer**

Naga Karthik Potru



**Frontend  
Developer**

Paul Morales



**Backend  
Developer & QA**

Nikitha Arpula



**Frontend  
Developer**

Nagalakshmi Narra

# 2. Project Description

Project Name:	Phytora
Team:	Salaar
Project Description:	<p><b>For</b> farmers and agricultural researchers <b>who</b> need an efficient and cost-effective method to detect apple tree diseases, <b>the</b> AI-based plant disease detection system <b>is a</b> deep learning-powered solution <b>that</b> identifies diseases like scab and rust early, preventing crop loss. <b>Unlike</b> traditional manual inspection methods, <b>our application</b> leverages Convolutional Neural Networks (CNNs) to provide fast, accurate, and automated disease classification, overcoming challenges like background noise and variable disease appearances.</p>
Benefit Outcomes:	<ul style="list-style-type: none"><li><b>Early Detection:</b> Enables quick identification of apple tree diseases, reducing yield loss.</li><li><b>Cost-Effective:</b> Eliminates the need for frequent manual inspections, saving labor costs.</li><li><b>High Accuracy:</b> Uses AI and deep learning models to minimize misdiagnosis.</li><li><b>Scalability:</b> Can be applied to large farms and integrated into existing agricultural systems.</li><li><b>User-Friendly:</b> Provides a simple interface for farmers to upload images and get instant results.</li></ul>
Github Link:	<a href="https://github.com/htmw/2025S-SALAAR/wiki">https://github.com/htmw/2025S-SALAAR/wiki</a>

# 3. Improvements



# Improvements

Team Introduction Slides Reordered - Correct order, follows checklist

Slide Titles Standardized - Added and fixed missing titles

Content Alignment Fixed - Text matches slide topic

Visual and Structural Cleanup - Clean format, removed clutter



# Team Working Agreement

04

## **Team Working Agreement – Phytoria**

### **1. Purpose & Scope**

This agreement establishes collaboration guidelines for the development of Phytoria by Team Salaar, an AI-powered plant disease detection system. It ensures smooth execution, accountability, and efficient teamwork while maintaining software development best practices.

### **2. Roles & Responsibilities**

- Frontend Developer (FD): Builds and optimizes the Next.js UI, ensuring seamless interaction with the backend.
- Backend Developer (BD): Develops Node.js APIs, integrates with FastAPI, and handles system architecture.
- Machine Learning Engineer (MLE): Trains and fine-tunes EfficientNet, manages datasets, and ensures accurate model predictions.
- Quality Assurance (QA): Conducts testing, detects bugs, ensures performance optimization, and validates deployment readiness.

All members are responsible for delivering assigned tasks, maintaining quality, and proactively communicating challenges.

### **3. Sprint Planning & Retrospective**

- Sprint Ends – Monday: Retrospective to review achievements, blockers, and process improvements.
- Sprint Planning – Wednesday: Define tasks, assign responsibilities, and set objectives for the new sprint.

### **4. Communication & Collaboration**

- WhatsApp – Daily updates and quick discussions.
- Google Meet – Sprint planning, retrospectives, and issue resolution meetings.
- Task Management – Tracked Trello for visibility and accountability.

### **5. Development Workflow & Code Management**

- Version Control – Use GitHub with feature branching and PR reviews before merging.
- Testing & Quality Assurance – Implement unit and integration tests, followed by manual validation.
- Deployment – Continuous testing and incremental deployment for stability.

### **6. Accountability & Conflict Resolution**

- Team members must proactively report blockers.
- Issues should be discussed and resolved within the team.
- Repeated delays or unresponsiveness may result in task reassignment.

### **7. Work Hours & Availability**

- Commitment: 10-15 hours per week per team member.
- Work flexibility, but tasks must align with sprint goals.

### **8. Agreement & Commitment**

All members agree to uphold this agreement to ensure efficient collaboration, high-quality development, and successful delivery of Phytoria.

# Personas

05

## Farmer - John Carter

- **Age:** 45
- **Occupation:** Apple Orchard Owner
- **Location:** Washington, USA
- **Technology Proficiency:** Low to Moderate
- **Farming Experience:** 20+ years
- **Pain Points:**
  - Relies on manual disease inspection, which is time-consuming and error-prone.
  - Finds it hard to differentiate between early-stage symptoms of apple scab and rust.
  - High costs for hiring agricultural experts and consultants.
  - Crop losses due to late detection of infections, leading to lower profits.
- **Needs & Expectations:**
  - A mobile-friendly app where he can take a picture of infected leaves and receive instant results.
  - An AI-driven advisory system suggesting treatment solutions for different disease stages.
  - Offline functionality, since farms often have poor internet connectivity.
  - A cost-effective solution that doesn't require expensive hardware.
- **How Our Product Helps:**
  - Automated disease detection reduces guesswork and ensures timely intervention.
  - Provides step-by-step guidance on disease management, reducing dependency on external experts.
  - Saves time and labor costs, increasing farm productivity and yield.



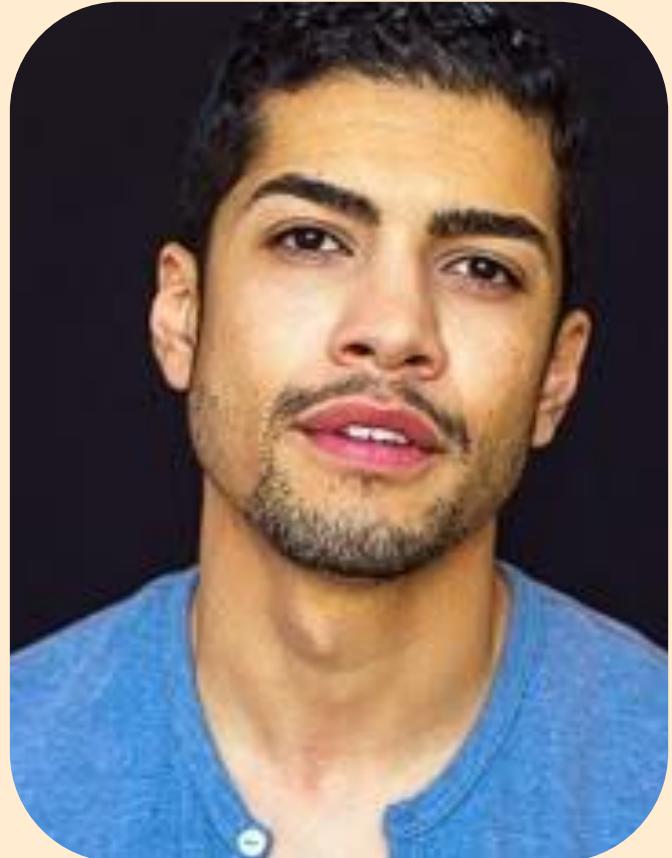
## Agricultural Researcher - Dr. Emmy Zhao

13

- **Age:** 38
- **Occupation:** Plant Pathologist & Researcher
- **Location:** California, USA
- **Technology Proficiency:** High
- **Research Focus:** AI-based plant disease prediction, crop disease resistance
- **Pain Points:**
  - Needs large, high-quality datasets to improve disease classification models.
  - Variability in disease symptoms (color, shape, size) makes model training challenging.
  - Lack of real-time field data for studying disease spread patterns.
  - Difficulty in integrating AI-based tools with existing agricultural research systems.
- **Needs & Expectations:**
  - Access to a labeled image dataset for improving AI accuracy.
  - A cloud-based dashboard with insights into disease occurrences across different regions.
  - Ability to contribute research data and collaborate with other scientists.
  - A customizable AI model that can be adapted for different crops and environmental conditions.
- **How Our Product Helps:**
  - Provides real-time disease detection data, aiding in research and policy recommendations.
  - Offers a machine-learning training platform for researchers to refine detection algorithms.
  - Helps predict disease outbreaks, leading to more proactive agricultural strategies.



- **Age:** 32
- **Occupation:** CEO of a Smart Farming Startup
- **Location:** Texas, USA
- **Technology Proficiency:** High
- **Business Focus:** AI-driven farm automation, precision agriculture
- **Pain Points:**
  - Farmers are hesitant to adopt AI-based solutions due to trust issues and cost concerns.
  - Needs a scalable AI model that can integrate with existing smart farming hardware.
  - Difficulty in monetizing AI-based agricultural solutions while keeping them affordable.
  - Requires accurate, real-world disease detection data to improve product effectiveness.
- **Needs & Expectations:**
  - A lightweight AI model that can be embedded in farm drones and IoT devices.
  - A subscription-based business model that balances affordability and profitability.
  - Market validation data to demonstrate AI's effectiveness in reducing crop losses.
  - A seamless API to integrate disease detection with smart irrigation and pesticide control systems.
- **How Our Product Helps:**
  - Provides real-time analytics and reporting tools for farm management.
  - Enables precision agriculture, optimizing pesticide use and reducing costs.





# Minimal Viable Product (MVP)

06

# Minimal Viable Product

Our MVP delivers the core capability — detecting apple leaf diseases from images using AI. It focuses only on essential features needed for farmers to quickly upload images and receive reliable disease identification.



## Image Upload

Farmers can upload apple leaf images directly into the application for disease analysis and classification.



## Disease Detection

System analyzes the uploaded image and automatically identifies if the leaf is healthy or diseased.



## AI Model Integration

The backend integrates a trained CNN model to detect and classify diseases like scab and rust accurately.



## Instant Diagnosis

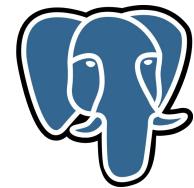
After analysis, the application displays the disease name and confidence score in real-time for the user.

07

# Technologies

# Technologies

- **Frontend:** Next.js - Fast, scalable, and SEO-friendly UI.
- **Backend:** Node.js & FastAPI - Efficient API handling and real-time processing.
- **Machine Learning:** PyTorch - Deep learning for accurate disease detection.
- **Database:** PostgreSQL - Reliable relational database for storing image data and results.
- **IDE/Development Tool:** Visual Studio Code - Primary code editor used by the team.
- **Version Control:** GitHub - For collaboration, versioning, and artifact storage.
- **Deployment:** Vercel - Hosting the frontend with automatic deployment from GitHub.



# Algorithms

08

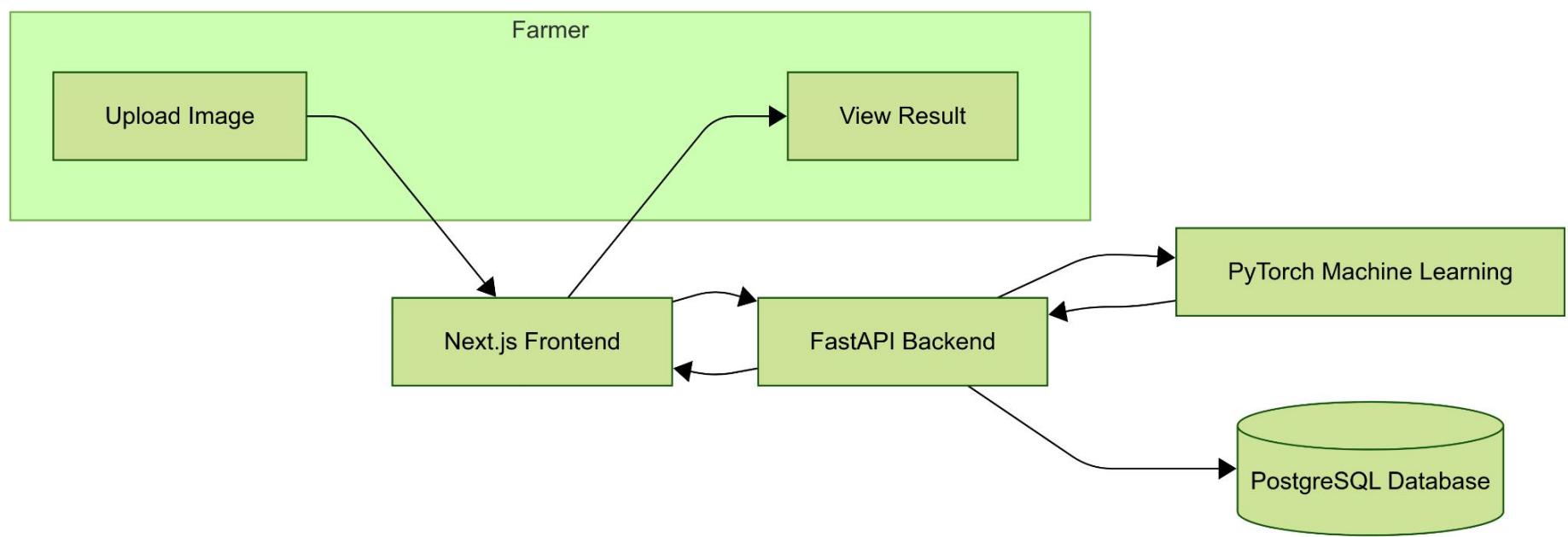
# Algorithms

Our system utilizes EfficientNet, a state-of-the-art CNN model, for highly accurate and optimized plant disease detection. EfficientNet balances speed and accuracy by scaling depth, width, and resolution efficiently. Transfer learning enhances performance, while FastAPI enables real-time predictions, ensuring fast, precise, and resource-efficient disease identification for farmers and researchers.

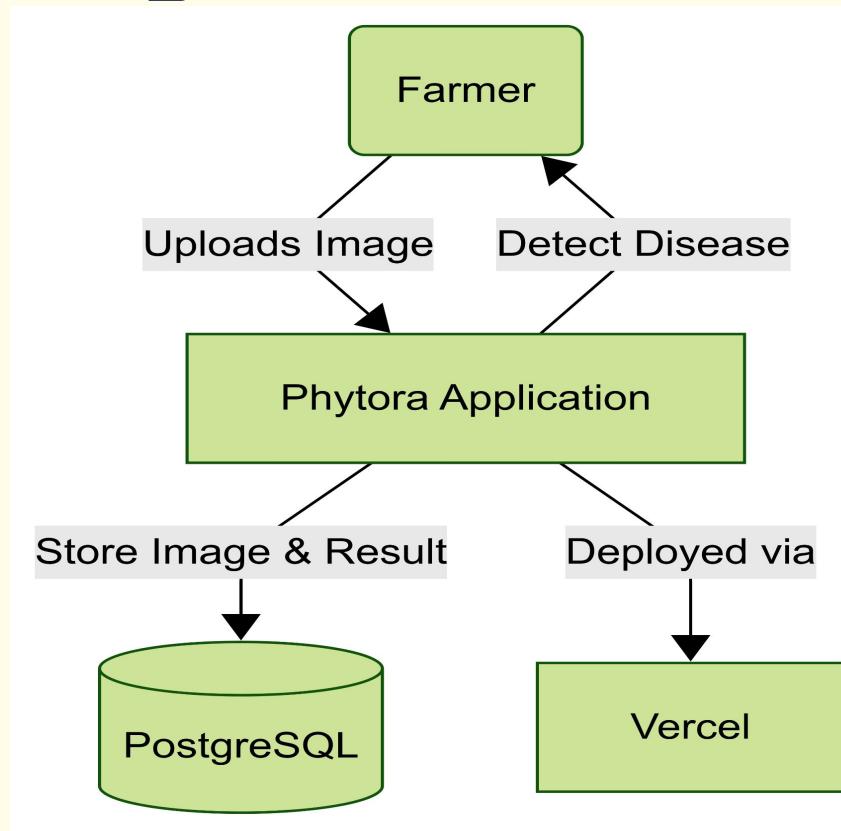
# 9. Diagrams



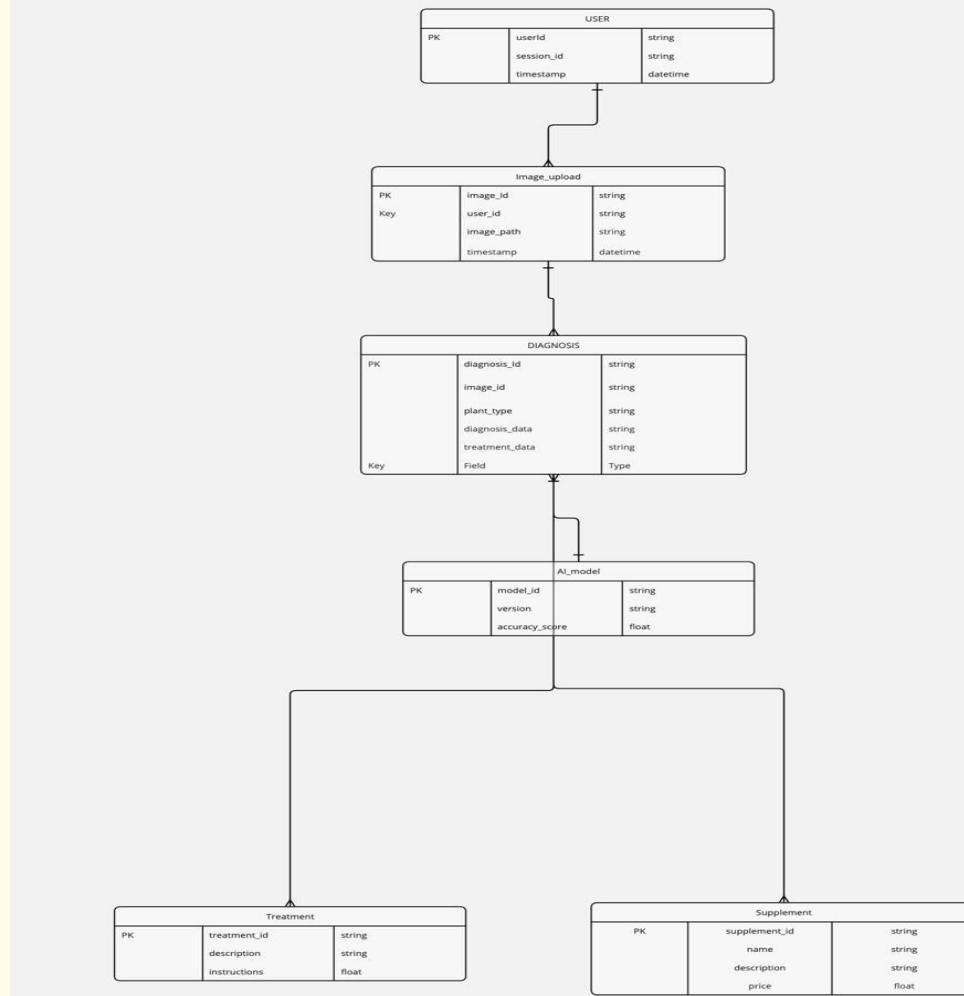
# Architecture Diagram



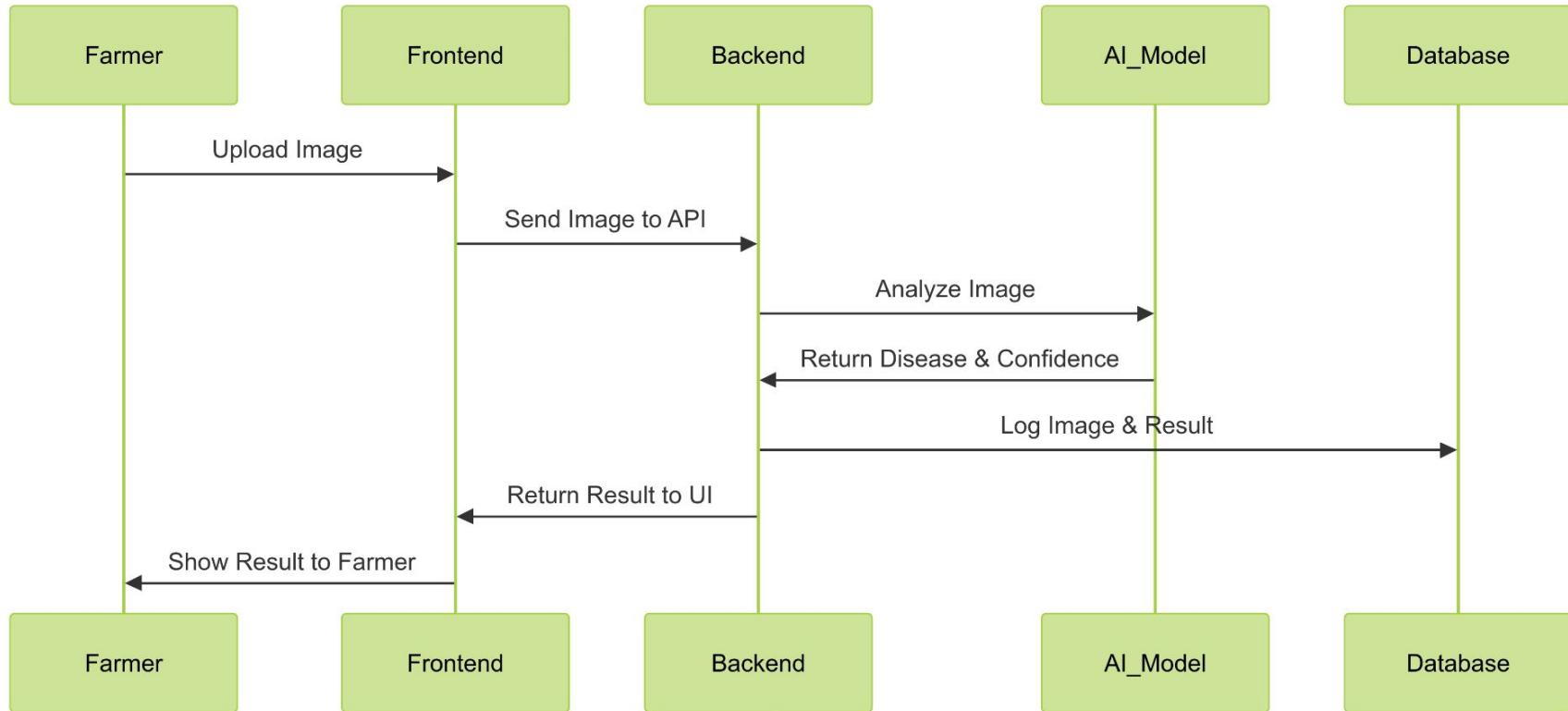
# Context Diagram



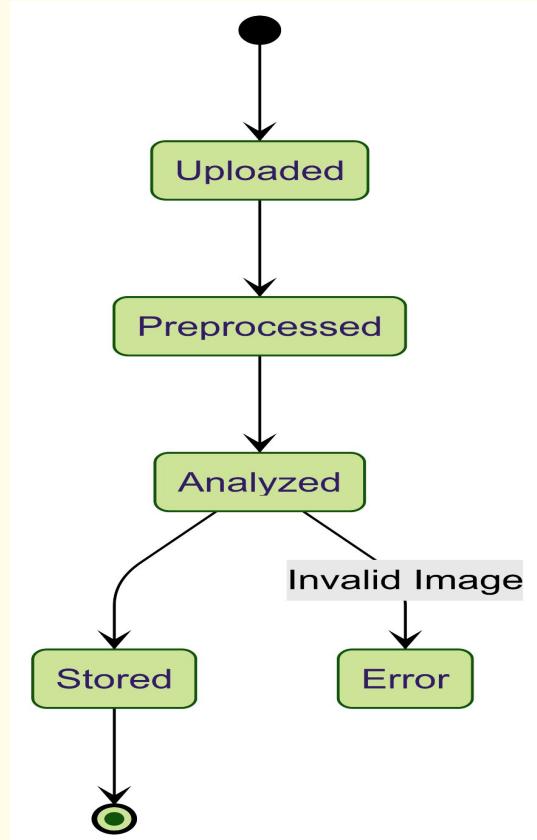
# ER Diagram



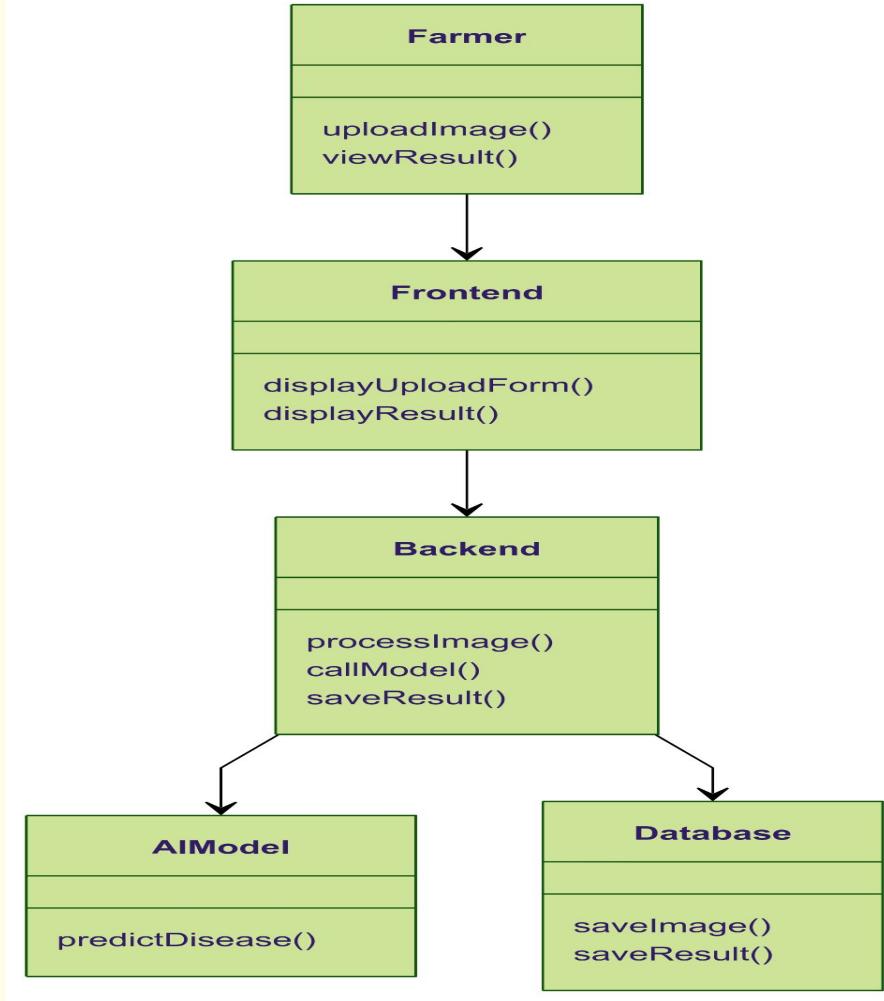
# Sequence Diagram



# State Diagram



# Class Diagram



# 10. Product Backlog



# Product Backlog

Sprint	ID	Story (User Story/Technical Story)	So That	User Benefit	Acceptance Criteria
Sprint 1	US1.1	As a farmer, I want to upload an image of my apple leaf	so that the system can analyze it for diseases	I can easily submit leaf images for disease detection	Image upload works from desktop and mobile; files saved to backend
Sprint 1	US1.2	As a farmer, I want the system to detect if my leaf is healthy or diseased	so that I can take action if needed	I get quick feedback on leaf health	System returns "Healthy" or "Diseased" after image upload
Sprint 1	TS1.1	Pre-process images to remove noise and resize for AI model	so that the AI model receives consistent input for reliable detection	Image quality is standardized for accurate detection	Images are resized and normalized before analysis
Sprint 1	TS1.2	Log uploaded images and results in PostgreSQL	so that we can track all scans for auditing and improvements	System can track all scans for future reference	Each entry saved with timestamp, image URL, and result

# Product Backlog

Sprint	ID	Story (User Story/Technical Story)	So That	User Benefit	Acceptance Criteria
Sprint 1	TS1.3	Set up CI/CD pipeline to deploy to Vercel	so that code changes automatically go live without manual deployment	Deployment is automated and reliable	Push to main triggers triggers deployment to Vercel
Sprint 2	US2.1	As a farmer, I want the system to identify the specific disease if my leaf is unhealthy	so that I know exactly what's wrong	I get clear diagnosis (scab, rust, etc.)	System classifies between scab and rust
Sprint 2	US2.2	As a farmer, I want to see a confidence score with the result	so that I know how reliable the result is	I understand how accurate the diagnosis is	Confidence score shown with each result
Sprint 2	US2.3	As a farmer, I want basic advice based on the detected disease	so that I know what to do next	I can take immediate corrective action	Text advice (e.g., "Apply fungicide") shown after detection

# Product Backlog

Sprint	ID	Story (User Story/Technical Story)	So That	User Benefit	Acceptance Criteria
Sprint 2	TS2.1	Validate and reject non-leaf images	so that the system doesn't waste resources analyzing invalid images	System only processes valid apple leaf images	Non-leaf images trigger clear error message
Sprint 2	TS2.2	Add test cases for all core APIs	so that future changes do not break existing functionality	System reliability improves and regressions are caught early	Each API (upload, detect, classify) has 3+ automated tests
Sprint 3	US3.1	As a farmer, I want to see past uploads and results	so that I can track disease history	I can monitor disease trends over time	History page shows past uploads with timestamps and results
Sprint 3	US3.2	As a farmer, I want to upload multiple images at once	so that I save time during scanning	I can submit several leaves in one go	Batch upload supports up to 5 images

# Product Backlog

Sprint	ID	Story (User Story/Technical Story)	So That	User Benefit	Acceptance Criteria
Sprint 3	US3.3	As a farmer, I want the app to work well on mobile	so that I can use it directly in the field	I can use the app comfortably from my phone	App fully responsive on mobile browsers
Sprint 3	TS3.1	Optimize AI model for faster detection	so that the user doesn't experience slow results	Disease detection is faster and smoother	Inference time is less than 3 seconds per image
Sprint 3	TS3.2	Log errors to PostgreSQL when detection fails	so that developers can quickly identify and resolve issues	Developers can debug issues faster	Each error log includes timestamp, image URL, and error message

# 11. Sprint 1 Backlog



# Sprint 1 Backlog

ID	Story (User Story/Technical Story)	So That	User Benefit	Acceptance Criteria	Story Points
US1.1	As a farmer, I want to upload an image of my apple leaf	so that the system can analyze it for diseases	I can easily submit leaf images for disease detection	Image upload works from desktop and mobile; files saved to backend	5
US1.2	As a farmer, I want the system to detect if my leaf is healthy or diseased	so that I can take action if needed	I get quick feedback on leaf health	System returns "Healthy" or "Diseased" after image upload	5
TS1.1	Pre-process images to remove noise and resize for AI model	so that the AI model receives consistent input for reliable detection	Image quality is standardized for accurate detection	Images are resized and normalized before analysis	3
TS1.2	Log uploaded images and results in PostgreSQL	so that we can track all scans for auditing and improvements	System can track all scans for future reference	Each entry saved with timestamp, image URL, and result	3
TS1.3	Set up CI/CD pipeline to deploy to Vercel	so that code changes automatically go live without manual deployment	Deployment is automated and reliable	Push to main triggers deployment to Vercel	2

# 12. Sprint 1 Test Cases



# Sprint 1 Test Cases

ID	Story	Test Case	Expected Result	Status
US1.1	Image Upload	Upload a valid apple leaf image	Image successfully uploads and stores in backend	Passed
US1.1	Image Upload	Upload an invalid file type (PDF/MP4)	System rejects file with error message	Passed
US1.2	Disease Detection	Upload image of healthy leaf	System returns "Healthy" result	Passed
US1.2	Disease Detection	Upload image of diseased leaf	System returns "Diseased" result	Passed
TS1.1	Image Pre-processing	Upload large image	Image is resized to predefined dimensions	Passed
TS1.2	Database Logging	Upload image and get result	Entry saved in PostgreSQL with timestamp and result	Passed
TS1.3	CI/CD	Push code to main branch	Vercel auto-deploys app successfully	Passed

# 13. Sprint 1 Stories Completed



# Sprint 1 Stories Completed

ID	Story	Status
US1.1	Image Upload	Completed
US1.2	Disease Detection	Completed
TS1.1	Image Pre-processing	Completed
TS1.2	Database Logging	Completed
TS1.3	CI/CD Setup	Completed

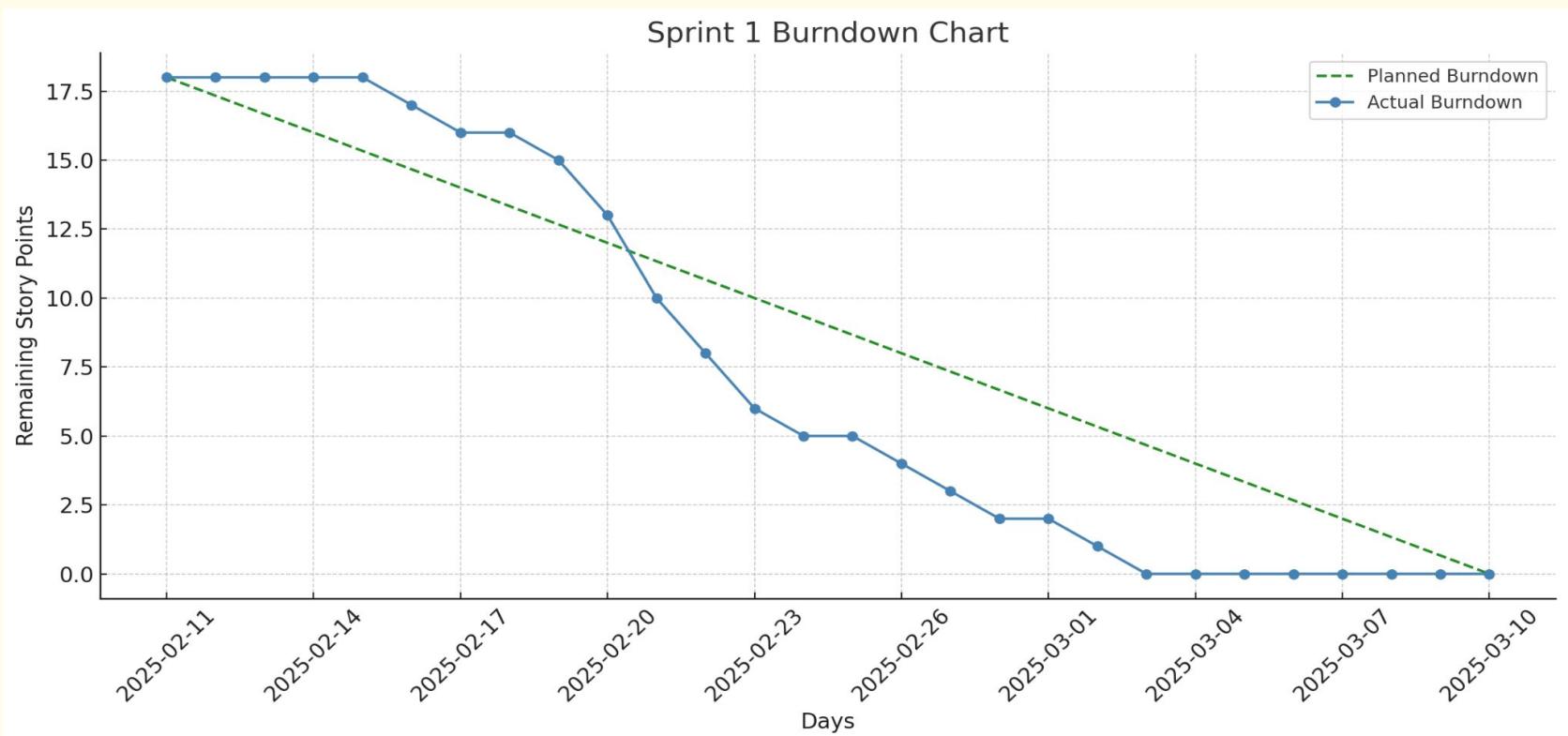
# Metrics

14

# Team Velocity

Total Story Points Completed: 18

# Burndown Chart Sprint 1



# 100%

## Completed/Committed Ratio

Committed Story Points: **18**

Completed Story Points: **18**

Completed/Committed Ratio: **18/18**

# Retrospective

15

start typing to filter stickies

View Section

All Sections

Sort By

created time

## Retrospective 1

## What went well

good understanding of the project goals  + 0	work load distribution  + 0
feedback went well  + 0	good understanding of the tools and the setup  + 0
Roles allocation  + 0	functional first version  + 0
user stories done perfectly  + 0	

## What can be improved

better Communication  + 0	More and clear clarification of the stories at the start of the sprint  + 0
better test cases and the inputs  + 0	long meetings  + 0
prioritizing the task activites  + 0	improving the story point estimation  + 0
more feedback for application improvement  + 0	Deadlines estimations  + 0

## Action Items

more test case  + 0	shorter meeting  + 0
better planning  + 0	feedback from the user  + 0
conducting risk management to avoid obstacles  + 0	

# 16. Sprint 2 Planning



# Sprint 2 Planning

ID	Story (User Story/Technical Story)	So That	Acceptance Criteria	Story Points
US2.1	As a farmer, I want the system to identify the specific disease if my leaf is unhealthy	so that I know exactly what's wrong	System classifies between scab and rust	5
US2.2	As a farmer, I want to see a confidence score with the result	so that I know how reliable the result is	Confidence score shown with each result	3
US2.3	As a farmer, I want basic advice based on the detected disease	so that I know what to do next	Text advice (e.g., "Apply fungicide") shown after detection	3
TS2.1	Validate and reject non-leaf images	so that the system doesn't waste resources analyzing invalid images	Non-leaf images trigger clear error message	3
TS2.2	Add test cases for all core APIs	so that future changes do not break existing functionality	Each API (upload, detect, classify) has 3+ automated tests	8

# 17. Project Demo



# Screenshots of Application

The screenshot displays the Phytora application interface, which includes a header bar, a main content area with sections for recent scans and a dashboard, and a footer.

**Header Bar:** The top navigation bar shows the URL "localhost:3000". It includes standard browser controls (back, forward, search) on the left, and a set of icons on the right: a download arrow, a shield, a triangle, a magnifying glass, a file, a star, a lock, and a menu icon.

**Main Content Area:**

- Welcome to Phytora:** A large heading followed by a subtext: "Monitor and protect your apple trees with our AI-powered disease detection system." Below this are two buttons: "Scan Now" (green rounded rectangle) and "View Reports" (white button with green border).
- Your Orchard Health Dashboard:** A light green rectangular box containing the text "Your Orchard Health Dashboard".
- Recent Scans:** A section titled "Recent Scans" featuring three placeholder cards for recent scans.

**Footer:** A small circular icon in the bottom-left corner contains the letter "N".

# Screenshots of Application

P Phytora

Home Detection About Diseases Contact

## Leaf Disease Detection

[Back to Dashboard](#)

Upload an image of your apple leaf for AI-powered disease detection.



Click to select an image or drag and drop here

Supports JPG and PNG

Analyze Leaf

# Screenshots of Application

Upload an image of your apple leaf for AI-powered disease detection.



Clear

Analyze Leaf

# Screenshots of Application

P Phytora

Home Detection About Diseases Contact

## Leaf Disease Detection

[Back to Dashboard](#)

Upload an image of your apple leaf for AI-powered disease detection.



⚠️ Diseased Leaf

[Scan Another Leaf](#)

[Save Report](#)

N

# 18. Wikipage



**[https://github.com/htmw/2025  
S-SALAAR/wiki](https://github.com/htmw/2025-S-SALAAR/wiki)**

# 19. Demo



# Thank

# You

