Phytora

By Team 3 SALAAR

SPRINT 2

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
- Sprint 1 Recap
- Product Backlog
- Sprint 2 Backlog
- Sprint 2 Test Cases
- Stories Completed (Sprint 2)
- Stories Not Completed (Sprint 2 Carry Over)
- Metrics

- Retrospective
- Sprint 3 Planning
- Project Demo - Sprint 2
- GitHub Link & Live Application Demo

Team Member Roles and Responsibilities

01

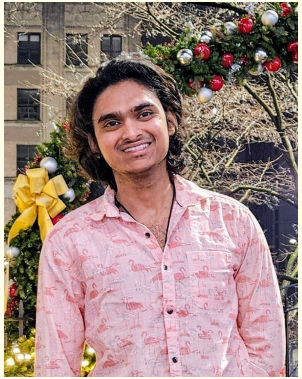
Team Members

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:	**For** farmers and agricultural researchers **who** need an efficient and cost-effective method to detect apple tree diseases**,** **the** AI-based plant disease detection system **is a** deep learning-powered solution **that** identifies diseases like scab and rust early, preventing crop loss**.** **Unlike** traditional manual inspection methods**,** **our application** leverages Convolutional Neural Networks (CNNs) to provide fast, accurate, and automated disease classification, overcoming challenges like background noise and variable disease appearances**.**
Benefit Outcomes:	• **Early Detection**: Enables quick identification of apple tree diseases, reducing yield loss. • **Cost-Effective**: Eliminates the need for frequent manual inspections, saving labor costs. • **High Accuracy**: Uses AI and deep learning models to minimize misdiagnosis. • **Scalability**: Can be applied to large farms and integrated into existing agricultural systems. • **User-Friendly**: Provides a simple interface for farmers to upload images and get instant results.
Github Link:	https://github.com/htmw/2025S-SALAAR/wiki

3. Improvements

Improvements

Change in the state diagram

Removal of user benefits

Team Working Agreement

04

Team Working Agreement – Phytora

1. Purpose & Scope

This agreement establishes collaboration guidelines for the development of Phytora 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 Phytora.

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

- **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

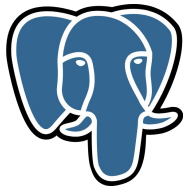
After analysis, the application displays the disease name and confidence score in real-time for the user.

Technologies

07

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

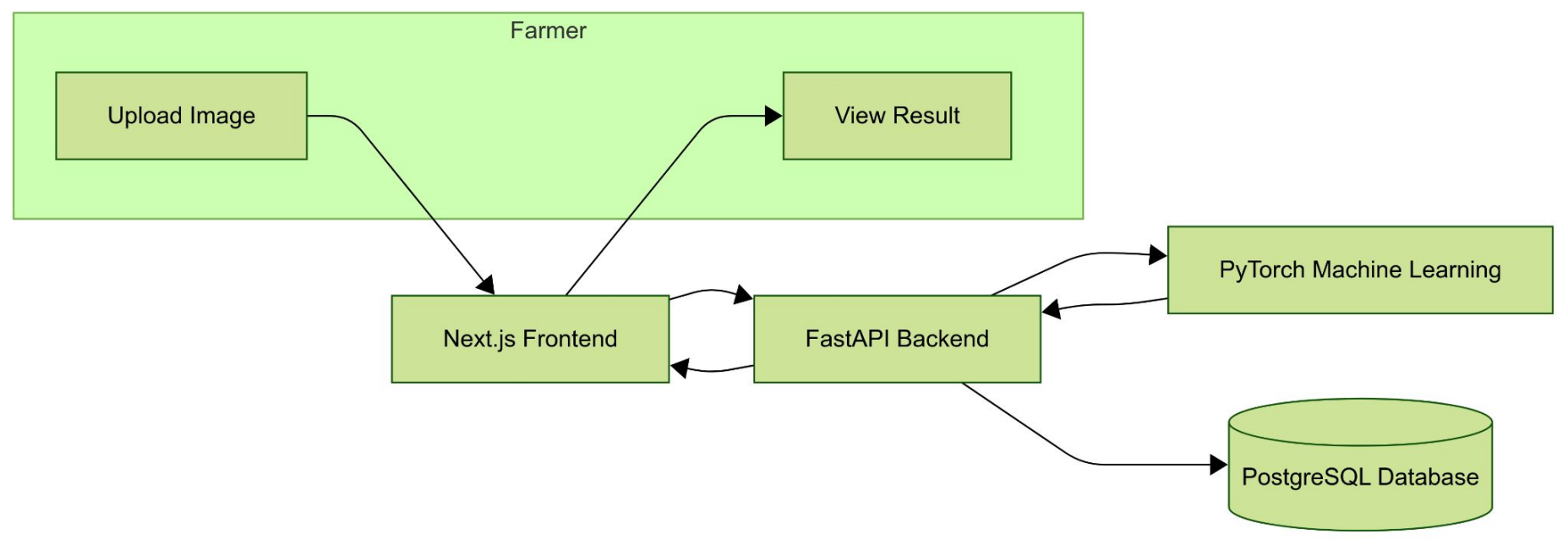
08

Algorithms

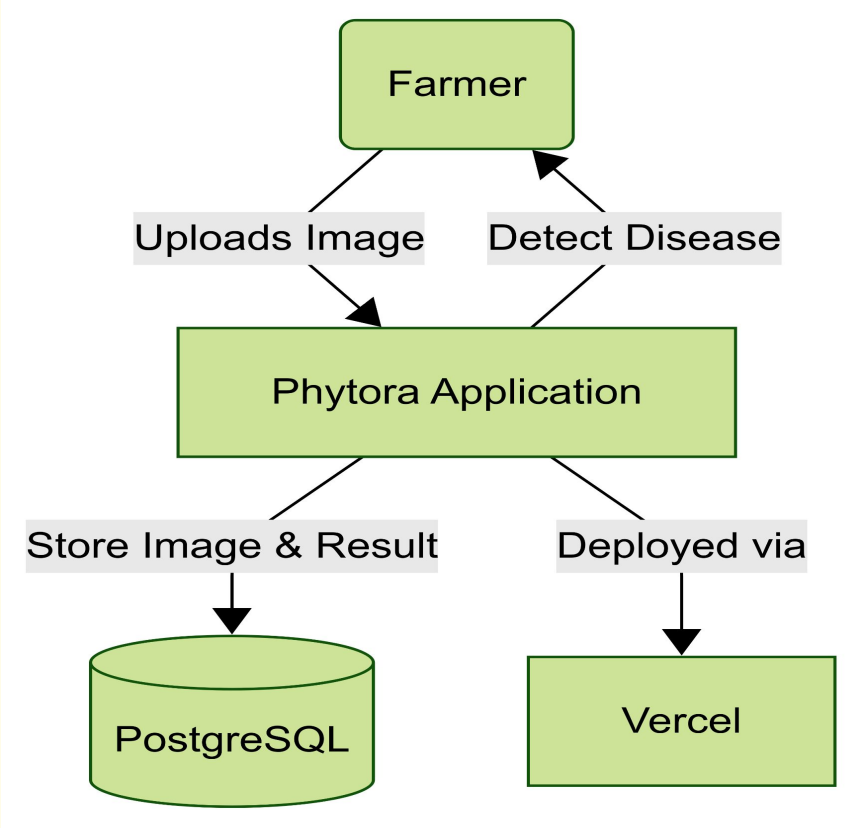
Our system utilizes EfficientNet, a state-of-the-art CNN model, for highly accurate and optimized plant disease detection. EfficientNet balances speed and accuracy by scaling depth, width, and resolution efficiently. Transfer learning enhances performance, while FastAPI enables real-time predictions,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

9. Diagrams

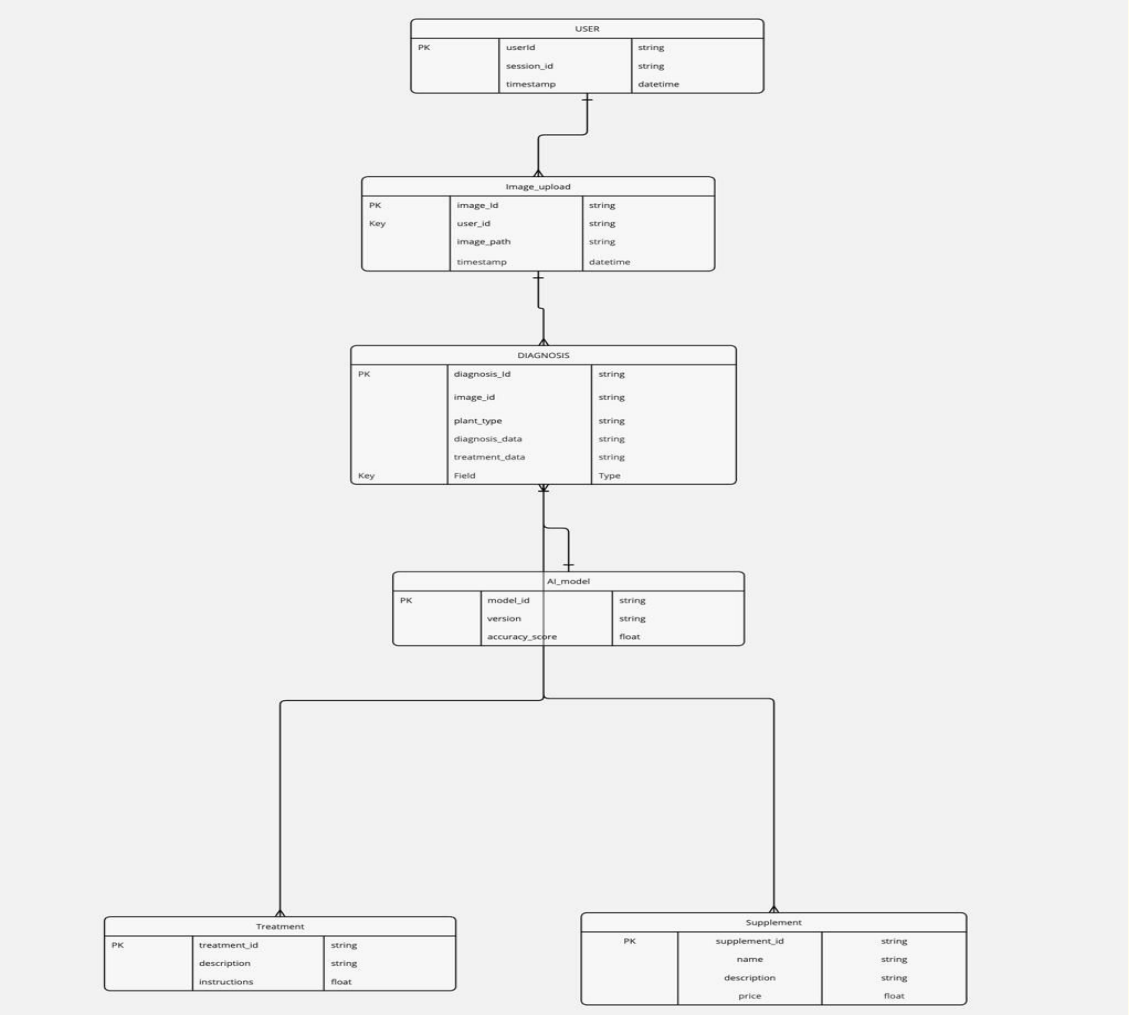
Architecture Diagram



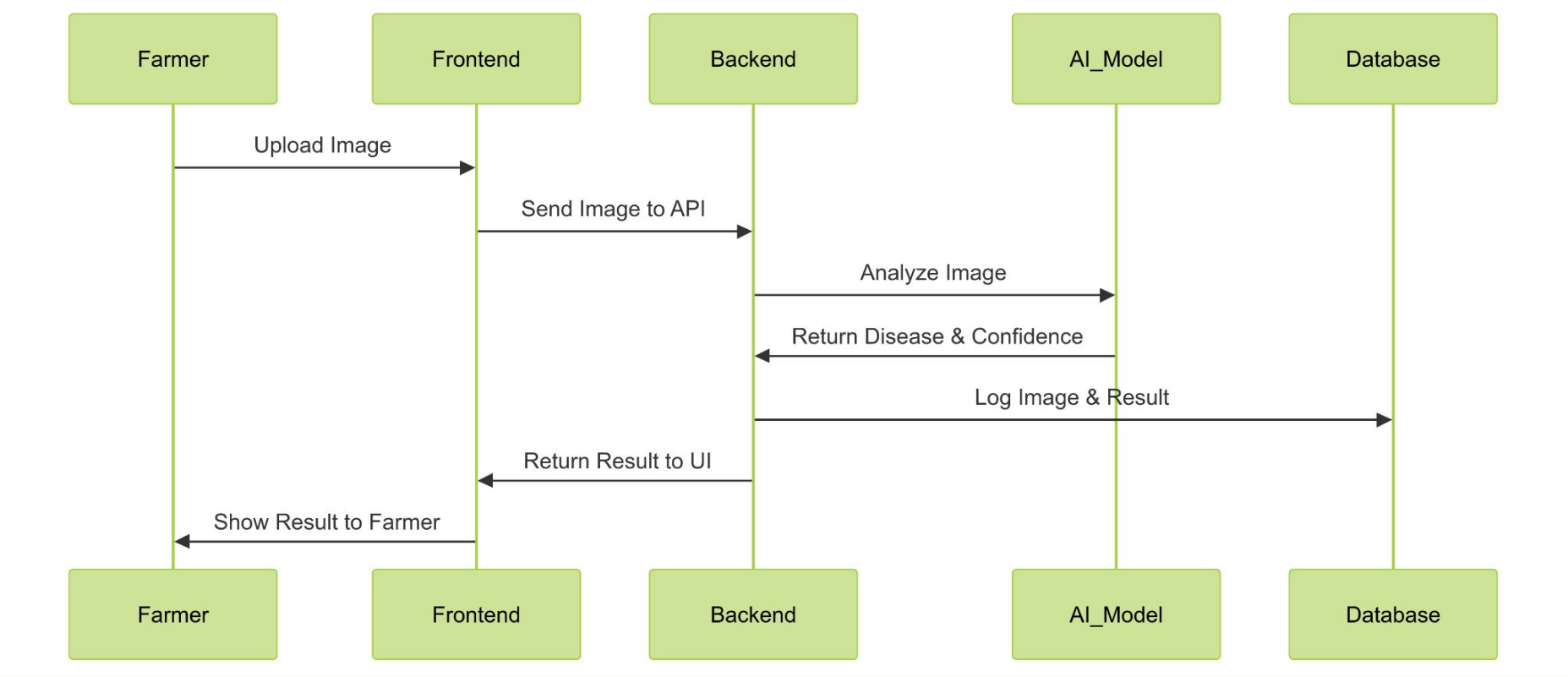
Context Diagram



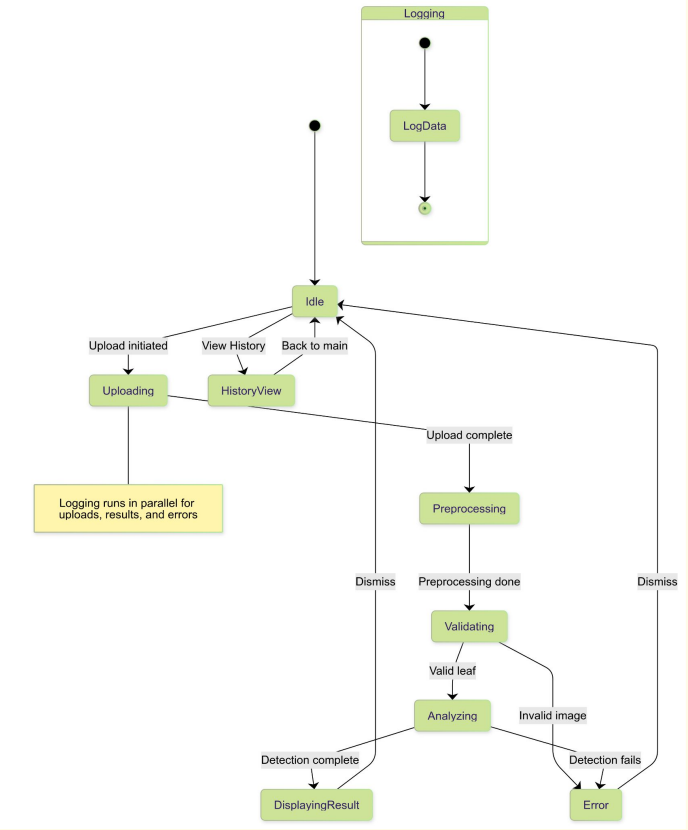
ER Diagram


USER
PK
userId
session_id
timestamp
string
string
datetime
Image_upload
PK
Key
image_id
user_id
image_path
timestamp
string
string
string
datetime
DIAGNOSIS
PK
Key
diagnosis_id
image_id
plant_type
diagnosis_data
treatment_data
Field
string
string
string
string
string
Type
AI_model
PK
model_id
version
accuracy_score
string
string
float
Treatment
PK
treatment_id
description
instructions
string
string
float
Supplement
PK
supplement_id
name
description
price
string
string
string
float

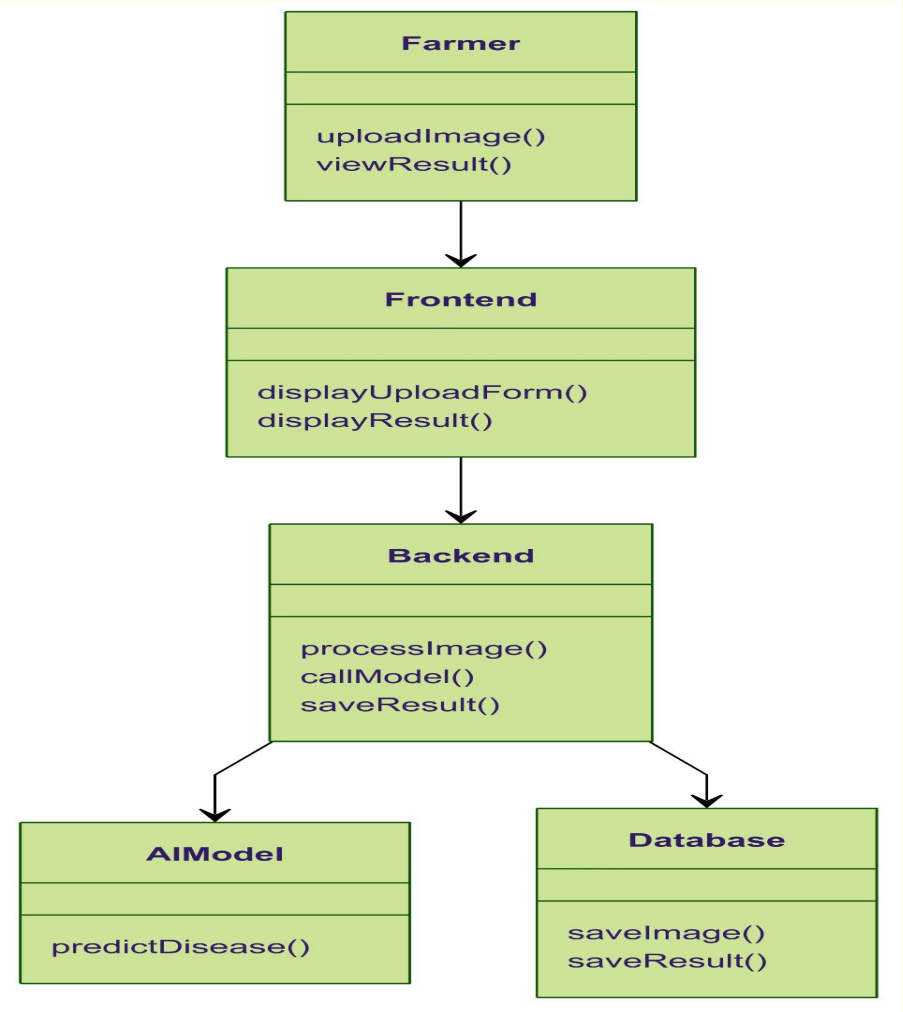
Sequence Diagram



State Diagram



Class Diagram



10.Recap

Sprint 1 Recap

In the previous sprint, basic user interface was developed and along with it, basic classification is done, just detecting either it is diseases or not and no confidence score.

Section | 12

11.Product Backlog

Product Backlog

Sprint	ID	Story (User Story/Technical Story)	So That	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	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	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	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	Each entry saved with timestamp, image URL, and result

Product Backlog

Sprint	ID	Story (User Story/Technical Story)	So That	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	Push to main 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	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	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	Text advice (e.g., "Apply fungicide") shown after detection

Product Backlog

Sprint	ID	Story (User Story/Technical Story)	So That	Acceptance Criteria
Sprint 2	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
Sprint 2	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
Sprint 3	US3.1	As a farmer, I want to see past uploads and results	so that I can track disease history	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	Batch upload supports up to 5 images

Product Backlog

Sprint	ID	Story (User Story/Technical Story)	So That	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	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	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	Each error log includes timestamp, image URL, and error message

12. Sprint 2 Backlog

Sprint 2 Backlog

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

13. Sprint 2 Test Cases

Sprint 2 Test Cases

ID	Story	Test Case	Expected Result	Status
US2.1	Disease Identification	Upload image of leaf with scab	System classifies disease as “Scab”	Passed
US2.1	Disease Identification	Upload image of leaf with rust	System classifies disease as “Rust”	Passed
US2.2	Confidence Score	Upload diseased leaf image	Result includes confidence score (e.g., 87%)	Passed
US2.2	Confidence Score	Upload ambiguous leaf image	System still returns a confidence score	Passed
US2.3	Advice Generation	Upload scab-affected leaf	Advice text shown: “Apply fungicide”	Passed
US2.3	Advice Generation	Upload rust-affected leaf	Advice text shown: “Use resistant variety”	Passed
TS2.1	Non-leaf Image Validation	Upload image of a hand	System shows error: “Invalid input. Please upload a leaf image”	Passed

Sprint 2 Test Cases

ID	Story	Test Case	Expected Result	Status
TS2.1	Non-leaf Image Validation	Upload image of a building	System shows error: “Invalid input. Please upload a leaf image”	Passed
TS2.2	API Test Coverage	Run unit tests on upload, detect, classify APIs	3+ tests per API pass	Passed
TS2.2	API Test Coverage	Add new endpoint and rerun tests	All existing tests pass (no regression)	Passed


14. Sprint 2 Stories
Completed

Sprint 2 Stories Completed

ID	Story	Status
US2.1	Disease Identification	Completed
US2.2	Confidence Score	Completed
US2.3	Advice Generation	Completed
TS2.1	Non-leaf Image Validation	Completed
TS2.2	API Test Coverage	Completed

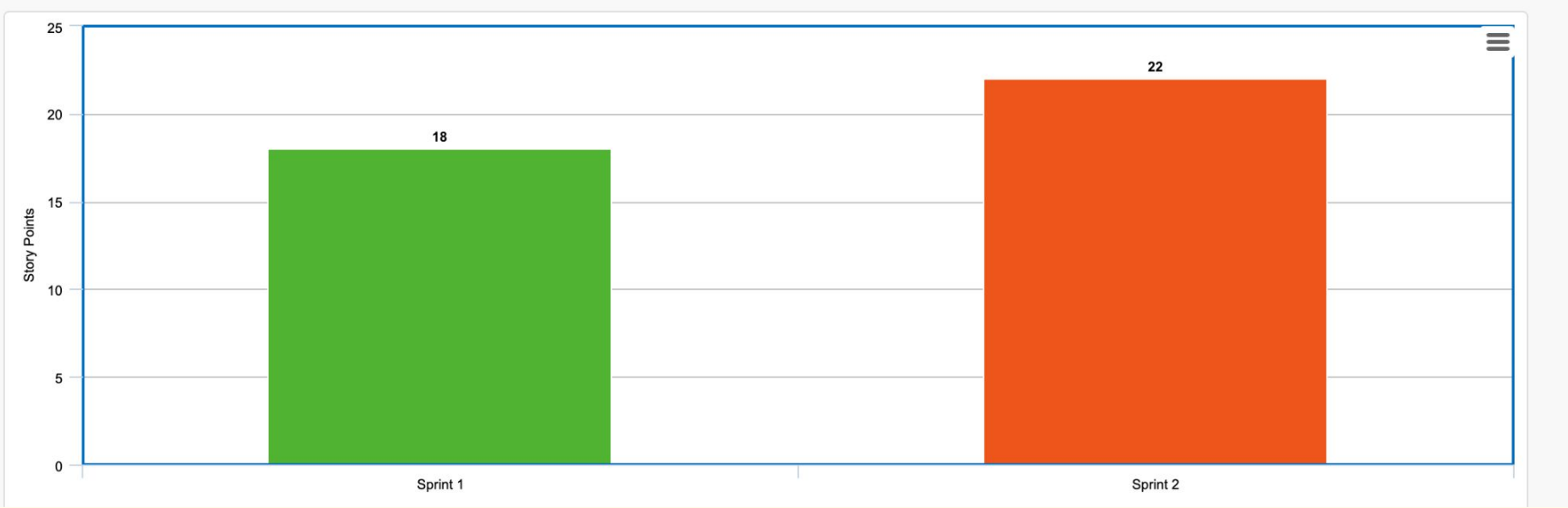
Metrics

15

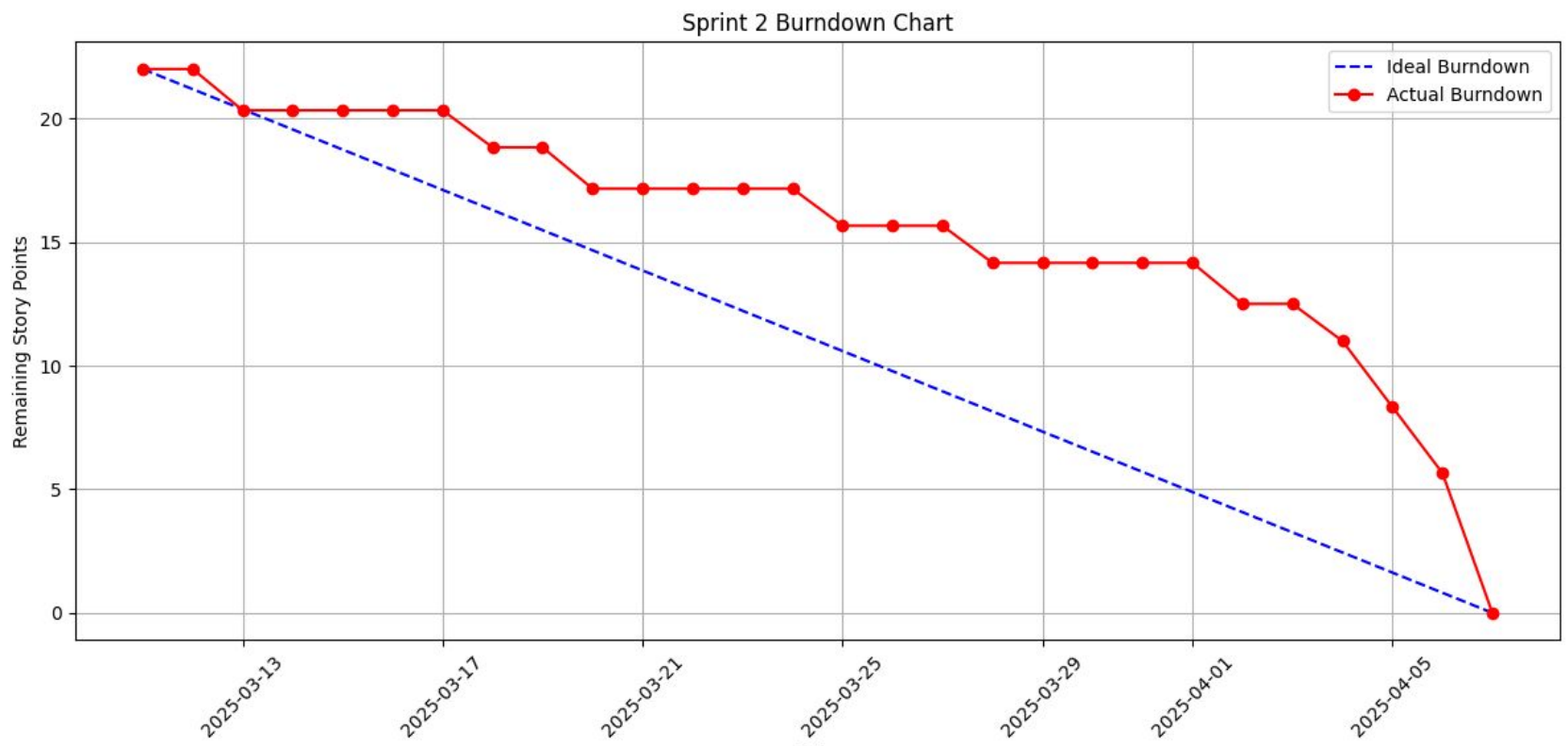
Team Velocity

Total Story Points Completed: 22

Historical Team Velocity



Burndown Chart Sprint 2



100%

Completed/Committed Ratio

Committed Story Points: **22**
Completed Story Points: **22**
Completed/Committed Ratio: **22/22**

Retrospective

16

Retrospective 2

CAPSTONE PROJECT

What went well

- Bugs got crushed across the board in both front end and back end (+0)
- Machine learning Predictions are going well with the project (+0)
- Deliver before the deadline (+0)
- allocation points well in this sprint (+0)
- Daily standup Calls (+0)
- Backend held strong under load (+0)
- Team collabaraton and commitment (+0)
- fun meetings to reduce stress (+0)

What can be improved

- Machine learning Trainng takes more time predict the outcome (+0)
- Frontend changes need to update quickly (+0)
- backend tweaks to get better response time (+0)
- Clear the tracking system (+0)
- Tasks should be allocated to multiple people (+0)

Action Items

- Split workload of frontend, backend, Machine learning Early to avoid collsion between tasks (+0)
- Keep One Proper Document for everyone to see the progress,updates and notes (+0)
- Need to add more days to testing cases and polish everyting (+0)
- Need to look for some build tools to make the production deployment easy (+0)
- Trail demo meeting after frequent intervals to get opnion for updates and feedbacks (+0)

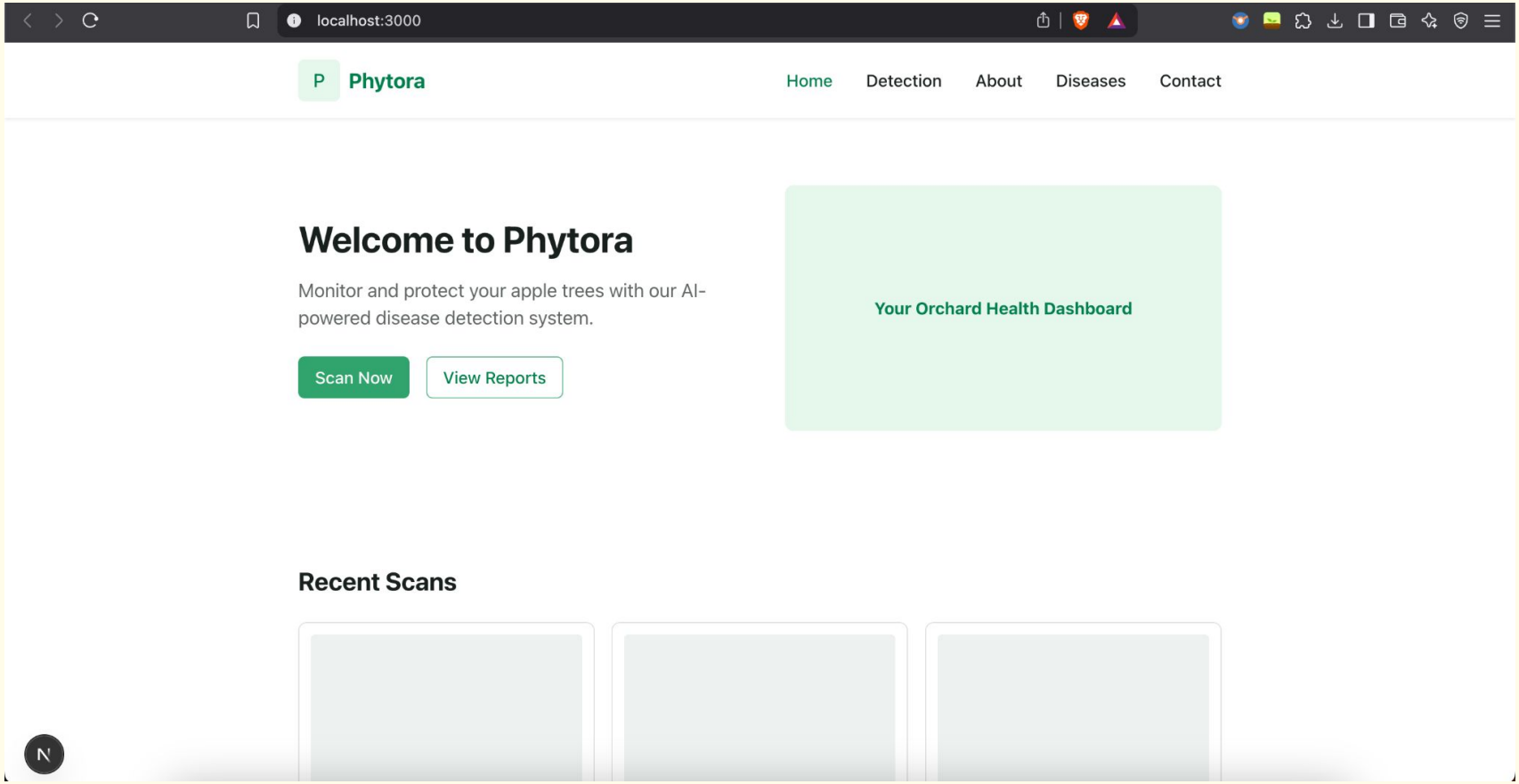
17. Sprint 3 Planning

Sprint 3 Planning

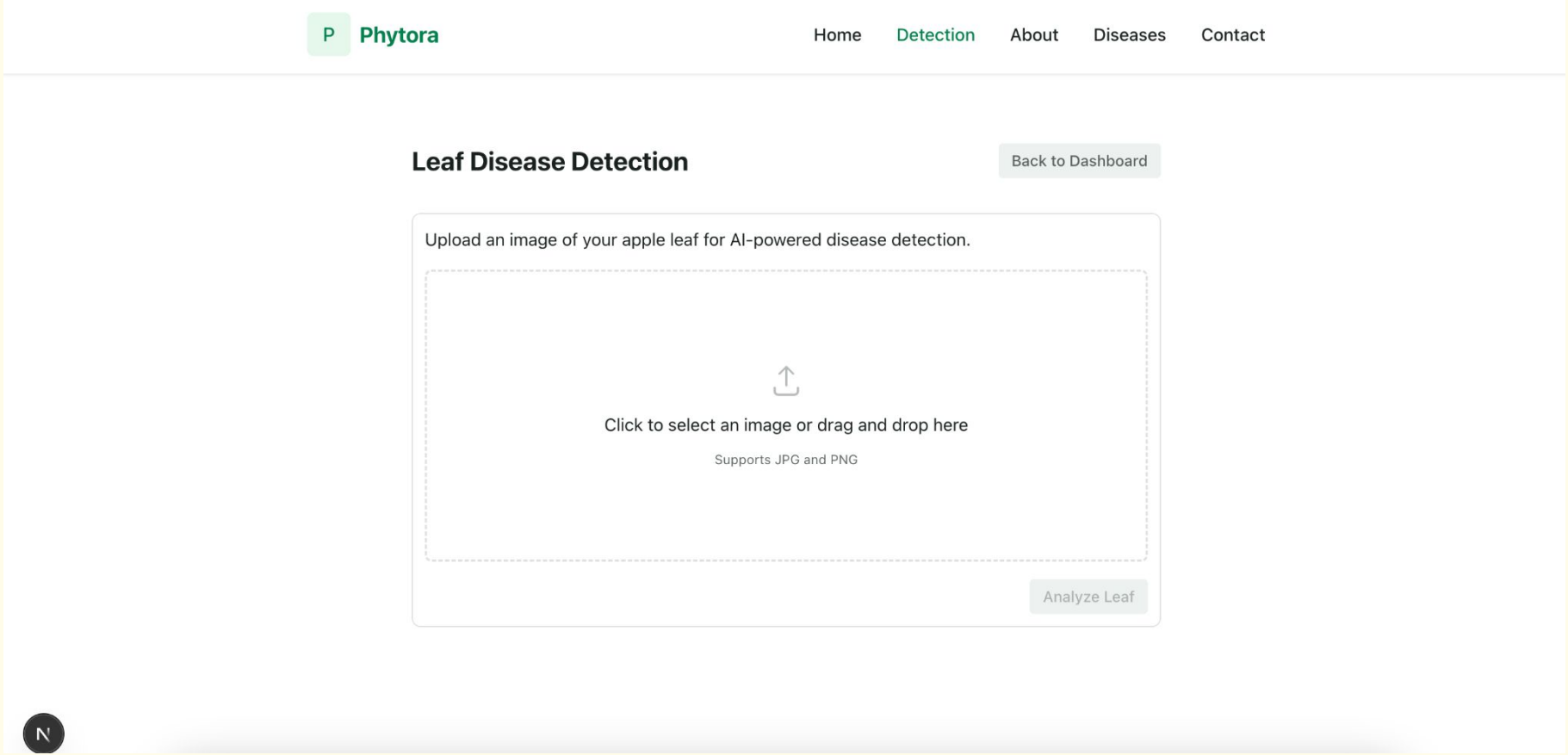
ID	Story (User Story/Technical Story)	So That	Acceptance Criteria	Story Points
US3.1	As a farmer, I want to see past uploads and results	so that I can track disease history	History page shows past uploads with timestamps and results	5
US3.2	As a farmer, I want to upload multiple images at once	so that I save time during scanning	Batch upload supports up to 5 images	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	App fully responsive on mobile browsers	3
TS3.1	Optimize AI model for faster detection	so that the user doesn't experience slow results	Inference time is less than 3 seconds per image	8
TS3.2	Log errors to PostgreSQL when detection fails	so that developers can quickly identify and resolve issues	Each error log includes timestamp, image URL, and error message	3


18. Project Demo

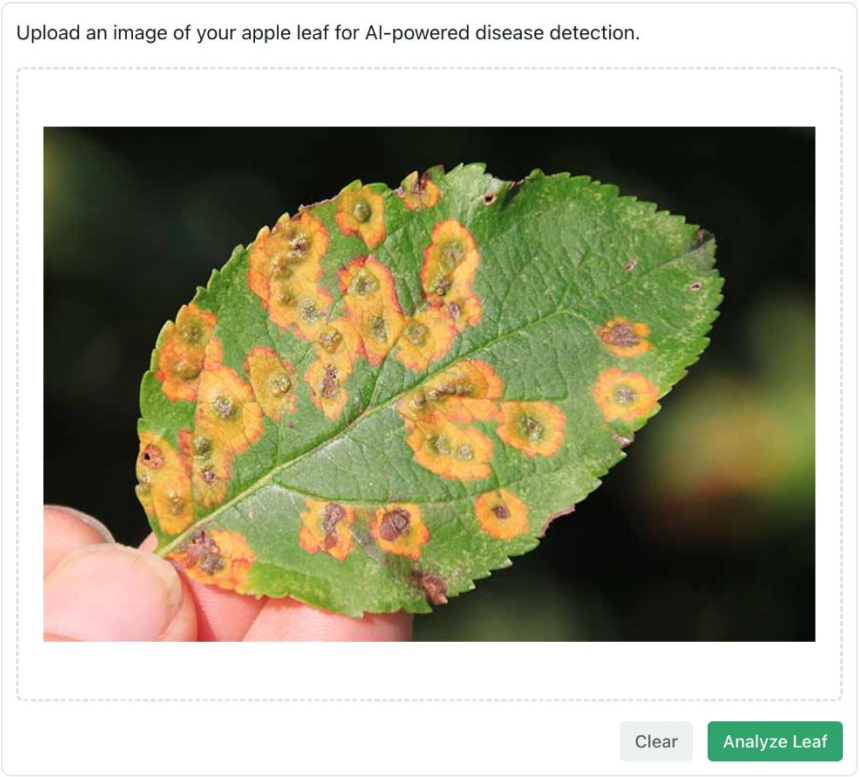
Screenshots of Application



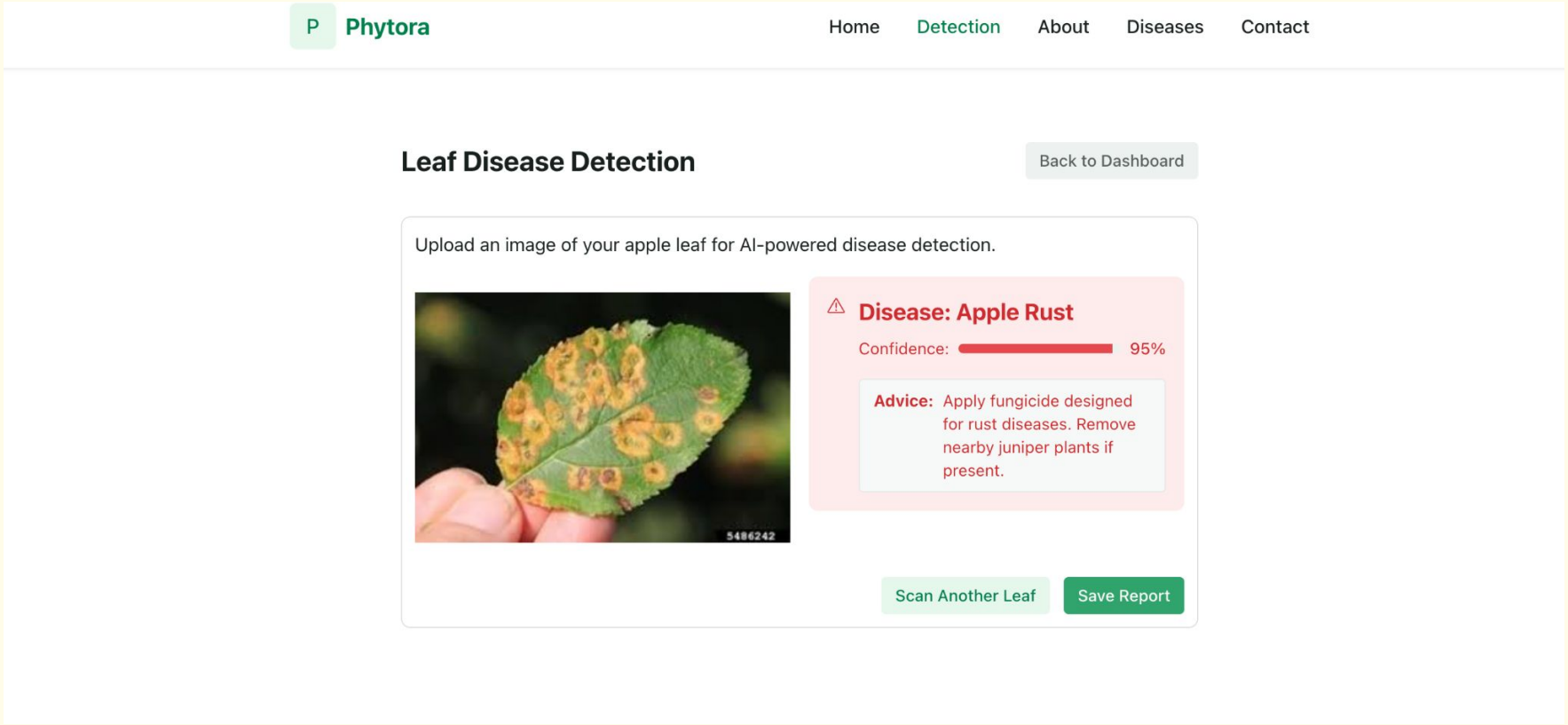
Screenshots of Application



Screenshots of Application



Screenshots of Application



19. Wikipage

https://github.com/htmw/2025S-SALAAR/wiki

20. Demo


Thank
You