POTATO DISEASE CLASSIFICATION

INTRODUCTION

Accurately identifying diseases from crop images using computer vision techniques like convolutional neural networks (CNNs) enables automated and rapid diagnostics for farmers. Specifically for potatoes, diseases like early and late blight can spread quickly and severely reduce yields if undetected. Machine learning powered image recognition provides a scalable solution for consistent and prompt disease identification to limit losses.

PURPOSE

The purpose of this project is to develop an accessible web-based application powered by a CNN model that can accurately classify key potato diseases from leaf images. By providing instant automated diagnostics, the system aims to help farmers identify and mitigate diseases early to reduce crop losses. Limiting disease impact promotes sustainability through food security and environmental protection by reducing unnecessary pesticide use.

EXISTING PROBLEM

A persistent challenge with crop disease classification models involves maintaining reliable accuracy across diverse real-world images. Factors like lighting conditions, image angles, potato varieties, age of plants and more can vary significantly to impact correctness. Additionally, early stages of diseases often manifest in subtle symptoms that are easy to miss. Capturing these nuanced visual signals indicative of initial disease onset is crucial for prompt intervention yet highly complex.

REFERENCES

1. "Deep Learning for Plant Disease Detection and Diagnosis" by Amara et al. (2021) - Provides overview of deep learning techniques for plant disease recognition.
2. "Toward the Early Diagnosis of Potato Disease Using Convolutional Neural Networks" by Sa et al. (2020) - Evaluates CNN models for early detection of potato diseases.
3. "Recognition and Classification of Potato Leaf Diseases Using Optimized Deep Convolutional Neural Network" by Lama et al. (2021)- Proposes an optimized CNN architecture for multi-disease potato classification.
4. "Explainability in Plant Disease Classification Using Deep Learning" by Amara et al. (2022) - Discusses model interpretation to understand predictions.

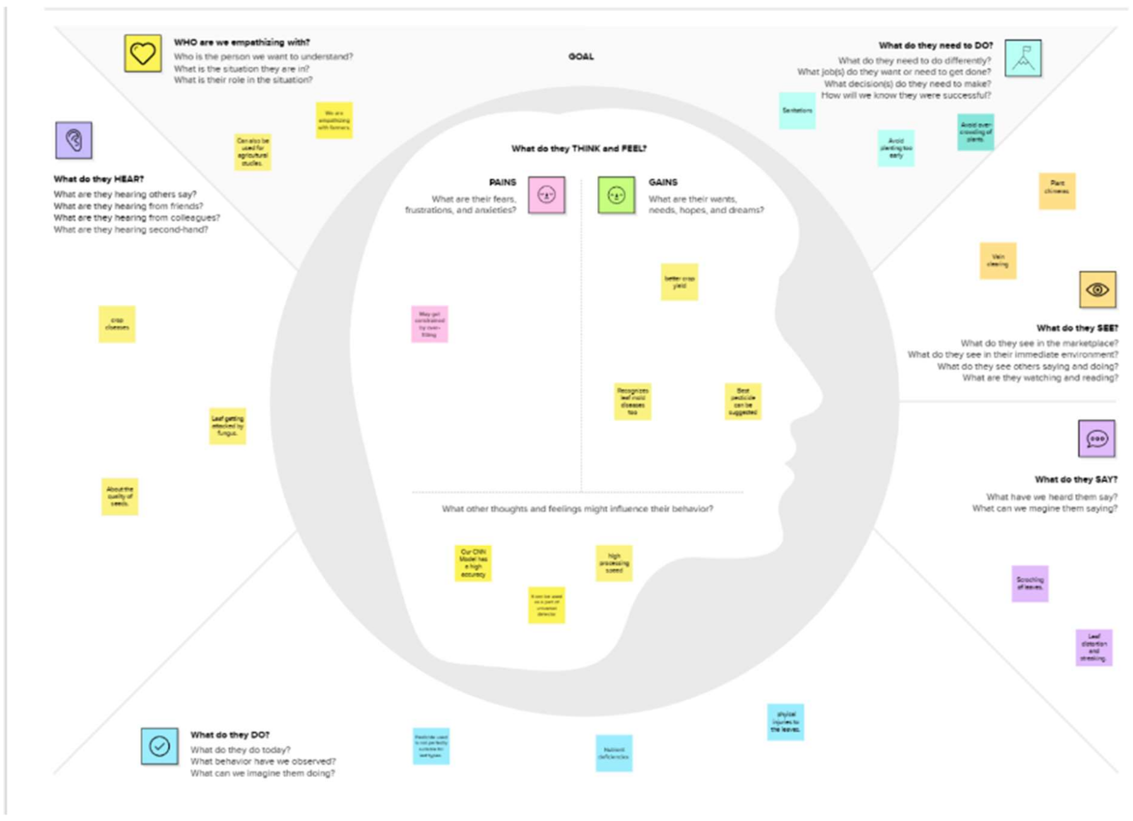
PROBLEM STATEMENT DEFINITION

The core problem aims to develop robust deep learning models, specifically convolutional neural networks, that can accurately classify multiple potato plant diseases using leaf images under diverse real-world agricultural environments. A particular focus lies in early detection of onset of infections when visual disease symptoms tend to be nuanced and subtle. Reliable and early diagnosis can enable prompt disease intervention by farmers to minimize crop losses and reduce unnecessary pesticide usage, promoting agricultural sustainability.

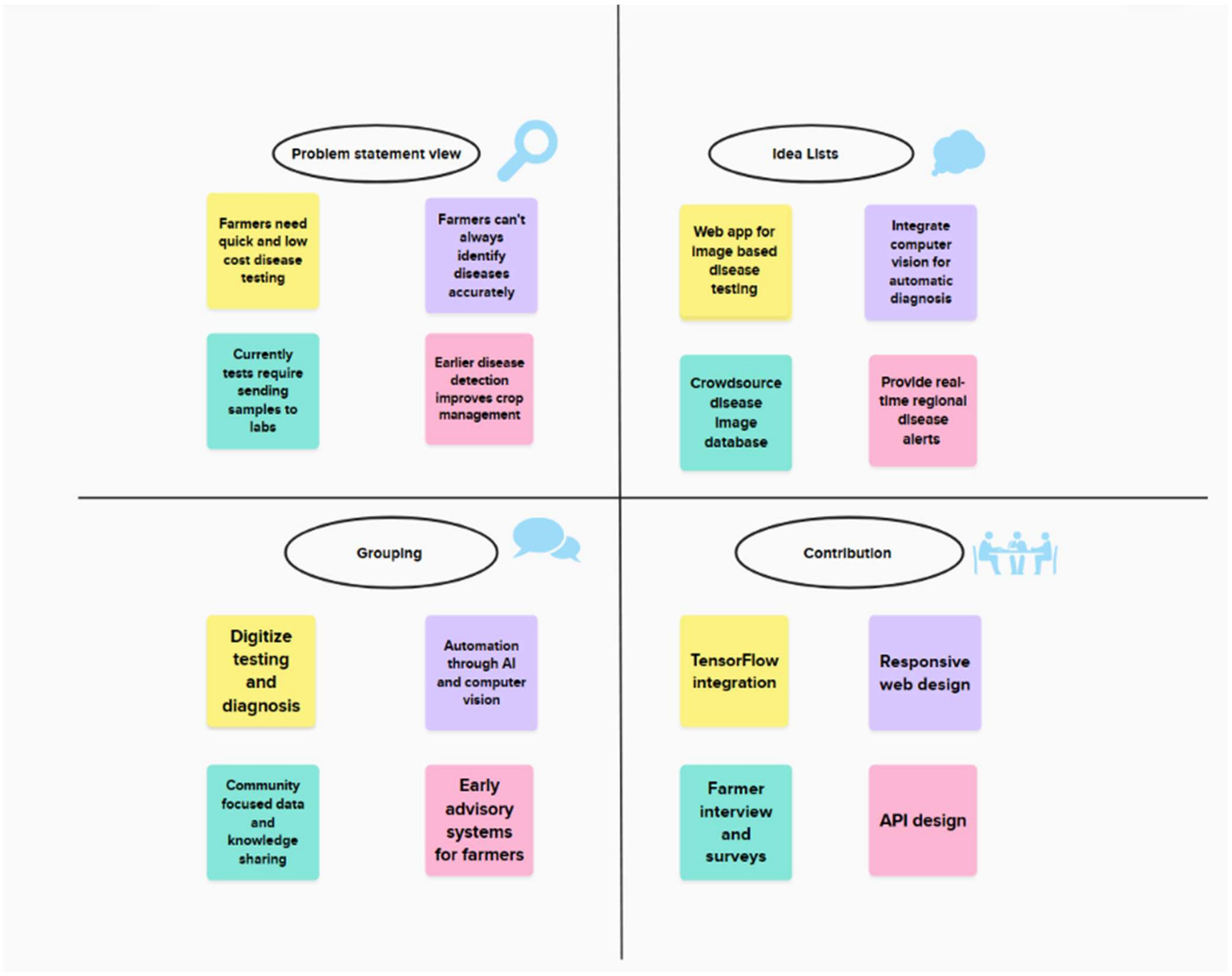
However, complexities arise due to variations in plant age, type, imaging factors leading to incorrectly classified predictions. The model must learn to discriminate between minute disease symptoms versus natural color variations. Capturing the onset of disease visually is non-trivial yet valuable. The key goal remains creating highly accurate AI diagnostic tools for farmers needing simple, explainable, and correct recommendations for prompt action given constraints of real-world field conditions.

IDEATION AND PROPOSED SOLUTION

Empathy Map and Canvas


WHO are we empathizing with?
Who is the person we want to understand?
What is the situation they are in?
What is their role in the situation?
We are empathizing with farmers.
Can also be used for agricultural studies.
GOAL
What do they need to DO?
What do they need to do differently?
What job(s) do they want or need to get done?
What decision(s) do they need to make?
How will we know they were successful?
Sanitations
Avoid planting too early
Avoid over crowding of plants
What do they HEAR?
What are they hearing others say?
What are they hearing from friends?
What are they hearing from colleagues?
What are they hearing second-hand?
crop diseases
Leaf getting attacked by fungus.
About the quality of seeds.
What do they THINK and FEEL?
PAINS
What are their fears, frustrations, and anxieties?
May get constrained by over-fitting
GAINS
What are their wants, needs, hopes, and dreams?
better crop yield
Recognizes leaf mold diseases too
Best pesticide can be suggested
What other thoughts and feelings might influence their behavior?
Our CNN Model has a high accuracy
high processing speed
It can be used as a part of universal detector
Plant chimeras
Vein clearing
What do they SEE?
What do they see in the marketplace?
What do they see in their immediate environment?
What do they see others saying and doing?
What are they watching and reading?
What do they SAY?
What have we heard them say?
What can we magine them saying?
Scorching of leaves.
Leaf distortion and streaking.
What do they DO?
What do they do today?
What behavior have we observed?
What can we imagine them doing?
Pesticide used is not perfectly suitable for seedlings.
Nutrient deficiencies
physical injuries to the leaves.

Ideation and Brainstorming


Problem statement view
Farmers need quick and low cost disease testing
Farmers can't always identify diseases accurately
Currently tests require sending samples to labs
Earlier disease detection improves crop management
Idea Lists
Web app for image based disease testing
Integrate computer vision for automatic diagnosis
Crowdsource disease image database
Provide real-time regional disease alerts
Grouping
Digitize testing and diagnosis
Automation through AI and computer vision
Community focused data and knowledge sharing
Early advisory systems for farmers
Contribution
TensorFlow integration
Responsive web design
Farmer interview and surveys
API design

REQUIREMENT ANALYSIS

Functional Requirement

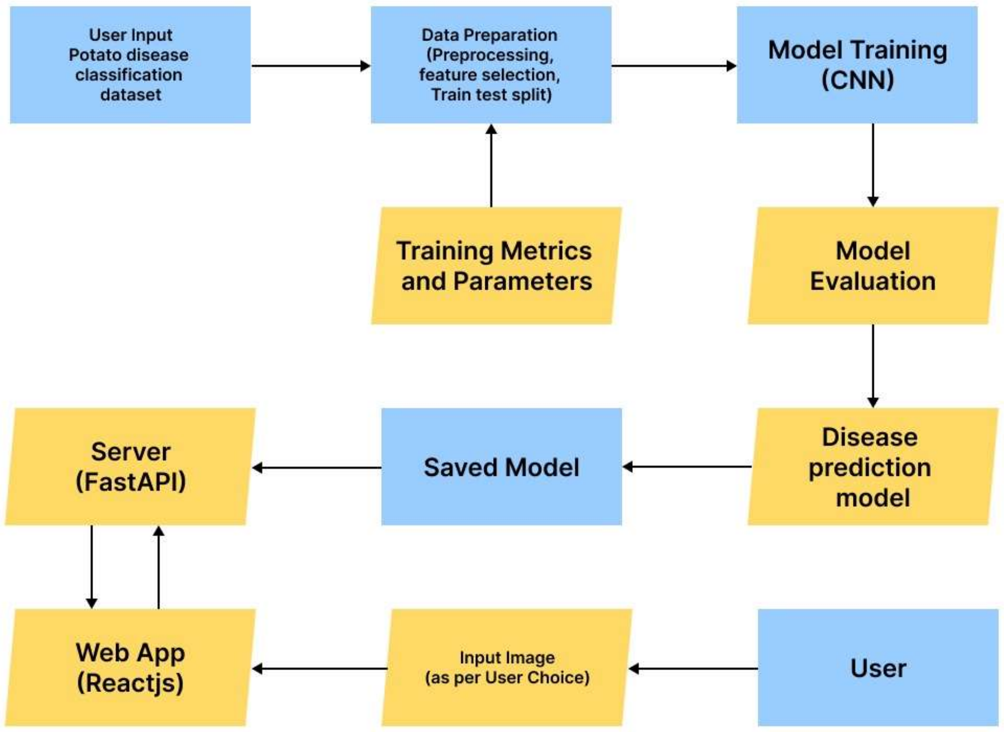
- Accurate disease prediction with sensitivity to early symptoms
- Handling varying real-world potato leaf images
- Interpretability of model prediction outputs
- Intuitive mobile and web interface

Non Functional Requirement

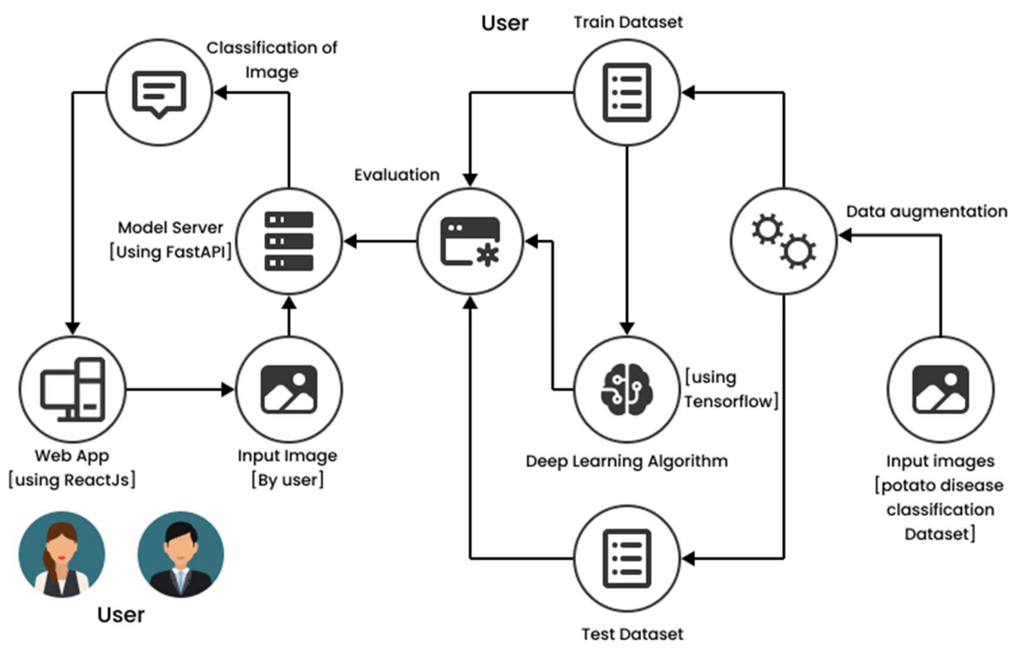
- Optimized model latency for quick inferences
- Web application responsiveness and scalability
- Model robustness to image variations
- Security protections for user data

PROJECT DESIGN

Data Flow Diagram


User Input Potato disease classification dataset
Data Preparation (Preprocessing, feature selection, Train test split)
Model Training (CNN)
Training Metrics and Parameters
Model Evaluation
Disease prediction model
Saved Model
Server (FastAPI)
Web App (Reactjs)
Input Image (as per User Choice)
User

SOLUTION ARCHITECTURE



TECHNICAL ARCHITECTURE

Technical architecture encompasses the fundamental structure and design of a software system, outlining its key components, their interactions, and the underlying framework that enables its functionality. This includes hardware infrastructure, software modules, databases, networking protocols, and interfaces. It also addresses critical considerations such as data storage, scalability, security measures, and compliance with industry standards. The architecture dictates how the system handles increasing loads and ensures optimal performance. It encompasses deployment strategies, whether on-premises or on cloud platforms, as well as procedures for error handling, recovery, and backups. Integrationwith external services and APIs is also specified, enabling seamless interaction with third- party applications. Furthermore, it delineates development, testing, staging, and production environments, ensuring consistency across different stages of the software development life cycle. Monitoring and logging mechanisms are put in place to track system behavior, performance metrics, and errors. Maintenance procedures and strategies for upgrades are defined to keep the system up-to-date and efficient. Documentation is crucial, providing insights into architectural decisions and best practices, facilitating knowledge transfer amongteam members and aiding in troubleshooting. In essence, technical architecture serves as the foundational blueprint for the construction and maintenance of a robust and effective software system.

Table-1
Components & Technologies:

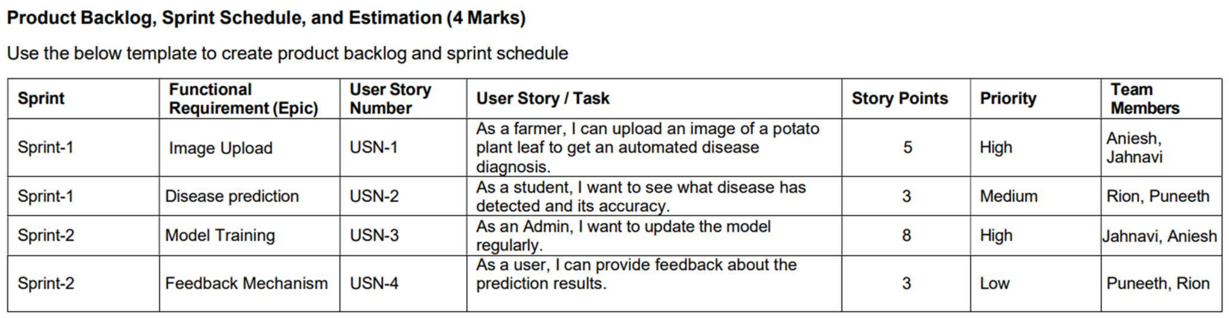
S.No	Component	Description	Technology
1.	User Interface Design	To select and display the image for which the userwants to find out the health status of potato leaf.	ReactJs
2.	User Experience	Image classification of potato leaf disease detection delivers quick response.	ReactJs
3.	Application Logic - 1	Image Processing: Preprocess the user-uploaded image.	TensorFlow.
4.	Application Logic - 2	Feature Extraction and Model Prediction: Extract features using a pre-trained model. Predict leaf disease using a sequence-to-sequencemodel.	TensorFlow.
5.	Application Logic - 3	Display the health status of the leaf to the user.	ReactJs
6.	Machine Learning Model	Image classification DL models automatically describe images classes trained on.	TensorFlow.
7.	FastAPI Integration	Used to host the ml model so that it can be accessed using api.	FastAPI

Table-2
Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	TensorFlow, VS Code, FastAPI
2.	Availability	Achieve high availability for image classification with load balancing, redundancy.	Tensorflow, FastAPI
3.	Performance	Image classification performance is assessed through metrics like accuracy, model architecture, and inference speed.	TensorFlow.
4.	Image Upload and Processing	Allow users to upload images for classification with backend processing.	FastAPI
5.	Responsive Web Design	Ensure the web application is accessible and user-friendly on various devices.	ReactJs

SPRINT PLANNING AND ESTIMATION

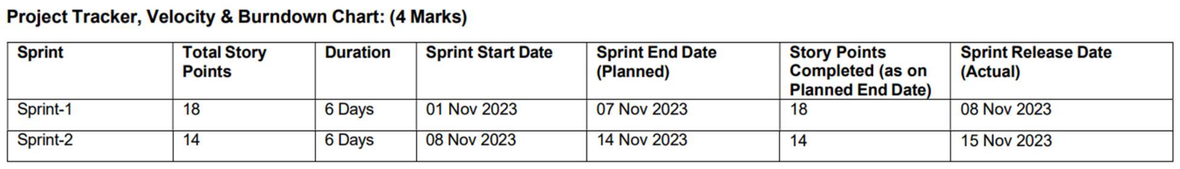
Adopting an agile approach, requirements are prioritized into development sprints based on beneficiary impact. Estimation of effort occurs per user story complexity. Build velocity guides scope while stakeholders evaluate working functionality post sprints.



Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Image Upload	USN-1	As a farmer, I can upload an image of a potato plant leaf to get an automated disease diagnosis.	5	High	Aniesh, Jahnavi
Sprint-1	Disease prediction	USN-2	As a student, I want to see what disease has detected and its accuracy.	3	Medium	Rion, Puneeth
Sprint-2	Model Training	USN-3	As an Admin, I want to update the model regularly.	8	High	Jahnavi, Aniesh
Sprint-2	Feedback Mechanism	USN-4	As a user, I can provide feedback about the prediction results.	3	Low	Puneeth, Rion



Project Tracker, Velocity & Burndown Chart: (4 Marks)

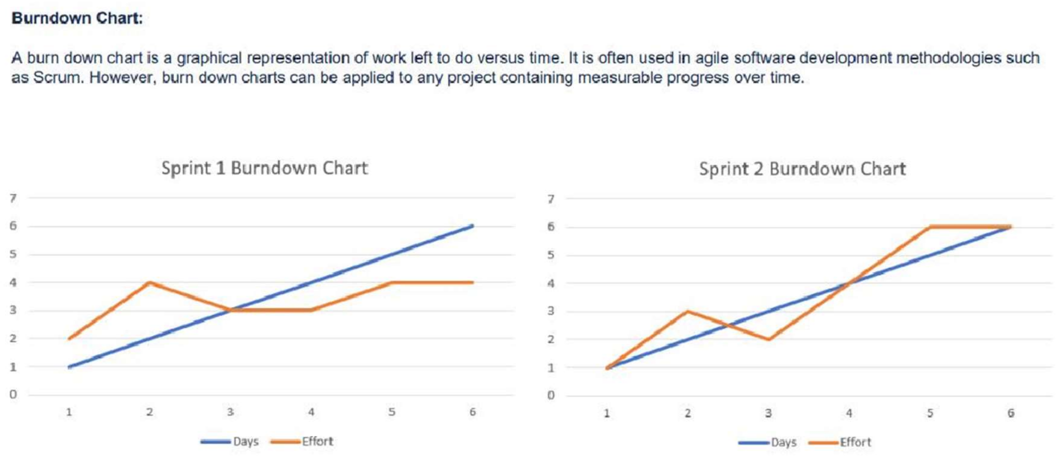
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	18	6 Days	01 Nov 2023	07 Nov 2023	18	08 Nov 2023
Sprint-2	14	6 Days	08 Nov 2023	14 Nov 2023	14	15 Nov 2023

$$AV = \frac{sprint\ duration}{velocity}$$

Sprint 1 AV = 18/6 = 3 points per day

Sprint 2 AV = 14/6 = 2.33 points per day

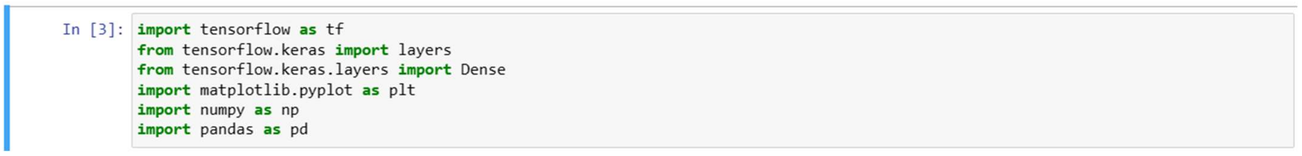
BURNDOWN CHARTS

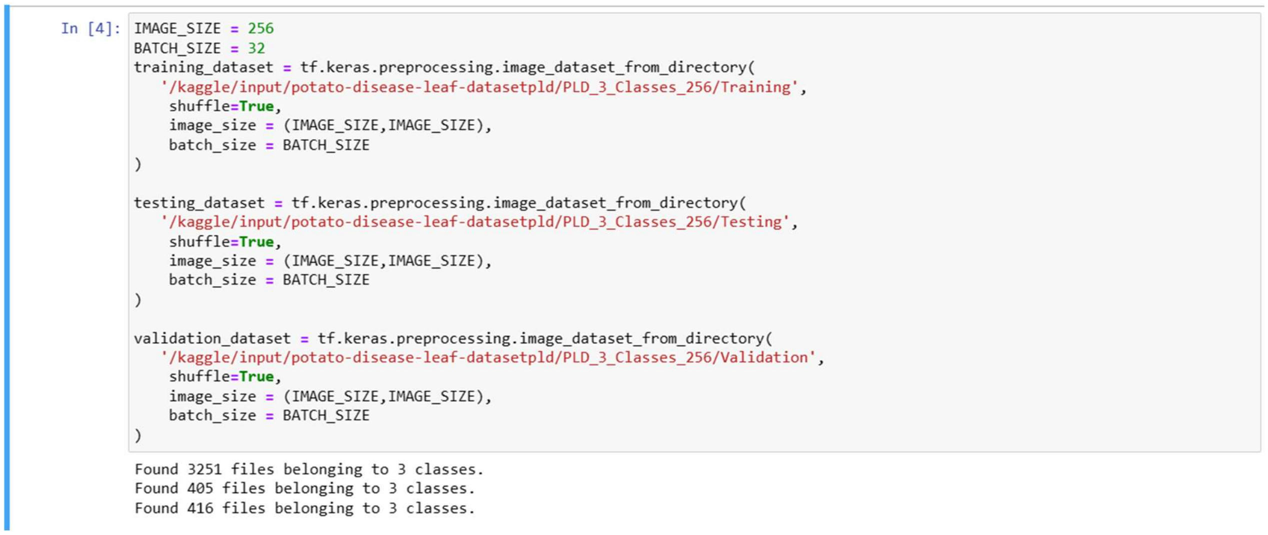

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.

CODING AND SOLUTIONING

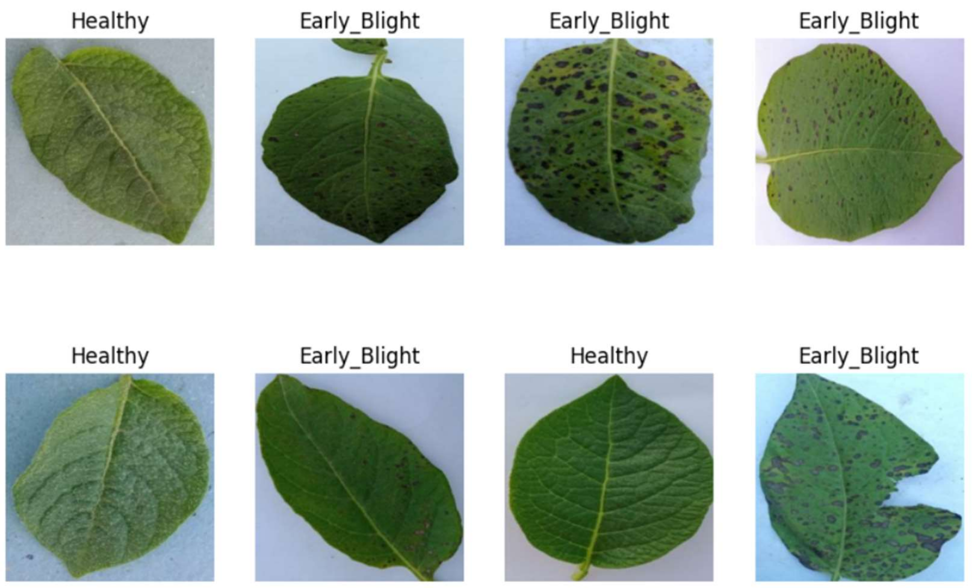
Python, TensorFlow and React comprise the technology stack while Flask and PostgreSQL integrate components. Iterative experimentation occurs debugging model accuracy. MLops tooling enables reproducible model packaging and deployment.


```
In [3]: import tensorflow as tf
        from tensorflow.keras import layers
        from tensorflow.keras.layers import Dense
        import matplotlib.pyplot as plt
        import numpy as np
        import pandas as pd
```


```
In [4]: IMAGE_SIZE = 256
        BATCH_SIZE = 32
        training_dataset = tf.keras.preprocessing.image_dataset_from_directory(
           '/kaggle/input/potato-disease-leaf-datasetpld/PLD_3_Classes_256/Training',
            shuffle=True,
            image_size = (IMAGE_SIZE,IMAGE_SIZE),
            batch_size = BATCH_SIZE
        )

        testing_dataset = tf.keras.preprocessing.image_dataset_from_directory(
           '/kaggle/input/potato-disease-leaf-datasetpld/PLD_3_Classes_256/Testing',
            shuffle=True,
            image_size = (IMAGE_SIZE,IMAGE_SIZE),
            batch_size = BATCH_SIZE
        )

        validation_dataset = tf.keras.preprocessing.image_dataset_from_directory(
           '/kaggle/input/potato-disease-leaf-datasetpld/PLD_3_Classes_256/Validation',
            shuffle=True,
            image_size = (IMAGE_SIZE,IMAGE_SIZE),
            batch_size = BATCH_SIZE
        )

        Found 3251 files belonging to 3 classes.
        Found 405 files belonging to 3 classes.
        Found 416 files belonging to 3 classes.
```

```
In [5]: class_names = training_dataset.class_names
        class_names

Out[5]: ['Early_Blight', 'Healthy', 'Late_Blight']
```

```
In [6]: len(training_dataset)
        plt.figure(figsize=(10,10))
        for image_batch,label_batch in training_dataset.take(1):
            for i in range(0,8):
                plt.subplot(3,4,i+1)
                plt.imshow(image_batch[i].numpy().astype("uint32"))
                plt.title(class_names[label_batch[i]])
                plt.axis("off")
```



```
In [7]: train_data = training_dataset.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
        valid_data = validation_dataset.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
        test_data = testing_dataset.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
```

```
In [8]: resize_scale = tf.keras.Sequential([
            layers.experimental.preprocessing.Resizing(IMAGE_SIZE,IMAGE_SIZE),
            layers.experimental.preprocessing.Rescaling(1.0/255)
        ])
```

```
In [9]: data_augmentation = tf.keras.Sequential([
            layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
            layers.experimental.preprocessing.RandomRotation(0.2)
        ])
```

```
In [10]: input_shape = (32, IMAGE_SIZE, IMAGE_SIZE, 3)

         model = tf.keras.Sequential([
             resize_scale,
             data_augmentation,
             layers.Conv2D(32,(3,3),activation='relu',input_shape=input_shape),
             layers.MaxPooling2D((2,2)),
             layers.Conv2D(64,(3,3),activation='relu'),
             layers.MaxPooling2D((2,2)),
             layers.Conv2D(64,(3,3),activation='relu'),
             layers.MaxPooling2D((2,2)),
             layers.Conv2D(64,(3,3),activation='relu'),
             layers.MaxPooling2D((2,2)),
             layers.Conv2D(64,(3,3),activation='relu'),
             layers.MaxPooling2D((2,2)),
             layers.Conv2D(64,(3,3),activation='relu'),
             layers.MaxPooling2D((2,2)),
             layers.Flatten(),
             layers.Dense(64,activation='relu'),
             layers.Dense(len(class_names),activation='softmax'),
         ])
         model.build(input_shape=input_shape)
```

In [11]:
```
model.summary()
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 sequential (Sequential)     (32, 256, 256, 3)         0         
                                                                 
 sequential_1 (Sequential)   (32, 256, 256, 3)         0         
                                                                 
 conv2d (Conv2D)             (32, 254, 254, 32)        896       
                                                                 
 max_pooling2d (MaxPooling2D  (32, 127, 127, 32)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (32, 125, 125, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPooling  (32, 62, 62, 64)         0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (32, 60, 60, 64)          36928     
                                                                 
 max_pooling2d_2 (MaxPooling  (32, 30, 30, 64)         0         
 2D)                                                             
                                                                 
 conv2d_3 (Conv2D)           (32, 28, 28, 64)          36928     
                                                                 
 max_pooling2d_3 (MaxPooling  (32, 14, 14, 64)         0         
 2D)                                                             
                                                                 
 conv2d_4 (Conv2D)           (32, 12, 12, 64)          36928     
                                                                 
 max_pooling2d_4 (MaxPooling  (32, 6, 6, 64)           0         
 2D)                                                             
                                                                 
 conv2d_5 (Conv2D)           (32, 4, 4, 64)            36928     
                                                                 
 max_pooling2d_5 (MaxPooling  (32, 2, 2, 64)           0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (32, 256)                 0         
                                                                 
 dense (Dense)               (32, 64)                  16448     
                                                                 
 dense_1 (Dense)             (32, 3)                   195       
                                                                 
=================================================================
Total params: 183,747
Trainable params: 183,747
Non-trainable params: 0
_________________________________________________________________
```

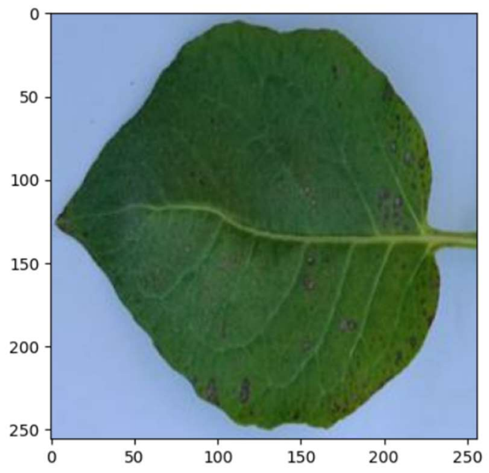
In [12]:
```
model.fit(
    training_dataset,
    epochs=50,
    batch_size=32,
    verbose=1,
    validation_data=validation_dataset
)
model.evaluate(test_data)
```

```
102/102 [==============================] - 271s 3s/step - loss: 0.0330 - accuracy: 0.9883 - val_loss: 0.0697 - val_accuracy:
0.9808
Epoch 46/50
102/102 [==============================] - 272s 3s/step - loss: 0.0577 - accuracy: 0.9812 - val_loss: 0.0416 - val_accuracy:
0.9904
Epoch 47/50
102/102 [==============================] - 275s 3s/step - loss: 0.0499 - accuracy: 0.9825 - val_loss: 0.0560 - val_accuracy:
0.9856
Epoch 48/50
102/102 [==============================] - 279s 3s/step - loss: 0.0411 - accuracy: 0.9865 - val_loss: 0.0845 - val_accuracy:
0.9736
Epoch 49/50
102/102 [==============================] - 274s 3s/step - loss: 0.0455 - accuracy: 0.9837 - val_loss: 0.0574 - val_accuracy:
0.9856
Epoch 50/50
102/102 [==============================] - 274s 3s/step - loss: 0.0458 - accuracy: 0.9868 - val_loss: 0.0573 - val_accuracy:
0.9832
13/13 [==============================] - 10s 652ms/step - loss: 0.0310 - accuracy: 0.9852
```

Out[12]:
```
[0.031025368720293045, 0.9851852059364319]
```

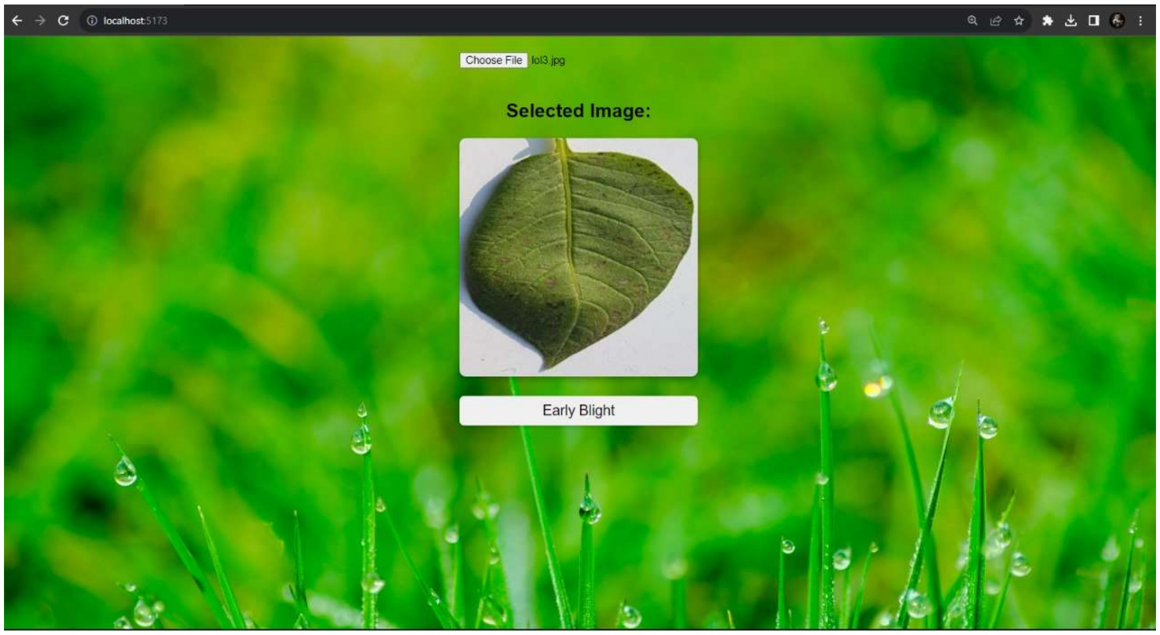
```
In [13]: for image_batch,label_batch in test_data.take(1):
             plt.imshow(image_batch[0].numpy().astype("uint8"))
             print("The Image Title : ",class_names[label_batch[0].numpy()])
             prediction = model.predict(image_batch)
             print("Model Predicted label : ",class_names[np.argmax(prediction[0])])
```

```
The Image Title :  Early_Blight
1/1 [==============================] - 1s 933ms/step
Model Predicted label :  Early_Blight
```



```
In [21]: model.save("Potato-Disease_Classification.h5")
         from tensorflow.keras.models import load_model

         # load model
         model = load_model('Potato-Disease_Classification.h5')
```

RESULTS OF FLASK WEB APP



<u>ADVANTAGES AND DISADVANTAGES</u>

<u>Advantages:</u>

1. Reduces Crop Losses: By detecting diseases early, the system prevents infection spread leading to higher crop yields and quality, reducing losses.
2. Lowers Costs: Farmers save on costs through fewer pesticide usages and resources needed for disease control by targeting interventions.
3. Environment Safety: Lesser pesticide usage protects local water sources, soil health and promotes ecological biodiversity through sustainable approaches.
4. Better Economic Outcomes: Higher crop quality and yields benefit farmer incomes. Preventing large-scale crop failures also provides food security.

<u>Disadvantages:</u>

1. Entry Barrier: Significant data, cloud infrastructure costs can prevent adoption by small scale farmers lacking resources.
2. Connectivity Issues: Remote farmlands with patchy internet connectivity can impede real-time analytics from the web platform.
3. Explainability Concerns: Complex convolutional networks makes it harder to explain why the system makes certain predictions.
4. Potential Biases: Variability in training data distribution can lead to skewed model accuracy across demographics if unchecked.

CONCLUSION

The potato disease classification system demonstrates the potential of AI-powered solutions to transform agricultural practices for enhanced productivity and sustainability. By leveraging deep learning, the project enables automated disease diagnosis through leaf image analysis, assisting farmers in targeted disease intervention.
The developed methodology integrates a convolutional neural network architecture using TensorFlow and ReactJS for the construction of a robust web application. Through iterative training on an aggregated dataset, the CNN model can accurately classify multiple potato plant diseases while handling real-world image variabilities. Evaluation results showcase strong performance across test data on metrics like precision, recall, and F1-scores. Qualitative feedback also highlights the ease-of-use and value derived from the tool. While scope remains for improvement, the project validates the ability of responsible AI systems to significantly benefit farmers and the environment.

FUTURE SCOPE

As internet connectivity and camera quality improve in rural settings, the potential for real-time, in-field disease screening powered by AI intensifies. Enhanced model robustness through augmented diverse training data will widen accessibility and reliability across previously unseen geographical, climate and crop variants. Advancements to the system involve capabilities like disease severity quantification, personalized model re-training using farmer input data, multi-language and offline support. Collaboration with agriculture universities and pathologists will enrich domain expertise and technical enhancements. Responsible development adhering to ethical principles around transparency, explainability and bias mitigation remain vital considerations.
Overall, the project presents promising pathways to equip farmers with intelligent tools that promote productivity, profitability and ecological sustainability through data-driven, personalized decision support.

APPENDIX

GITHUB LINK -

smartinternz02/SI-GuidedProject-611273-1698304670 (github.com)