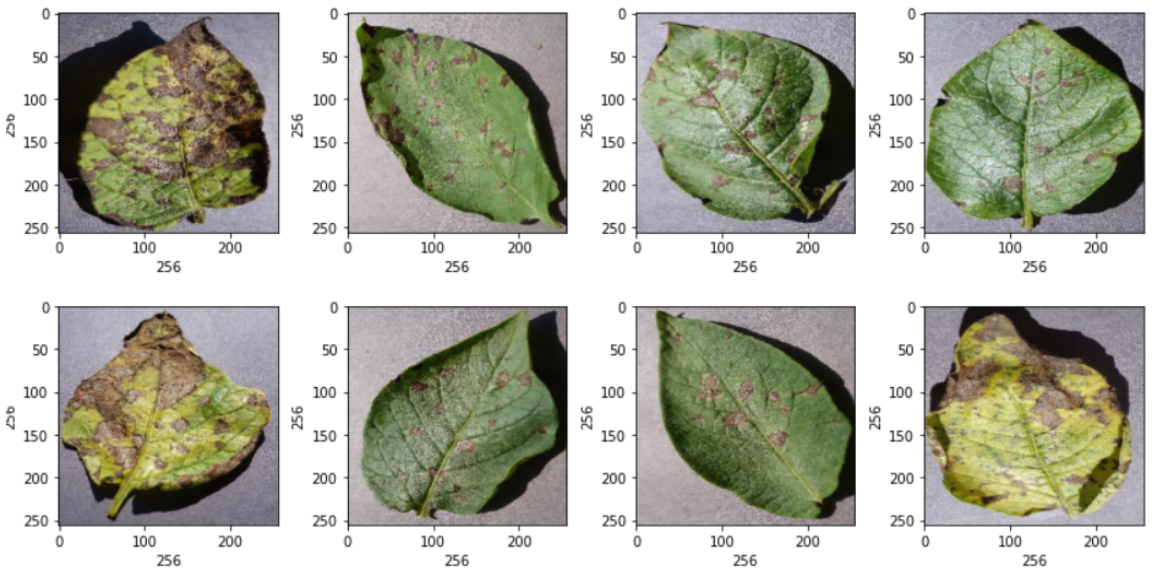
Product Requirement Document (PRD)
CropGuardian: Digital Sentinel for Crops Health
PM: Chinmay Dhamapurkar

1. Overview:
The Plant Diseases Prediction System is designed to predict and classify diseases in various plants, leveraging advanced deep learning techniques. The project's primary objective is to assist farmers and agriculturalists in identifying diseases at an early stage, enabling timely intervention and reducing crop losses.

2. Objectives:
- Predict and classify plant diseases with a high degree of accuracy.
- Provide actionable insights for users based on predictions.
- Reduce economic losses due to undetected or late-detected plant diseases.

3. Data:
- Source: The data will be collected from reliable agricultural datasets containing labeled images of plant leaves.
- Features:
    - High-resolution images of plant leaves.
    - Disease labels for each image.
    - Additional meta-data such as date of capture, geographic location



4. Scope:

i. Input:

High-resolution images of plant leaves, preferably in varied lighting conditions and from multiple angles.

ii. Process:

- Data Collection: Extract data from provided agricultural datasets.
- Data Preprocessing: Clean, augment, and preprocess images for the neural network.
- Model Building: Construct a deep learning model architecture tailored for image classification tasks.
- Model Training: Train the model using the preprocessed dataset, optimizing accuracy.
- Validation and Testing: Evaluate model performance on unseen data.
- Deployment: Integrate the model into a user-friendly application.

iii. Output:

Predicted disease label with a confidence score and, if possible, recommended interventions.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 256, 256, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 85, 85, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 85, 85, 16)        4624
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 42, 42, 16)        0
_________________________________________________________________
flatten (Flatten)            (None, 28224)             0
_________________________________________________________________
dense (Dense)                (None, 8)                 225800
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 27
=================================================================
Total params: 231,347
Trainable params: 231,347
Non-trainable params: 0
_________________________________________________________________
```

5. Detailed Model Architecture:

i. Data Collection:

- Use web scraping tools and APIs to fetch plant images and related data.
- Ensure diversity in the dataset to make the model robust to various conditions.

ii. Data Preprocessing:

- Image Augmentation: Introduce variations such as rotations, zooming, and horizontal flips to increase the dataset's size and diversity.
- Scaling: Normalize pixel values to be between 0 and 1.
- Train-Test Split: 80% for training and 20% for testing/validation.

iii. Model Building:

- Convolutional Neural Network (CNN) layers: Extract features from the images.
- Pooling layers: Reduce dimensionality and extract dominant features.
- Dropout layers: Prevent overfitting.
- Fully Connected (Dense) layers: Perform classification.

iv. Model Training:

- Optimizer: Adam optimizer for adaptive learning rates.
- Loss Function: Cross-entropy for multi-class classification.
- Epochs and Batch Size: Based on training convergence - typical values might be 100 epochs with a batch size of 64.

v. Model Evaluation:

- Accuracy: Percentage of correctly predicted labels to total predictions.
- Confusion Matrix: Understand misclassifications.
- Loss Curves: Ensure the model isn't overfitting or underfitting.

vi. Deployment:

- Deploy using platforms like Flask/Django for web applications or integrate into mobile applications.
- Ensure the application provides a user-friendly interface for uploading images and receiving predictions.

```
[ ]  # Finding max value from predition list and comaparing original value vs predicted
     print("Originally : ",all_labels[np.argmax(y_test[10])])
     print("Predicted : ",all_labels[np.argmax(y_pred[10])])
```

```
Originally :  Potato-Early_blight
Predicted :  Potato-Early_blight
```

6. Additional Features:

- Feedback Loop: Allow users to confirm if the prediction was correct or provide the correct label, enhancing the model over time.
- Informational Database: Upon disease detection, provide users with more information about the disease and potential interventions or treatments.

7. Constraints:

- Model inference time should be kept minimal to ensure real-time feedback for users.
- Ensure data privacy; user-uploaded images should not be stored without explicit consent.

8. Future Enhancements:

- Incorporate more diverse datasets covering additional plants and diseases.
- Use Transfer Learning by leveraging pre-trained models like ResNet, VGG, etc., for better accuracy.
- Provide a mobile application for on-the-go disease detection.

9. Risks & Challenges:

- Inaccurate predictions could lead to incorrect interventions, potentially causing more harm.
- The diversity of plant diseases and similarities between some might pose challenges in achieving high accuracy.