1. Introduction

1.1 Purpose

Crop leaf diseases have a significant impact on agricultural productivity, leading to significant yield losses for farmers. Early detection and timely management of crop leaf diseases are essential to minimize the damage caused to crops. With the advancements in deep learning techniques and computer vision, the use of artificial intelligence for crop leaf disease prediction has gained significant attention in recent years. Therefore, the motivation behind this thesis is to develop an accurate and efficient crop leaf disease prediction model using deep learning algorithms.

The study focuses on evaluating the performance of five different deep learning architectures, namely MobileNetV2, Resnet152V2, VGG19, Inception V3, and DenseNet201, for crop leaf disease prediction. The dataset used for the study comprises images of healthy and diseased leaves of four different types of crops, namely Apple, Grape, Potato, and Tomato. By using multiple architectures, we aim to identify the best-performing model for each crop and compare their performance against each other.

The motivation behind deploying the developed model using Flask is to make it easily accessible to farmers. The deployed model provides a web-based interface that allows users to upload images of crop leaves and obtain predictions on whether the leaves are healthy or diseased. This would enable farmers to identify crop diseases early, make informed decisions, and take the necessary actions to prevent further damage.

We develop a crop leaf disease prediction model that can accurately and efficiently identify crop diseases, thereby helping farmers in making informed decisions and reduce yield losses. The deployment of the developed model using Flask would make it easily accessible to farmers, thus contributing to the development of sustainable agricultural practices.

By using Five architectures, the goal is to compare their performance in identifying crop leaf diseases accurately. The trained models will be evaluated on a test set, and the one with the highest accuracy will be considered the best model for crop leaf disease prediction. The project aims to provide a useful tool for farmers to identify crop diseases early and take appropriate measures to prevent the spread of the disease and minimize yield losses.

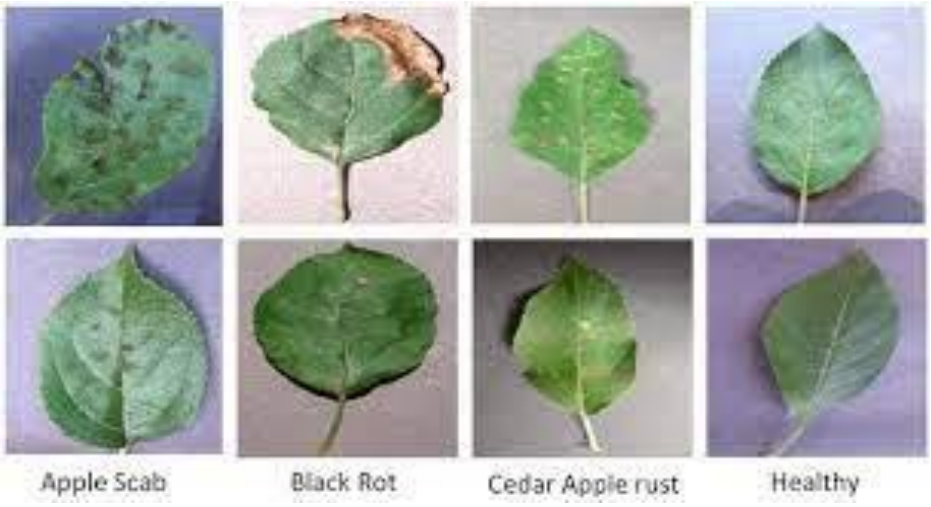


Fig 1.2 Apple Healthy and Types of Disease [13]

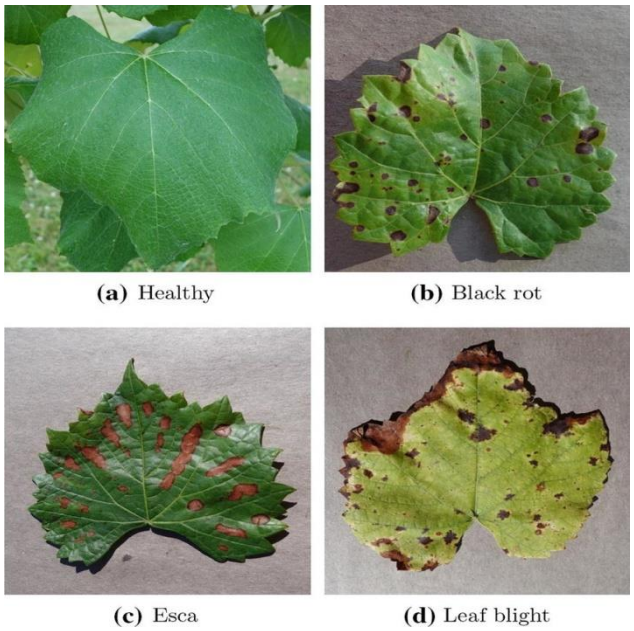


Fig 1.3 Grape Healthy and Types of Disease [14]

Fig 1.4 Potato Healthy and Types of Disease [15]

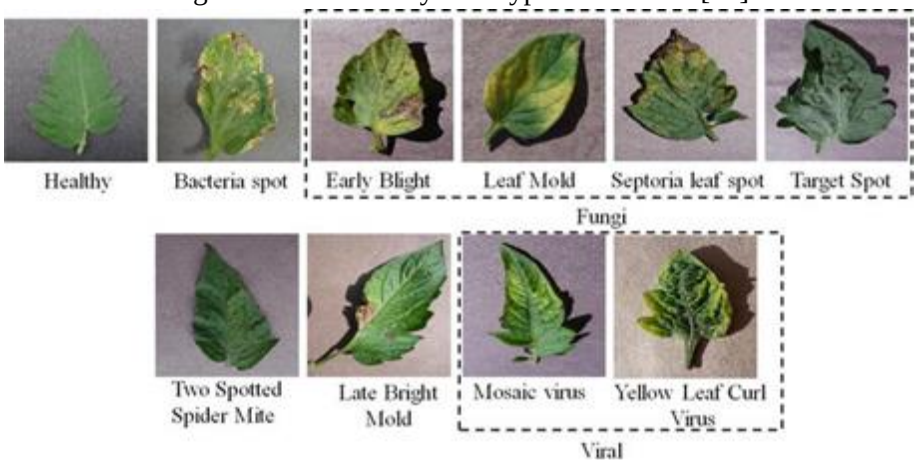


Fig 1.5 Tomato Healthy and Types of Disease [16]

1.2 Document Conventions

This section outlines the writing conventions and standards employed in this Software Requirements Specification (SRS) document to ensure clarity, consistency, and effective communication.

1. Document Format:

This SRS is authored using Microsoft Word (version X).

The document will be distributed in PDF format for ease of sharing and printing.

2. Title and Version:

The title of this document follows the format 'Software Requirements Specification - [Tomato Leaf Disease Prediction].'

The version number and date are located in the document header for easy reference.

3. Headings and Subheadings:

Headings are formatted in 16-point Arial font, bold, and centered.

Subheadings are in 12-point Arial font, bold, and left-aligned.

These conventions are adopted to enhance the readability and consistency of this SRS document. Stakeholders, reviewers, and team members are encouraged to adhere to these conventions when interacting with this document.

1.3 Intended Audience and Reading Suggestions

This Software Requirements Specification (SRS) is intended for a diverse audience of stakeholders involved in the software project. The document is tailored to meet the needs of the following reader types:

Developers: This section contains technical details and requirements necessary for the software development process. Developers should focus on understanding functional and technical specifications.

Project Managers: Project managers will find information related to project scope, timelines, and resource allocation. They should pay special attention to project constraints, dependencies, and scheduling details.

Marketing Staff: Marketing staff may be interested in understanding the features and functionalities of the software to develop marketing materials and strategies.

Users: End-users will benefit from the user requirements section, which describes how the software will meet their needs and what functionalities they can expect.

Testers: Testers should focus on functional requirements, non-functional requirements, and any specific testing criteria outlined in the document.

Documentation Writers: Those responsible for creating user manuals or help documentation will want to review user requirements and system functionalities in detail.

1.4 Product Scope

Project Overview:
The Tomato Leaf Disease Prediction project aims to develop a software solution that assists in the early detection and prediction of diseases affecting tomato plants. The primary focus is on providing a reliable and user-friendly tool for farmers, horticulturists, and agricultural experts to identify and manage diseases affecting tomato leaves.

In Scope:
Leaf Disease Identification: The system will accurately identify various diseases that affect tomato plant leaves, including but not limited to early blight, late blight, Septoria leaf spot, and bacterial spot.

Image Input: Users will be able to input images of tomato leaves, and the system will process these images for disease detection.

Prediction and Diagnosis: The system will provide a diagnosis based on the image input, indicating the likelihood of specific diseases being present on the tomato plant leaves.

User Interface: A user-friendly interface will be designed to facilitate easy interaction with the system.

Mobile Compatibility: The system will be compatible with mobile devices, enabling users to capture and analyze images directly from smartphones.

User Accounts: Users will have the option to create accounts to store and track their disease prediction history.

Educational Resources: The system may include educational resources and recommendations for disease management and prevention.

Out of Scope:
Physical Hardware: The project does not involve the development or provision of specialized hardware for image capture.

Crop Management: The project does not address broader crop management issues beyond disease identification and prediction.

Commercialization: The project is not involved in commercializing the software or creating revenue-generating models.

Cross-Platform Compatibility: While mobile compatibility is in scope, the project will not initially support multiple mobile platforms (e.g., iOS and Android) but can be extended in the future.

Advanced Machine Learning Models: The project will use established machine learning models for disease prediction and will not delve into research and development of new models.

Constraints:
The product development is subject to certain constraints:

Data Quality: The accuracy of disease identification is dependent on the quality and quantity of the training dataset.

Performance: The speed and performance of disease prediction may be influenced by the hardware and network capabilities of the user's device.

Regulatory Compliance: The system should comply with relevant data protection and privacy regulations.

Limited Resources: The project is constrained by time, budget, and human resources.

Dependencies:
The successful development and deployment of the Tomato Leaf Disease Prediction project depend on the following factors:

Data Collection: The availability and collection of a comprehensive dataset of tomato leaf images with labeled disease conditions.

Technological Dependencies: Availability and compatibility of the necessary software, libraries, and hardware for machine learning and image processing.

User Input: The project's success relies on user engagement and their willingness to provide images for disease prediction.

Regulatory Approvals: Compliance with local and national data privacy and agricultural regulations.

This product scope section provides a clear understanding of what is included and excluded in your Tomato Leaf Disease Prediction project and outlines the project's constraints and dependencies. Adapt this template to the specific details of your project and any additional considerations that may apply.

2. Overall Description

2.1 Product Perspective

The Tomato Leaf Disease Prediction system is designed to operate within a specific context. It is part of a larger ecosystem involving various stakeholders, data sources, and potential integrations. The following elements define the system's context:

Stakeholders: The primary users of the system include farmers, horticulturists, and agricultural experts who rely on the system for tomato leaf disease prediction.

Data Sources: The system relies on a dataset of tomato leaf images with labeled disease conditions for its machine learning model. Users provide these images for analysis.

Integration Possibilities: The system may offer integration options with other agricultural and crop management tools and platforms for a more comprehensive solution.

Hardware Requirements: Users require access to a compatible device (e.g., smartphone or computer) with internet connectivity to interact with the system.

2.2 Product Functions

FUNCTIONAL REQUIREMENTS

Functional Requirements:

1. The system should be able to accurately predict whether the crop leaf is healthy or diseased.

2. The system should be able to handle images of crop leaves of four different types of crops, namely Apple, Grape, Potato, and Tomato.
3. The system should be able to identify the best-performing deep-learning architecture for each crop.
4. The system should be able to provide a web-based interface for users to upload images of crop leaves and obtain predictions on whether the leaves are healthy or diseased.
5. The system should be able to store and retrieve the preprocessed dataset of images of healthy and diseased leaves of crops.

NON- FUNCTIONAL REQUIREMENTS

Non - Functional Requirements:

1. The system should have a high level of accuracy and precision in predicting crop leaf diseases.
2. The system should have a low processing time for prediction, even with large datasets.
3. The system should be scalable and able to handle an increasing number of users and requests.
4. The system should be user-friendly and intuitive, with clear instructions and feedback for users.
5. The system should ensure the security and privacy of user data and predictions.
6. The system should be reliable, with minimal downtime or errors, and be easy to maintain and update.

2.3 User Classes and Characteristics

The software requirements for the thesis of crop leaf disease prediction using five architectures (MobileNetV2, Resnet152V2, VGG19, Inception V3, and DenseNet201) include the usage of the Google Colaboratory platform for training the models. Google Colab provides a free cloud-based platform with powerful hardware resources, such as GPUs and TPUs, that can significantly accelerate the model training process. Additionally, Colab allows easy access to various libraries and frameworks, such as TensorFlow and Keras, that are essential for deep learning tasks.

It also requires the installation of Python 3.7 or later versions, TensorFlow, Keras, Flask, NumPy, Pandas, matplotlib, and Scikit-learn libraries. These libraries can be easily installed using the pip package manager. Additionally, the use of a web browser such as Google Chrome, Firefox, or Safari is necessary to access the Google Colaboratory platform.

2.4 Operating Environment

The hardware requirements include a stable internet connection with sufficient bandwidth to support data transfer during model training and deployment. As the models are trained on Google Colaboratory, a compatible device with a web browser is required. However, for optimal performance, it is recommended to use a device with a multi-core CPU, a minimum of 16 GB of RAM, and a dedicated GPU, preferably Nvidia GPUs with CUDA support.

2.5 Design and Implementation

In terms of agriculture preservation, our farmers have to check the leaves and illness of the plants by getting the sample itself or checking on the field. There might be a shot at a blunder because of the absence of information and numerous different variables. So we need to automate this with the goal that farmers can rapidly build their creations.

This research aims to use a transfer learning DenseNet201, Inception v3, MobileNetV2 ResNet152V2, VGG19 model that has previously been trained on a significant quantity of data to identify disease in Crop leaves autonomously.

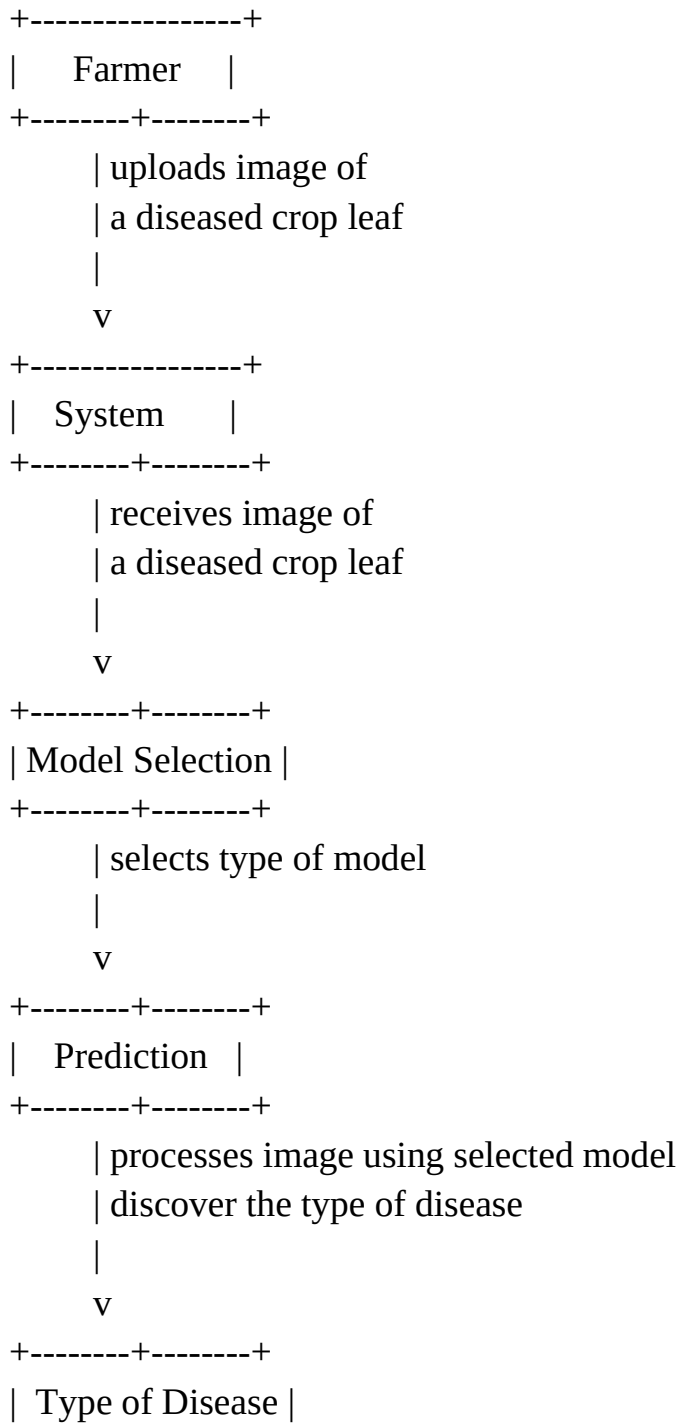
Once the models are trained, they will be evaluated using various metrics such as accuracy, precision, recall, and F1 score. The best-performing model will be selected and deployed using Flask, a microweb framework in Python. The Flask application will allow users to upload an image of a crop leaf and get a prediction of whether it is healthy or diseased, along with the type of disease it is diseased.

To ensure the smooth functioning of the system, various non-functional requirements such as scalability, availability, and security will also be considered during this phase. The system will be designed to handle a large number of requests concurrently and provide high availability. Security measures such as user authentication and data encryption will also be implemented to ensure the privacy and confidentiality of user data.

Deep learning increased the learning capacity of the features directly in highly dimensional unprocessed s data, the deep learning algorithms in images and extraction of specific audio segments, and supervised learning in general. Hence, as a strong candidate for classification task modulation, an integrated understanding of deep Learning algorithms solves the central problem as the characteristics of the samples are selected and extracted. Therefore, it shows the combination of simple functions in more efficient and more complex features to achieve classification more efficiently and complicated. Moreover, deep neural networks have a multilayer structure that can better extract the signal properties by avoiding the lengthy manual selection of data properties. In this project, we create a small-scale deep convolutional neural network (DCNN), assess its performance, and then develop a more advanced deep learning network based on various art data and techniques.

2.5.1 USE CASE DIAGRAM

A use case diagram is a graphical representation of the system's functionality and actors involved in it. The use case diagram for the crop leaf disease prediction system is given below.



+--------+--------+

The primary actors involved in the system are the farmer and the system itself. The farmer can perform the following actions:

Upload an image of a diseased crop leaf.
Select the type of model.
Click on predict button.
Discover the type of disease in the crop leaf.

The pictorial use case diagram is given below :

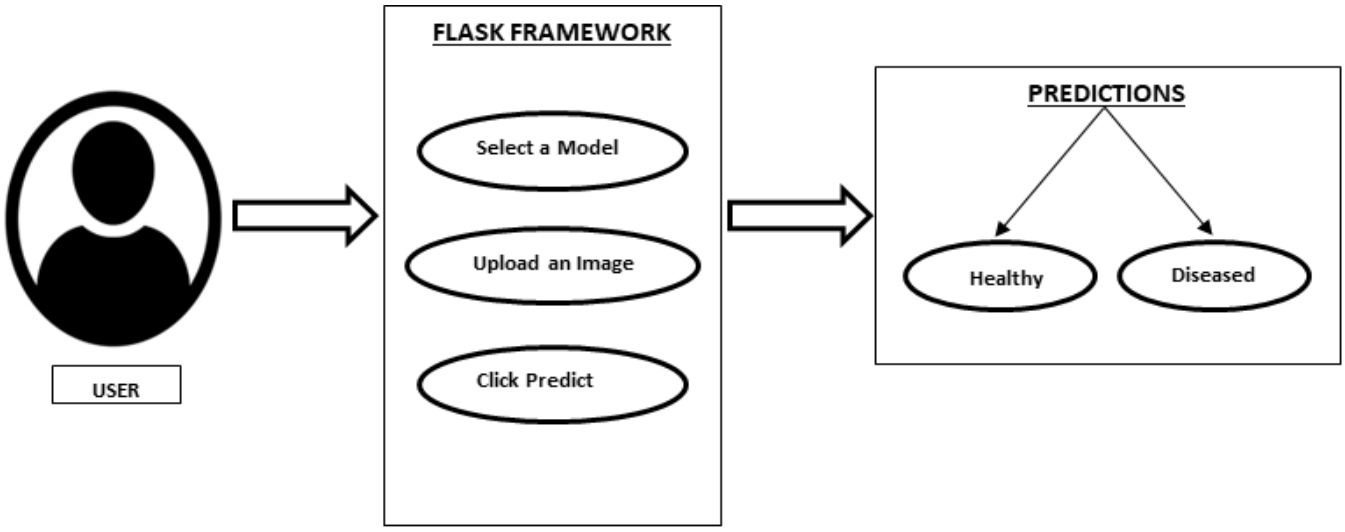


Fig 3.1 Use Case Diagram

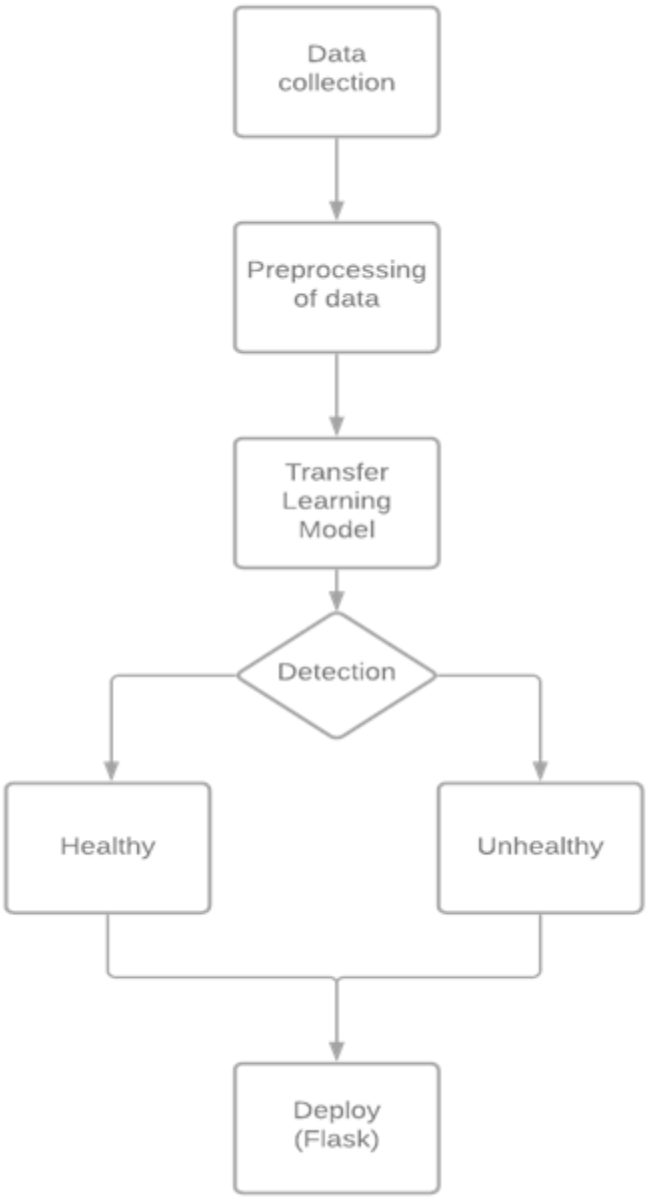


Fig 3.1 Proposed Methodology of the System

2.5.2 TYPES OF MODELS USED

The thesis of crop leaf disease prediction uses five different types of convolutional neural network (CNN) architectures: MobileNetV2, ResNet152V2, VGG19, Inception V3, and DenseNet201. These architectures have been chosen for their effectiveness in image classification tasks and have been trained using the dataset of four different crops: Apple, Grape, Potato, and Tomato. Each model has

its own advantages and limitations, and by using multiple models, the thesis aims to achieve better accuracy in disease prediction. The deployed model uses Flask, a Python web framework, to provide a user-friendly interface for predicting diseases in crop leaves.

2.5.2.i DenseNet201

DenseNet201 is a convolutional neural network (CNN) architecture that is commonly used for image classification tasks. DenseNet stands for "Densely Connected Convolutional Networks" and it is designed to improve the flow of information through the network, which can lead to better performance and reduced complexity.

In DenseNet201, each layer is connected to every other layer in a feed-forward fashion, resulting in a densely connected network. This is achieved by concatenating the feature maps of all previous layers, rather than simply adding or averaging them.

The architecture of DenseNet201 consists of a series of dense blocks, which are composed of multiple convolutional layers. Each dense block is connected to a transition layer, which includes a pooling layer and a convolutional layer, that reduces the spatial dimensions of the feature maps.

One of the advantages of DenseNet201 is its efficient use of parameters, as the dense connectivity reduces the number of parameters needed compared to other CNN architectures. It has been shown to perform well on a variety of image classification tasks, including in the field of agriculture for crop disease detection.

In terms of accuracy, DenseNet201 has achieved state-of-the-art results on several image classification benchmarks. However, the performance of the model can vary depending on the specific dataset and task at hand.

2.5.2.ii Inception V3

Inception V3 is a convolutional neural network (CNN) architecture that was developed by Google. It was designed to be computationally efficient while achieving high accuracy in image classification and object detection tasks.

The key feature of Inception V3 is the inception module, which allows for more efficient computation by performing convolutions at multiple scales within the same layer. This enables the network to capture information at different levels of abstraction and to process images of varying sizes.

In addition to its efficiency, Inception V3 has achieved state-of-the-art results on a number of image classification and detection tasks. Its success has made it a popular choice for a wide range of applications, including crop leaf disease prediction.

One of the advantages of Inception V3 is its ability to handle large and complex datasets. It can also be trained with relatively small amounts of data, making it ideal for applications where data is scarce or expensive to collect.

In terms of accuracy, Inception V3 has achieved high performance on several benchmark datasets. For example, it achieved a top-5 error rate of 3.46% on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2016 dataset, which consists of over a million images from 1,000 different classes.

In the context of crop leaf disease prediction, Inception V3 can be used to accurately classify different types of diseases affecting crops such as Apple, Grape, Potato, and Tomato.

2.5.2.iii MobileNetV2

MobileNetV2 is a convolutional neural network architecture that is designed to be fast and efficient while still maintaining high accuracy. It achieves this by using a combination of depthwise separable convolutions and pointwise convolutions, which greatly reduce the number of parameters in the model. This makes it a popular choice for mobile and embedded applications where computational resources are limited.

In the context of crop leaf disease prediction, MobileNetV2 can provide accurate predictions while requiring fewer resources compared to other architectures. It can handle the complex features of crop leaves and make predictions in real-time, which is essential for quick and effective disease management.

MobileNetV2 has been widely used in various computer vision tasks, including image classification, object detection, and segmentation. It has achieved high accuracy on benchmark datasets such as ImageNet, and it is also a popular choice in the field of transfer learning, where pre-trained models are used as a starting point for training on other datasets.

The maximum accuracy achieved by MobileNetV2 on the crop leaf disease prediction task will depend on the specific dataset and training conditions used. However, it has been shown to achieve competitive accuracy compared to other architectures while requiring significantly fewer resources.

2.5.2.iv ResNet152V2

ResNet152V2 is a deep convolutional neural network architecture that was introduced as an improvement over its predecessor, ResNet. It has 152 layers, making it deeper than its predecessor, and is capable of achieving high accuracy in image recognition tasks. ResNet152V2 is characterized

by residual connections that help to mitigate the vanishing gradient problem, allowing for deeper networks to be trained more effectively.

In the context of crop leaf disease prediction, ResNet152V2 has shown promising results due to its ability to learn and extract features from images. Its deep architecture enables it to capture intricate details of the leaves and the diseases that affect them. Additionally, ResNet152V2 has shown high accuracy rates in image classification tasks in various domains, making it a viable option for crop leaf disease prediction.

The advantages of ResNet152V2 include its ability to handle deeper architectures and its high accuracy rates in image recognition tasks. It has been successfully applied in various fields such as object detection, image segmentation, and face recognition.

The max accuracy achieved with ResNet152V2 on the crop leaf disease prediction dataset will depend on the specific implementation and dataset used. However, in general, ResNet152V2 has achieved high accuracy rates on various image classification tasks, and it can be expected to perform well in crop leaf disease prediction as well.

2.5.2.v VGG19

VGG19, or the 19-layered Visual Geometry Group model, is a convolutional neural network architecture that was first introduced in 2014. This model was developed by the Visual Geometry Group at Oxford University, and it achieved outstanding results on the ImageNet dataset, which contains over 14 million images.

The VGG19 architecture consists of 19 layers, which includes 16 convolutional layers and 3 fully connected layers. The convolutional layers have filters of size 3x3 with a stride of 1 and a padding of 1, while the max-pooling layers have a filter size of 2x2 with a stride of 2. This architecture is deeper than its predecessor, VGG16, which only has 16 layers.

One of the main advantages of the VGG19 model is its simplicity and ease of implementation. It has a uniform architecture with a small filter size, which allows for a smaller number of parameters compared to other deep learning models. This makes the model easier to train and faster to converge.

The VGG19 model has been applied to a wide range of computer vision tasks, such as image classification, object detection, and segmentation. In crop leaf disease prediction, it has been used to classify diseased and healthy leaves of different crops, including tomato, potato, and apple.

The maximum accuracy achieved by the VGG19 model in crop leaf disease prediction may vary depending on the dataset and the preprocessing techniques used. However, it has been reported to achieve accuracy values ranging from 90% to 98%.

2.5.3 TRANSFER LEARNING

Transfer learning is the practice of reusing a previously learned model for a replacement task; it is popular in deep learning since it allows deep neural networks to be trained with a small amount of data. It is instrumental in data science because most real problems do not have many data points marked to coach these complex models. Let us look at transfer learning, how it works, why it is valid, and when it should be used. Several resources for models who have been previously trained in learning transfers are included.

The general idea behind transfer learning is to apply knowledge obtained from projects with many marked data to situations where just-named data is available. Because creating named data is expensive, it is critical to make use of existing datasets whenever possible. The primary goal of a standard machine learning model is to summarize inconspicuous information based on designs derived from preparation data. You use transfer learning to begin this speculative encounter by starting with strategies learned for a separate mission. Essentially, instead of starting the taking-in process with a (usually randomly introduced) transparent sheet, you start with designs that have been planned out to answer an alternate assignment.

Although deep learning is frequently employed in research, many real-world situations lack many named data points on which to create a model. To tune the high number of boundaries in neuronal architecture, profound learning approaches need large amounts of data. This necessitates a large amount of (expensive) labeled information, especially in the case of controlled learning. Although it may seem minor, master information is required to create a large named dataset in Natural Language Processing (NLP).

Transfer learning has both advantages and disadvantages. Understanding these drawbacks is critical for successful AI applications. Information transfer is only possible when it is 'appropriate.' It's difficult to determine the best approaches in this case, so you'll have to try a lot.

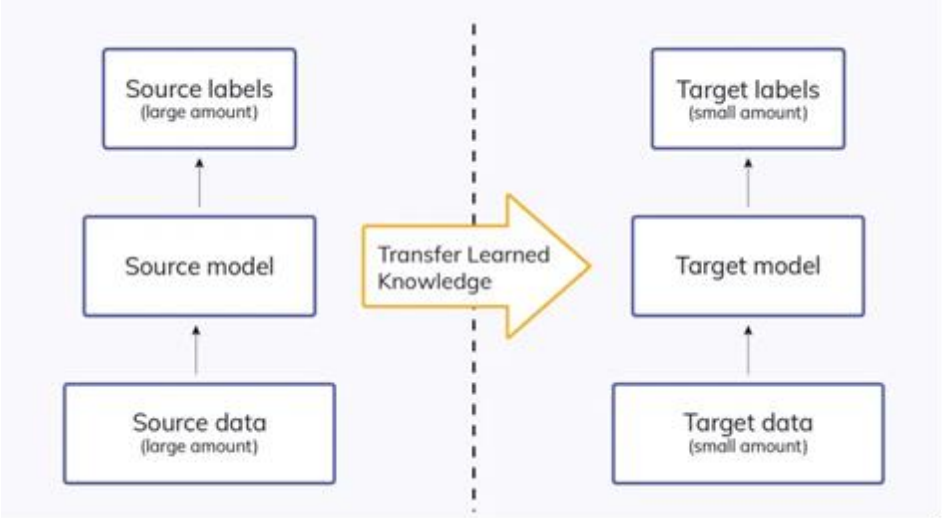


Fig 3.2 Transfer Learning [14]

2.5.4 DATA ANALYSIS

The dataset used for crop leaf disease prediction contains a total of 15,760 images of diseased and healthy leaves from four different types of crops: Apple, Grape, Potato, and Tomato. The dataset is divided into a training set and a validation set, with 75% of the images used for training and 25% for validation.

Each class contains an approximately equal number of images, 1000 images per class. Our model contains 21 classes, every class describes the disease of crops (Apple, Grape, Potato, and Tomato). The images are of varying sizes.

The dataset includes images of both diseased and healthy leaves for each crop type. The diseased leaves are labeled with the specific disease present, with a total of 21 different diseases represented in the dataset. The classes of the crop leaves are labeled as such-
Apple: 4 classes– Healthy, apple scab, black rot, and cedar apple rust.
Grape: 4 classes– Healthy, black rot, black measles, isariopsis, and leaf spot.
Potato: 3 classes– Healthy, early blight, and late blight.
Tomato: 10 classes– Healthy, early blight, late blight, septoria leaf spot, bacterial spot, yellow leaf curl virus, leaf mold, mosaic virus, spider bites, target spot.

The dataset includes images of both diseased and healthy leaves for each crop type. The diseased leaves are labeled with the specific disease present, with a total of 21 different diseases represented in the dataset. The healthy leaves are labeled as such.

The dataset was collected from various sources and includes images with different lighting conditions, camera angles, and backgrounds. The images were preprocessed to remove any unwanted artifacts

and normalize the image sizes. The dataset was then split into training and validation sets using a stratified sampling approach to ensure that each class was represented equally in both sets.

2.5.5 STEPS OF MODEL TRAINING

Crop leaf disease prediction involves several steps, including data collection, preprocessing, model training and testing, and deployment using Flask.

- Data collection: The first step in the project was to collect a dataset of images of healthy and diseased crop leaves for the three crops: apple, grape, and tomato. The dataset was curated from publicly available sources and augmented to increase its size and diversity.

- Data preprocessing: The collected dataset was preprocessed by resizing the images to a uniform size and normalizing the pixel values. The images were also split into training, validation, and test sets.

- Model selection: Three deep learning architectures were selected for the project: InceptionV3, MobileNetV2, and ResNet152V2. These architectures were chosen for their high accuracy and efficiency in image recognition tasks.

- Model training: The selected architectures were trained on the preprocessed dataset using transfer learning. Transfer learning involves using pre-trained models and fine-tuning them on a new dataset. The training was performed on a GPU to speed up the process.

- Model evaluation: The trained models were evaluated on the test set to measure their performance in predicting the type of disease and the affected crop plant. Metrics such as accuracy, precision, recall, and F1-score were used to evaluate the models.

- Model comparison: The performance of the three models was compared, and the one with the highest accuracy was selected as the best model for crop leaf disease prediction.

- Results and analysis: The results of the project were analyzed, and insights were drawn regarding the effectiveness of deep learning in crop leaf disease prediction. The project showed that deep learning models can accurately predict the type of disease and the affected crop plant, thus helping farmers take appropriate measures to prevent the spread of the disease and minimize yield losses.

3. External Interface Requirements

3.1 Hardware Interfaces

The hardware requirements include a stable internet connection with sufficient bandwidth to support data transfer during model training and deployment. As the models are trained on Google Colaboratory, a compatible device with a web browser is required. However, for optimal performance, it is recommended to use a device with a multi-core CPU, a minimum of 16 GB of RAM, and a dedicated GPU, preferably Nvidia GPUs with CUDA support.

3.2 Software Interfaces

The software requirements for the thesis of crop leaf disease prediction using five architectures (MobileNetV2, Resnet152V2, VGG19, Inception V3, and DenseNet201) include the usage of the Google Colaboratory platform for training the models. Google Colab provides a free cloud-based platform with powerful hardware resources, such as GPUs and TPUs, that can significantly accelerate the model training process. Additionally, Colab allows easy access to various libraries and frameworks, such as TensorFlow and Keras, that are essential for deep learning tasks.

It also requires the installation of Python 3.7 or later versions, TensorFlow, Keras, Flask, NumPy, Pandas, matplotlib, and Scikit-learn libraries. These libraries can be easily installed using the pip package manager. Additionally, the use of a web browser such as Google Chrome, Firefox, or Safari is necessary to access the Google Colaboratory platform.

4. System Features

<This template illustrates organizing the **functional requirements** for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

4.1 CONVOLUTIONAL NEURAL NETWORK(CNN):

CNN stands for Convolutional Neural Network, which is a type of neural network that is commonly used in image and video analysis tasks such as image classification, object detection, and facial recognition.

CNNs are inspired by the structure and function of the human visual system, which processes visual information through layers of cells called neurons. Similarly, CNNs are designed to process visual information through layers of neurons that are arranged in a hierarchical manner.

In CNNs, the input image is passed through several layers of filters, which apply convolution operations to extract important features from the image. These filters help to identify edges, corners, and other patterns that are useful for recognizing objects in the image.

The output of each convolutional layer is then passed through a nonlinear activation function, such as ReLU (rectified linear unit), which helps to introduce nonlinearity and increase the model's ability to learn complex patterns.

Finally, the output of the last convolutional layer is passed through one or more fully connected layers, which help to classify the image into different categories or outputs. The weights of the fully connected layers are learned during the training process, where the network is trained on a dataset of labeled images.

4.2 DEEP CONVOLUTIONAL NEURAL NETWORK(CNN)

As an information processing paradigm, the neural network was inspired by the biological nervous system. It comprises a large number of neurons, which are densely coupled processing components that generate a series of real-valued activations. For example, when given input, early neurons are activated, and weighted connections from previously active neurons activate other neurons.

Depending on how neurons are employed and coupled, large causal chains and links between computational stages may be required. Deep neural networks (DNNs) are neural networks with a large number of hidden layers.

As a result of its growth, Deep Learning has made significant progress in image classification. Deep learning algorithms aim to learn the feature hierarchy automatically. At different degrees of deliberation, this self-learned component permits the framework to dissect complex contributions to yield planning capacities straightforwardly from getting to information without depending on human-made highlights.

4.3 HOW CNN WORKS?

A Convolution, Neural Network layers include Convolution, ReLU Layer, Pooling, Fully Connected, Flatten, and Normalization. The photographs would be compared piece by piece using CNN. The item is referred to as a feature or filter. CNN employs the weight matrix to extract particular characteristics from the input picture without losing information about the image's spatial organization. CNN adheres to the following layers:

The layer of Convolution is the first layer. It aligns the feature and picture before multiplying each image pixel by the feature pixel. After completing the multiplication of the associated matrix, CNN adds and divides the result by the total number of pixels, creates a map, and places the filter's value there. The feature is then moved to every other place in the picture, and the matrix output is obtained. Next, the process is repeated for the other filters.

Thus, this layer repositions the filter on the picture in every conceivable location.

1. ReLU Layer:

2. ReLU is an acronym for Rectified Linear Unit

3. Every negative value in the filter pictures will be eliminated and replaced with zeros in this layer. The function only activates a node when the input is zero and the output is zero. However, the dependent variable has a linear connection if the input grows. It enhances the neural network by increasing the amount of training.

4. Polling Layer:

5. This layer reduces the size of the picture by extracting the maximum value from the filtered image and converting it to a matrix. It also keeps overfitting at bay.

6. All Fully Connected Layer

7. This is the final layer of CNN, and it is here that the absolute categorization takes place. First, a single list is created with all of the filtered and compressed photos.

4.4 THE ARCHITECTURE OF DCNN

We developed "IDENTIFICATION of crop leaf disease" using Deep learning. We just took a step and started to collect lots of images of crop leaves. We require space capability to get the correct information. Then, at that point, we pick which calculation is ideal for tackling this issue, and we choose "Convolution Neural Network" not surprisingly (CNN). Be that as it may, we gain less precision utilizing move learning engineering, which admirably prepares and tests datasets.

Pre-processing & feature extraction, we select leaf data such as color, shape, and texture that are helpful in pattern recognition, and classification.
It gives us more than 97% accuracy on training data, more than 90% accuracy on test data, and more than 85% accuracy on validation data sets in 50 epochs. After that, we deployed this model on the flask.

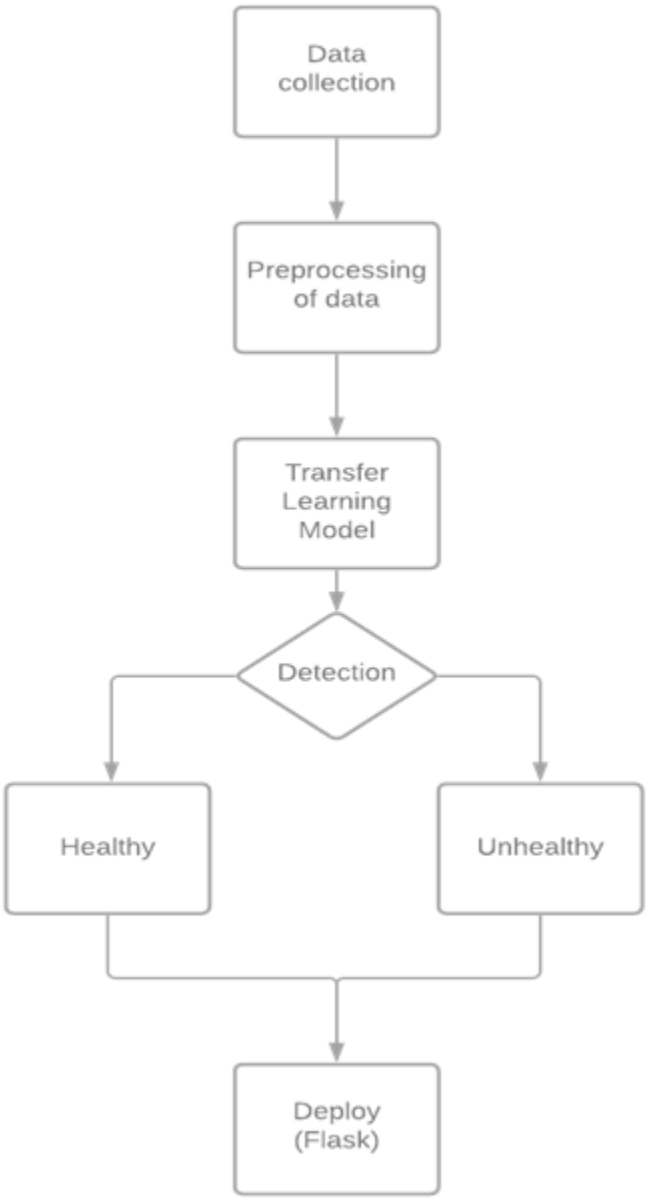


Fig 3.1 Proposed Methodology of the System

5. Nonfunctional Requirements

5.1 Software Quality Attributes

The Tomato Leaf Disease Prediction software is expected to exhibit various quality attributes to ensure its effectiveness, reliability, and usability. The following quality attributes have been identified for this software:

Reliability:

The software must consistently provide accurate disease predictions with a minimum accuracy rate of [specify percentage].
It should operate without frequent failures or crashes, ensuring reliable performance for users.
Performance:

Response time for disease predictions should not exceed [specify maximum response time].
The system should be capable of processing a minimum of [specify number] image analysis requests per minute.
Load testing will be performed to ensure optimal performance under high user demand.
Scalability:

The system should be designed to scale horizontally, accommodating a growing number of users and image analysis requests.
Availability:

The system should maintain at least a 99% uptime, allowing for scheduled maintenance periods.
High availability is crucial to ensure users can access the service when needed.
Security:

User data, especially images, must be stored and transmitted securely, following industry standards for encryption and data protection.
The system should implement robust authentication and authorization mechanisms to prevent unauthorized access.
Regular security assessments and audits will be conducted to identify and address vulnerabilities.
Maintainability:

Code should follow coding standards and be well-documented to facilitate ease of maintenance and updates.
The software will be version-controlled, ensuring a structured approach to development and maintenance.
Usability:

The user interface should be intuitive, accessible, and designed for a diverse range of users.
It should meet accessibility standards to ensure users with disabilities can easily use the system.

6. Project Plan

6.1 Team Members

Divya Purohit [EN20CS3030]
Jasnoor Singh Mac [EN20CS303029]

6.2 Division of Work

The development and implementation of the Tomato Leaf Disease Prediction system will be a collaborative effort among the project team members. The responsibilities and division of work are as follows:

Team Member 1: Divya Purohit

Responsibilities:

Data Collection and Preparation:

will be responsible for collecting and curating the dataset of tomato leaf images with labeled disease conditions for machine learning model training.
Data preprocessing tasks, including data cleaning, labeling, and augmentation
Machine Learning Model Development.

will focus on developing and fine-tuning the Convolutional Neural Network (CNN) model for disease prediction.
This includes model architecture design, hyperparameter tuning, and model training.
Model Evaluation and Testing:

will lead the testing and validation of the machine learning model's accuracy and performance.
The team member will work on defining and implementing performance metrics.

Team Member 2: Jasnoor Singh Mac

Responsibilities:

Software Development:

will be responsible for the development of the software application, including the user interface and backend components.
This includes designing the user interface, implementing image upload features, integrating the machine learning model, and creating the user management system (if applicable).

Documentation and User Support:

Team Member 2 will oversee the creation of user documentation components, including the user manual, online help system, and tutorials.
User support, such as addressing user queries and issues, will also fall under the purview of this team member.
Deployment and Maintenance:

will be involved in deploying the software to the selected platform and ensuring its continuous operation.
Ongoing maintenance and updates to the software will be managed by this team member.

Collaborative Tasks:
Both Team Member 1 and Team Member 2 will collaborate on the following aspects:

User Interface Design:

The design of the user interface will be a collaborative effort, combining Team Member 2's software development skills and Team Member 1's expertise in user experience design.
Integration and Testing:

The integration of the machine learning model into the software and comprehensive testing will involve joint efforts to ensure seamless functionality.
Project Management:

Both team members will be responsible for project management tasks, including regular team meetings, progress tracking, and issue resolution.
This division of work is designed to leverage the strengths and expertise of each team member and ensure a coordinated effort to successfully deliver the Tomato Leaf Disease Prediction system.

6.3 Time Schedule

<Tentative time requirement for a part of project to be completed.>

Appendix A: Glossary

This glossary provides definitions for key terms and acronyms used throughout the Tomato Leaf Disease Prediction Software Requirements Specification (SRS).

Disease Prediction: The process of identifying and diagnosing diseases that affect tomato plant leaves based on image analysis.

UI: Abbreviation for User Interface, which refers to the visual elements and design that users interact with to use the software.

UX: Abbreviation for User Experience, which encompasses the overall experience of users while interacting with the software, including usability, accessibility, and satisfaction.

Machine Learning: The branch of artificial intelligence (AI) that focuses on developing algorithms and models that enable software to learn and make predictions or decisions without being explicitly programmed.

Dataset: A collection of data used for training machine learning models, consisting of labeled images of tomato plant leaves in this context.

API: Abbreviation for Application Programming Interface, which defines the methods and protocols for software components to interact with each other.

Horizontal Scalability: The ability to handle increased load or demand by adding more servers or nodes to the system.
Vertical Scalability: The ability to handle increased load or demand by increasing the resources (e.g., CPU, RAM) of the existing server or node.

Authentication: The process of verifying the identity of a user or system component.

Authorization: The process of granting or denying access to specific resources or functionalities based on a user's permissions.

Backup: The process of creating a copy of data to ensure its availability in case of data loss or system failure.

Data Encryption: The process of converting data into a secure format to prevent unauthorized access.

Accessibility: The degree to which software, websites, and applications are usable by people with disabilities.

Load Testing: The process of testing a system's ability to handle a specific amount of load or traffic.

Usability Testing: A testing process that evaluates the ease with which users can interact with and navigate the software.

Compliance: The adherence to legal, regulatory, or industry-specific standards or guidelines.
Audit Trails: Logs or records that track user actions and system changes for security and accountability purposes.

Scalability: The system's ability to adapt and perform well as load or demand increases.

Usability: The measure of how user-friendly and accessible the software is for its intended users.

Interoperability: The ability of the software to work seamlessly with other systems or components.

Extensibility: The ease with which the software can be extended or expanded with new features or capabilities.

Cultural and Internationalization Requirements: Considerations for language, currency, date and time formats, and other cultural factors.

Appendix B: To Be Determined List

TBD-01: Image Dataset Selection

Rationale: The specific source and characteristics of the image dataset to be used for training the Convolutional Neural Network (CNN) are yet to be determined.
TBD-02: Model Architecture and Hyperparameters

Rationale: The exact architecture of the CNN model and the hyperparameters, such as the number of layers, filter sizes, and learning rate, are pending finalization.
TBD-03: Data Preprocessing Techniques

Rationale: The preprocessing techniques, including image resizing, normalization, and data augmentation, need to be defined.
TBD-04: Disease Categories

Rationale: The specific disease categories that the CNN model will be trained to detect are yet to be determined. This includes the list of diseases and their labels.
TBD-05: Performance Metrics

Rationale: The selection of performance metrics for evaluating the model's accuracy, precision, recall, and F1 score is pending finalization.
TBD-06: Deployment Platform

Rationale: The choice of deployment platform or environment, such as a web application or mobile app, is yet to be determined.
TBD-07: User Authentication

Rationale: Details regarding user authentication methods and security considerations are awaiting finalization.
TBD-08: User Interface Design

Rationale: The design and layout of the user interface, including the color scheme, user interactions, and accessibility features, require further development.
TBD-09: Legal and Compliance Requirements

Rationale: The legal and compliance requirements, such as data privacy regulations and intellectual property considerations, need to be specified.
TBD-10: Maintenance and Support Plan

Rationale: The plan for ongoing maintenance, updates, and user support is pending definition.
These TBD references represent areas where further decisions and details are needed for the successful execution of the "Tomato Leaf Disease Prediction using CNN" project. Each TBD item will be addressed, finalized, and documented as part of the project's development and review process.