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 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.

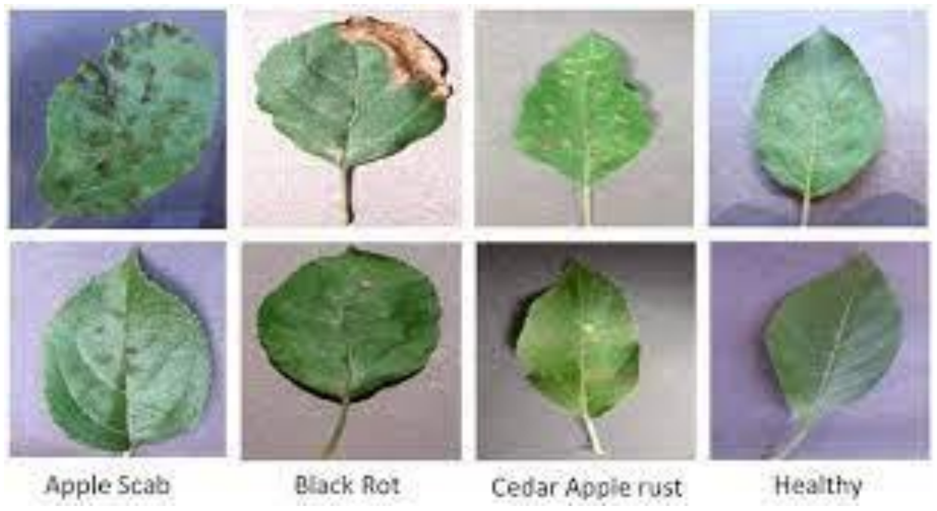

Fig 1.2 Apple Healthy and Types of Disease [13]

1.2 Document Conventions

While writing the SRS for “Tomato Leaf Disease Prediction”, we have used The Software Specification Requirement template [Section 1.5(3)]. This document uses Times font throughout the entire document. Major header lines use size 18 font. Subsections use size 14 font. All other portions of this document use size 12 font. All sections and subsections are numbered accordingly.

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:

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.

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.
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.

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

1.5 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:

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.

1.6 Product Functions

The project aims to accurately predict diseases based on user-input symptoms utilizing advanced machine learning algorithms. Its core functions include seamless user registration and authentication, intuitive symptom input, real-time analysis, and instant disease predictions. The interface provides detailed information about predicted diseases, fosters user interaction through chat-like features, and incorporates visualizations for better understanding. Users can offer real-time feedback, enhancing the system's accuracy over time. The software ensures data security, offers account management features, and maintains compatibility across devices, providing a user-friendly and reliable disease prediction experience.

1.7 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.

1.8 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.

1.9 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.

2.5 DESIGN AND IMPLEMENTATION CONSTRAINTS

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.

```
+-----------------+
|      Farmer     |
+--------+--------+
         | uploads image of
         | a diseased crop leaf
         |
         v
+-----------------+
|    System       |
+--------+--------+
         | receives image of
         | a diseased crop leaf
         |
         v
+--------+--------+
| Model Selection |
+--------+--------+
         | selects type of model
         |
         v
+--------+--------+
|    Prediction   |
+--------+--------+
         | processes image using selected model
         | discover the type of disease
         |
         v
+--------+--------+
|  Type of Disease |
 +--------+--------+
```

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 :

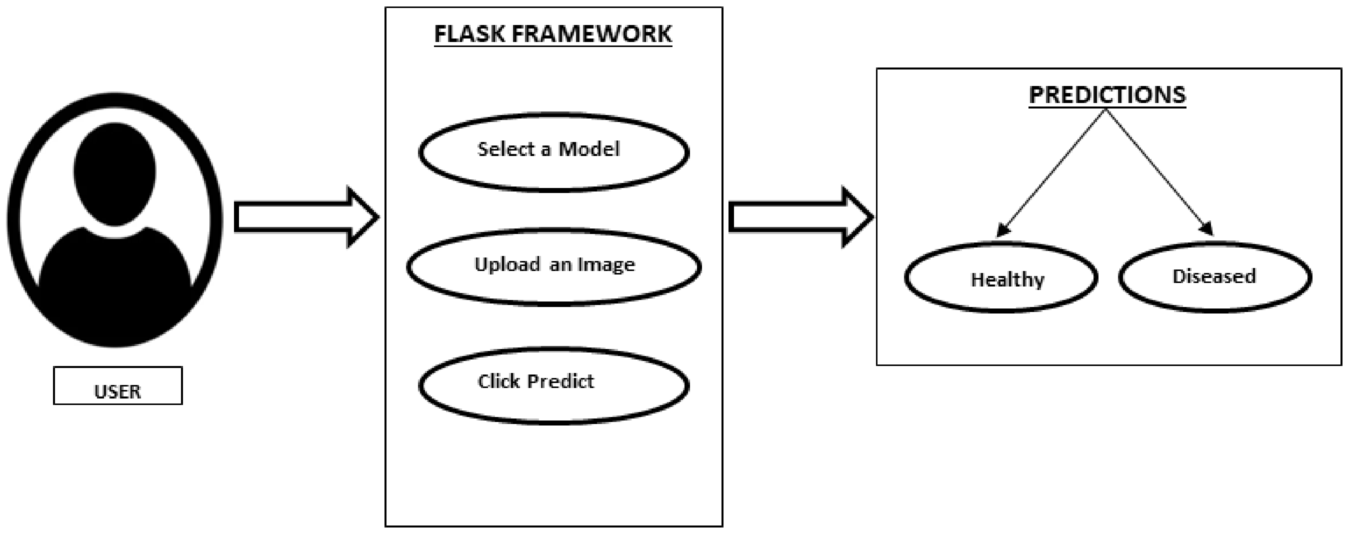

Fig. Use Case Diagram

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.

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.

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.

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.

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.

2.5.3 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.

2.6 User Documentation

As this application product is being designed and marketed to be clean, simple and easy to use, ideally, the application will require no additional explanation or support.

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.

3.2 Communications Interfaces:

A web browser is the basic requirement for this application. It includes a communicational interface in which the user has to enter the symptoms as input and based on that it will predict the disease as an output and also store the information of the user.

4. System Features

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 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.

5. Nonfunctional Requirements

5.1 Performance Requirements

• Response Time: The system should respond to user queries within 2 seconds.

• System Resource Utilization: the system should utilize system resources (CPU, memory, disk space) efficiently, with a maximum utilization rate of 70% under normal operating conditions..

5.2 Safety Requirements

Fault Tolerance: The system should continue to provide accurate disease predictions even if one of its components fails.

Accuracy and Reliability: Disease predictions made by the system must be highly accurate and reliable, with a minimum confidence level to prevent incorrect diagnoses and ensure patient safety.

Redundancy and Disaster Recovery: Implement redundancy and disaster recovery measures to minimize data loss and downtime, ensuring patient data and access are not compromised in case of system failures.

5.3 Security Requirements

Data Encryption: Requirement: User data must be encrypted both in transit and at rest using industry-standard encryption algorithms to prevent unauthorized access and ensure data confidentiality.

Secure APIs: the system integrates with external services or databases, and secure APIs (Application Programming Interfaces) must be implemented, ensuring data integrity and preventing data breaches through API endpoints.

5.4 Software Quality Attributes

Accuracy: The accuracy of disease predictions is paramount. The software must consistently provide correct diagnoses and minimize false positives and false negatives.

Scalability: As the user base and data volume grow, the software should be able to scale to handle increased demand while maintaining performance.

Performance: The system must be responsive and provide quick predictions, especially in time-sensitive healthcare situations.

Usability: The user interface should be intuitive and user-friendly, making it easy for patients to input symptoms and for healthcare professionals to interpret results.

6. Project Plan

6.1 Team Members

Divya Purohit [EN20CS3030]
Jasnoor Singh Mac [EN20CS303029]

6.2 Division of Work

The project team for "Tomato Leaf Disease Prediction" employs a cohesive approach to system development, with every member sharing equal responsibilities in tasks such as analysis, coding, and design. Their collective efforts extend to gathering functional and non-functional requirements, creating architectural designs, and designing user interfaces. This collaborative synergy is most pronounced during the coding phase, where the entire team combines their skills and dedication to bring the project to fruition.

6.3 Time Schedule

The development and implementation of the ‘Tomato Leaf Disease Prediction’ project are projected to be concluded by the conclusion of the current academic semester as specified in the project timeline.

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.