

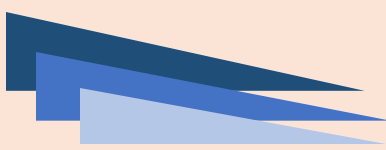
*Software Requirements Specification*  
*Version 1.0*

# LeafGuard

Theme: Green Environment

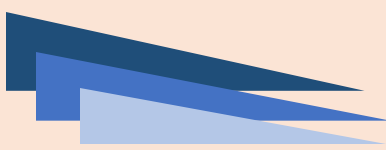
Project Name: LeafGuard

Category: Transforming Big Data with Data Science



## Contents

1.1	Background and Necessity for the Application.....	2
1.2	Proposed Solution.....	3
1.3	Purpose of the Document .....	6
1.4	Scope of Project.....	6
1.5	Constraints.....	7
1.6	Functional Requirements.....	8
1.7	Non-Functional Requirements.....	10
1.8	Interface Requirements .....	11
1.8.1	Hardware .....	11
1.8.2	Software.....	11
1.9	Project Deliverables .....	12

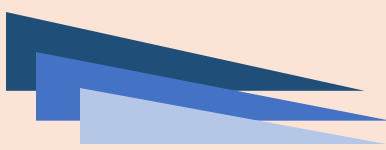


## 1.1 Background and Necessity for the Application

Leaf diseases are a significant concern for agricultural productivity and sustainability worldwide. They can cause substantial yield losses and affect the quality of produce, thereby impacting food security and economic stability for farmers. Traditional methods of leaf disease detection involve manual inspection by experts, which is time-consuming, labor-intensive, and impractical for large-scale farming operations. With the advent of technology, there has been a growing interest in leveraging Data Science and Big Data processing to address this challenge. By analyzing data from sources such as images, weather patterns, and soil conditions, more efficient methods for detecting leaf diseases can be developed. This approach also enhances the accuracy of disease detection.

The application of Big Data processing and Data Science in leaf disease detection is not just a technological advancement, but a necessity for modern agriculture. The increasing global population demands higher agricultural productivity, while resources such as land and water remain limited. Early and accurate detection of leaf diseases can help in timely intervention, reducing the spread of diseases and minimizing crop losses.





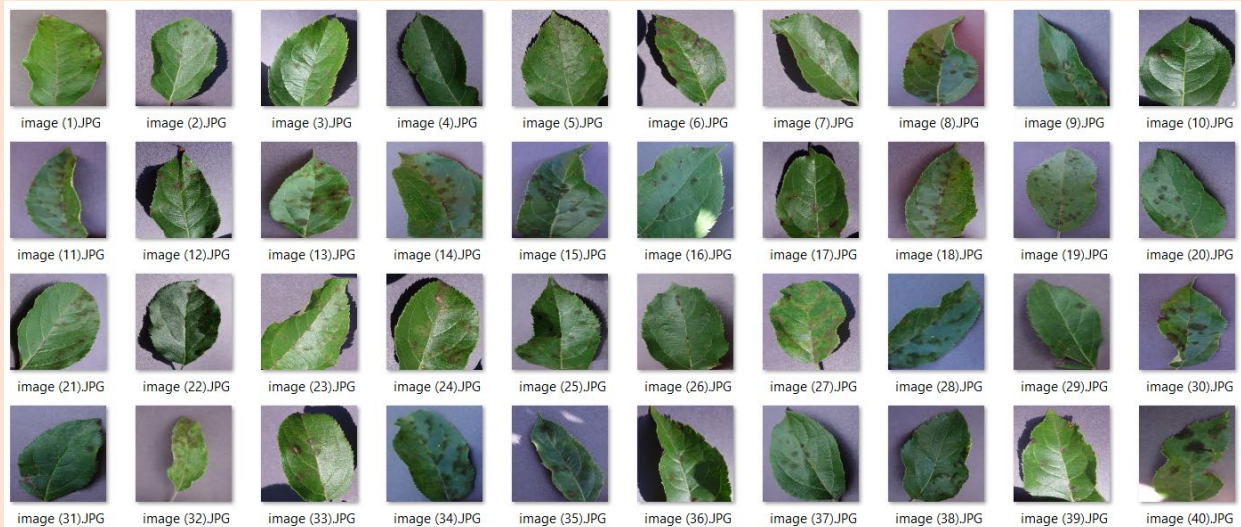
## 1.2 Proposed Solution

The **LeafGuard** application has the potential to completely transform the way plant health is monitored and diseases are managed. From modest gardens to enormous industrial farms, it can be applied in a variety of environments. The adoption of this technology will help promote environmentally responsible farming practices. To stop the spread of plant diseases and reduce their impacts, there is a necessity for reliable methods to identify plant illnesses. The **LeafGuard** application aims to address this issue by using Big Data processing and Data Science algorithms to diagnose plant diseases. It offers farmers and gardeners a quick, dependable, and cost-effective way to monitor the health of their plants and crops.

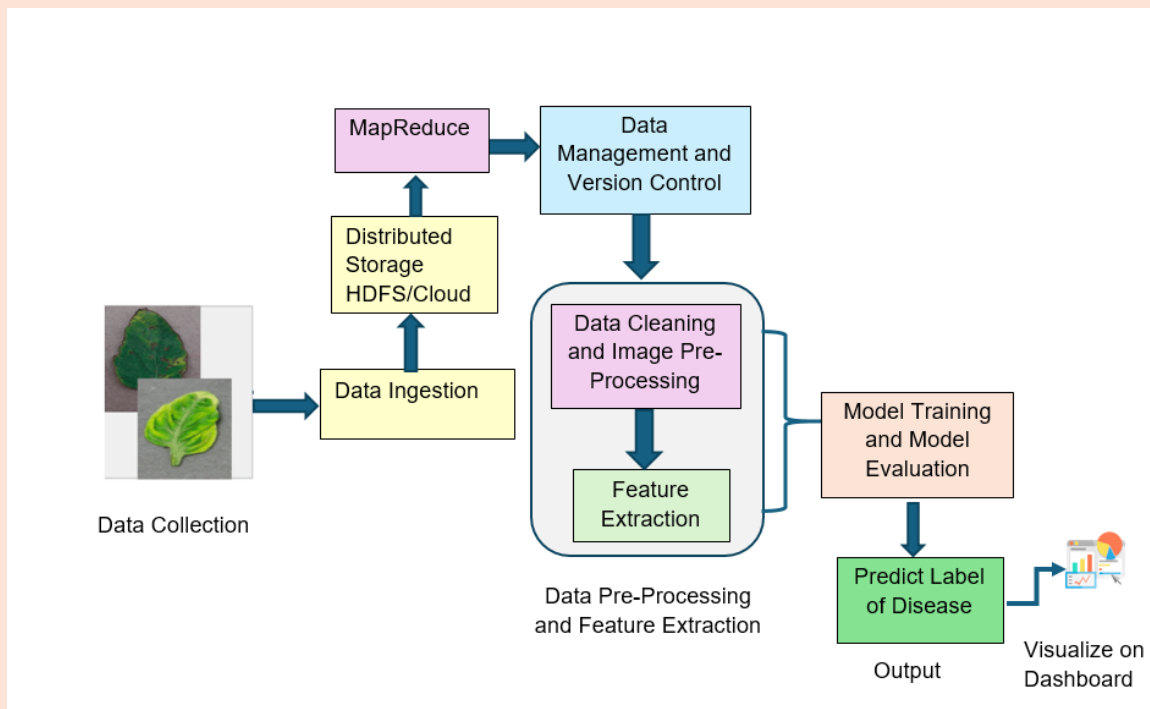
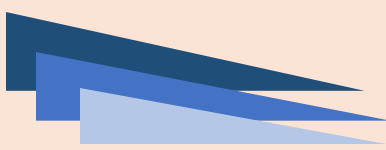
- **Data Collection:** Initially, the process begins with data collection and ingestion where high-quality images of leaves, potentially affected by various diseases, are sourced from repositories such as Kaggle. These images are then ingested into a distributed storage system such as HDFS or cloud-based storage which provides scalable and reliable data storage capabilities. The dataset is provided to you. Alternatively, you can download the dataset from:

<https://www.kaggle.com/datasets/vipooool/new-plant-diseases-dataset>.

**Hint:** Dataset of images for plant disease detection downloaded from Kaggle for implementation purpose is as follows:



The sample architecture for **LeafGuard** application can be as follows:

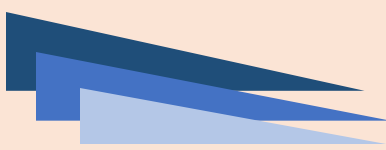


### *Sample Architecture of the Application*

**LeafGuard** can be effectively accomplished using Big Data processing and Data Science techniques. Here is how these techniques can be applied:

- **Data Pre-Processing and Feature Extraction:** Following ingestion, the data undergoes a preprocessing phase that includes cleaning and image enhancement to remove noise and standardize the dataset. Feature extraction is performed to isolate key characteristics such as color, texture, and shape which are crucial for accurate disease identification. This step is critical as it prepares the data for effective analysis by machine learning models.
- **Data Management and Version Control:** Data management and version control are integrated into the application to track and manage the datasets and preprocessing workflows, ensuring consistency and reproducibility of results. Techniques such as MapReduce are employed for efficient processing and management of large datasets, allowing for the handling of complex data transformations and analysis tasks.
- **Model Training:** Model training involves using the preprocessed data to train machine learning models, employing techniques such as cross-



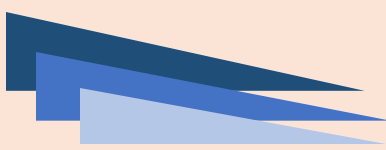


validation to fine-tune the model's performance, and ensure robustness. The evaluation phase focuses on assessing the models' accuracy, precision, and other metrics ensuring that they meet the desired performance standards.

- **Model Evaluation:** The final stage involves deploying these models to predict leaf diseases, with the results visualized on an interactive dashboard. This dashboard provides a user-friendly interface for users, enabling them to interpret the results and make informed decisions based on the data insights. The entire methodology is designed to be scalable, secure, and efficient capable of handling large volumes of data and delivering precise disease diagnostics.

In conclusion, the suggested methodology entails gathering plant picture data, preprocessing the data to eliminate any inconsistencies, and choosing suitable methods. It also involves training the model, assessing its performance, and deploying the model for usage in a Web-based application. This methodology aims to create a machine learning-based model for plant disease identification that is both precise and effective. This model can lead to early diagnosis and treatment, improving agriculture and reducing environmental degradability.





## 1.3 Purpose of the Document

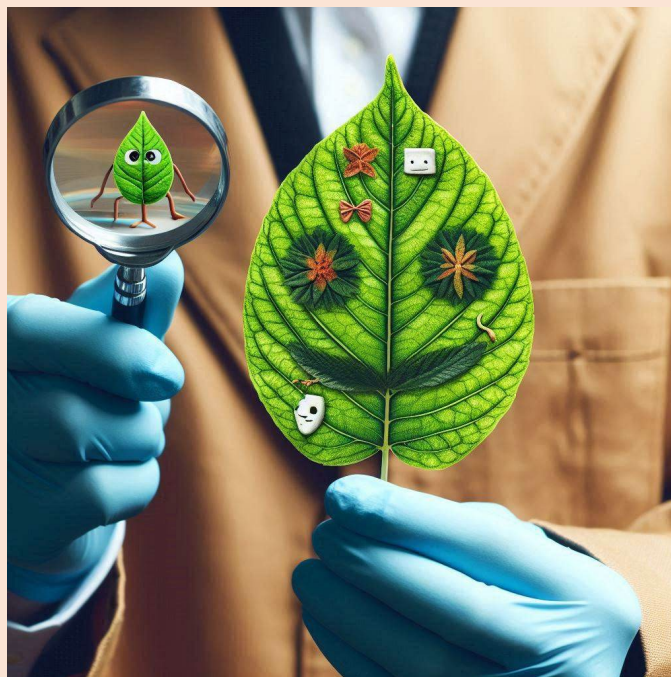
The purpose of this document is to present a trained and tested interactive model titled '**LeafGuard**'.

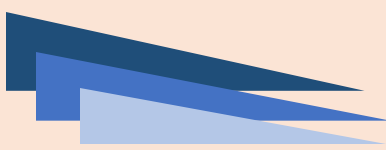
This document explains the purpose and features of Big Data Processing Using Data Science and the constraints under which it must operate. This document is intended for both stakeholders and developers of the system.

## 1.4 Scope of Project

The scope of the project is to develop a comprehensive application that can accurately and efficiently identify various leaf diseases in crops. By leveraging large datasets comprising high-resolution images, weather data, and soil conditions the project will utilize Machine Learning algorithms to analyze and classify different disease patterns.

The scope includes creating a scalable data pipeline for continuous data collection and implementing robust preprocessing techniques. Additionally, it involves developing a user-friendly interface for farmers to easily upload images and receive diagnostic results.





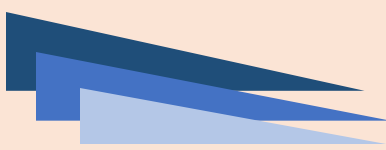
## 1.5 Constraints

Implementing an effective '**LeafGuard**' application faces various non-technical constraints that can impact its performance.

- Regulatory compliance and ethical considerations around data collection, particularly concerning agricultural practices and environmental impact, can influence the application's design and deployment.
- Operational constraints, such as limited access to consistent Internet connectivity in remote agricultural areas, may hinder data transmission and application's updates.
- Furthermore, integrating the application with existing agricultural workflows and practices poses compatibility challenges that require careful planning and stakeholder engagement.
- Additionally, seasonal variations in leaf appearance and disease patterns introduce complexity necessitating adaptive algorithms capable of handling dynamic environmental conditions.



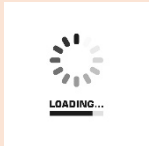




## 1.6 Functional Requirements

Functional requirements for a leaf disease detection system are crucial to ensure its effectiveness in agricultural applications. These requirements define the specific capabilities and behaviors the system must exhibit to accurately identify and manage plant diseases. Key functional requirements of the application are as follows:

- i. **Data Collection and Ingestion:** Data Collection and Ingestion involve importing and storing leaf images from multiple sources. It also includes efficiently ingesting this data into a distributed storage system such as HDFS or the cloud for easy access and processing.



- ii. **Data Storage:** Data Storage involves using distributed storage systems to securely and reliably manage large volumes of data. This approach ensures that data is accessible and protected across various locations and systems.

- iii. **Data Management and Version Control:** Data Management and Version Control involve implementing systems to track and manage different data versions, ensuring consistency and reproducibility. Additionally, MapReduce is used for processing large datasets and handling computational tasks efficiently.

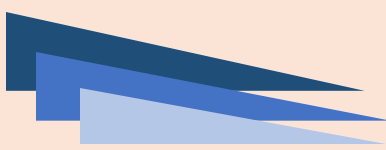


- iv. **Data pre-processing:** The application shall preprocess the collected data by cleaning, normalizing, and augmenting the images to enhance their quality. The system shall handle missing data, filter out noise, and standardize input formats to ensure consistency.

- v. **Feature Extraction:** The application shall extract relevant features from the images using techniques such as edge detection, color analysis, and texture analysis. The system shall also extract features from contextual data (for example, weather and soil conditions) to aid in disease detection.



- vi. **Model Training and Evaluation:** The application should train machine learning models using labeled datasets that include various leaf diseases. The application should support different types of models, such as Convolutional Neural Networks (CNNs), for image analysis and decision



trees for contextual data. After training the models, they are evaluated to ensure accuracy and reliability in disease detection.

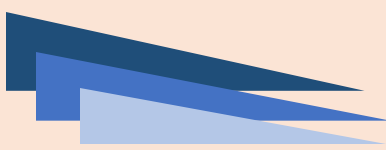
- vii. Disease Detection:** The application should analyze new leaf images and contextual data to detect and classify diseases accurately. The application should provide probability scores or confidence levels for the detected diseases. It should be able to predict disease labels based on model outputs.



- viii. User Interface:** The application should offer a user-friendly interface for farmers to upload leaf images and input additional data. The application should display the results of the disease detection including the type of disease, confidence level, and recommended actions.




- ix. Reporting and Visualization:** The application should generate reports on disease occurrences, trends, and patterns over time. The application should also provide visualizations such as heatmaps and charts to help users understand the spread and impact of diseases.





## 1.7 Non-Functional Requirements

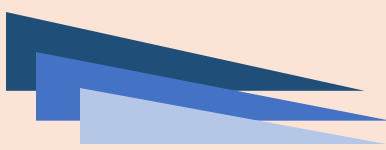
There are several non-functional requirements that should be fulfilled by the application. The application should be:

- 
1. **Usable:** The application should be designed with a clear and intuitive interface ensuring ease of use for all users.
  2. **Scalable:** The leaf disease detection application should be scalable, capable of efficiently processing large volumes of image data, and accommodating a growing number of concurrent users without compromising performance or accuracy.
  3. **Accuracy:** The application must achieve a high level of accuracy in identifying and classifying diseases, based on benchmark datasets and standard evaluation metrics. This requirement ensures reliable disease diagnosis and effective management strategies for farmers, supporting sustainable agricultural practices and minimizing crop losses.
  4. **Robustness:** The application must demonstrate robustness by maintaining consistent performance across varying environmental conditions such as different lighting conditions, diverse plant species, and varying leaf orientations.
  5. **Reliable:** The application should consistently provide accurate results and be reliable in various situations and environments.
- 
- 



These are the bare minimum expectations from the project. It is a must to implement the FUNCTIONAL and NON-FUNCTIONAL requirements given in this SRS.

Once they are complete, you can use your own creativity and imagination to add more features if required.



## 1.8 Interface Requirements

### 1.8.1 Hardware

---

Intel Core i5/i7 Processor or higher  
8 GB RAM or higher  
Color SVGA  
500 GB Hard Disk space  
Mouse  
Keyboard

### 1.8.2 Software

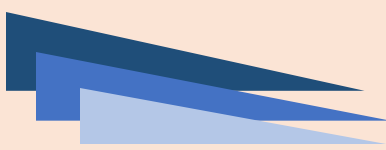
---

Technologies to be used:

1. **Data Store:** HDFS, Apache HBase, MongoDB, or CSV
2. **Backend:** Apache Spark or Apache Hive
3. **Programming/IDE:** R programming/ Python 3.11.4 or higher , Jupyter Notebook, Anaconda 23.1.0 or higher, or Google Colab
4. **Libraries:** OpenCV, TensorFlow, scikit-learn, Pandas, NumPy, PyTorch, Matplotlib, and Seaborn
5. **Visualization:** Tableau Desktop







## 1.9 Project Deliverables

You will design and build the project and submit it along with a complete project report that includes:

- Problem Definition
- Design Specifications
- Diagrams such as User Flow Diagram/User Journey Map
- Test Data Used in the Project
- Project Installation Instructions
- Link of GitHub for accessing the uploaded project code (Link should have public access)
- Detailed Steps to Execute the Project
- Link of Published Blog

The source code, including .ipynb files for Jupyter Notebook and Google Colab, should be shared via GitHub. Appropriate access permissions should be granted to users to allow testing for Jupyter Notebook and Google Colab. The consolidated project must be submitted on GitHub with a ReadMe.doc file listing assumptions (if any) made at your end.

Provide the GitHub URL where the project has been uploaded for sharing. The repository on GitHub should have public access. Documentation is a very important part of the project; hence, all crucial aspects of the project must be documented properly. Ensure that documentation is complete and comprehensive.

You should publish a blog of minimum 2000 words on any free blogging Website such as Blogger, Tumblr, Ghost or any other blogging Website. The link of the published blog should be submitted along with the project documentation.

**Submit a video (.mp4 file) demonstrating the working of the application, including all the functionalities of the project. This is MANDATORY.**

~~~ End of Document ~~~