AN AI & CLOUD-BASED COLLABORATIVE PLATFORM FOR PLANT DISEASE IDENTIFICATION, TRACKING AND FORECASTING FOR FARMERS

A Mini Project Report submitted to

JNTU Hyderabad in partial fulfilment

of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

G. SRINITH REDDY	**22S11A05J4**
P. AKSHARA	**22S11A05D4**
D. MANISHA	**22S11A05F4**
K. HIMA BINDU	**22S11A05E9**

Under the Guidance of

Dr. M. JAGANATHAN

B.E., M.E., PhD.

Associate Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MALLA REDDY INSTITUTE OF TECHNOLOGY AND SCIENCE

(Approved by AICTE New Delhi and Affiliated to JNTUH)

(Accredited by NBA & NAAC with "A" Grade)

An ISO 9001: 2015 Certified Institution

Maisammaguda, Medchal (Dist), Hyderabad -500100, Telangana.

MAY – 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MALLA REDDY INSTITUTE OF TECHNOLOGY AND SCIENCE

(Approved by AICTE New Delhi and Affiliated to JNTUH)
(Accredited by NBA & NAAC with "A" Grade)
An ISO 9001: 2015 Certified Institution
Maisammaguda, Medchal (Dist), Hyderabad -500100, Telangana.
MAY – 2025

CERTIFICATE

This is to certify that the mini Project entitled **"AN AI & CLOUD-BASED COLLABORATIVE PLATFORM FOR PLANT DISEASE IDENTIFICATION, TRACKING AND FORECASTING FOR FARMERS"** has been submitted by **Gaddam Srinith Reddy (22S11A05J4),Patlolla Akshara (22S11A05D4), Daroor Manisha(22S11A05F4) and Koyada HimaBindu (22S11A05E9)** in partial fulfilment of the requirements for the award of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE & ENGINEERING.** This record of bonafide work was carried out by them under my guidance and supervision. ***The result embodied in this Mini project report has not been submitted to any other University or Institute for the award of any degree.***

Dr. *M. JAGANATHAN*
Associate Professor
Project Guide

Mrs. K. MAMATHA
Head of the Department

External Examiner

ACKNOWLEDGEMENT

The Mini Project work carried out by our team in the Department of Computer Science and Engineering, Malla Reddy Institute of Technology and Science, Hyderabad. ***This work is original and has not been submitted in part or full for any degree or diploma of any other university.***

We wish to acknowledge our sincere thanks to our project guide **Dr. M. Jaganathan**, Associate Professor of Computer Science & Engineering for formulation of the problem, analysis, guidance and her continuous supervision during work.

We acknowledge our sincere thanks to **Dr.Vaka Murali Mohan**, Principal and **Mrs. K. Mamatha**, Head of the Department and Coordinator, faculty members of CSE Department for their kind cooperation in making this Mini Project work a success.

We extend our gratitude to **Sri. Ch. Malla Reddy**, Founder Chairman **MRGI** and **Sri. Ch. Mahender Reddy**, Secretary MRGI, **Dr.Ch. Bhadra Reddy**, President MRGI, **Sri. Ch. Shalini Reddy**, Director MRGI, **Sri. P. Praveen Reddy**, Director MRGI, for their kind cooperation in providing the infrastructure for completion of our Mini Project.

We acknowledge our special thanks to the entire teaching faculty and non-teaching staff members of the Computer Science & Engineering Department for their support in making this project work a success.

G. SRINITH REDDY (22S11A05J4) _____

P. AKSHARA (22S11A05D4) _____

D. MANISHA (22S11A05F4) _____

K. HIMA BINDU (22S11A05E9) _____

INDEX

ABSTRACT

Tomato plant diseases significantly impact global agricultural productivity, causing major crop losses. Early and accurate detection is essential but often hampered by limited access to expert diagnosis. This project presents an efficient deep learning-based approach using a lightweight Convolutional Neural Network (CNN) for automatic identification of tomato leaf diseases. The model leverages a reduced number of layers to ensure low computational complexity, making it suitable for real-time applications. To enhance the dataset without increasing data collection efforts, augmentation techniques such as rotation, shift, shear, zoom, and flipping are applied. The model is trained and tested on the Plant Village dataset, which includes ten tomato leaf classes—Bacterial Spot, Early Blight, Healthy, Late Blight, Leaf Mold, Septoria Leaf Spot, Target Spot, Tomato Mosaic Virus, Tomato Yellow Leaf Curl Virus, and Two-Spotted Spider Mite. The proposed solution achieves high classification accuracy while consuming less storage and computational resources compared to traditional deep CNN models, making it ideal for mobile and edge device deployment in agricultural settings.

LIST OF FIGURES

1. INTRODUCTION

Agriculture remains the backbone of many economies, and tomato is one of the most widely cultivated crops due to its nutritional value and economic demand. However, tomato crops are highly vulnerable to numerous leaf diseases, which reduce yield and quality, causing significant economic losses. The timely detection of these diseases is critical to managing crop health and optimizing productivity.

Traditionally, disease identification has relied on visual inspection by agricultural experts, which is time-consuming, subjective, and often unavailable in rural regions. With the rise of digital agriculture, image-based disease recognition has gained traction, but conventional machine learning models demand handcrafted features and expert preprocessing, limiting scalability and performance.

Deep learning, particularly Convolutional Neural Networks (CNNs), has revolutionized image classification by learning features automatically from raw data. This project leverages a custom lightweight CNN to detect tomato leaf diseases efficiently. The model is trained on a publicly available dataset—Plant Village—which includes images for ten tomato leaf conditions: Bacterial Spot, Early Blight, Healthy, Late Blight, Leaf Mold, Septoria Leaf Spot, Target Spot, Tomato Mosaic Virus, Tomato Yellow Leaf Curl Virus, and Two-Spotted Spider Mite.

To address data scarcity and improve generalization, the dataset is augmented using techniques such as flipping, rotating, shearing, shifting, and zooming. These transformations simulate real-world variations in leaf images without collecting additional samples. The model is built with fewer convolutional and pooling layers to reduce training time and memory usage while maintaining high accuracy.

This work focuses on achieving a balance between classification performance and computational efficiency. The ultimate goal is to develop a disease detection system that can be embedded into mobile applications or low-power handheld devices, allowing farmers to take a photo of a tomato leaf and receive an instant diagnosis. This would eliminate the dependence on expert consultation and enable timely disease management, especially in remote areas.

1.1 Objective, Scope, and Goal

Objective:

- To develop a lightweight CNN model capable of accurately classifying tomato leaf diseases using image data.
- To reduce the computational overhead while maintaining high accuracy.
- To deploy a solution suitable for mobile or low-resource environments.

Scope:

- Image-based classification using Plant Village dataset with 10 tomato disease categories.
- Application of image preprocessing and augmentation to enhance training.
- Evaluation of model performance using accuracy, precision, recall, and confusion matrix.
- Deployment on devices with limited processing power (e.g., smartphones).

Goal:

- Provide farmers with an accessible and efficient disease detection tool.
- Reduce dependence on manual inspection and expert advice.
- Increase agricultural productivity and reduce crop loss through early detection.

2. SYSTEM ANALYSIS

2.1 EXISTING SYSTEM:

Description:

- Traditional CNNs such as VGG16, ResNet50, and Inception are used for crop disease classification.
- Requires powerful hardware (GPUs) and large storage.
- Often trained on large datasets with complex architectures.

2.1.1 Disadvantages:

- High computational and memory requirements.
- Not suitable for mobile or real-time applications.
- Longer training and inference times.

2.2 PROPOSED SYSTEM:

Description:

- Custom lightweight CNN model with fewer layers.
- Trained on an augmented Plant Village dataset.
- Focus on optimizing speed and size while preserving accuracy.

2.2.1 Advantages:

- Low memory footprint and faster execution.
- High accuracy (~90–95%) with simple architecture.
- Ideal for deployment on mobile and handheld devices.
- Less dependence on external hardware or cloud servers.

3. LITERATURE REVIEW

Tomato leaf disease detection has been a widely researched area due to its direct impact on food security and agricultural economics. Several studies have explored both traditional and modern techniques for automated detection of diseases in crops, particularly using image processing and machine learning techniques.

One of the foundational works in this domain is by Mohanty et al. (2016), where a deep CNN based on Alex Net and Google Net architectures was used to classify plant diseases across 38 classes from the Plant Village dataset. The model achieved over 99% accuracy, but its high computational requirement made it unsuitable for real-time deployment on edge devices.

Ferentinos (2018) further extended this work by evaluating various CNN architectures (Alex Net, LeNet, and VGG) for crop disease recognition. Although VGG-based models offered high accuracy, they required considerable training time and hardware resources. These models are effective but not practical for farmers in rural areas who may not have access to high-end systems.

To address these limitations, several researchers have explored lightweight CNNs. For example, To (2019) proposed using Mobile Net and Squeeze Net for plant disease classification. These models significantly reduced inference time and model size, making them suitable for mobile applications, though sometimes at the cost of a slight drop in accuracy.

Another important contribution came from Amara et al. (2017), who used LeNet architecture on banana leaf disease images. The model demonstrated that even simple CNNs can yield effective classification results if trained well with preprocessing and augmentation.

Data augmentation is another important component in disease classification. Shorten and Khosh go (2019) surveyed various augmentation strategies and concluded that simple operations like flipping, zooming, rotating, and shifting significantly improve model generalization. These techniques are particularly useful when working with limited datasets, as they help prevent overfitting and improve robustness.

In the context of tomato disease classification, researchers like Brahimi et al. (2017) utilized transfer learning with deep networks such as Res Net and Dense Net to identify multiple tomato leaf diseases. While these models achieved impressive accuracy levels (~97%), their deployment on low-resource platforms remained a challenge.

Recent trends show a shift towards hybrid or ensemble models that combine CNNs with other techniques like attention mechanisms or fuzzy logic to improve interpretability and precision. However, such methods often reintroduce complexity that conflicts with real-time needs.

The proposed work stands out by focusing on a custom-built lightweight CNN model specifically tailored for 10 tomato disease classes. Instead of leveraging pre-trained heavyweight models, this approach builds a minimal yet effective architecture from scratch. Additionally, extensive data augmentation ensures that the model generalizes well to unseen data.

In summary, while many approaches focus on accuracy, fewer address the balance between performance and resource constraints. This project fills that gap by offering a model with strong classification capability and low resource demand, ideal for practical, real-world deployment.

4. SYSTEM DESIGN

4.1 System Architecture

User Interface Layer

- Mobile and Web-based interface where farmers:
    - Upload images of affected plants
    - Receive disease diagnoses and treatments
    - View outbreak maps and forecasts

Data Acquisition and Preprocessing Layer

- Captures images and metadata (GPS, timestamp)
- Performs preprocessing like resizing, noise reduction, color normalization

AI-Based Disease Detection

- Uses Convolutional Neural Networks (CNNs) like Efficient Net or YOLOv8 to classify plant diseases from leaf images
- Outputs probability, disease type, and severity

Tracking Module

- Maps disease occurrence using GPS coordinates
- Identifies clusters of outbreaks using clustering algorithms (e.g., DBSCAN)
- Powered by Geo Pandas, folium

Benefits of This Architecture

- Scalable: Modular design supports expansion for more crops/diseases
- Real-time: Integration with weather APIs and live map updates
- Farmer-Friendly: Simple interface with backend automation
- Data-Driven: Continuous learning from user-submitted data

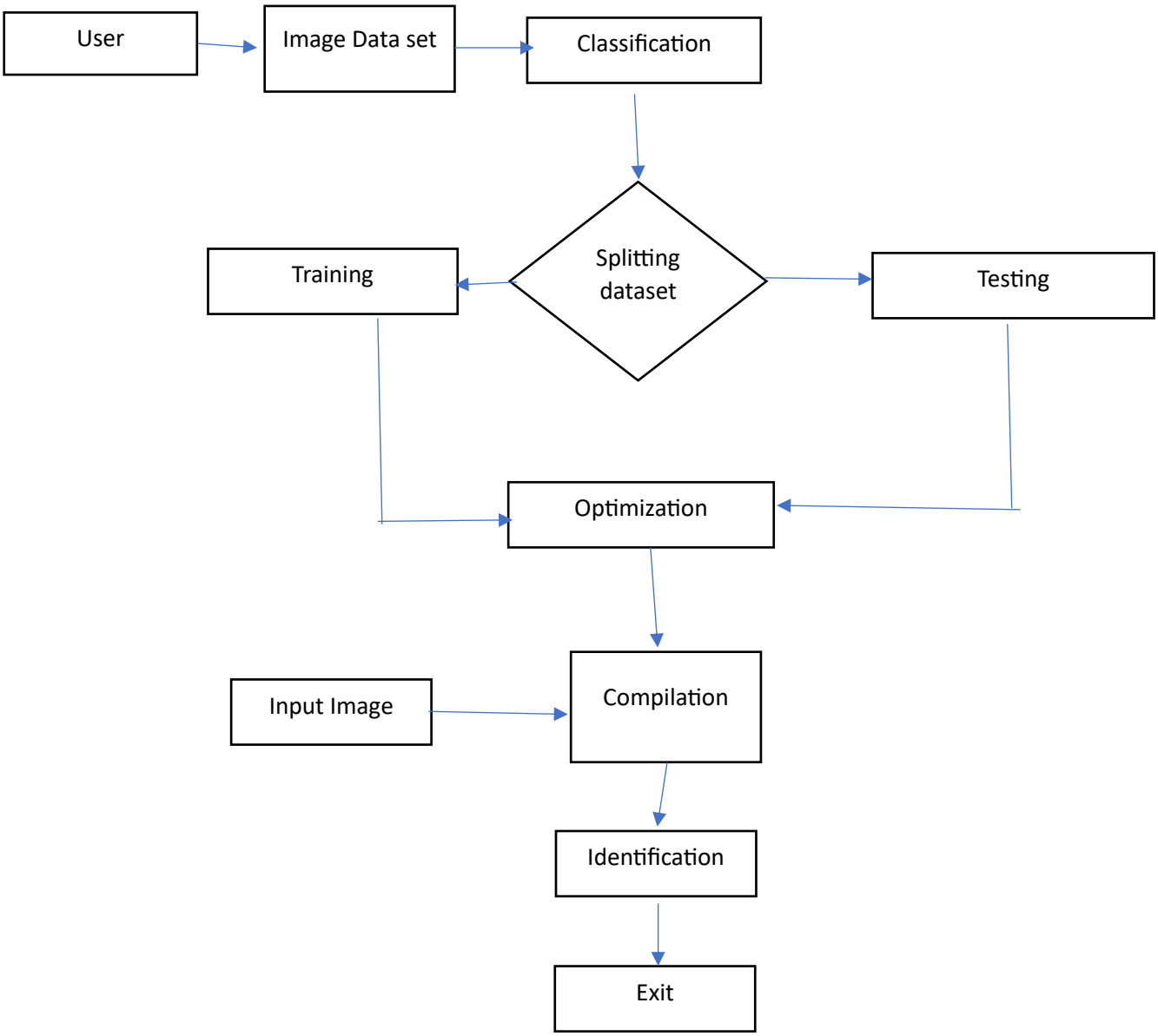


Fig 4.1: SYSTEM ARCHITECTURE

4.2 MODULES USED IN PROJECT

TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

NumPy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with Python. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or

tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

4.3 UML diagram

Visualizes the system before actual development.

Clarifies requirements and processes for developers and stakeholders.

Improves maintainability and understanding of complex systems.

Helps in modular design, reuse, and scalability.

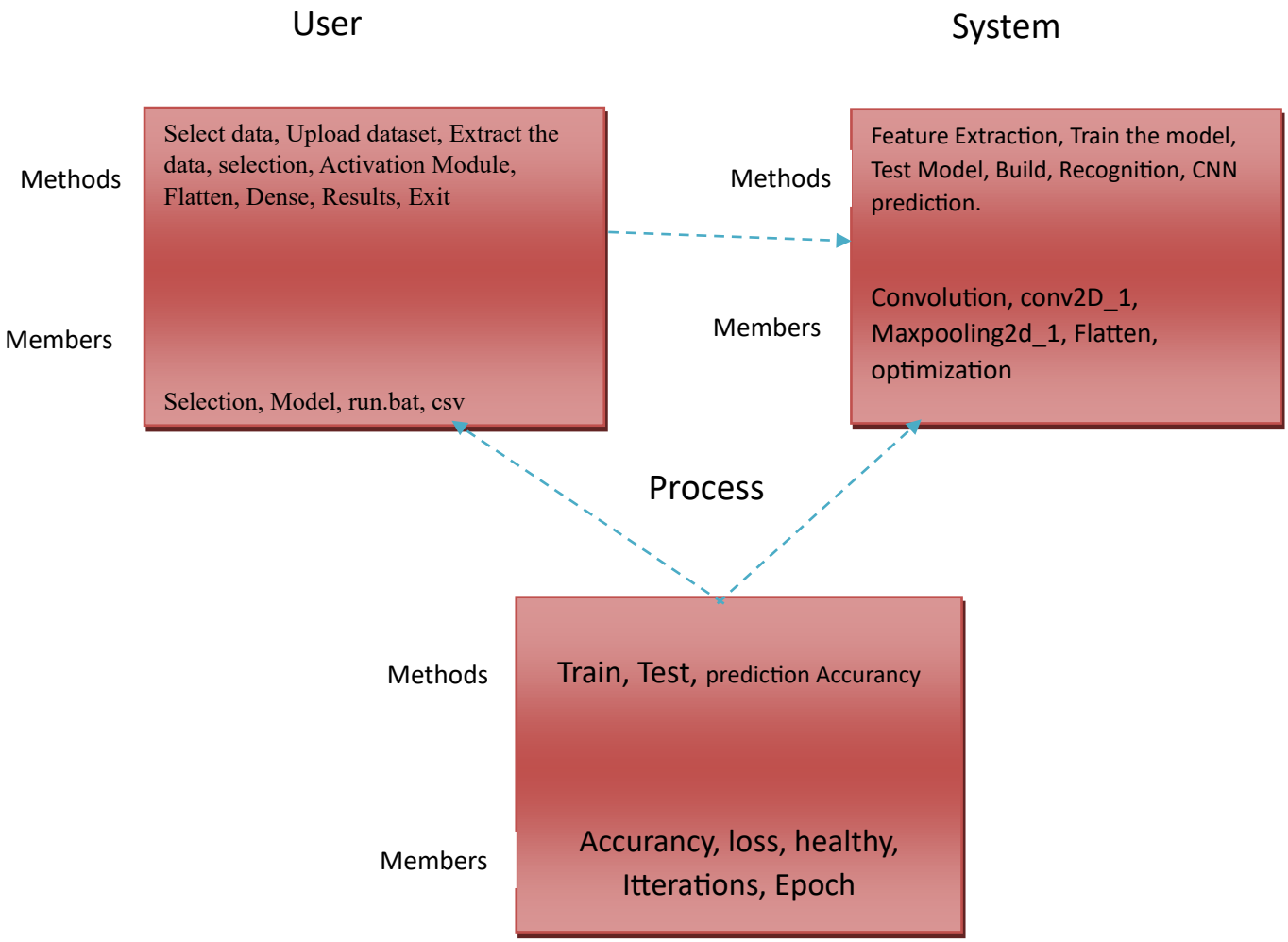


Fig 4.2: UML DIAGRAM

4.3.1 USE CASE Diagram

About Use Case Diagram

A Use Case Diagram is one of the most fundamental diagrams in UML (Unified Modeling Language). It visually represents the functional requirements of a system from the user's perspective. It shows who interacts with the system (called actors) and what they do (called use cases).

Purpose of a Use Case Diagram

- Define the scope of the system.
- Show interactions between users and system functions.
- Serve as a communication tool between stakeholders and developers.
- Capture functional requirements clearly and simply

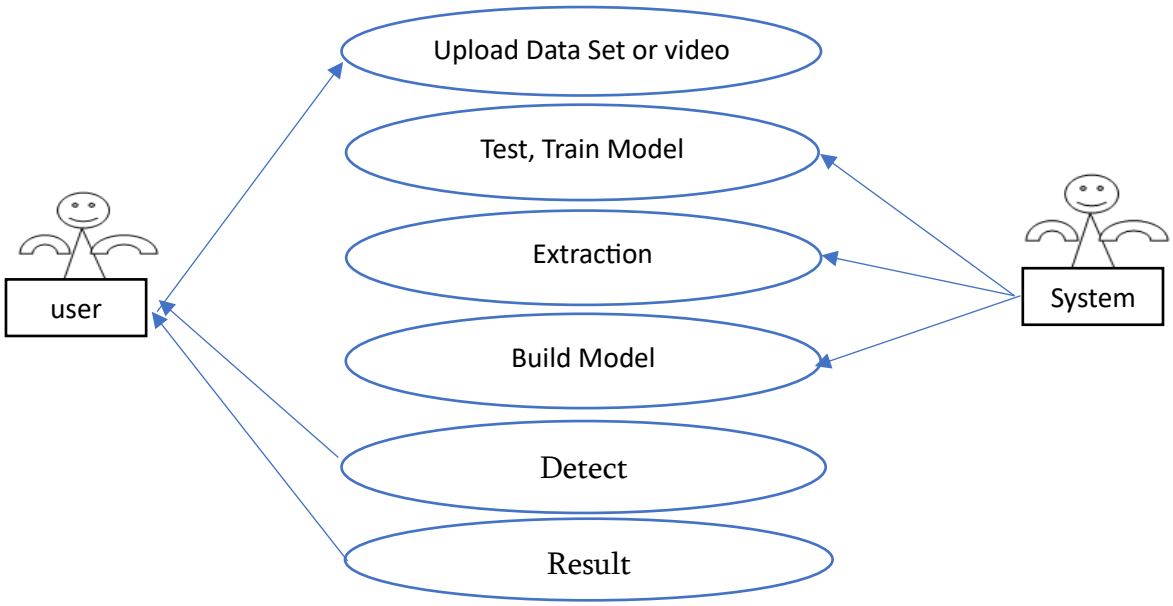


Fig 4.3: Use Case Diagram

Actors:

- Farmer
- System (AI Model, Cloud Database)
- Admin/Agri Expert

Main Use Cases:

- Upload plant image
- Identify disease
- View diagnosis result
- Track disease spread
- View forecasted outbreaks

4.3.2 SEQUENCE DIAGRAM

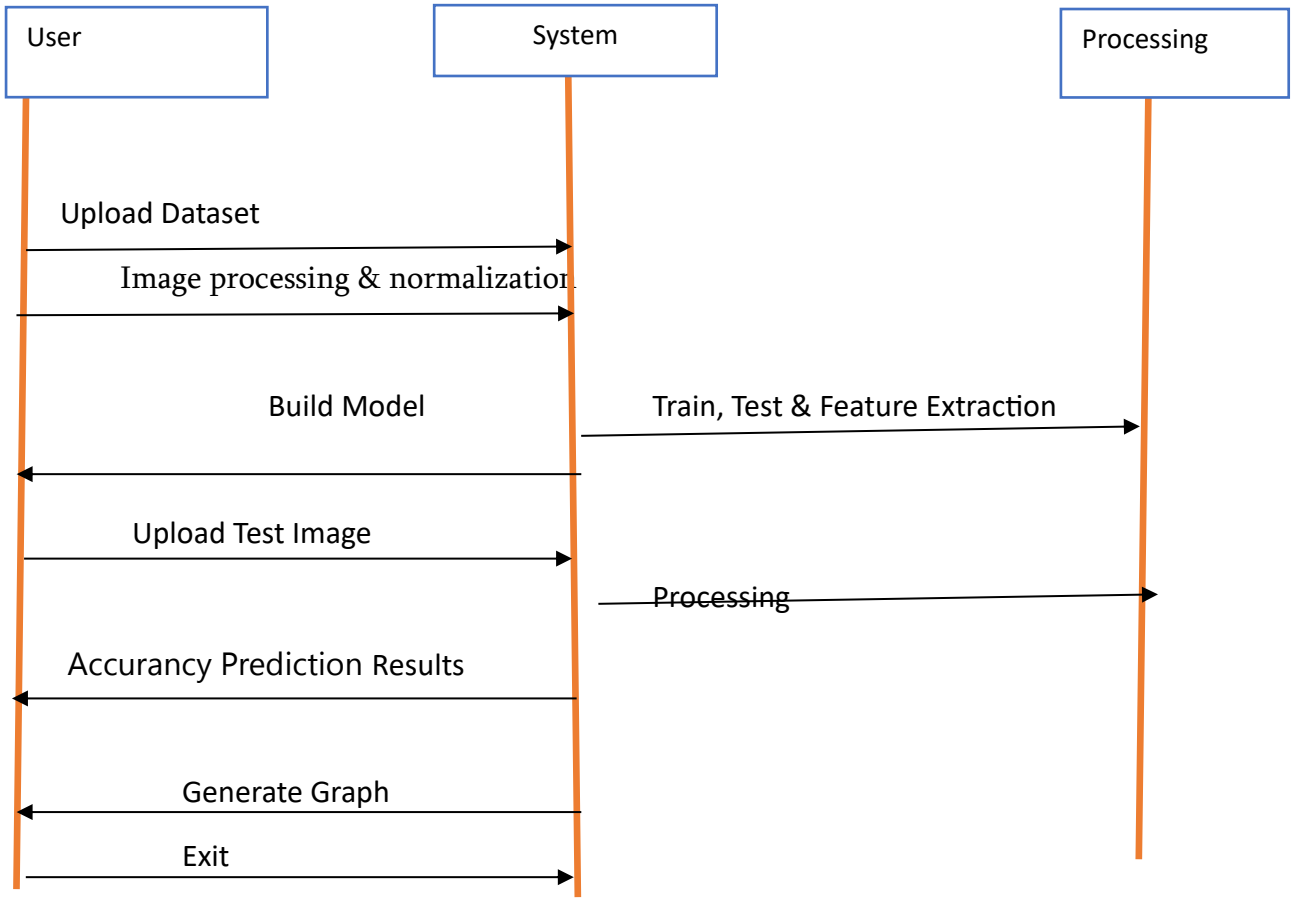


Fig 4.4: Sequence Diagram

4.4 SYSTEM REQUIREMENT

4.4.1 Software Requirements

Component	Description
Operating System	Windows 10/11, Linux (Ubuntu), or macOS
Programming Language	Python 3.7 or later
Development Tools	Jupyter Notebook, VS Code, PyCharm
Libraries & Frameworks	TensorFlow, Keras, OpenCV, Scikit-learn, Geo Pandas, Flask/Stream lit
Database	SQLite, Firebase, or PostgreSQL (for storing user data and disease history)
Browser	Chrome, Firefox, or Edge (for accessing dashboards)
Cloud Platform	AWS / Azure / Google Cloud (optional for deployment and storage)

4.4.2. Hardware Requirements

Component	Minimum Requirement	Recommended Specification
Processor	Intel Core i3 or AMD equivalent	Intel i5/i7 or equivalent
RAM	4 GB	8 GB or more
Storage	100 GB (HDD or SSD)	256 GB SSD or more
Graphics	Integrated GPU (for testing only)	Dedicated GPU (NVIDIA GTX 1050 or higher)
Camera	Smartphone or webcam for image capture	HD camera
Internet	Required for cloud services & updates	Stable broadband connection

5. INPUT & OUTPUT DESIGN

5.1 Input Design

Input Design is concerned with how data enters the system ensuring accuracy, efficiency, and ease of use.

Major Inputs in the System:

Input Type	Source	Description
Leaf Image Upload	Farmer / User	High-quality image of the infected plant/leaf captured via camera/smartphone
Crop Information	User Form	Crop type, variety, and planting date
Location Data (GPS)	Mobile / Web App	Geo-tagging to locate disease outbreaks geographically
Weather Data (API)	External APIs (optional)	Used for forecasting disease spread based on environmental conditions
User Credentials	Login/Register form	To authenticate farmers and system administrators

Input Validation:

- File type/image size checks for uploads
- Mandatory field checks in forms
- Real-time feedback for missing/incorrect entries

5.2 Output Design

Output Design defines how the processed data is presented to users, ensuring it is understandable and actionable.

Major Outputs of the System:

Output Type	Medium	Description
Disease Diagnosis Result	Web/Mobile Interface	Displays predicted disease name, severity level, and suggested treatment
Outbreak Heat Map	Map Dashboard	Shows geolocated clusters of plant disease cases
Forecast Report	Web Dashboard / Email	Displays predicted risk levels based on time and region
Notifications/Alerts	SMS / App Notification	Sends alerts for new outbreaks or high-risk forecasts
User Report	Downloadable PDF/Excel	Summary of user uploads, diagnoses, and disease history

Output Considerations:

- User-friendly UI/UX with icons and color codes
- Language support for local farmers (optional)
- Printable/downloadable reports for agronomists

6. SOFTWARE ENVIRONMENT

6.1 What is Python?

Below are some facts about Python.

- Python is currently the most widely used multi-purpose, high-level programming language.
- Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
- Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
- Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, Py Qt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Open cv, Pillow)
- Web scraping (like Scrapy, Beautiful Soup, Selenium)
- Test frameworks
- Multimedia

Advantages of Python

Let's see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading,

databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet of Things. This is a way to connect the language with the real world.

6. Simple and Easy

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory.

8. Object-Oriented

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world.

9. Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Advantages of Python Over Other Languages

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbon.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like JDBC (Java Data Base Connectivity) and ODBC (Open Data Base Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java,I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

History of Python

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers.

6.2 What is Machine Learning

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based

"learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

Categories of Machine Leaning

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate, and solve complex problems. On the other side, AI is still in its initial stage and have not surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, “to make decisions, based on data, with efficiency and scale”.

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process.

These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently.

Challenges in Machines Learning

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

1. Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.
2. Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.
3. Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.
4. No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.
5. Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.
6. Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.
7. Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.

Applications of Machines Learning

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML

- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction

- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

How to Start Learning Machine Learning?

Arthur Samuel coined the term "Machine Learning" in 1959 and defined it as a "Field of study that gives computers the capability to learn without being explicitly programmed".

And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is The Best Job of 2019 with a 344% growth and an average base salary of $146,085 per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let's get started!!!

How to start learning ML?

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don't know these, never fear! You don't need a Ph.D. degree in these topics to get started but you do need a basic understanding.

(a) Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

(b) Learn Statistics

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!! Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c) Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python

So if you want to learn ML, it's best if you learn Python! You can do that using various online resources and courses such as Fork Python available Free on GeeksforGeeks.

Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

(a) Terminologies of Machine Learning

- Model – A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.

- Feature – A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
- Target (Label) – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
- Training – The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- Prediction – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

(b) Types of Machine Learning

- Supervised Learning – This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
- Unsupervised Learning – This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
- Semi-supervised Learning – This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- Reinforcement Learning – This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

Advantages of Machine learning

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like

Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

Disadvantages of Machine Learning

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

   ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time.

Python Development Steps

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

- Print is now a function.
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e., int. long is int as well.

- The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
- Text Vs. Data Instead of Unicode Vs. 8-bit

Purpose

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

6.3 Install Python Step-by-Step in Windows and Mac

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

How to Install Python on Windows and Mac

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system..

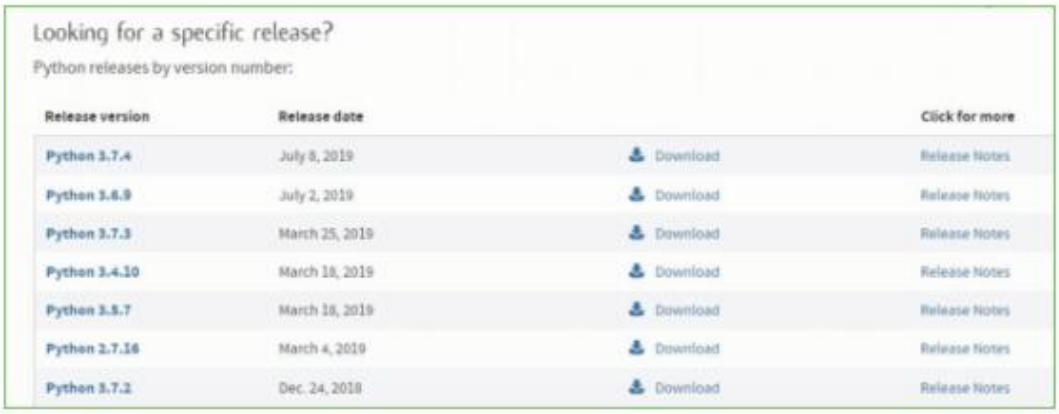
Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.org

Now, check for the latest and the correct version for your operating system.

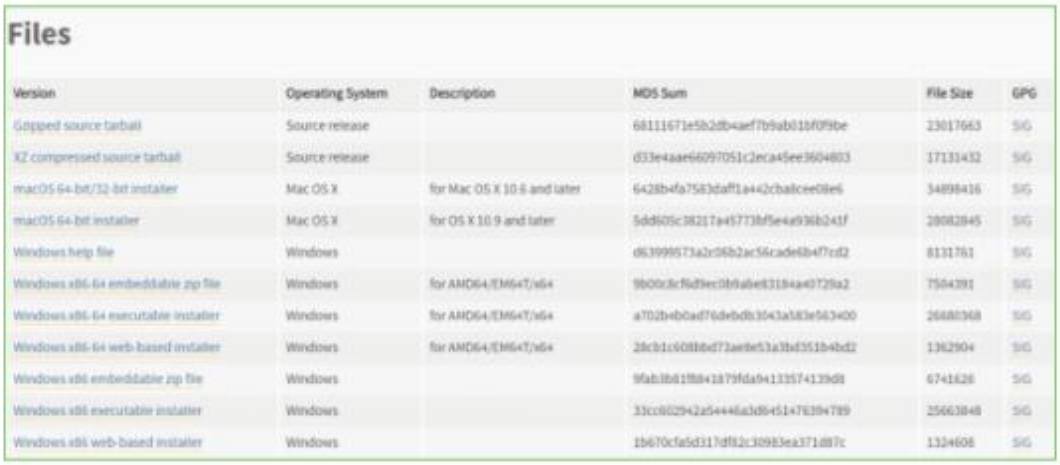
Step 2: Click on the Download Tab.

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4



Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.



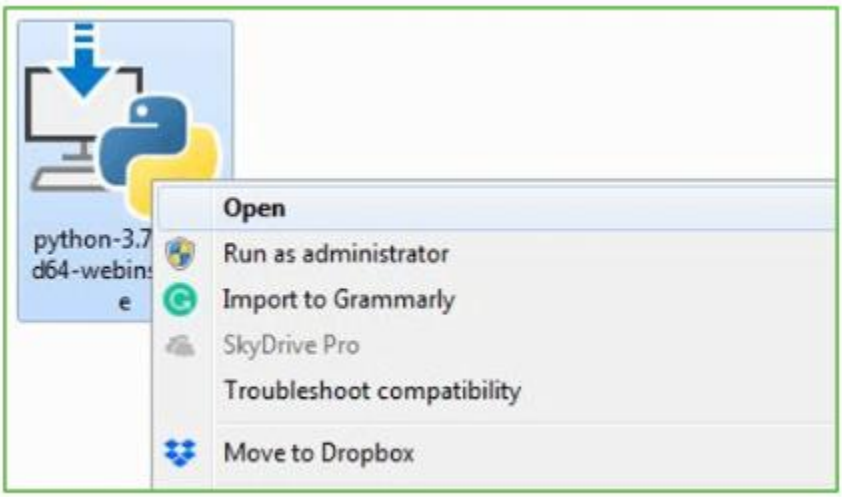
- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.

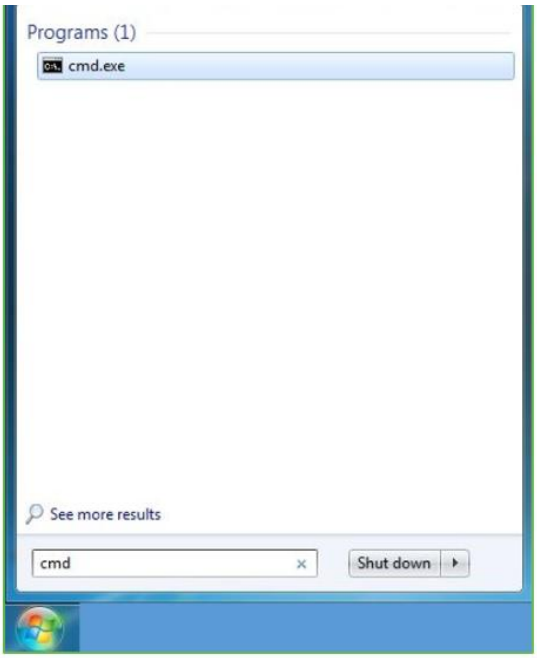
Step 3: Click on Install NOW After the installation is successful. Click on Close.

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Verify the Python Installation

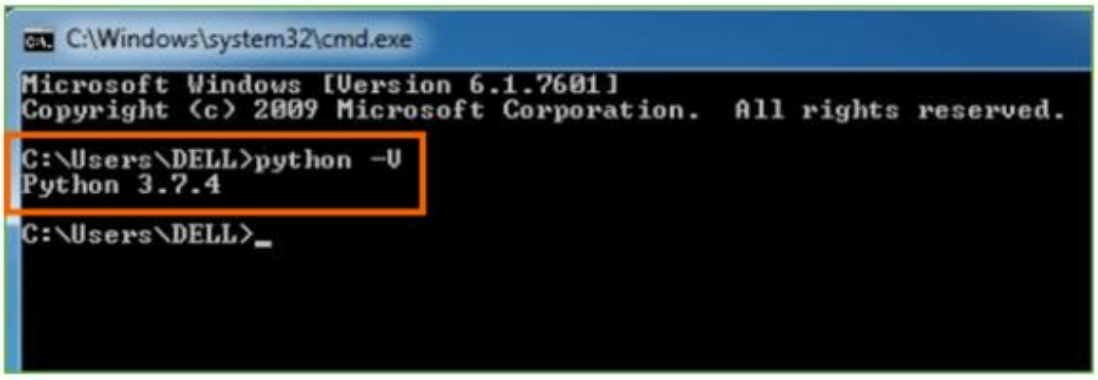
Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python –V and press Enter.



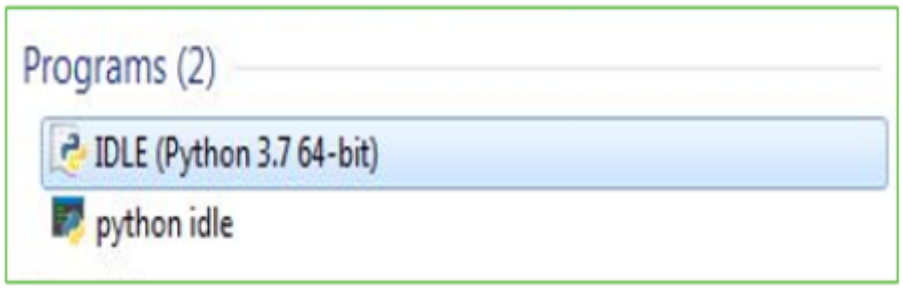
Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

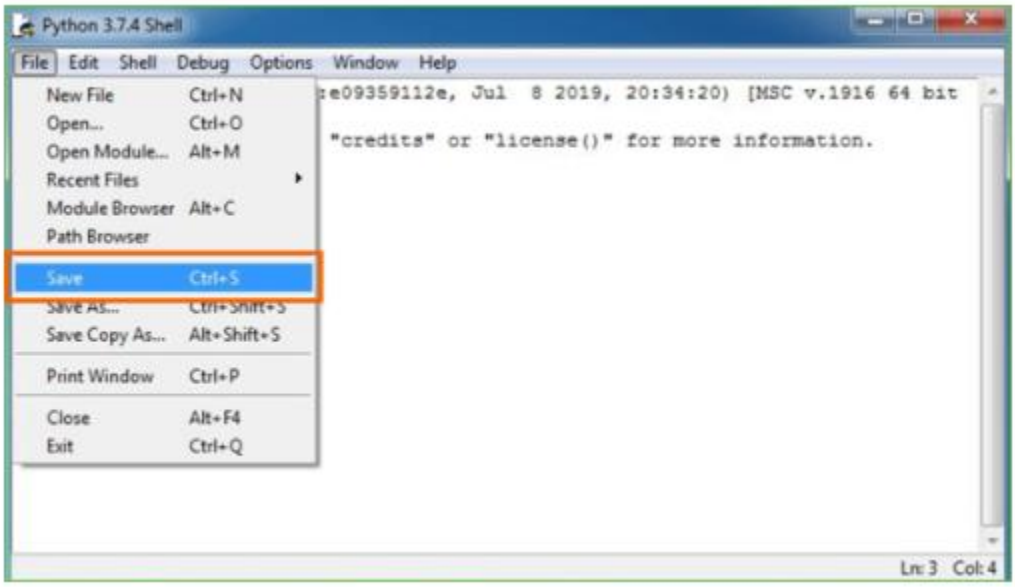
Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.



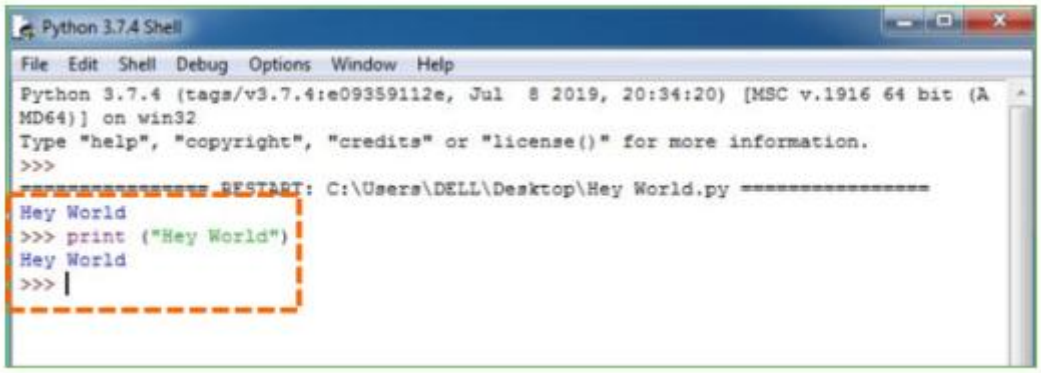
Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save



Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. enter print ("Hey World") and Press Enter.



You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won't work.

7. SYSTEM STUDY

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

1. ECONOMICAL FEASIBILITY
2. TECHNICAL FEASIBILITY
3. SOCIAL FEASIBILITY

7.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

7.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

7.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

8. SYSTEM TEST

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

8.1 TYPES OF TESTS

8.1.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

8.1.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

8.1.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

8.1.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

8.1.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

8.1.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

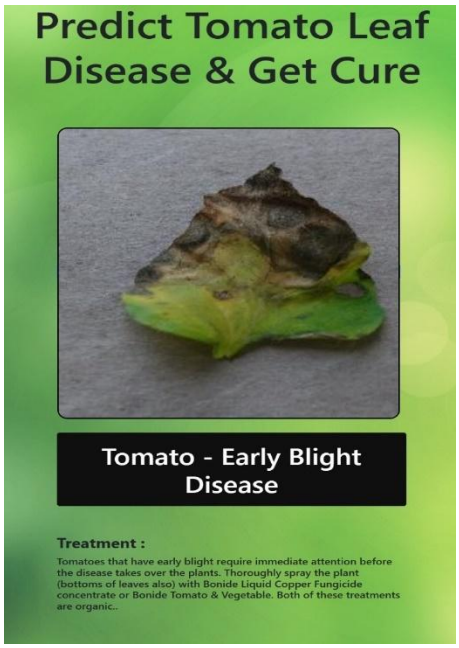
- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

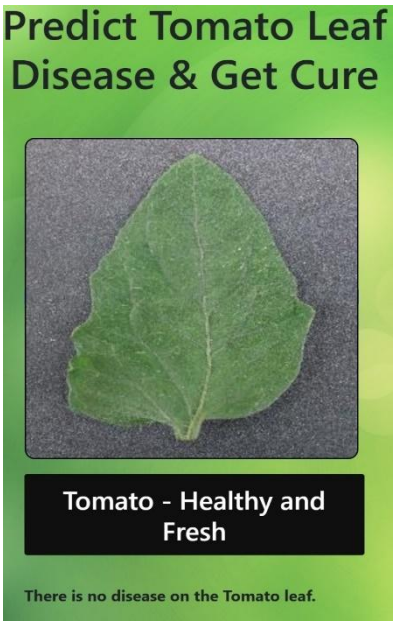
.8.1.7 Acceptance Testing

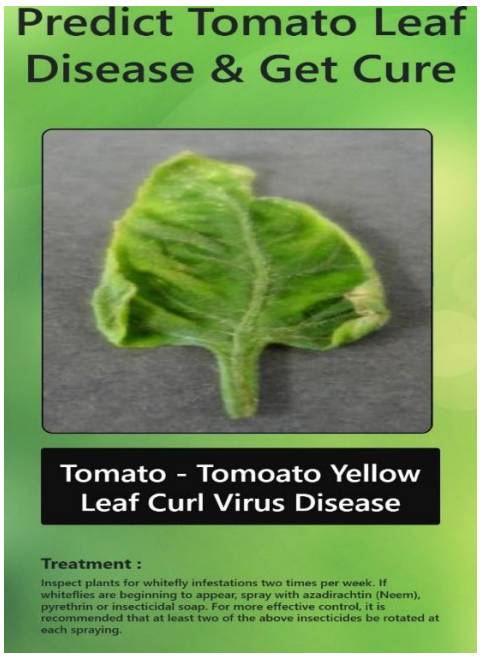
User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

8.2 Test Results: All the test cases mentioned above passed successfully. No defects encountered.

9. RESULT

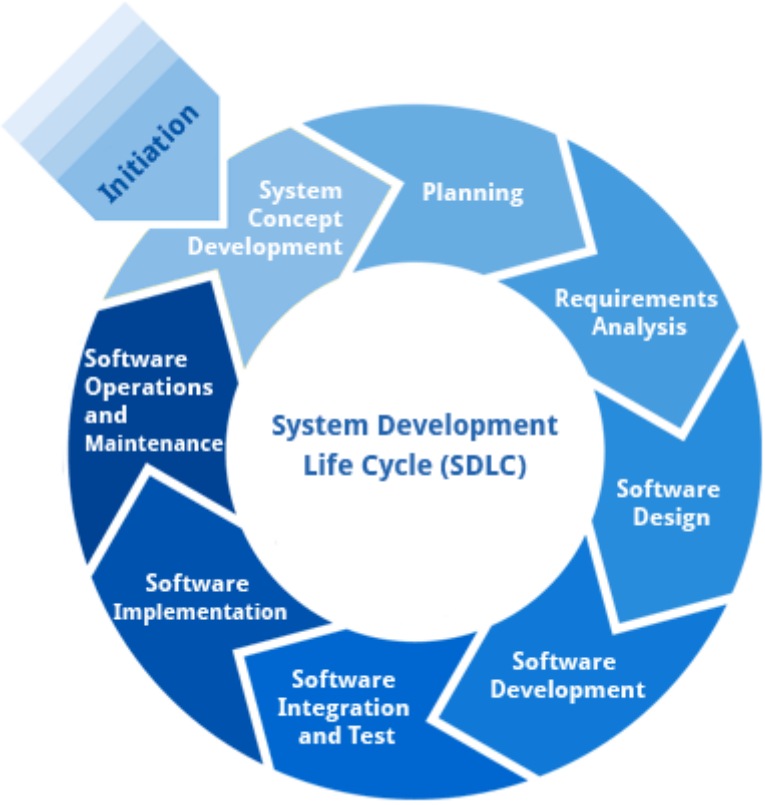

Predict Tomato Leaf Disease & Get Cure
Tomato - Early Blight Disease
Treatment :
Tomatoes that have early blight require immediate attention before the disease takes over the plants. Thoroughly spray the plant (bottoms of leaves also) with Bonide Liquid Copper Fungicide concentrate or Bonide Tomato & Vegetable. Both of these treatments are organic..


Predict Tomato Leaf Disease & Get Cure
Tomato - Healthy and Fresh
There is no disease on the Tomato leaf.


Predict Tomato Leaf Disease & Get Cure
Tomato - Tomoato Yellow Leaf Curl Virus Disease
Treatment :
Inspect plants for whitefly infestations two times per week. If whiteflies are beginning to appear, spray with azadirachtin (Neem), pyrethrin or insecticidal soap. For more effective control, it is recommended that at least two of the above insecticides be rotated at each spraying.

10. SOFTWARE DEVELOPMENT LIFE CYCLE

The **Systems Development Life Cycle (SDLC)**, or Software Development Life Cycle in systems engineering, information systems and software engineering, is the process of creating or altering systems, and the models and methodologies use to develop these systems.



Requirement Analysis and Design

Analysis gathers the requirements for the system. This stage includes a detailed study of the business needs of the organization. Options for changing the business process may be considered. Design focuses on high level design like, what programs are needed and how are they going to interact, low-level design (how the individual programs are going to work), interface design (what are the interfaces going to look like) and data design (what data will be required). During these phases, the software's overall structure is defined. Analysis and Design are very crucial in the whole development cycle. Any glitch in the design phase could be very expensive to solve in the later stage of the software development. Much care is taken during this phase. The logical system of the product is developed in this phase.

Implementation

In this phase the designs are translated into code. Computer programs are written using a conventional programming language or an application generator. Programming tools like Compilers, Interpreters, and Debuggers are used to generate the code. Different high level programming languages like C, C++, Pascal, Java, .Net are used for coding. With respect to the type of application, the right programming language is chosen.

Testing

In this phase the system is tested. Normally programs are written as a series of individual modules, this subject to separate and detailed test. The system is then tested as a whole. The separate modules are brought together and tested as a complete system. The system is tested to ensure that interfaces between modules work (integration testing), the system works on the intended platform and with the expected volume of data (volume testing) and that the system does what the user requires (acceptance/beta testing).

Maintenance

Inevitably the system will need maintenance. Software will definitely undergo change once it is delivered to the customer. There are many reasons for the change. Change could happen because of some unexpected input values into the system. In addition, the changes in the system could directly affect the software operations. The software should be developed to accommodate changes that could happen during the post implementation period.

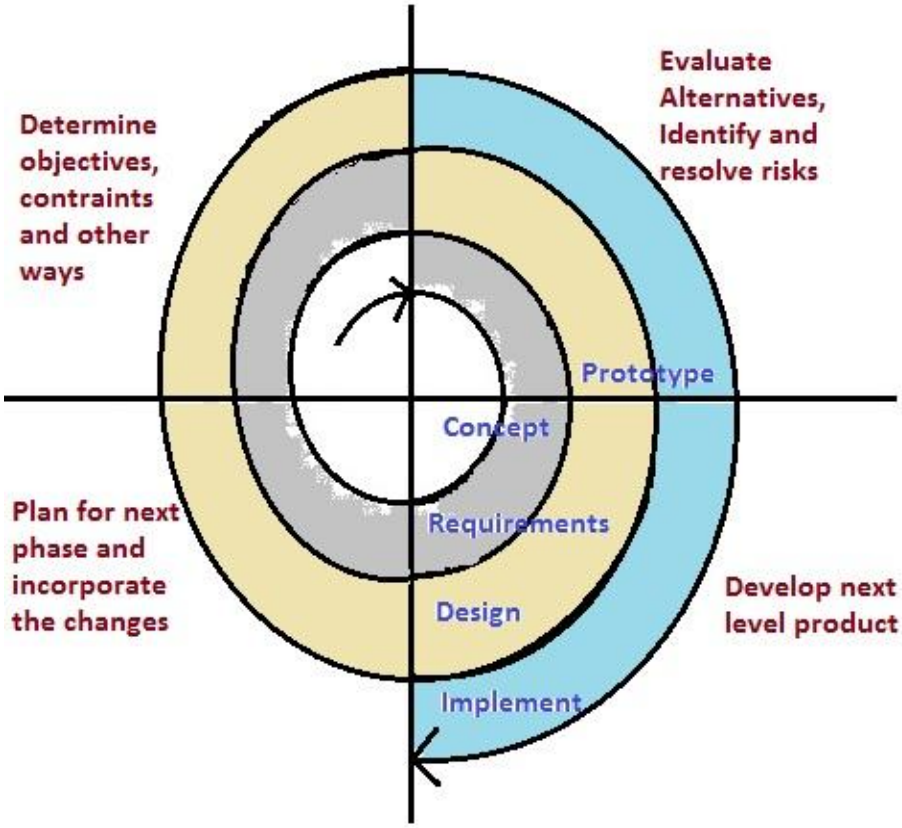
SDLC METHDOLOGIES

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article, “A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far.

The following diagram shows how a spiral model acts like:



The steps for Spiral Model can be generalized as follows:

- The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
- A preliminary design is created for the new system.
- A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
- A second prototype is evolved by a fourfold procedure:

1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
2. Defining the requirements of the second prototype.
3. Planning an designing the second prototype.

- At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involved development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product.
- The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
- The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
- The final system is constructed, based on the refined prototype.
- The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

11. CONCLUSION

This project presents a practical and efficient solution for the automatic detection of tomato leaf diseases using a lightweight convolutional neural network. By focusing on minimizing computational overhead and maximizing accuracy, the proposed system enables real-time deployment on mobile and edge devices. Leveraging data augmentation, the model is trained on a diverse dataset representing 10 disease categories. Experimental results demonstrate high accuracy and efficiency, confirming its suitability for agricultural applications. Ultimately, this work empowers farmers with a fast, reliable, and accessible tool for early disease diagnosis, thereby enhancing crop health and agricultural productivity.

12. BIBLOGRAPHY

1. **Mohanty, S. P., Hughes, D. P., & Salathe , M.** (2016). *Using deep learning for image-based plant disease detection. Frontiers in Plant Science*, 7, 1419. https://doi.org/10.3389/fpls.2016.01419
2. **Ferentinos, K. P.** (2018). *Deep learning models for plant disease detection and diagnosis. Computers and Electronics in Agriculture*, 145, 311-318. https://doi.org/10.1016/j.compag.2018.01.009
3. **Barbedo , J. G. A.** (2013). *Digital image processing techniques for detecting, quantifying and classifying plant diseases. Springer Plus*, 2, 660. https://doi.org/10.1186/2193-1801-2-660
4. **TensorFlow** – Open-source machine learning framework by Google https://www.tensorflow.org/
5. **Keras** – Deep learning API for building and training models https://keras.io/
6. **OpenCV** – Open-source computer vision and machine learning software library https://opencv.org/
7. **Py Torch** – Deep learning framework by Facebook https://pytorch.org/
8. **Scikit-learn** – Machine learning in Python https://scikit-learn.org/
9. **Geo Pandas & Folium** – Python libraries for geospatial data visualization https://geopandas.org/
10. **LSTM and Prophet Models** – For time series forecasting in agriculture

- Hochreiter, S., & Schmidhuber, J. (1997). *Long Short-Term Memory*.
- Taylor, S. J., & Letham, B. (2018). *Forecasting at Scale*. The American Statistician

13. YUKTHI CERTIFICATE