A Real Time Project Report on

Plant Disease Detection Using CNN

A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the academic requirements for the award of the degree.

Bachelor of Technology

In

Electronics & Communication Engineering

Submitted by

K SHASHANK 22H51A0430
K SIDDARATHA 22H51A0431
K LASYA PRIYA 22H51A0432

Under the esteemed guidance of

Guide Name – Ms. K. DEEPA RAO
(Assistant Professor)

Department of Electronics & Communication Engineering

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC Autonomous)

***Approved by AICTE *Affiliated to JNTUH * NAAC Accredited with A+ GRADE**

Kandlakoya (V), Medchal Road, Hyderabad, Telangana State -501 401.

2023- 2024

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC Autonomous)

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

Department of Electronics & Communication Engineering

CERTIFICATE

This is to certify that the Real Time Project report entitled **"Plant Disease Detection Using CNN"** being submitted by **K SHASHANK-22H51A0430,KSIDDARTHA-22H51A0431, K LASYA PRIYA-22H51A0432** in partial fulfillment for the award of Bachelor of Technology in Electronics & Communication Engineering**,** submitted to the Department of Electronics & Communication Engineering, CMR College of Engineering &Technology, Hyderabad during the Academic Year 2023-24

Ms.K.Deepa Rao
Assistant Professor

Head of the Department
Dr. P. Raveendra Babu
Professor & Head of Dept of ECE, CMRCET.

ACKNOWLEDGEMENT

With great pleasure we want to take this opportunity to express our heartfelt gratitude to all the people who helped in making this project a grand success.

We are grateful to **Ms.K.Deepa Rao (Assistant Professor)** , Department of Electronics & Communication Engineering for his/her valuable technical suggestions and guidance during the execution of this Real Time Project work.

We would like to thank**, Dr. P. Raveendra Babu,** Head of the Department of Electronics & Communication Engineering, CMR College of Engineering & Technology, who is the major driving forces to complete our Real Time Project work successfully.

We are very grateful to **Dr. Ghanta Devadasu**, Dean-Academics, CMR College of Engineering & Technology, for his constant support and motivation in carrying out the Real Time Project work successfully.

We are highly indebted to **Major Dr. V A Narayana,** Principal, CMR College of Engineering & Technology, for giving permission to carry out this Real Time Project in a successful and fruitful way.

We would like to thank the **Teaching & Non - Teaching** staff of Department of Electronics & Communication Engineering for their co-operation.

We express our sincere thanks to **Shri. Ch. Gopal Reddy**, Secretary& Correspondent, CMR Group of Institutions, and Shri Ch Abhinav Reddy, CEO, CMR Group of Institutions for their continuous care and support.

Finally, we extend thanks to our parents who stood behind us at different stages of this Project. We sincerely acknowledge and thank all those who gave support directly or indirectly in completion of this project work.

K SHASHANK 22H51A0430
K SIDDARTHA 22H51A0431
K LASYA PRIYA 22H51A0432

TABLE OF CONTENTS

List of Figures

List of Tables

ABSTRACT

The project presents plant disease detection using image processing techniques for automated vision system used at agriculture field. In agriculture research of automatic plant disease detection is essential one in monitoring large fields of crops, and thus automatically detects symptoms of disease as soon as they appear on plant leaves. For this approach, colour transformations, masking green pixels, segmentation are used for classification based on learning with some training samples of that category. Finally, the simulated result shows that used network classifier provides minimum error during training and better accuracy in classification.

CHAPTER 1
INTRODUCTION

CHAPTER 1
INTRODUCTION

1.1.Problem Statement

The agricultural industry faces significant challenges due to crop diseases and pests, which lead to substantial economic losses. Early detection and accurate diagnosis of these issues are crucial for effective management and mitigation. Traditional methods of disease identification rely heavily on expert knowledge and manual inspection, which can be time-consuming and prone to human error.

1.2.Research Objective

The project presents plant disease detection using image processing techniques for automated vision system used at agriculture field. In agriculture research of automatic plant disease detection is essential one in monitoring large fields of crops, and thus automatically detects symptoms of disease as soon as they appear on plant leaves. For this approach, colour transformations, masking green pixels, segmentation are used for classification based on learning with some training samples of that category. Finally, the simulated result shows that used network classifier provides minimum error during training and better accuracy in classification.

1.3 Project Scope and Limitations

The scope of this project encompasses the development and implementation of a deep learning model, specifically RESNET, for the automated identification of crop diseases and pests using image data. The project includes data preprocessing, model training, evaluation, and testing on a diverse dataset sourced from the AI Challenger Competition. It aims to provide a reliable and efficient tool for early detection of plant diseases to mitigate economic losses in agriculture. However, limitations exist, such as the potential need for a large, annotated dataset

for effective model training, the computational resources required for processing and training deep neural networks, and the model's dependency on the quality and variety of input images. Additionally, the model may face challenges in accurately identifying diseases in real-world conditions where images could vary significantly from those in the dataset due to differences in lighting, angles, and environmental factors.

CHAPTER 2
LITERATURE REVIEW

CHAPTER 2

BACKGROUND WORK

‘‘Deep learning in neural networks: An overview,’

In recent years, deep artificial neural networks (including recurrent ones) have won numerous contests in pattern recognition and machine learning. This historical survey compactly summarises relevant work, much of it from the previous millennium. Shallow and deep learners are distinguished by the depth of their credit assignment paths, which are chains of possibly learnable, causal links between actions and effects. I review deep supervised learning (also recapitulating the history of backpropagation), unsupervised learning, reinforcement learning & evolutionary computation, and indirect search for short programs encoding deep and large networks.

‘‘Acoustic modeling using deep belief networks,’’

Gaussian mixture models are currently the dominant technique for modeling the emission distribution of hidden Markov models for speech recognition. We show that better phone recognition on the TIMIT dataset can be achieved by replacing Gaussian mixture models by deep neural networks that contain many layers of features and a very large number of parameters. These networks are first pre-trained as a multi-layer generative model of a window of spectral feature vectors without making use of any discriminative information. Once the generative pre-training has designed the features, we perform discriminative fine-tuning using backpropagation to adjust the features slightly to make them better at predicting a probability distribution over the states of monophone hidden Markov models.

‘‘On the expressive power of deep architectures,’’

Deep architectures are families of functions corresponding to deep circuits. Deep Learning algorithms are based on parametrizing such circuits and tuning their parameters so as to approximately optimize some training objective. Whereas it was thought too difficult to train

deep architectures, several successful algorithms have been proposed in recent years. We review some of the theoretical motivations for deep architectures, as well as some of their practical successes, and propose directions of investigations to address some of the remaining challenges.

''A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion,''

We develop and present a novel deep convolutional neural network architecture, where heterogeneous pooling is used to provide constrained frequency-shift invariance in the speech spectrogram while minimizing speech-class confusion induced by such invariance. The design of the pooling layer is guided by domain knowledge about how speech classes would change when formant frequencies are modified. The convolution and heterogeneous-pooling layers are followed by a fully connected multi-layer neural network to form a deep architecture interfaced to an HMM for continuous speech recognition. During training, all layers of this entire deep net are regularized using a variant of the "dropout" technique. Experimental evaluation demonstrates the effectiveness of both heterogeneous pooling and dropout regularization. On the TIMIT phonetic recognition task, we have achieved an 18.7% phone error rate, lowest on this standard task reported in the literature with a single system and with no use of information about speaker identity. Preliminary experiments on large vocabulary speech recognition in a voice search task also show error rate reduction using heterogeneous pooling in the deep convolutional neural network.

''Monitoring and detection of agricultural disease using wireless sensor network,'

Wireless sensor network technology is widely used in the western world for improving agriculture output. However, in the developing countries, the adaptation of technology is very slow due to various factors such as cost and unawareness of farmers with the technology. There are reports in the literature related to the precision agriculture and hopefully, this paper will add to the knowledge of the use of Wireless sensor network (WSN) for monitoring agriculture fields

Using CNN

for pest detection. The literature related to pest monitoring and detection using wireless sensor networking technologies are reviewed. Then, the advanced sensing technologies are currently in use for the detection of a pest has been described. The existing techniques about pest detection and disease monitoring are evaluated on the basis of some key parameters such as the type of sensors used, their cost, processing tools, etc. Finally, the sensing technologies and the possibility of using third generation sensing technology for monitoring and detection of cotton crops are analyzed.

‘‘Monitoring pest insect traps by means of low-power image sensor technologies,’’

Monitoring pest insect populations is currently a key issue in agriculture and forestry protection. At the farm level, human operators typically must perform periodical surveys of the traps disseminated through the field. This is a labor-, time- and cost-consuming activity, in particular for large plantations or large forestry areas, so it would be of great advantage to have an affordable system capable of doing this task automatically in an accurate and a more efficient way. This paper proposes an autonomous monitoring system based on a low-cost image sensor that it is able to capture and send images of the trap contents to a remote control station with the periodicity demanded by the trapping application. Our autonomous monitoring system will be able to cover large areas with very low energy consumption. This issue would be the main key point in our study; since the operational live of the overall monitoring system should be extended to months of continuous operation without any kind of maintenance (i.e., battery replacement). The images delivered by image sensors would be time-stamped and processed in the control station to get the number of individuals found at each trap. All the information would be conveniently stored at the control station, and accessible via Internet by means of available network services at control station (WiFi, WiMax, 3G/4G, etc.).

‘‘Pest Monitor and control system using WSN with special referance to acoustic device,

-Agriculture has increasingly become dependent on chemical pesticides to control the pests that damage the crops. Heightened concern over the environmental effects of pesticides, coupled

with increased pest resistance and secondary pest outbreaks, severely limits the effective

pesticides available to farmers. As weather patterns change, crops mature, and cattle graze pastures for food, farmers must decide when to irrigate pastures, apply fertilizer, or move cattle to another pasture. A farmer relies on a combination of experience, visual observation, intuition and he has to perform periodical surveys over a widespread plantation which is a time consuming activity. Acoustic detection technology is a far better substitute to these laborious tasks. Sugarcane takes 10-18 months to grow and is therefore liable to be attacked by a number of insects, pests and diseases. Its production declines by 19 - 20 % by their attacks. To increase the crop productivity, management of insect-pest and diseases is of great significance. Sugarcane is infested by about 288 insects of which nearly two dozen causes heavy losses to the quality as well as quantity of the crop. In this project we will be focusing on a pest control and monitoring system for efficient sugarcane crop production, which is a staple crop grown in Pune. The main pests that affect sugarcane are top shoot borer, stalk borer, rood borer and sugarcane wooly aphid. Apart from this, the main diseases that affect sugarcane crop are Red rot, Smut, Grassy Shoot and Wilt. The system uses an acoustic device sensor which monitors the noise level of the pests and gives an indication to the farmer through an alarm when the noise crosses a threshold. The dissemination of is done via a network of wireless sensors connected to a control room computer. Transmission and reception of field data is through ZigBee 802.15.4 digital communication device standard. The system covers large areas with very low energy consumption.

2.1 Deep Artificial Neural Networks:

2.1.1 Introduction

Deep artificial neural networks, including recurrent ones, have significantly advanced pattern recognition and machine learning, distinguished by the depth of their credit assignment paths.

2.1.2 Merits, Demerits, and Challenges

Merits include superior performance in complex tasks; demerits involve high computational costs and data requirements; challenges consist of training stability and interpretability issues.

2.1.3 Implementation of Deep Artificial Neural Networks

Implementation involves training deep architectures using supervised, unsupervised, reinforcement learning, or evolutionary computation techniques, optimizing them through backpropagation and other advanced algorithms.

2.2 Monitoring and Detection of Agricultural Disease Using Wireless Sensor Network:

2.2.1 Introduction

Explores the use of WSN technology for pest detection and monitoring in agriculture to improve crop output.

2.2.2 Merits, Demerits, and Challenges

Merits are automation,efficiency in pest detection; demerits involve high costs ,slow adaptation in developing countries; challenges are sensor technology advancements and cost reduction.

2.2.3 Implementation of Agricultural Disease Monitoring

Involves deploying sensors in fields to collect data, processed by advanced sensing technologies, and transmitted via wireless networks for pest detection and monitoring.

CHAPTER 3
PROPOSED SYSTEM

CHAPTER 3

PROPOSED SYSTEM

3.1. Objective of Proposed Model

In this paper, a convolution neural network (CNN) is used to automatically identify crop diseases. The dataset comes from the public dataset of the AI Challenger Competition, which includes images of crop diseases. The deep learning CNN, specifically the RESNET model, is employed to predict crop diseases and pests, aiming to reduce economic losses in the crop business. By using the RESNET CNN model, the accuracy of disease recognition is increased.

3.2. Block diagram

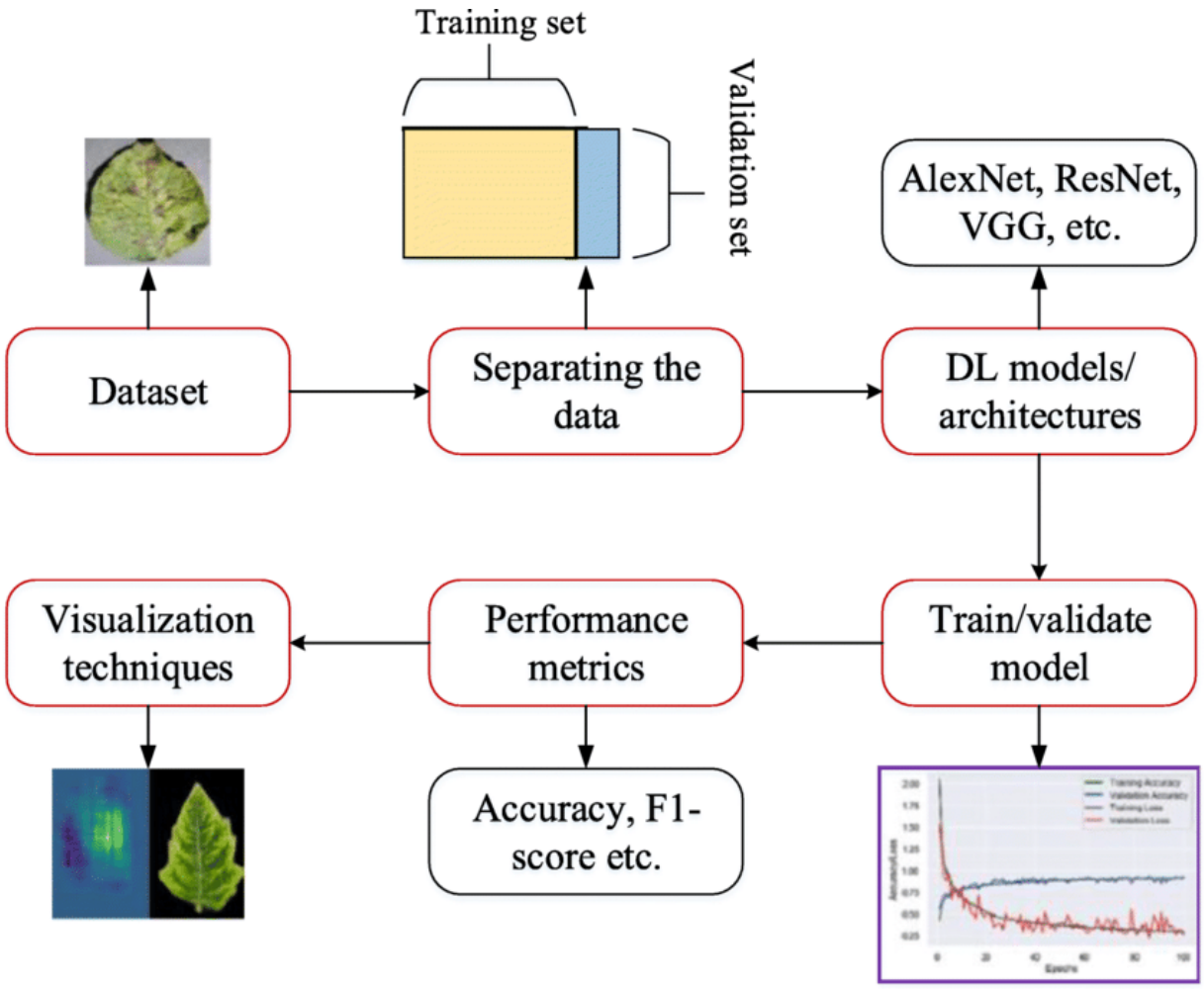


Figure 3.1 Block Diagram

3.3. Algorithms Used for Proposed Model

The RESNET convolution neural network (CNN) model is applied to build the disease recognition model. This model enhances the accuracy of predicting crop diseases and pests.

SDLC (Umbrella Model):

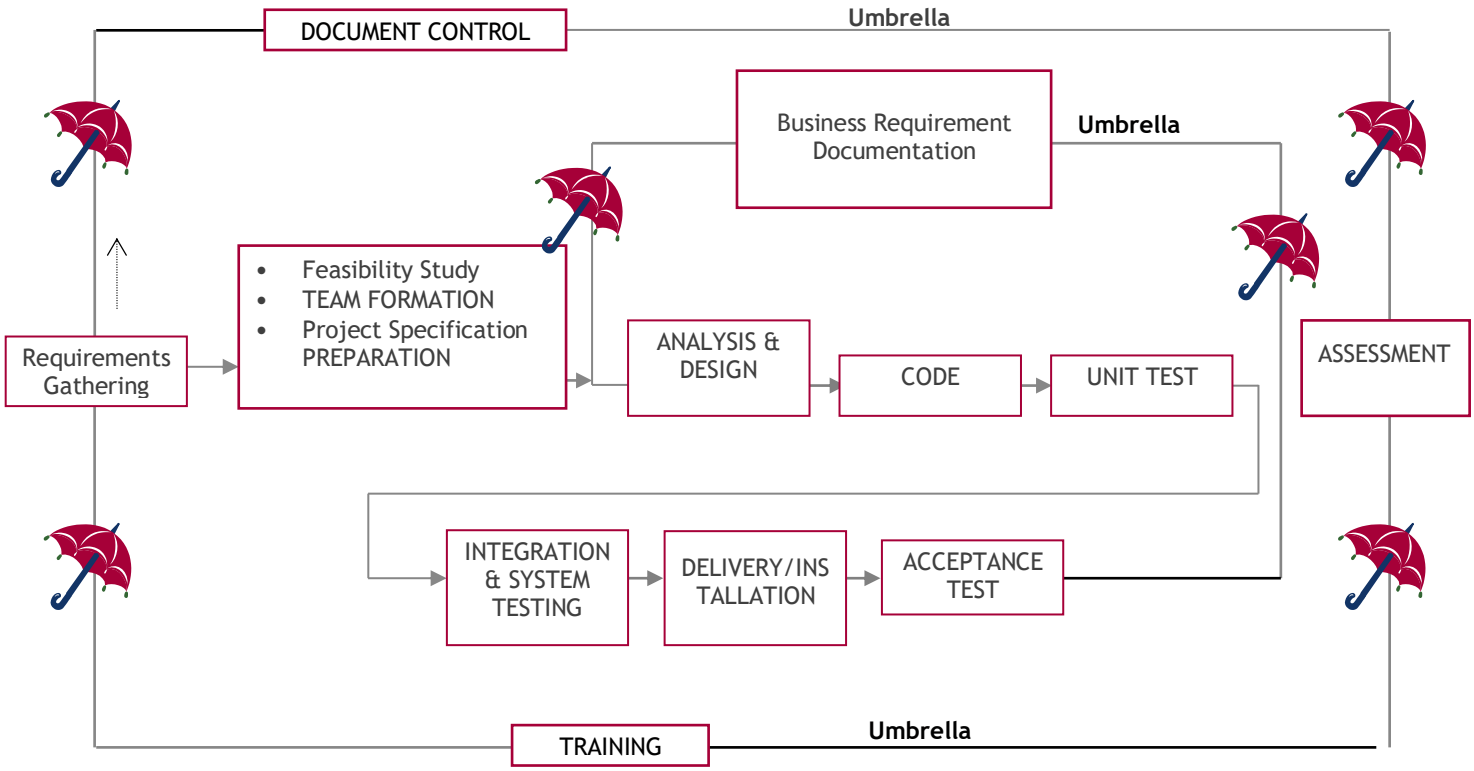


Figure 3.2 SDLC Life Cycle

SDLC is nothing but Software Development Life Cycle.

Stages in SDLC:

- Requirement Gathering
- Analysis
- Designing
- Coding
- Testing

- Maintenance

Requirements Gathering stage:

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.

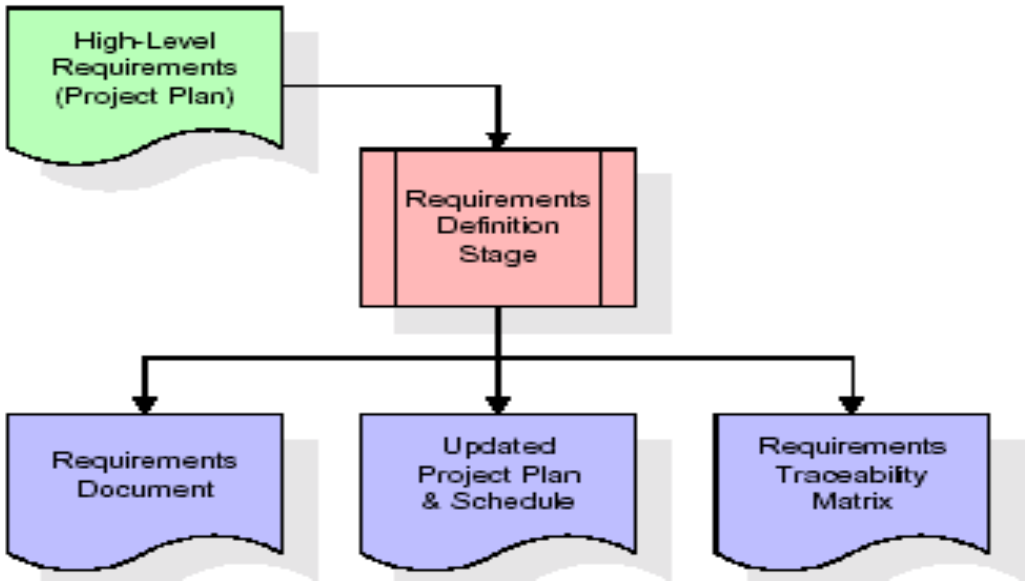


Figure 3.3 Requirment gathering stage

These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the

title components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

- Feasibility study is all about identification of problems in a project.
- No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
- Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

Analysis Stage:

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.

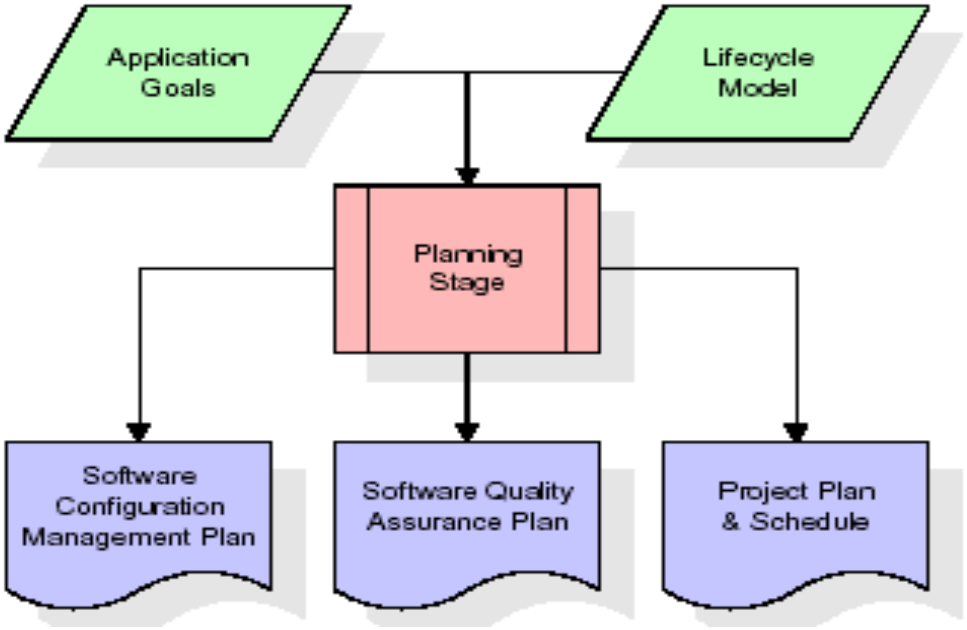


Figure 3.4 Analysis stage

The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

Designing Stage:

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elementsdiagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo

code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

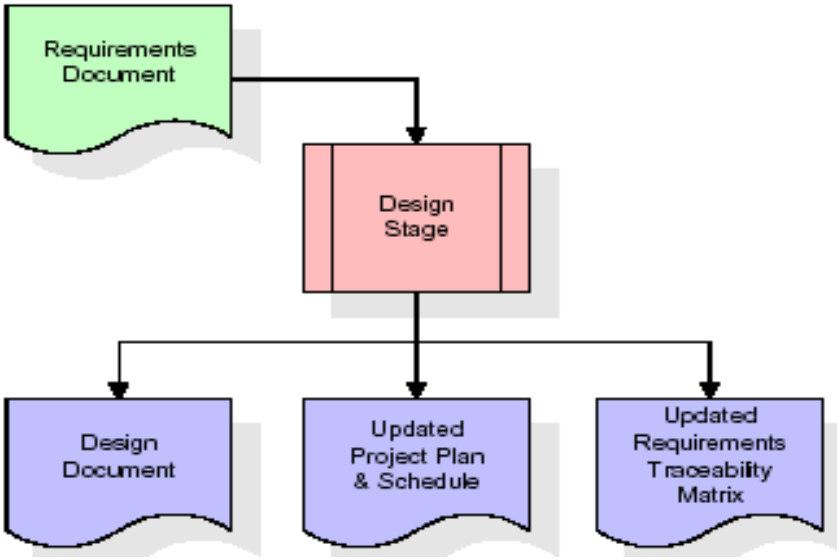


Figure 3.5 Designing stage

When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

Development (Coding) Stage:

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.

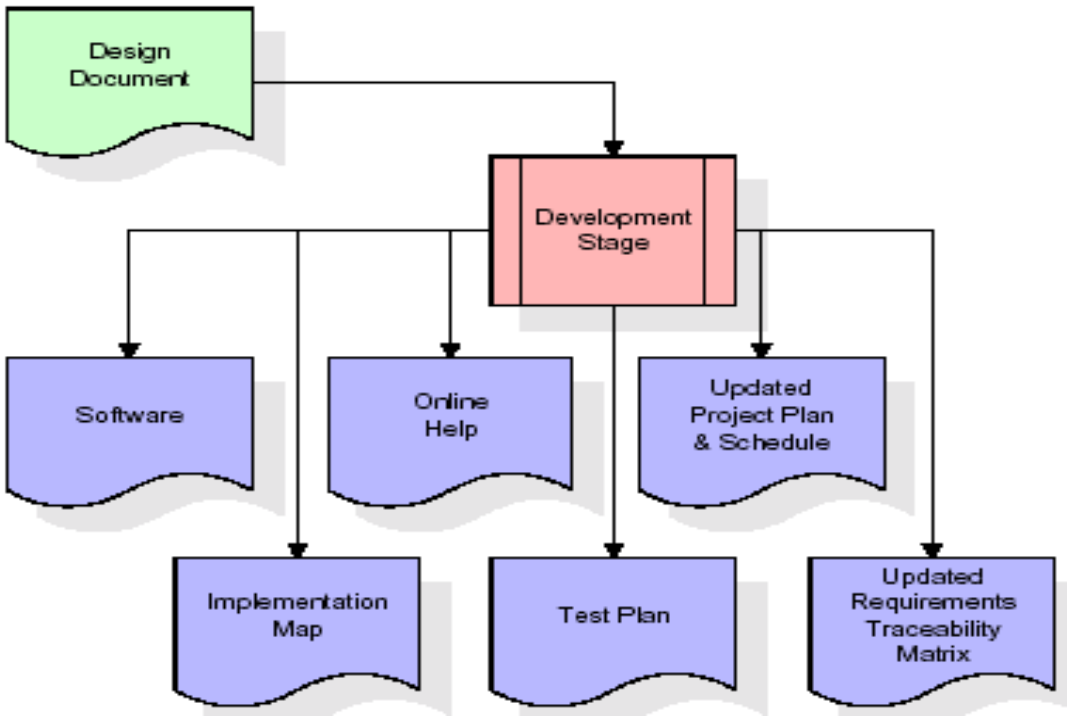


Figure 3.6 Development stage

The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

Integration & Test Stage:

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful

execution of the test suite confirms a robust and complete migration capability. During this

stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.

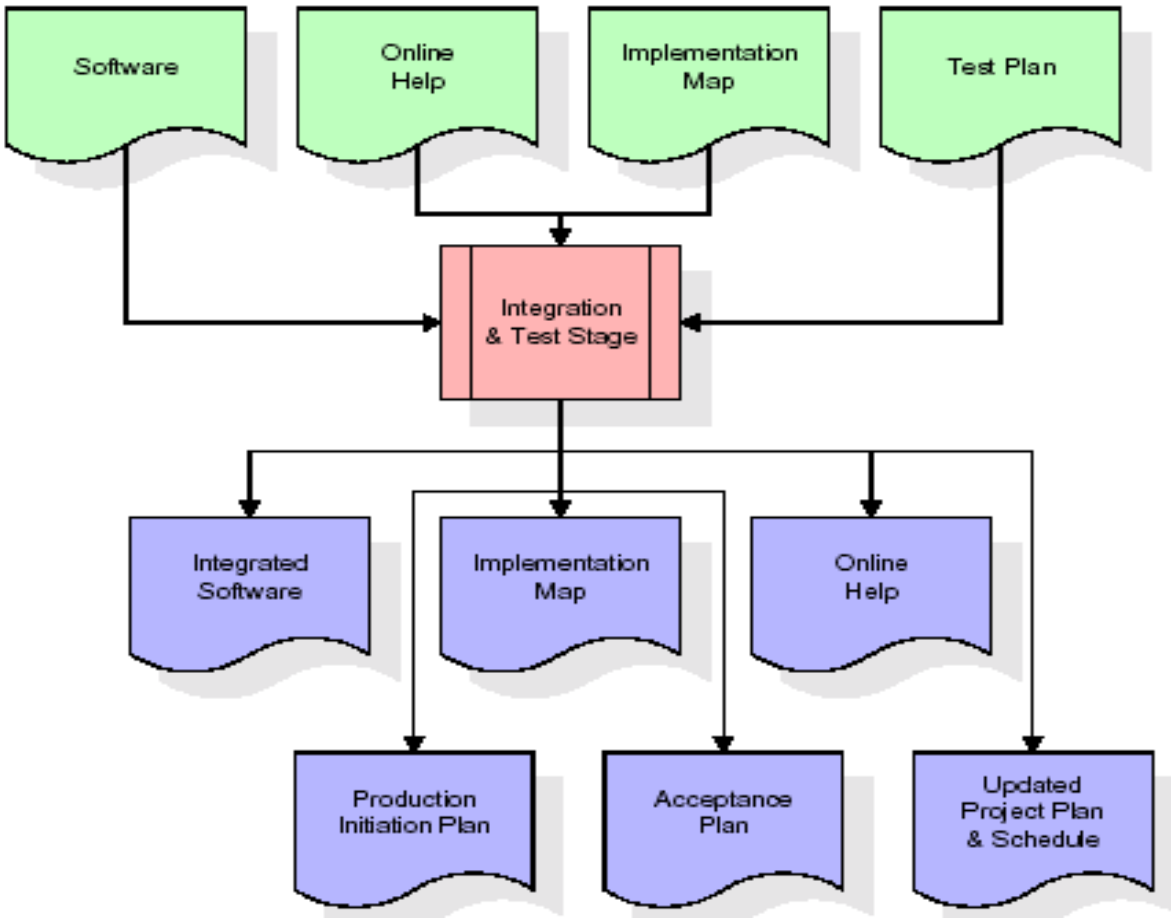


Figure 3.7 Integration& test stage

The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

- **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial

production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.

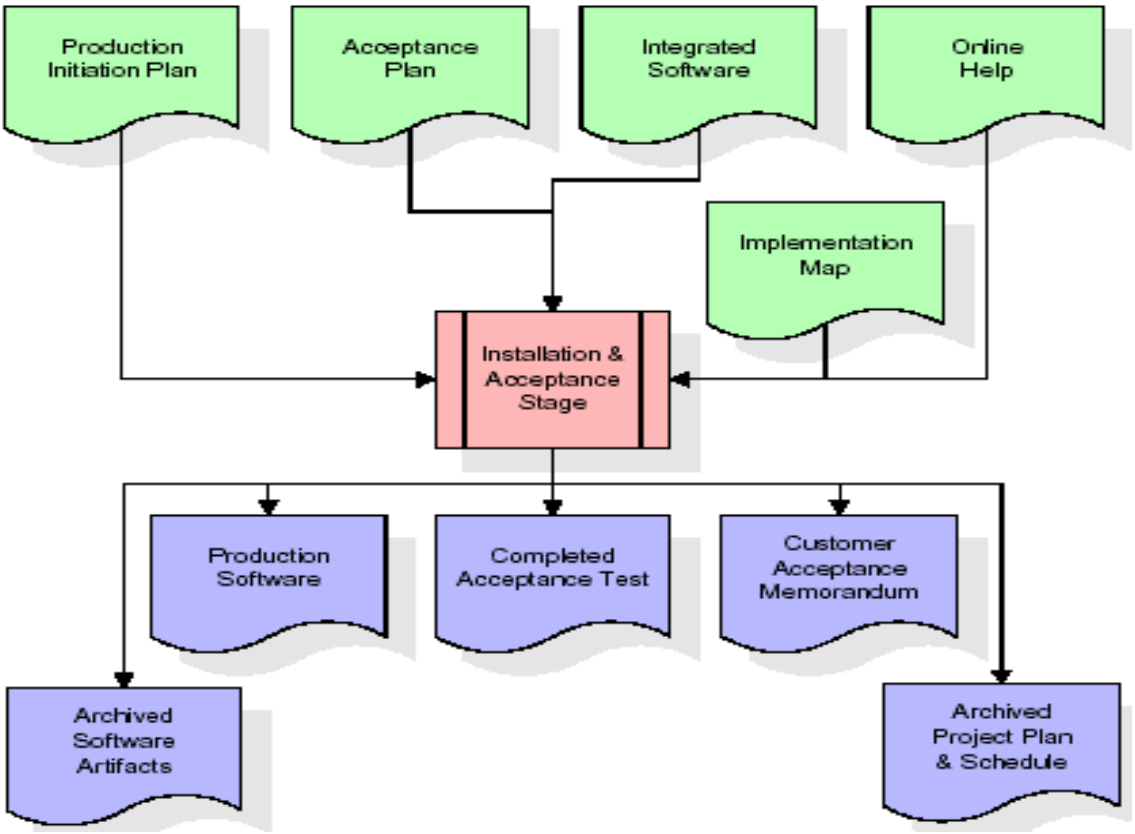


Figure 3.8 Installation test

The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

Maintenance:

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

Software Requirement Specification

Overall Description

A Software Requirements Specification (SRS) – a requirements specification for a software system is a complete description of the behavior of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Nonfunctional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the systems development lifecycle domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

- Business requirements describe in business terms *what* must be delivered or accomplished to provide value.
- Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
- Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited

resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- **ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

- **OPERATIONAL FEASIBILITY**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

- **TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to .the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

External Interface Requirements

User Interface

The user interface of this system is a user friendly python Graphical User Interface.

Hardware Interfaces

The interaction between the user and the console is achieved through python capabilities.

Software Interfaces

The required software is python.

Operating Environment

Windows XP.

HARDWARE REQUIREMENTS:

- **Processor - Pentium –IV**
- Speed - 1.1 Ghz
- RAM - 256 MB(min)
- Hard Disk - 20 G
- Key Board - Standard Windows Keyboard
- Mouse - Two or Three Button Mouse
- Monitor - SVGA

SOFTWARE REQUIREMENTS:

- Operating System - Windows7/8
- Programming Language - Python (python 3.6.3)

SYSTEM DESIGN

UML Diagram:

Class Diagram:

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

- The upper part holds the name of the class
- The middle part contains the attributes of the class
- The bottom part gives the methods or operations the class can take or undertake

Class Diagram:

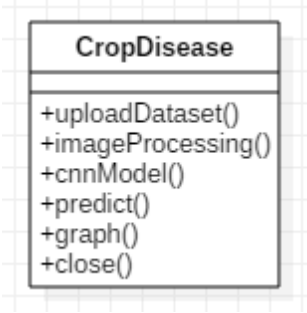


Figure 3.9 Class Diagram

Use case Diagram:

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

Use case Diagram:

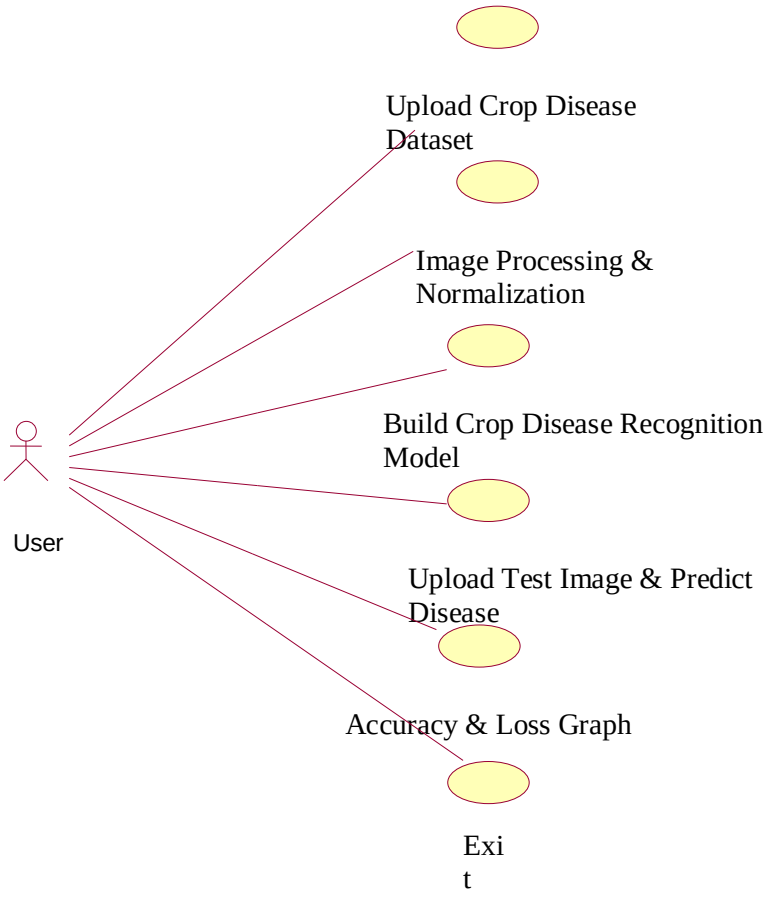


Figure 3.10 Use case Diagram

Sequence diagram:

A **sequence diagram** is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams**, **event scenarios**, and timing diagrams.

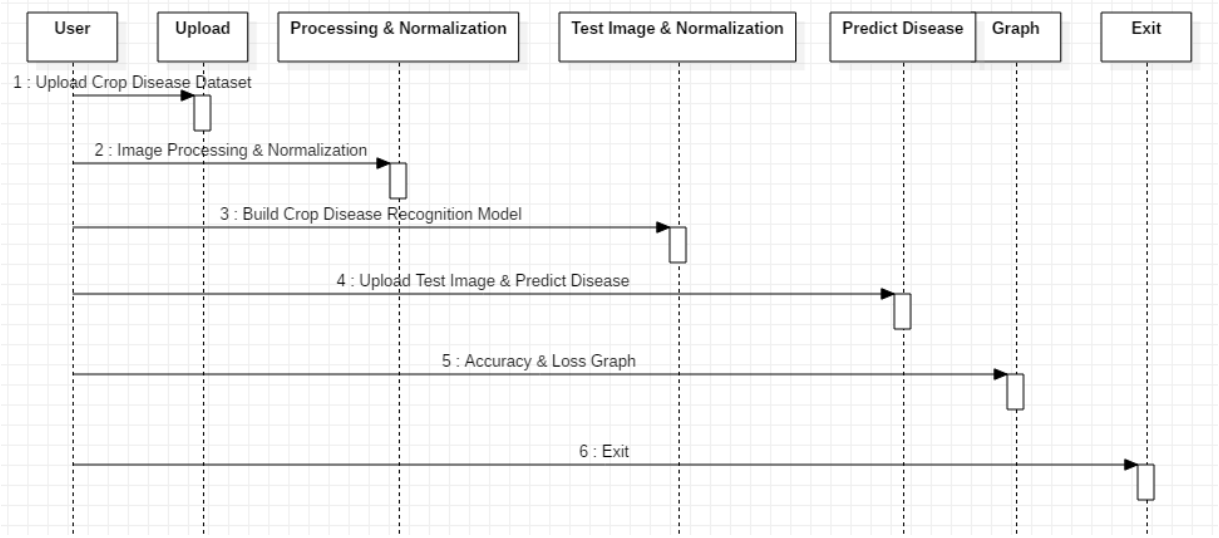


Figure 3.11 Sequence Diagram

Component Diagram:

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

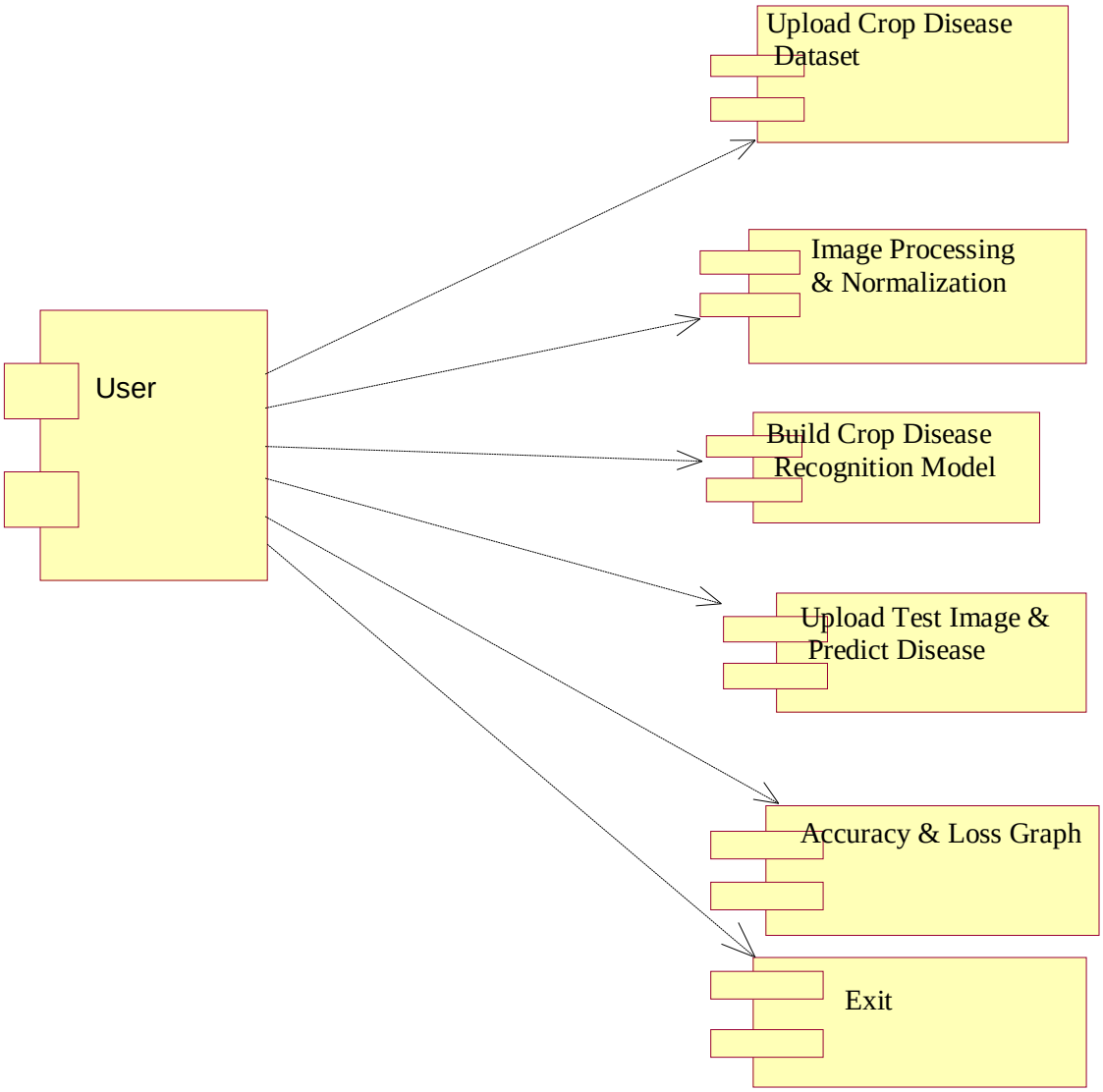


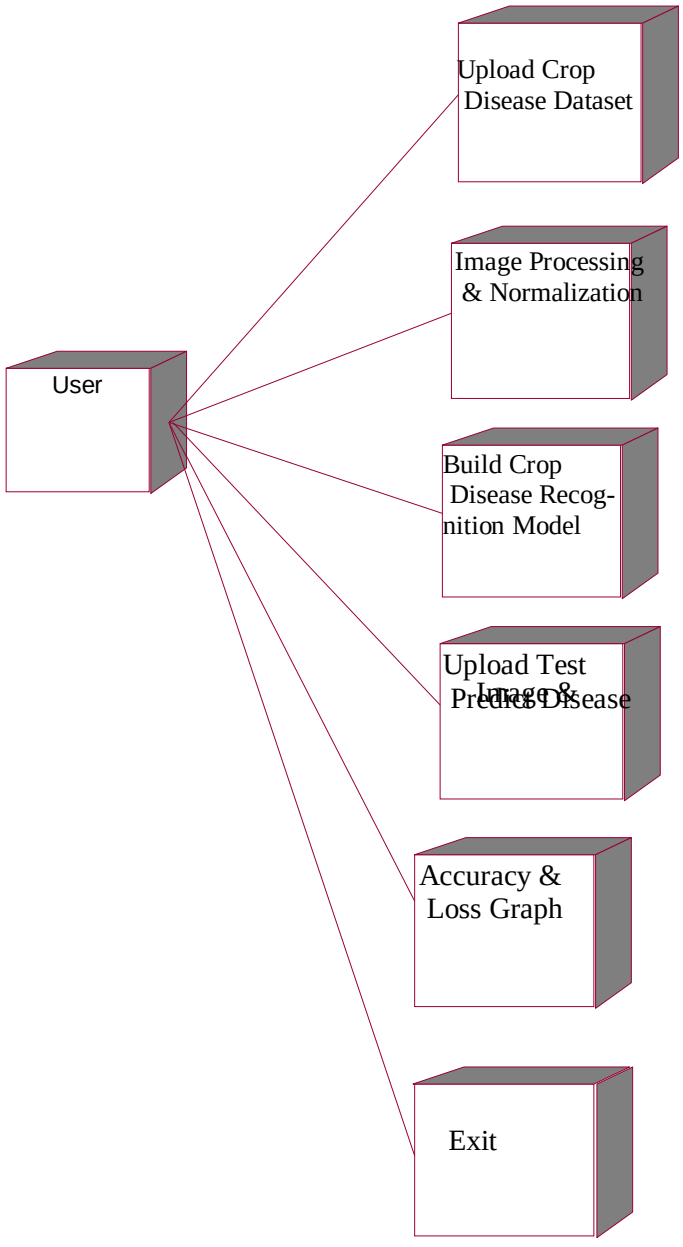
Figure 3.12 Component Diagram

Deployment Diagram:

A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application

server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of



database servers.

Figure 3.13 Deployment Diagram

3.4 Flowchart for Proposed Model

Activity Diagram:

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent

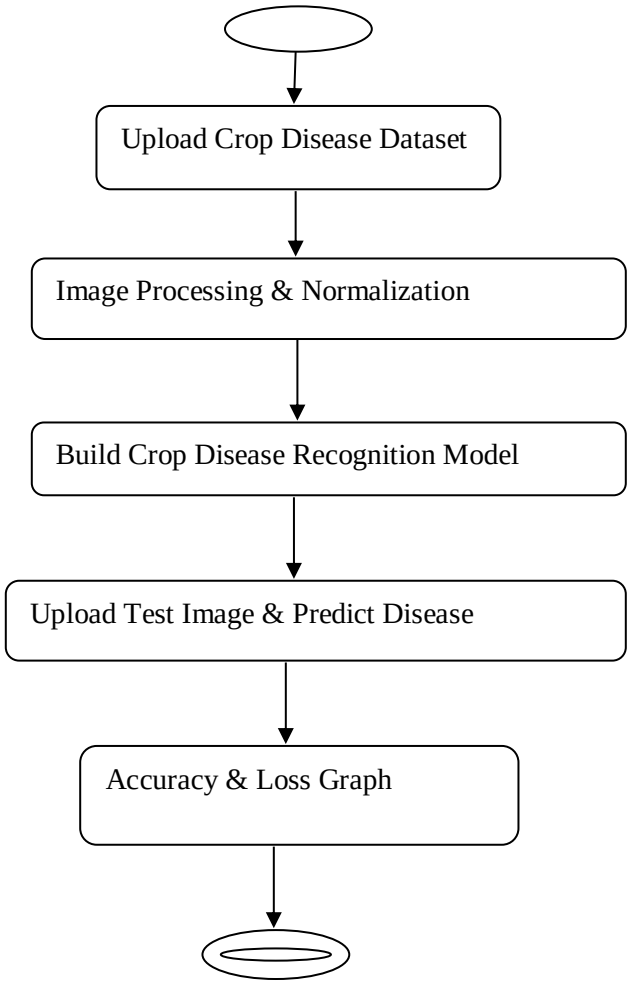


Figure 3.14 Activity Diagram

Data Flow Diagram:

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

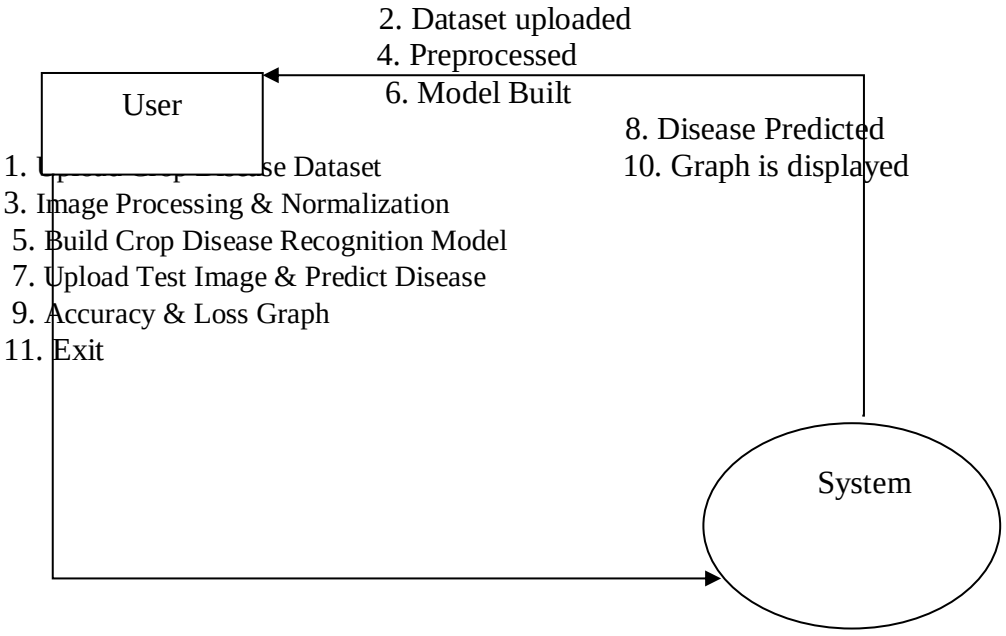


Figure 3.15 Data flow diagram

CHAPTER 4
RESULTS AND DISCUSSION

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Performance metrics

Table 4.1 Performance Metrics

Parameter	Performance Metrics
Accuracy	98.63
Precision	100
Recall	55
F-measure	68.76
Specificity	100

To run project double click on 'run.bat' file to get below screen

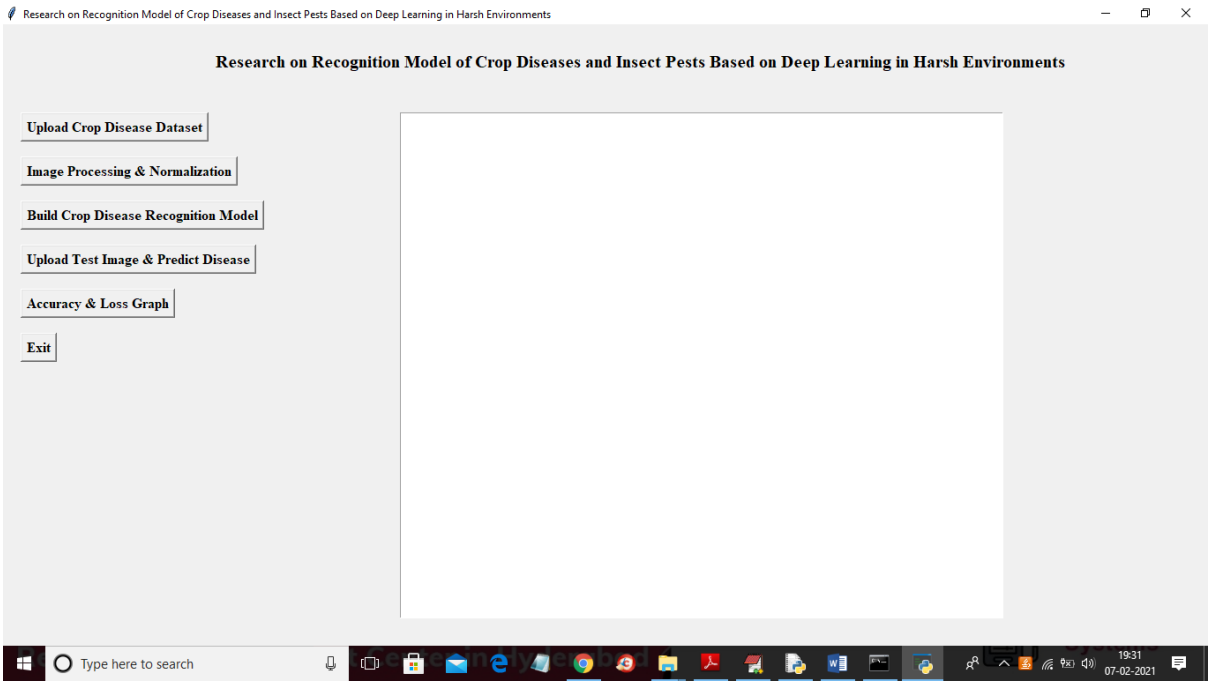


Figure 4.1 Initial frame in GUI

In above screen click on 'Upload Crop Disease Dataset' button to upload dataset images

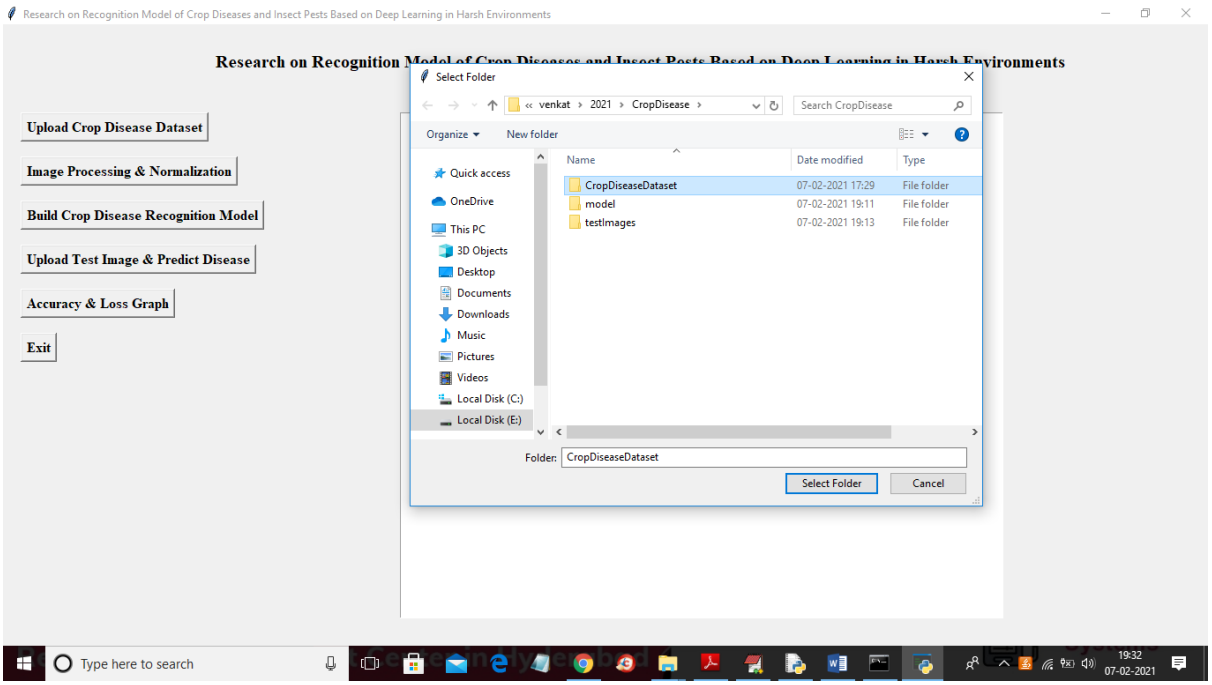


Figure 4.2 Upload crop disease dataset

In above screen selecting and uploading 'CropDiseaseDataset' folder and then click on 'SelectFolder' button to load dataset and to get below screen.

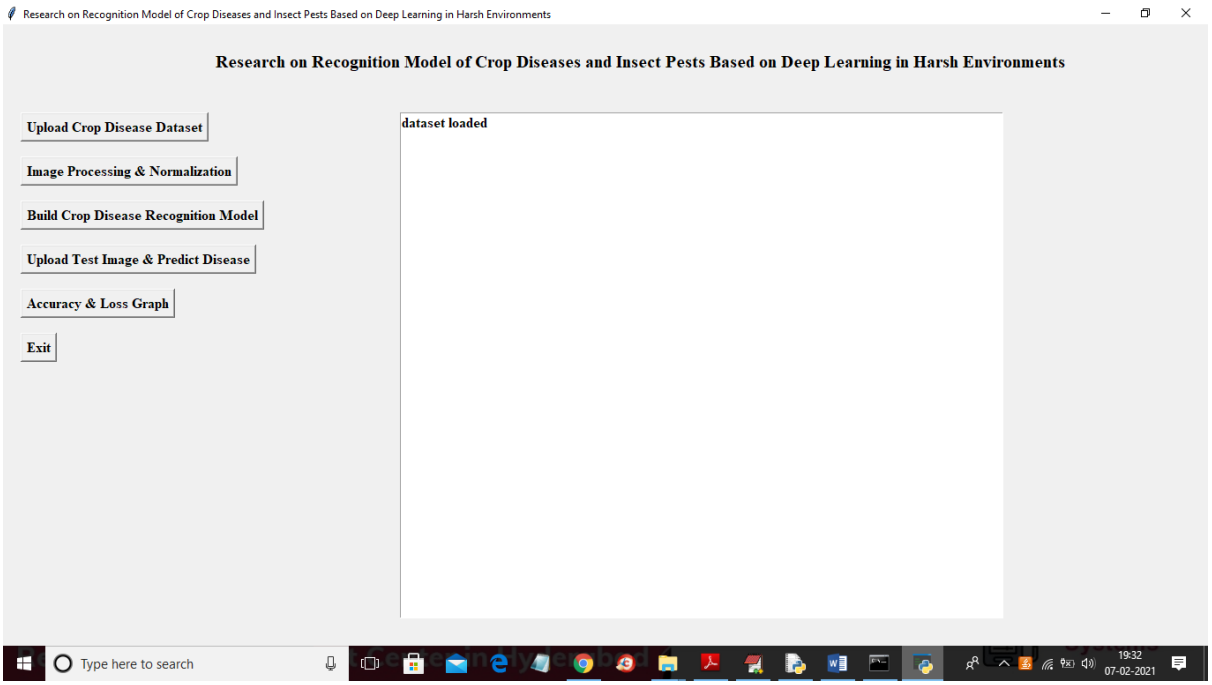


Figure 4.3 After select folder

In above screen dataset loaded and now click on 'Image Processing & Normalization' button to read all images and then process images to normalize by converting each image pixel value between 0 and 1 and for that normalization we will divide image pixels with 255 and then get

value as 0 or 1 as all images pixel value will be between 0 to 255.

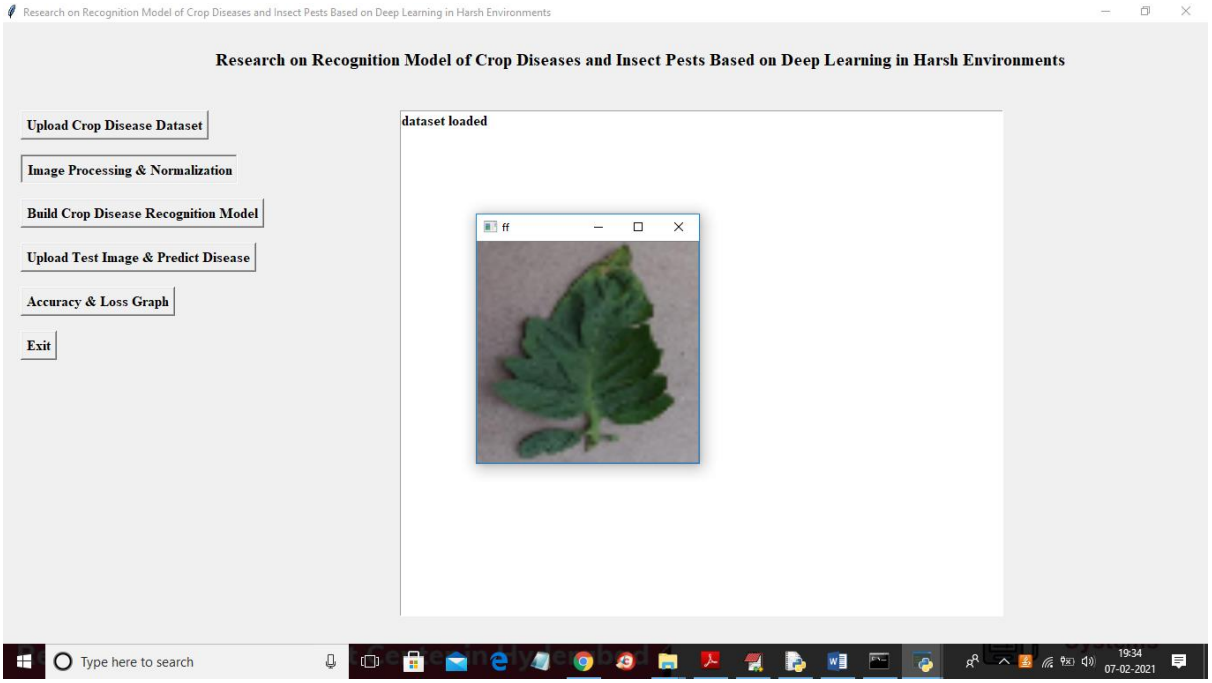


Figure 4.4 Image processing & normalisation

In above screen after applying normalization we are just displaying one random image from dataset to check whether images loaded and process properly or not and now you close above image to get below screen.

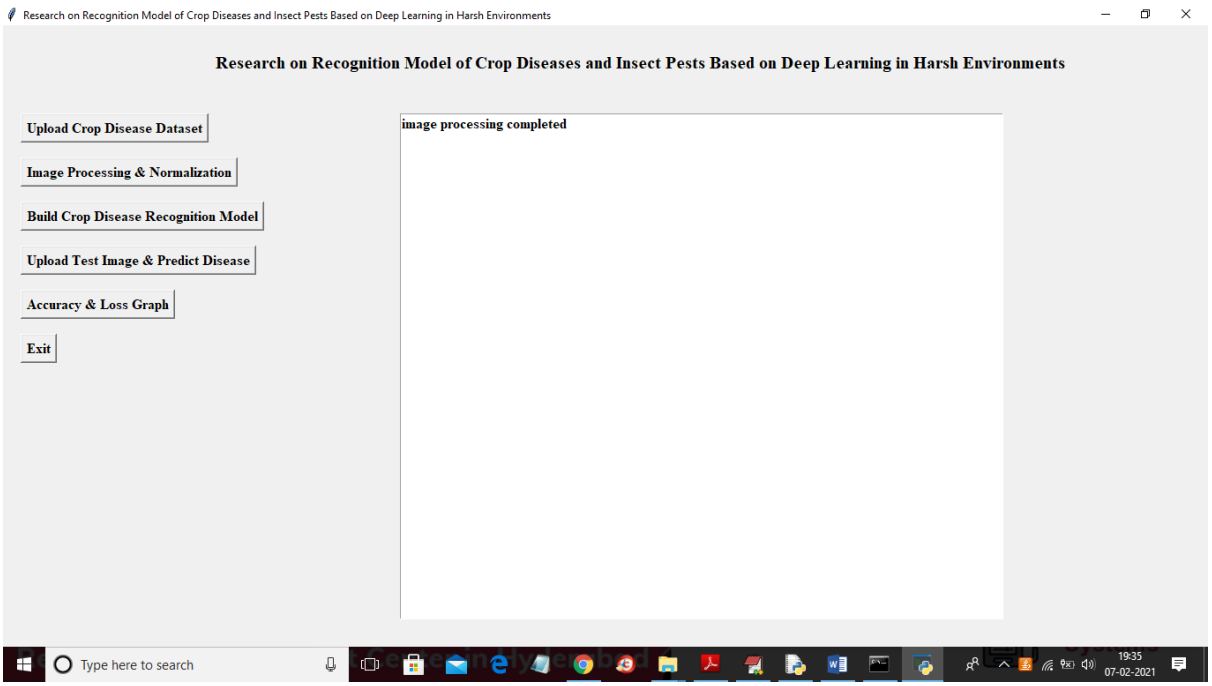


Figure 4.5 Applying normalisation

In above screen all images process successfully and now dataset images are ready and now click on 'Build Crop Disease Recognition Model' button to build CNN model.

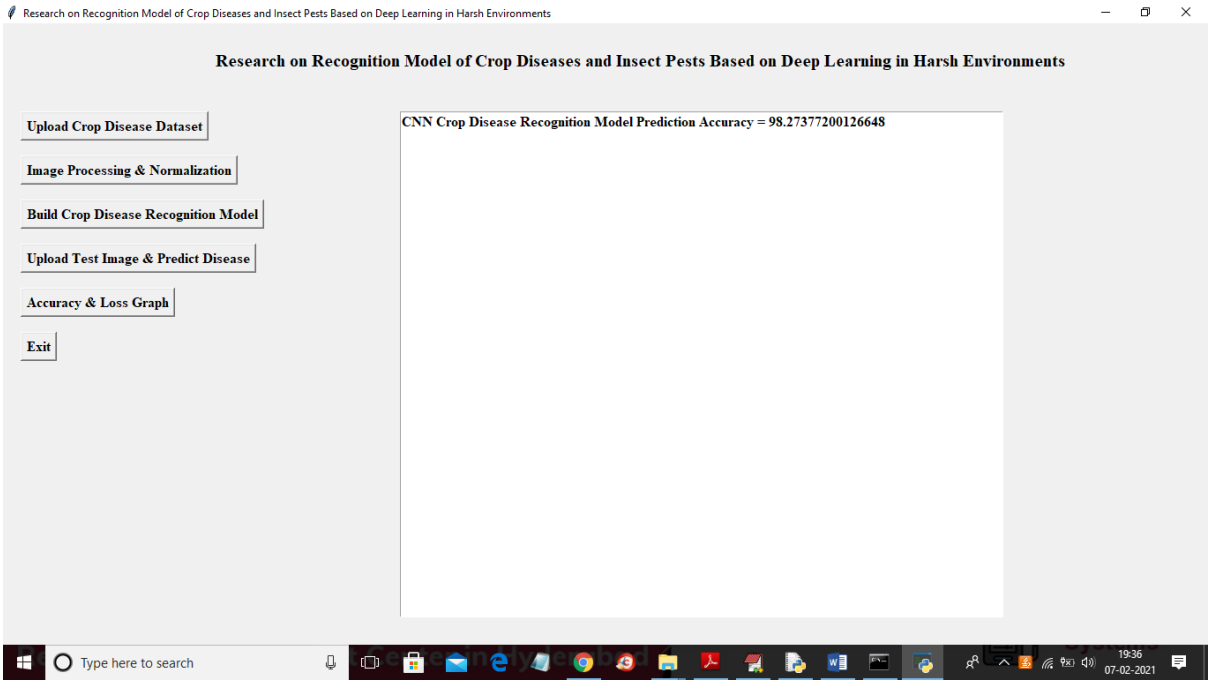


Figure 4.6 Build crop disease Recognition model

In above screen CNN model generated and its prediction accuracy is 98% and in below console screen we can see all CNN layers details.

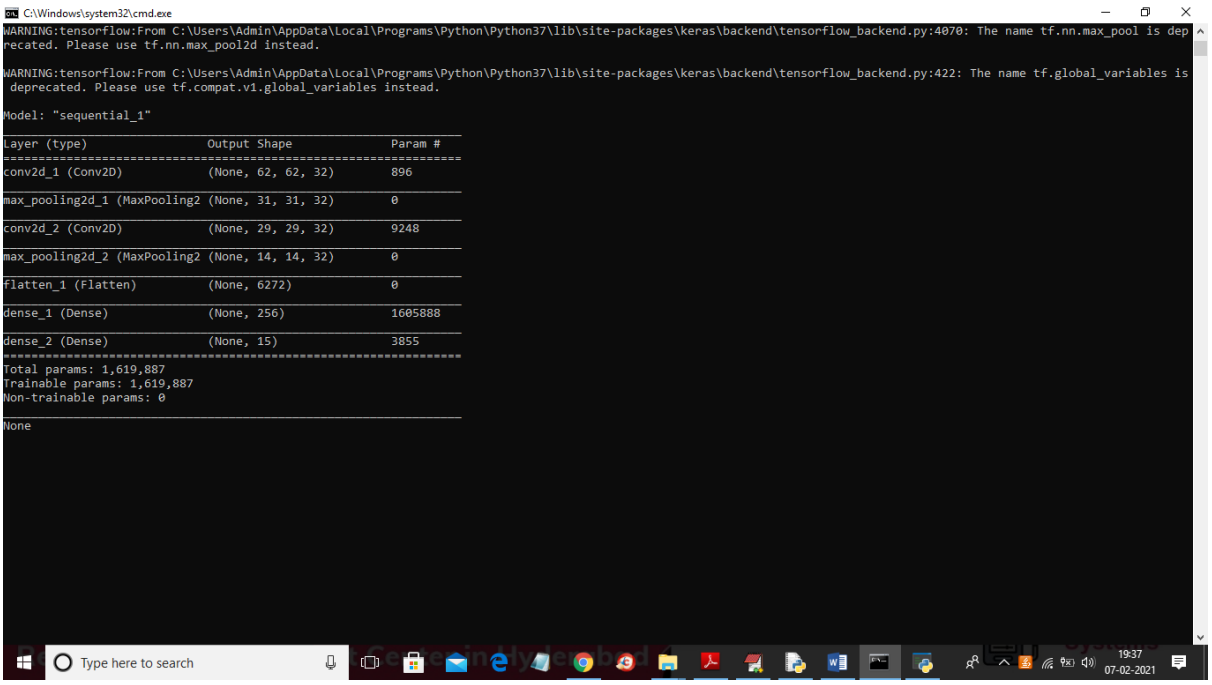


Figure 4.7 Accuray

In above screen we can see we have used CONV2D, MAXPOOLING, FLATTEN and

DENSE layer to build crop disease recognition model and RELU details you can see in code. Now model is ready and now click on 'Upload Test Image & Predict Disease' button to upload any test image and then application will predict disease or healthy from that image

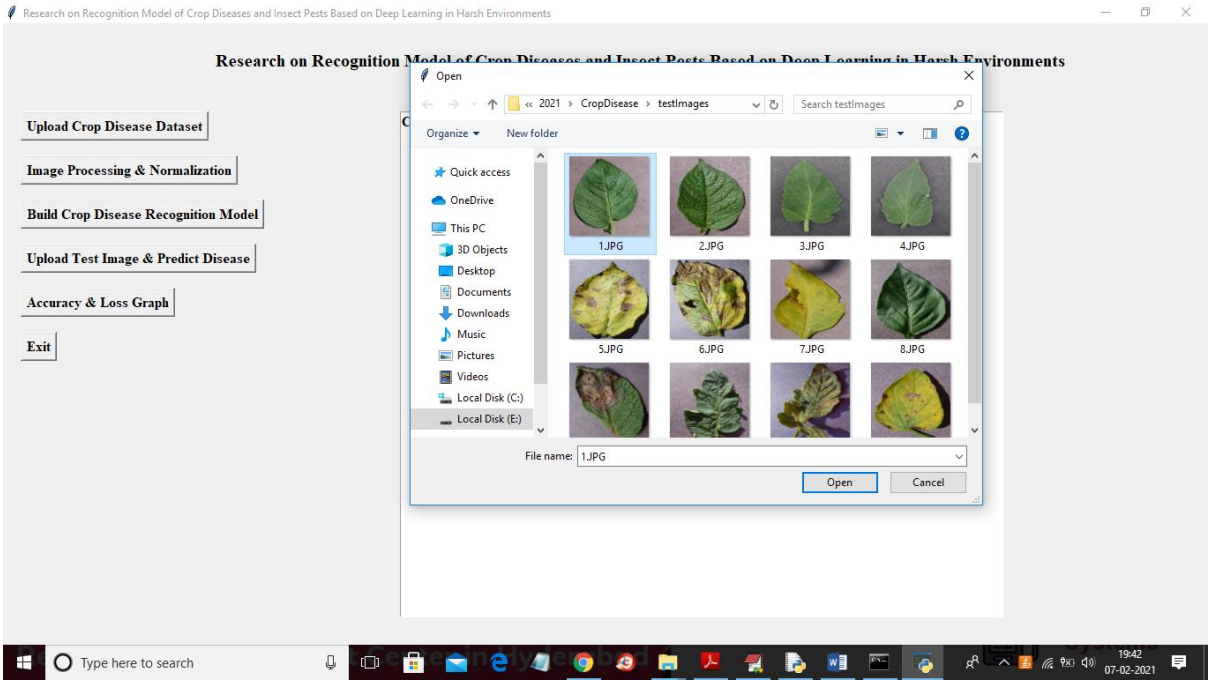


Figure 4.8 Figure 3.15 Data flow diagram

In above screen selecting and uploading '1.JPG' image file and then click on 'Open' button to get below prediction result

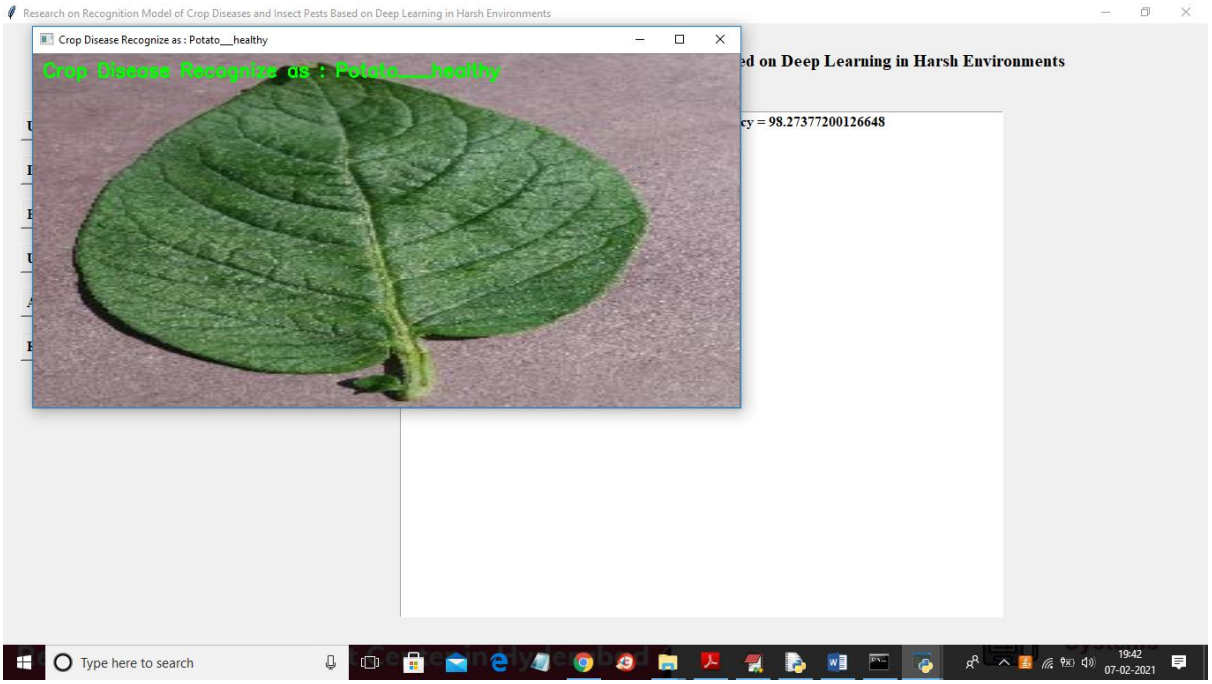


Figure 4.9 Upload & open Image

In above screen potato leaf predicted as healthy and now try with other image

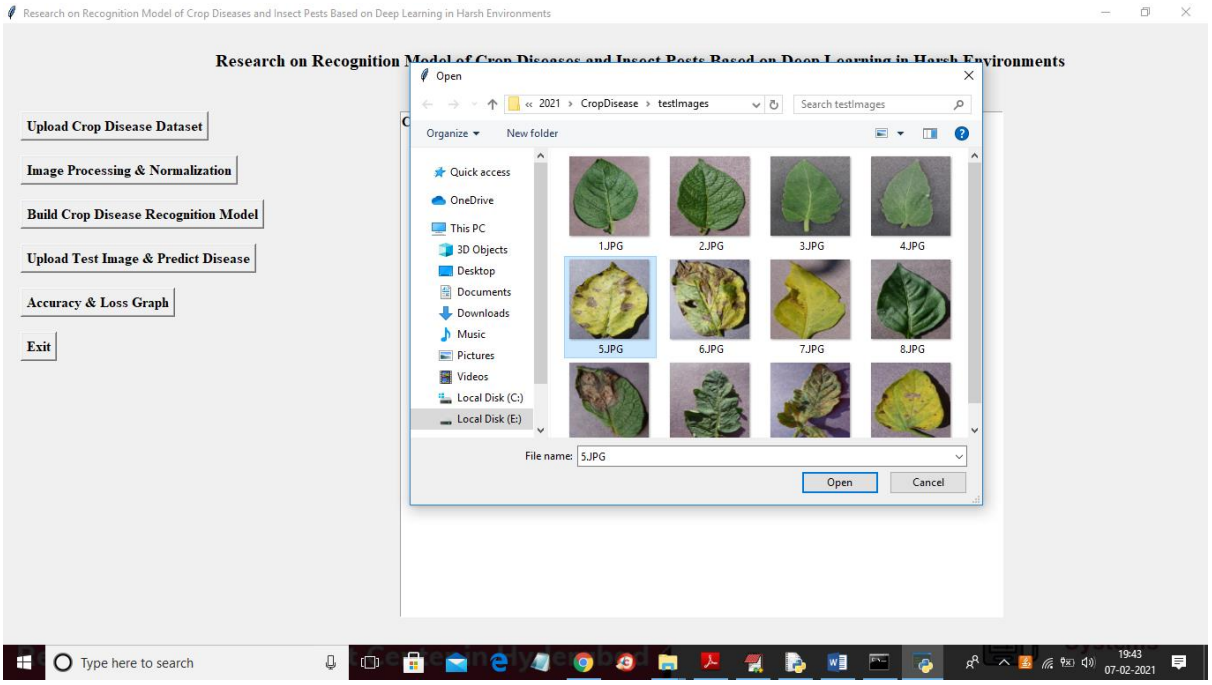


Figure 4.10 Predicted result

In above screen selecting and uploading '5.JPG' file and click 'Open' button to get below result.

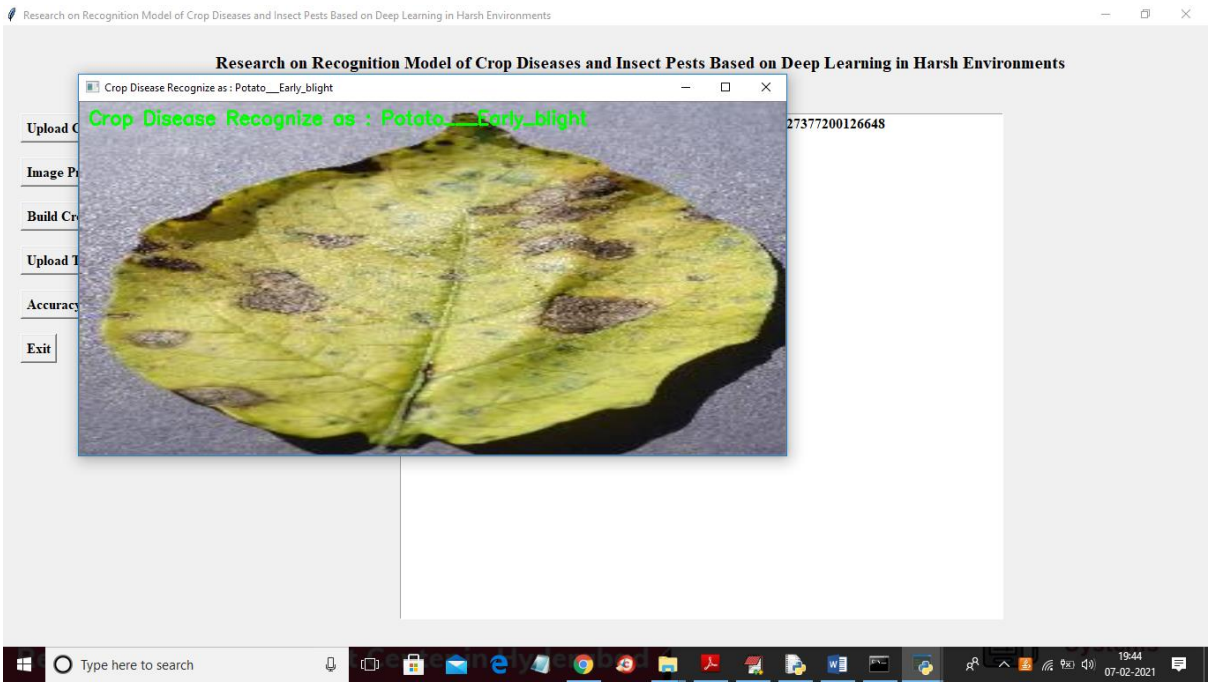


Figure 4.11 Testing another Image

In above image potato EARLY BLIGHT disease is detected or recognize and similarly you can upload any other image and get result and now click on ‘Accuracy & Loss Graph’ button to get below graph

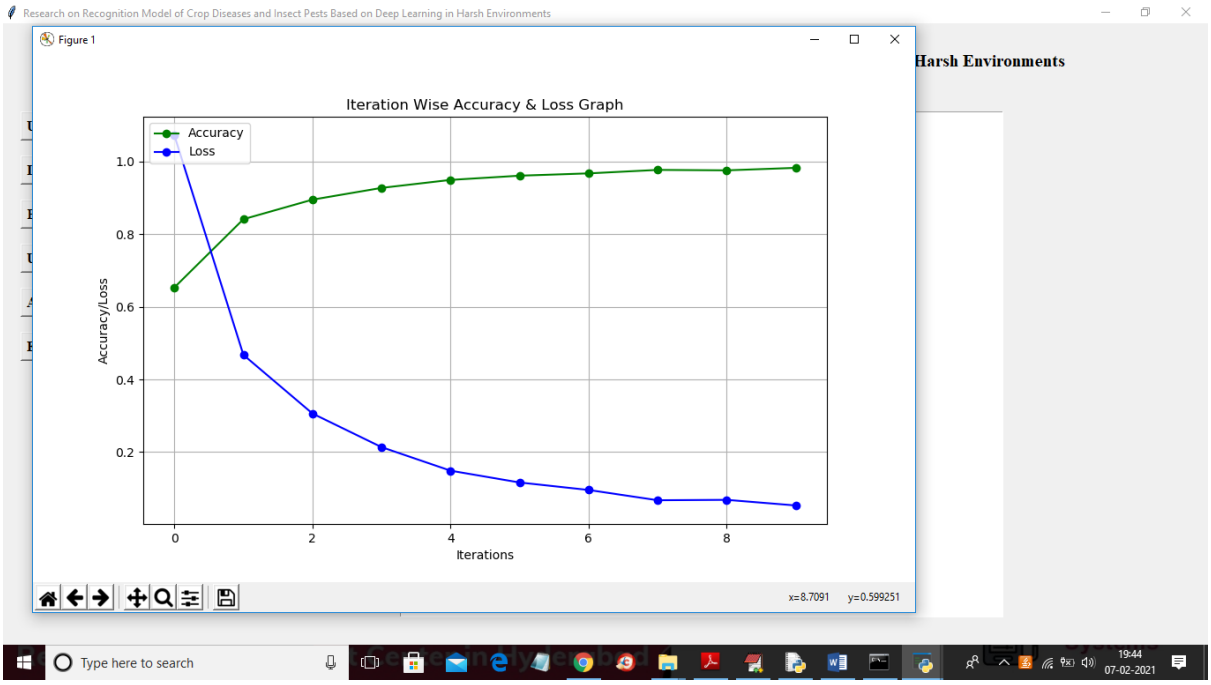


Figure 4.12 Accuracy & loss Graph

In above graph x-axis represents epoch/iterations and y-axis represents accuracy/loss and green line represents accuracy and blue line represents loss and from above graph we can see with each increasing iteration accuracy is getting better and better and loss getting decrease.

CHAPTER 5
CONCLUSION

CHAPTER 5

CONCLUSION

This paper describes a classification-based model for the detection of diseases in plants using a convolutional neural network (CNN). Leveraging a dataset from the AI Challenger Competition, which includes over 10,000 images of diseased crops, the RESNET CNN model was employed to automatically identify crop diseases and pests. This approach aims to significantly reduce economic losses in the agriculture sector by providing accurate and efficient disease recognition. In future work, the proposed system could be enhanced by integrating additional services, such as information on nearby government stores, pesticide price lists, and local open markets. Moreover, the system could be made accessible through a mobile application, offering farmers a comprehensive tool for managing crop health and optimizing their yield.

APPENDIX- A
CODE

IMPLEMETATION

Python

Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

History of Python:

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

Why Python was created?

In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python.

Why the name Python?

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late seventies. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

Features of Python:

A simple language which is easier to learn

Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax.

If you are a newbie, it's a great choice to start your journey with Python.

Free and open-source

You can freely use and distribute Python, even for commercial use. Not only can you use and distribute softwares written in it, you can even make changes to the Python's source code.

Python has a large community constantly improving it in each iteration.

Portability

You can move Python programs from one platform to another, and run it without any changes.

It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

Extensible and Embeddable

Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code.

This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

A high-level, interpreted language

Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on.

Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.

Large standard libraries to solve common tasks

Python has a number of standard libraries which makes life of a programmer much easier

since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server? You can use MySQLdb library using import MySQLdb .

Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.

Object-oriented

Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.

With OOP, you are able to divide these complex problems into smaller sets by creating objects.

Applications of Python:

1. Simple Elegant Syntax

Programming in Python is fun. It's easier to understand and write Python code. Why? The syntax feels natural. Take this source code for an example:

```
a = 2

b = 3

sum = a + b

print(sum)
```

2. Not overly strict

You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement.

Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

3. Expressiveness of the language

Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of Tic-tac-toe game with a graphical interface and a smart computer opponent in less than 500 lines of code. This is just an example. You will be amazed how much you can do with Python once you learn the basics.

4. Great Community and Support

Python has a large supporting community. There are numerous active forums online which can be handy if you are stuck.

5.2 Sample Code;

CropDisease.py:

```
from tkinter import messagebox

from tkinter import *

from tkinter import simpledialog

import tkinter

import matplotlib.pyplot as plt

import numpy as np

from tkinter import ttk

from tkinter import filedialog

from keras.utils.np_utils import to_categorical

from keras.models import Sequential

from keras.layers.core import Dense,Activation,Dropout, Flatten

from sklearn.metrics import accuracy_score

import os
```

```
import cv2

from keras.layers import Convolution2D

from keras.layers import MaxPooling2D

import pickle

from keras.models import model_from_json



main = Tk()

main.title("Research on Recognition Model of Crop Diseases and Insect Pests Based on Deep Learning in Harsh Environments")

main.geometry("1300x1200")



global filename

global X, Y

global model

global accuracy



plants = ['Pepper__bell___Bacterial_spot', 'Pepper__bell___healthy', 'Potato___Early_blight', 'Potato___healthy', 'Potato___Late_blight', 'Tomato__Target_Spot',

        'Tomato__Tomato_mosaic_virus', 'Tomato__Tomato_YellowLeaf__Curl_Virus', 'Tomato_Bacterial_spot', 'Tomato_Early_blight', 'Tomato_healthy',

        'Tomato_Late_blight', 'Tomato_Leaf_Mold', 'Tomato_Septoria_leaf_spot', 'Tomato_Spider_mites_Two_spotted_spider_mite']
```

```
def uploadDataset():
    global X, Y
    global filename
    text.delete('1.0', END)
    filename = filedialog.askdirectory(initialdir=".")
    text.insert(END,'dataset loaded\n')


def imageProcessing():
    text.delete('1.0', END)
    global X, Y
    X = np.load("model/myimg_data.txt.npy")
    Y = np.load("model/myimg_label.txt.npy")
    Y = to_categorical(Y)
    X = np.asarray(X)
    Y = np.asarray(Y)
    X = X.astype('float32')
    X = X/255
    indices = np.arange(X.shape[0])
    np.random.shuffle(indices)
```

```
    X = X[indices]
    Y = Y[indices]
    text.insert(END,'image processing completed\n')
    img = X[20].reshape(64,64,3)
    cv2.imshow('ff',cv2.resize(img,(250,250)))
    cv2.waitKey(0)


def cnnModel():
    global model
    global accuracy
    text.delete('1.0', END)
    if os.path.exists('model/model.json'):
        with open('model/model.json', "r") as json_file:
            loaded_model_json = json_file.read()
            model = model_from_json(loaded_model_json)
        json_file.close()
        model.load_weights("model/model_weights.h5")
        model._make_predict_function()
        print(model.summary())
        f = open('model/history.pckl', 'rb')
```

```
        accuracy = pickle.load(f)
        f.close()
        acc = accuracy['accuracy']
        acc = acc[9] * 100
        text.insert(END,"CNN Crop Disease Recognition Model Prediction Accuracy = "+str(acc))
    else:
        model = Sequential() #resnet transfer learning code here
        model.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation = 'relu'))
        model.add(MaxPooling2D(pool_size = (2, 2)))
        model.add(Convolution2D(32, 3, 3, activation = 'relu'))
        model.add(MaxPooling2D(pool_size = (2, 2)))
        model.add(Flatten())
        model.add(Dense(output_dim = 256, activation = 'relu'))
        model.add(Dense(output_dim = 15, activation = 'softmax'))
        model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
        print(model.summary())
        hist = model.fit(X, Y, batch_size=16, epochs=10, validation_split=0.2, shuffle=True, verbose=2)
        model.save_weights('model/model_weights.h5')
        model_json = model.to_json()
```

```
        with open("model/model.json", "w") as json_file:
            json_file.write(model_json)
        json_file.close()
        f = open('model/history.pckl', 'wb')
        pickle.dump(hist.history, f)
        f.close()
        f = open('model/history.pckl', 'rb')
        accuracy = pickle.load(f)
        f.close()
        acc = accuracy['accuracy']
        acc = acc[9] * 100
        text.insert(END,"CNN Crop Disease Recognition Model Prediction Accuracy = "+str(acc))


def predict():
    global model
    filename = filedialog.askopenfilename(initialdir="testImages")
    img = cv2.imread(filename)
    img = cv2.resize(img, (64,64))
    im2arr = np.array(img)
    im2arr = im2arr.reshape(1,64,64,3)
```

```
    test = np.asarray(im2arr)

    test = test.astype('float32')

    test = test/255

    preds = model.predict(test)

    predict = np.argmax(preds)

    img = cv2.imread(filename)

    img = cv2.resize(img, (800,400))

    cv2.putText(img, 'Crop Disease Recognize as : '+plants[predict], (10, 25), cv2.FONT_HERSHEY_SIMPLEX,0.7, (0, 255, 0), 2)

    cv2.imshow('Crop Disease Recognize as : '+plants[predict], img)

    cv2.waitKey(0)


def graph():

    acc = accuracy['accuracy']

    loss = accuracy['loss']

    plt.figure(figsize=(10,6))

    plt.grid(True)

    plt.xlabel('Iterations')

    plt.ylabel('Accuracy/Loss')

    plt.plot(acc, 'ro-', color = 'green')

    plt.plot(loss, 'ro-', color = 'blue')
```

```
    plt.legend(['Accuracy', 'Loss'], loc='upper left')
    #plt.xticks(wordloss.index)
    plt.title('Iteration Wise Accuracy & Loss Graph')
    plt.show()


def close():
    main.destroy()
    text.delete('1.0', END)


font = ('times', 15, 'bold')
title = Label(main, text='Research on Recognition Model of Crop Diseases and Insect Pests Based on Deep Learning in Harsh Environments')
#title.config(bg='powder blue', fg='olive drab')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)


font1 = ('times', 13, 'bold')
ff = ('times', 12, 'bold')


uploadButton = Button(main, text="Upload Crop Disease Dataset", command=uploadDataset)
```

```
uploadButton.place(x=20,y=100)

uploadButton.config(font=ff)

processButton = Button(main, text="Image Processing & Normalization", command=imageProcessing)

processButton.place(x=20,y=150)

processButton.config(font=ff)


modelButton = Button(main, text="Build Crop Disease Recognition Model", command=cnnModel)

modelButton.place(x=20,y=200)

modelButton.config(font=ff)


predictButton = Button(main, text="Upload Test Image & Predict Disease", command=predict)

predictButton.place(x=20,y=250)

predictButton.config(font=ff)


graphButton = Button(main, text="Accuracy & Loss Graph", command=graph)

graphButton.place(x=20,y=300)

graphButton.config(font=ff)


exitButton = Button(main, text="Exit", command=close)
```

```
exitButton.place(x=20,y=350)

exitButton.config(font=ff)

font1 = ('times', 12, 'bold')

text=Text(main,height=30,width=85)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=450,y=100)

text.config(font=font1)



main.config()

main.mainloop()
```

REFERENCES

REFERENCES

[1]https://www.researchgate.net/publication/327065422_Plant_Disease_Detection_Using_Machine_Learning.

[2] https://link.springer.com/chapter/10.1007%2F978-981-15-7219-7_23.

[3] J. Schmidhuber, ‘‘Deep learning in neural networks: An overview,’’ Neural Netw., vol. 61, pp. 85–117, Jan. 2015.

[4] A.-R. Mohamed, G. E. Dahl, and G. Hinton, ‘‘Acoustic modeling using deep belief networks,’’ IEEE Trans. Audio, Speech, Lang. Process., vol. 20, no. 1, pp. 14–22, Jan. 2012.

[5] Y. Bengio and O. Delalleau, ‘‘On the expressive power of deep architectures,’’ in Proc. 14th Int. Conf. Discovery Sci. Berlin, Germany, 2011, no. 1, pp. 18–36.

[6] L. Deng, O. Abdel-Hamid, and D. Yu, ‘‘A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion,’’ in Proc. IEEE Int. Conf. Acoust., Speech Signal Process., May 2013, pp. 6669–6673.

[7] S. Datir and S. Wagh, ‘‘Monitoring and detection of agricultural disease using wireless sensor network,’’ Int. J. Comput. Appl., vol. 87, no. 4, pp. 1–5, Feb. 2014.

[8] O. López, M. Rach, H. Migallon, M. Malumbres, A. Bonastre, and J. Serrano, ‘‘Monitoring pest insect traps by means of low-power image sensor technologies,’’ Sensors, vol. 12, no. 11, pp. 15801–15819, Nov. 2012.

[9] N. Srivastav, G. Chopra, P. Jain, and B. Khatter, ‘‘Pest Monitor and control system using WSN with special referance to acoustic device,’’ in Proc. 27th ICEEE, Jan. 2013, pp. 92–99.

[10] G. Athanikar and M. P. Badar, ‘‘Potato leaf diseases detection and classification system,’’ Int. J. Comput. Sci. Mobile Comput., vol. 5, no. 2, pp. 76–88, 2016.

[11] H. Wang, G. Li, Z. Ma, and X. Li, ‘‘Application of neural networks to image recognition of plant diseases,’’ in Proc. Int. Conf. Syst. Informat. (ICSAI), May 2012, pp. 2159–2164.

[12] D. Samanta, P. P. Chaudhury, and A. Ghosh, ‘‘Scab diseases detection of potato using

image processing,'' Int. J. Comput. Trends Technol., vol. 3, no. 1, pp. 109–113, 2012.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ''ImageNet classification with deep convolutional neural networks,'' in Proc. Int. Conf. Neural Inf. Process. Syst. Red Hook, NY, USA: Curran Associates, 2012, pp. 1097–1105.

[14] K. He, X. Zhang, S. Ren, and J. Sun, ''Deep residual learning for image recognition,'' in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2016, pp. 770–778.

[15] E. C. Too, L. Yujian, S. Njuki, and L. Yingchun, ''A comparative study of fine-tuning deep learning models for plant disease identification,'' Comput. Electron. Agricult., vol. 161, pp. 272–279, Jun. 2019.

[16] S. P. Mohanty, D. P. Hughes, and M. Salathé, ''Using deep learning for image-based plant disease detection,'' Frontiers Plant Sci., vol. 7, p. 1419, Sep. 2016.

[17] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, ''Deep neural networks based recognition of plant diseases by leaf image classification,'' in Proc. Comput. Intell. Neurosci., May 2016, Art. no. 3289801.