**----College Emblem----**

**Department of Computer Science & Engineering**

**CERTIFICATE**

This is to certify that the project entitled **“Detection of Diseases in Blueberry Leaves using Machine Learning.”** has been submitted by **----,----,----,----,** to the partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering from Jawaharlal Nehru Technological University, Hyderabad. The results embodied in this project have not been submitted to any other University or Institution for the award of any degree or diploma.

**Internal Guide Head of the Department**

**External Examiner**

**DECLARATION**

We hereby declare that the project entitled **“Detection of Diseases in Blueberry Leaves using Machine Learning.”** is the work done during the period from –-- **to –--** and is submitted to the partial fulfillment of the requirements for the award of degree of Bachelor of technology in Computer Science and Engineering from Jawaharlal Nehru Technology University, Hyderabad. The results embodied in this project have not been submitted to any other University or Institution for the award of any degree or diploma.

**–--Name1**

**(–--Rollno1)**

**–--Name2**

**(–--Rollno2)**

**–--Name3**

**(–--Rollno3)**

**–--Name4**

**(–--Rollno4)**

**ACKNOWLEDGEMENT**

There are many people who helped us directly and indirectly to complete our project successfully. We would like to take this opportunity to thank one and all.

First of all, we would like to express our deep gratitude towards our internal guide –--**,** Department of CSE for his support in the completion of our dissertation. We wish to express our sincere thanks to **–--, HOD Dept. of CSE** and also to our principal **–--** for providing the facilities to complete the dissertation.

We would like to thank all our faculty and friends for their help and constructive criticism during the project period. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

–--name1 (roll no.1)

–--name2 (roll no.2)

–--name3 (roll no.3)

–--name4 (roll no.4)

**ABSTRACT**

**Agriculture plays a major role in the present society. Developing effective technologies to improve agriculture, is important. One such technology is detection of plant diseases using leaf images. This project uses an algorithm to take the image of a leaf of a plant under study, analyzing the leaf image and then determining the disease in the plant, if there is any. The algorithm is based on the image template matching technique using Normalized Square Differences Based Image Matching formula.**

**The system has three modules. The DTD (Disease types module) is used to store the details of different types of plant diseases in the Database, the DA (Disease Analysis) module is used to analyze the disease of a given plant, using its leaf image and the IP (Image Processing) module is used to pin point the affected area of the leaf.**

**This Project uses PyQt tool to create the needed Graphical User Interfaces, PyUIC module to automatically generate the automated code .**

**The images of various types of disease effected leaves are stored in an image file system. The file system will be stored at a specific location in directory structure. The location will be fed as in input to the system through a GUI screen.**

**Algorithm Used: Normalized Square Differences method.**

**Outputs from the project:**

**(1) A set of graphical User Interfaces to control project operations.**

**(2) Three different leaf images with disease**

**(3) Images showing the effected part of the leves.**

**CHAPTER 1**

**INTRODUCTION**

**Our project is “Detection of Diseases in Blueberry Leaves using Machine Learning.”, for this we have chosen python which is robust, easy to understand and we can minimize the code through various predefined functions. To develop this, we have used different tools like PYQT Designer, PYUIC and python. Each and every tool plays a major role. PYQT Designer tool is used to design user interfaces, PYUIC tool is helpful to generate python code for user interfaces. Detection of Diseases in Blueberry Leaves using Machine Learning. is developed using python, pyuic and pyqt designer.PyQt is a Python binding of the cross-platform GUI toolkit Qt, implemented as a Python plug-in.**

**PyQt implements different classes and methods including:classes for accessing SQL databases (ODBC, MySQL, PostgreSQL, Oracle,SQLite),Scintilla-based rich text editor widget, data aware widgets that are automatically populated from a database, an XML parser and SVG support. Scalable Vector Graphics (SVG) is an XML-based vector image format for graphics with support for interactivity. All the above mentioned features of PyQt, are extensively used in this project, to create the needed Graphical User Interfaces.**

**PyUic tool is used automatically generate the code for the Front end user interfaces created by PyQt. All the front end python code is automatically generated by this tool, by converting the user interface(.ui) files into .py files.**

**1.1 ADVANTAGES**

**Following are the technical advantages:**

**Using Python, which is chosen as the best programming language, by the Programming Community.**

**More Functionality can be implemented with less no.of lines of code in Python.**

**PyQt tool is used to create the Graphical User interfaces.**

**All the Front end code is generated automatically by PyUIC.**

**Following are the other project advantages:**

**The project is very useful to the farmers, as they need not go for the plant specialist consultation towards the identification of the disease.**

**The project is also useful to the plant specialists, as they need not remember the details about each and every plant disease.**

**This project finally leads to the enhancement average plant life.**

**1.2 PROBLEM DEFINITION**

**Vision:**

* The project aims at developing a tool for the Detection of Diseases in Blueberry Leaves using Machine Learning., with all the above mentioned advantages.

**Mission:**

* **This tool is developed by using Python along with its layout toolkit PyQt & PyUIC.**

**1.3 Literature Survey**

# Types of pests and insects

Following are different types of pests.

* Beneficial Insects
* Garden Pests
* Household Pests
* Houseplant Pests
* Landscape/ Lawn Pests
* Orchard & Tree Pests

**Beneficial Insects:**

Not all insects are pests. In fact, there are a great many good bugs for your garden. Many growers recognize the help provided by beneficial insects as well as the injury done by harmful ones. As per C.B Huffaker "When we kill off the natural enemies of a pest, we inherit their work.”

Following are some of the beneficial insects:

Aphid Parasite,Aphid Predator,Beneficial Nematode,Bumblebee,Fly Predator,Fungus Gnat Predator,Green Lacewing,Lady Beetle,Leafminer Parasite,Mealybug Destroyer,Moth Egg Parasite,Pirate Bug,Praying Mantis,Soldier Bug,Spider Mite Predator,Thrips Predator & Whitefly Parasite.

### Garden Pests:

One bug does not make a problem! In nature, there are always some garden pests chewing on plants; that’s just the way it is. However, not all pest damage is significant enough to warrant action. Even the healthiest gardens encounter bugs at one time or another, yet they still produce a beautiful harvest. As gardeners, we must each consider the level of pest activity that we are willing to tolerate.

The best way to maintain a healthy garden is to educate yourself and learn to identify common “bad bugs.” Inspect your garden regularly to detect problems early. The sooner a pest is identified the easier it will be to manage using earth-friendly methods.

Following are some of the Garden Pests:

Animal,Aphid,Armyworm,Asparagus Beetle,Bean Beetle,Blister Beetle,Cabbage Looper,Cabbage Worm,Corn Earworm,Cucumber Beetle,Cutworm, Deer,Earwig,European Corn Borer, Flea Beetle,Grasshopper,Leafhopper,Leafminer,Potato Beetle,,Psyllid,Root Maggot,Slug & Snail,Sowbug & Pillbug,Spider Mite,Squash Bug,Stink Bug,Thrips,Tomato Hornworm,

Whitefly & Wireworm.

### Household Pests:

All homes occasionally run into problems with household pests. While most are merely a nuisance, some may bite, sting or transmit disease. A few may even cause serious structural damage which can impact the value of your house. While it may seem easier to reach for a can of bug spray, this may not be the best way to fix the problem. Many homeowners today are leery of the harmful effects of chemical pesticides and are turning to safer, least-toxic solutions for protection.

Understanding the problem is the first step in finding a healthy solution. Once a pest is identified, you will be better able to determine the factors which limit its reproduction and survival potential. For example, all pests need water, shelter and food to survive. By limiting one or more of these basic necessities you can significantly impact the number of pests present. Combined with natural control measures, a longer lasting, more significant impact is made.

Following are some of the Household Pests.

Ant,Bed Bug,Boxelder Bug,Carpenter Ant,Clothes Moth,Cockroach,Dust Mite,Flea,Fly,Fruit Fly,Home Remedies,Mice,Pantry Moth,Spider,Stink Bug & Termite.

### Houseplant Pests:

Like all plants, houseplants will occasionally come under attack from pests. These insects can be just as voracious as their outdoor counterparts but have the added benefit of developing and reproducing in near ideal conditions. As a result, houseplant pests can multiply very quickly, so you have to be diligent about checking for symptoms. If you notice a plant that suddenly begins to look ill, take a closer look — chances are an insect is responsible. Infestations can be very severe and plants that have had more than half of their leaves damaged are probably not worth saving. For this reason, it is important to identify and control indoor plant pests as quickly as possible.

Following are some of the Houseplant Pests

Aphid,Fungus Gnat,Leafminer,Mealybug,Root Aphid,Russet Mite,Scale Insect,Spider Mite,Springtail,Thrips & Whitefly

### Landscape/ Lawn Pests:

Healthy lawns and landscapes are teaming with insect life. Yet, very few of the species you see in your backyard are damaging (it is estimated that less than 1% of all insect species are pests). The point here, is that once lawn pests are discovered don’t go crazy and spray all kinds of harmful chemicals to eradicate the problem. Broad-spectrum insecticides kill all insects, both good and bad.

One of the first steps to controlling lawn pests,requires a basic understanding of the insect itself. Knowing the life-cycle can help determine what control measures should be used and when they will be most effective. A common-sense approach to pest control involves using least-toxic methods first — barriers & repellents, beneficial insects, biological pesticides, soaps and oils — with the more toxic (but short lived) botanical insecticides used only if necessary.

Following are some of the Landscape/ Lawn Pests:

Animal,Ant,Aphid,Black Vine Weevil,Blister Beetle,Boxelder Bug,Carpenter Ant,Chinch Bug,Cutworm,Deer,Dollar Spot,Earwig,Fairy Ring,Fire Ant,Flea,Fly,Grasshopper,

Gypsy Moth,Japanese Beetle,Leafhopper,Leafminer,Mealybug,Mole,Mosquito,Sawfly,Scale Insect,Slug & Snail,Snow Mold,Sod Webworm,Sowbug & Pillbug,Spider Mite,

Springtail,Tent Caterpillar,Thrips,Tick,Whitefly & Yellow Jacket

### Orchard & Tree Pests:

Trees and ornamental shrubs will eventually encounter a variety of bugs. The difficulty is determining when, or if, many of these problems will reach a level where something must be done about it. Some tree pests are present every few growing seasons and require immediate attention, while many others are found each year but cause little or no harm.Maintaining healthy, productive trees means knowing about common pests. Survey your backyard regularly to identify problems at an early stage. Identifying the problem and evaluating its severity will help decide if control is necessary and, if so, what management strategy to take. The tree pests listed below are some (of the many) that homeowners are likely to encounter.

Following are some of the Orchard & Tree Pests:

Apple Maggot,Codling Moth,Gypsy Moth,Peachtree Borer,Pecan Weevil,Psyllid,Sawfly,Scale Insect,Spruce Budworm & Tent Caterpillar

**CHAPTER 2**

**SYSTEM ANALYSIS**

**2.1 Requirement Analysis**

**2.1.1 Hardware Requirements**

1. It requires a minimum of 2.16 GHz processor.

2. It requires a minimum of 4 GB RAM.

3. It requires 64-bit architecture.

4. It requires a minimum storage of 500GB.

**2.1.2 Software Requirements**

1. It requires a 64-bit Ubuntu Operating System.

2. Python Qt Designer for designing user interface.

3. MY SQL server for storing database Entities.

4. Pyuic for converting the layout designed user interface (UI) to python code.

5. Python language for coding.

**2.2 Feasibility Analysis**

As the name implies, a feasibility study is used to determine the viability of an idea, such as ensuring a project is legally and technically feasible as well as economically justifiable. It tells us whether a project is worth the investment—in some cases, a project may not be doable. There can be many reasons for this, including requiring too many resources, which not only prevents those resources from performing other tasks but also may cost more than an organization would earn back by taking on a project that isn’t profitable.

**2.2.1 Economical Feasibility**

This assessment typically involves a cost/ benefits analysis of the project, helping organizations determine the viability, cost, and benefits associated with a project before financial resources are allocated. It also serves as an independent project assessment and enhances project credibility—helping decision makers determine the positive economic benefits to the organization that the proposed project will provide. Our project is economically feasible because in this we have used “UBUNTU”, “PYTHON”, “PYQT” designer tool and “PYUIC” which are all available as an open source.

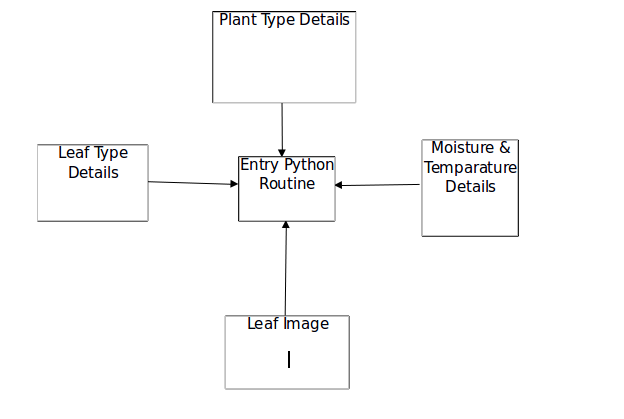
**2.2.2 Technical Feasibility**

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity. Technical feasibility also involves evaluation of the hardware, software, and other technology requirements of the proposed system. A prototype of the tool was developed to verify the technical feasibility. The prototype is working successfully and hence the project is feasible.

**CHAPTER 3**

**SYSTEM DESIGN**

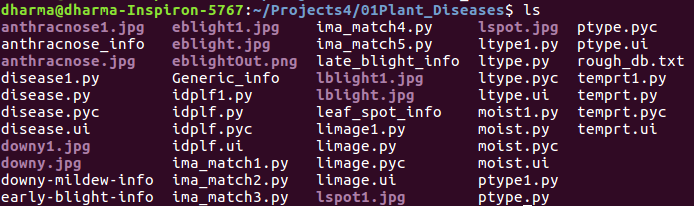
**3.1 Module Description**

****

**Fig 1: Data Flow Diagram**

Details like plant type, leaf type, moisture and temperature are to be provided as Input to the system, using the corresponding user interfaces. Th following section describe the screen shots of these interfaces.

**Screen shots and descriptions:**



Pic1: Screen shot showing the files created during the project

The above screen shot shows different files created during this projects. There are four different types of files: (1) .jpg files (2) .ui files (3).py files and (4).txt files

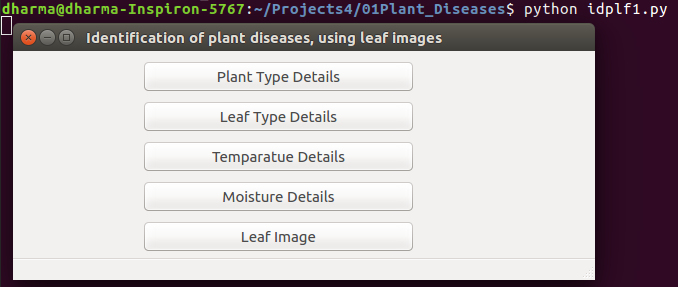
.jpg files contains the needed images like leaf and disease images that are needed

in this project.

.ui files are the user interface files, created by using PyQt layout editor

.py files are python program files created either manually, or automatically. For instance, each .ui file has a corresponding .py file that is created automatically by using the PyUIC tool. .txt files contains the generic useful information about the project.

Idplf1.py, is the entry program for this project. Execution of this python program leads to the entry screen as follows:



This entry screen consists of five push buttons. Upon clicking the first button, ptype1.py program is instantiated resulting in a screen , by using which the user can enter the details of various types of plants.

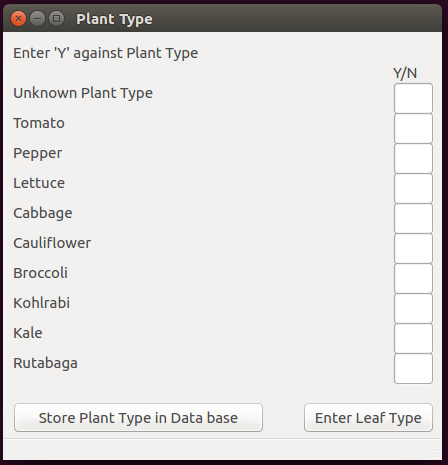
The second button leads to the invoking of ltype1.py program, which results in a screen, where the user can enter the details of the leaves.

Upon clicking the third button, temprt1.py program is instantiated resulting in a screen , by using which the user can enter the temperature details.

Upon clicking the fourth button, moist1.py program is instantiated resulting in a screen , by using which the user can enter the moisture details.

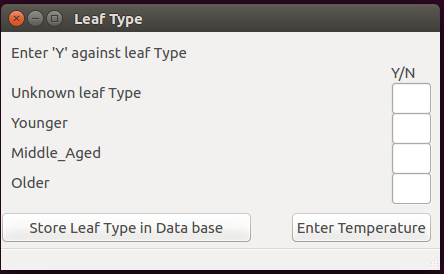
The fifth button leads to the invoking of ltype1.py program, which results in a screen, where the user can enter the details of the leaves.

ptype1.py program leads to the following screen.

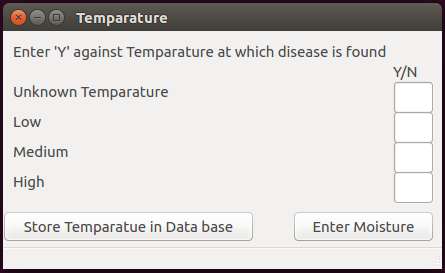


The user can enter the type of the plant, and store it in the DB, by using the above screen.

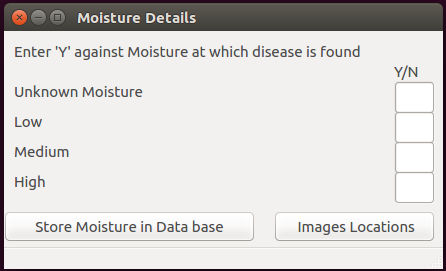
The user can enter the leaf details like whether the leaf is young/middle\_aged/old/unknown, by using the following screen.



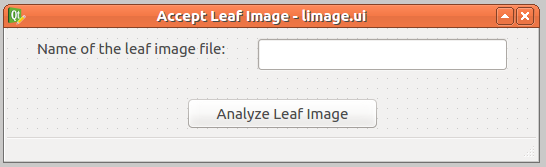
The temperature details can be entered and stored in the system, by using the following screen.



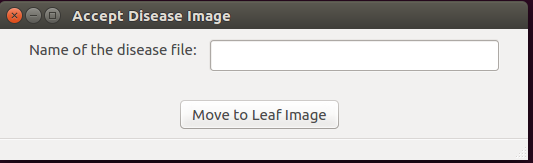
The following screen is used to store the moisture details in the Database.



The name and location of the leaf image file, can be provided as input to the system by using the following GUI screen.



The name and location of the disease image files, can be provided as input to the system by using the following GUI screen.



**PYTHON:** Python is a widely used high level programming language for general purpose programming, created by Guido Van Rossum and first released in 1991. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale. Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library. Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python was conceived in the late 1980s, and its implementation began in December 1989 by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL) capable of exception handling and interfacing with the operating system Amoeba. Van Rossum is Python's principal author, and his continuing central role in deciding the direction of Python is reflected in the title given to him by the Python community, Benevolent Dictator for Life (BDFL). About the origin of Python, Van Rossum wrote in 1996.

Python 2.0 was released on 16 October 2000 and had many major new features, including a cycle-detecting garbage collector and support for Unicode. With this release the development process was changed and became more transparent and community-backed. Python 3.0 (initially described as Python 3000 or py3k), is a major, backward-incompatible release that was released after a long period of testing on 3 December 2008. Many of its major features have been back ported to the backwards-compatible Python 2.6.x and 2.7.x version series.

The End of Life date (EOL, sunset date) for Python 2.7 was initially set at 2015, then postponed to 2020 out of concern that a large body of existing code cannot easily be forward ported to Python 3. In January 2017, Google announced work on a Python 2.7 to Go trans compiler, which The Register speculated was in response to Python 2.7's planned end-of life but Google cited performance under concurrent workloads as their only motivation.

**3.2 Tools Used**

**Python Qt Designer**

**Qt:**

Qt is designed for developing applications and user interfaces once and deploying them across several desktop and mobile operating systems.

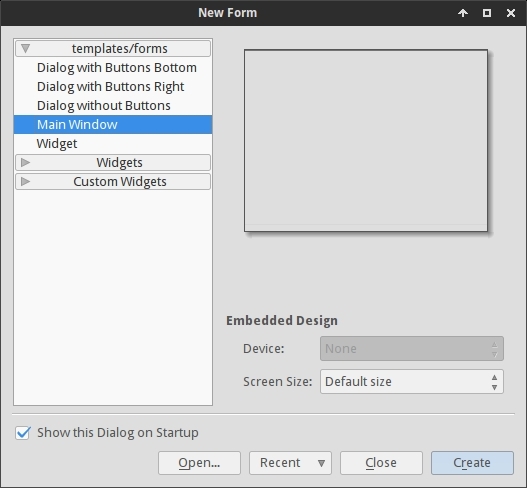
The easiest way to start application development with Qt is to [download](http://qt.io/download) and install Qt 5. It contains Qt libraries, examples, documentation, and the necessary development tools, such as the [Qt Creator](http://doc.qt.io/qtcreator/index.html) integrated development environment (IDE).

Qt Creator provides you with tools for accomplishing your tasks throughout the whole application development life-cycle, from creating a project to deploying the application on the target platforms. Qt Creator automates some tasks, such as creating projects, by providing wizards that guide you step-by-step through the project creation process, create the necessary files, and specify settings depending on the choices you make. Also, it speeds up some tasks, such as writing code, by offering semantic highlighting, checking code syntax, code completion, refactoring actions, and other useful features.

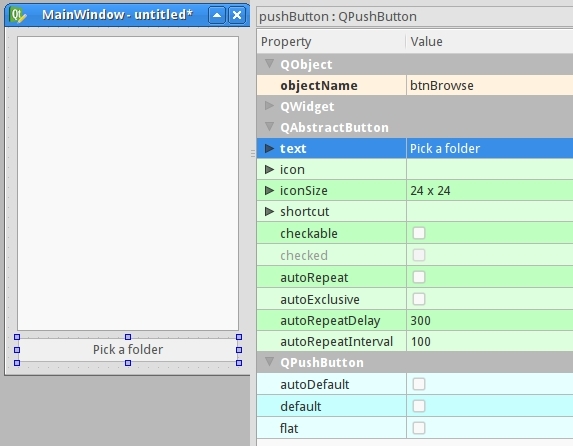
**Python Qt Designer**

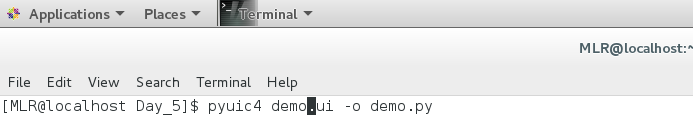
The PyQt installer comes with a GUI builder tool called Qt Designer. Using its simple drag and drop interface, a GUI interface can be quickly built without having to write the code. It is however, not an IDE such as Visual Studio. Hence, Qt Designer does not have the facility to debug and build the application.

Creation of a GUI interface using Qt Designer starts with choosing a top-level window for the application



You can then drag and drop required widgets from the widget box on the left pane. You can also assign value to properties of widget laid on the form.



The designed form is saved as demo.ui. This ui file contains XML representation of widgets and their properties in the design. This design is translated into Python equivalent by using pyuic4 command line utility. This utility is a wrapper for ui module. The usage of pyuic4 is as follows

|  |  |
| --- | --- |
| **S. No.** | **Widgets & Description** |
| 1 | **QLabel**  A QLabel object acts as a placeholder to display non-editable text or image, or a movie of animated GIF. It can also be used as a mnemonic key for other widgets. |
| 2 | **QLineEdit**  QLineEdit object is the most commonly used input field. It provides a box in which one line of text can be entered. In order to enter multi-line text, QTextEdit object is required. |
| 3 | **QPushButton**  In PyQt API, the QPushButton class object presents a button which when clicked can be programmed to invoke a certain function. |

**QPushButton**

**The QPushButton widget provides a command button.**

The push button, or command button, is perhaps the most commonly used widget in any graphical user interface. Push (click) a button to command the computer to perform some action, or to answer a question. Typical buttons are OK, Apply, Cancel, Close, Yes, No and Help.

A command button is rectangular and typically displays a text label describing its action. A shortcut key can be specified by preceding the preferred character with an ampersand in the text. For example:

QPushButton \*button = new QPushButton("&Download", this);

In this example the shortcut is Alt+D. See the QShortcut documentation for details (to display an actual ampersand, use '&&').

Push buttons display a textual label, and optionally a small icon. These can be set using the constructors and changed later using setText() and setIcon(). If the button is disabled the appearance of the text and icon will be manipulated with respect to the GUI style to make the button look "disabled".

A push button emits the signal clicked() when it is activated by the mouse, the Spacebar or by a keyboard shortcut. Connect to this signal to perform the button's action. Push buttons also provide less commonly used signals, for example, pressed() and released().

Command buttons in dialogs are by default auto-default buttons, i.e. they become the default push button automatically when they receive the keyboard input focus. A default button is a push button that is activated when the user presses the Enter or Return key in a dialog. You can change this with setAutoDefault(). Note that auto-default buttons reserve a little extra space which is necessary to draw a default-button indicator. If you do not want this space around your buttons, call setAutoDefault(false).

Being so central, the button widget has grown to accommodate a great many variations in the past decade. The Microsoft style guide now shows about ten different states of Windows push buttons and the text implies that there are dozens more when all the combinations of features are taken into consideration.

The most important modes or states are:

 Available or not (grayed out, disabled).

 Standard push button, toggling push button or menu button.

 On or off (only for toggling push buttons).

 Default or normal. The default button in a dialog can generally be "clicked" using the Enter or Return key.

 Auto-repeat or not.

 Pressed down or not.

As a general rule, use a push button when the application or dialog window performs an action when the user clicks on it (such as Apply, Cancel, Close and Help) and when the widget is supposed to have a wide, rectangular shape with a text label. Small, typically square buttons that change the state of the window rather than performing an action (such as the buttons in the top-right corner of the QFileDialog) are not command buttons, but tool buttons. Qt provides a special class (QToolButton) for these buttons.

If you need toggle behavior (see setCheckable()) or a button that auto-repeats the activation signal when being pushed down like the arrows in a scroll bar (see setAutoRepeat()), a command button is probably not what you want. When in doubt, use a tool button.

A variation of a command button is a menu button. These provide not just one command, but several, since when they are clicked they pop up a menu of options. Use the method setMenu() to associate a popup menu with a push button.

Other classes of buttons are option buttons (see QRadioButton) and check boxes (see QCheckBox).

In Qt, the QAbstractButton base class provides most of the modes and other API, and QPushButton provides GUI logic. See QAbstractButton for more information about the API.

**QLineEdit**

The QLineEdit widget is a one-line text editor.

A line edit allows the user to enter and edit a single line of plain text with a useful collection of editing functions, including undo and redo, cut and paste, and drag and drop.

By changing the echoMode() of a line edit, it can also be used as a "write-only" field, for inputs such as passwords.

The length of the text can be constrained to maxLength(). The text can be arbitrarily constrained using a validator() or an inputMask(), or both. When switching between a validator and an input mask on the same line edit, it is best to clear the validator or input mask to prevent undefined behaviour.

A related class is QTextEdit which allows multi-line, rich text editing.

You can change the text with setText() or insert(). The text is retrieved with text(); the displayed text (which may be different, see EchoMode) is retrieved with displayText(). Text can be selected with setSelection() or selectAll(), and the selection can be cut(), copy()ied and paste()d. The text can be aligned with setAlignment().

When the text changes the textChanged() signal is emitted; when the text changes other than by calling setText() the textEdited() signal is emitted; when the cursor is moved the cursorPositionChanged() signal is emitted; and when the Return or Enter key is pressed the returnPressed() signal is emitted.

When editing is finished, either because the line edit lost focus or Return/Enter is pressed the editingFinished() signal is emitted.

Note that if there is a validator set on the line edit, the returnPressed()/editingFinished() signals will only be emitted if the validator returns QValidator.Acceptable.

By default, QLineEdits have a frame as specified by the Windows and Motif style guides; you can turn it off by calling setFrame(false).

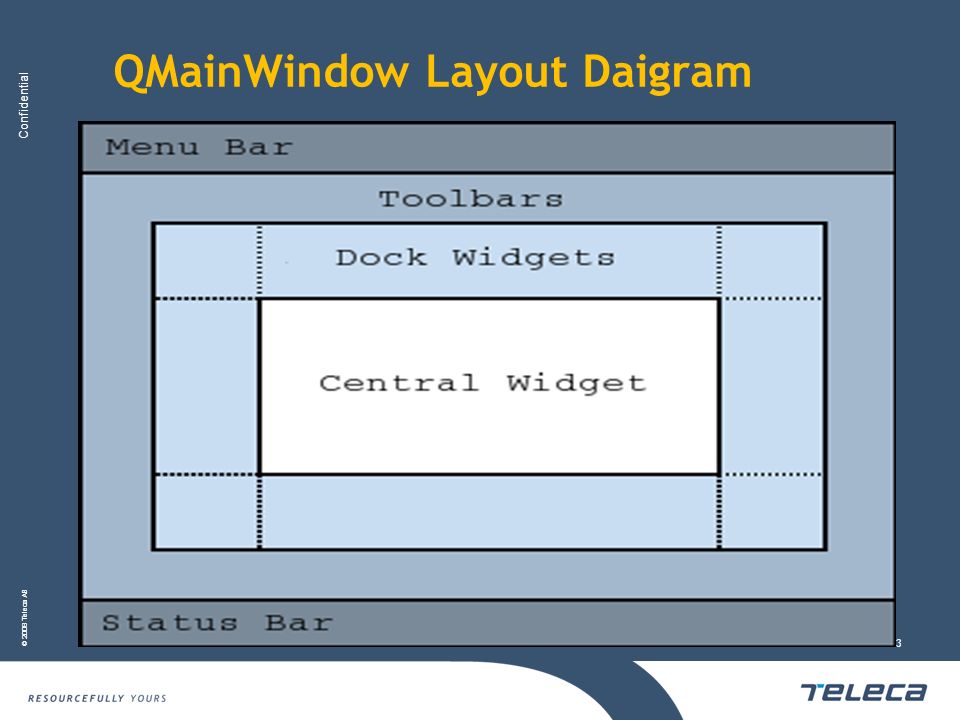
The default key bindings are described below. The line edit also provides a context menu (usually invoked by a right mouse click) that presents some of these editing options.

**QMainWindow**

**The QMainWindow class provides a main application window.**

Qt Main Window Framework

A main window provides a framework for building an application's user interface. Qt has QMainWindow and its related classes for main window management. QMainWindow has its own layout to which you can add QToolBars, QDockWidgets, a QMenuBar, and a QStatusBar. The layout has a centre area that can be occupied by any kind of widget. You can see an image of the layout below



**Note:** Creating a main window without a central widget is not supported. You must have a central widget even if it is just a placeholder.

Creating Main Window Components

A central widget will typically be a standard Qt widget such as a QTextEdit or a QGraphicsView. Custom widgets can also be used for advanced applications. You set the central widget with setCentralWidget().

Main windows have either a single (SDI) or multiple (MDI) document interface. You create MDI applications in Qt by using a QMdiArea as the central widget.

We will now examine each of the other widgets that can be added to a main window. We give examples on how to create and add them.

**Creating Menus**

Qt implements menus in QMenu and QMainWindow keeps them in a QMenuBar. QActions are added to the menus, which display them as menu items.

You can add new menus to the main window's menu bar by calling menuBar(), which returns the QMenuBar for the window, and then add a menu with QMenuBar::addMenu().

QMainWindow comes with a default menu bar, but you can also set one yourself with setMenuBar(). If you wish to implement a custom menu bar (i.e., not use the QMenuBar widget), you can set it with setMenuWidget().

Creating Main Window Components

A central widget will typically be a standard Qt widget such as a QTextEdit or a QGraphicsView. Custom widgets can also be used for advanced applications. You set the central widget with setCentralWidget().

Main windows have either a single (SDI) or multiple (MDI) document interface. You create MDI applications in Qt by using a QMdiArea as the central widget.

We will now examine each of the other widgets that can be added to a main window. We give examples on how to create and add them.

An example of how to create menus follows:

void MainWindow.createMenus()

{

fileMenu = menuBar()->addMenu(tr("&File"));

fileMenu->addAction(newAct);

fileMenu->addAction(openAct);

fileMenu->addAction(saveAct);

The createPopupMenu() function creates popup menus when the main window receives context menu events. The default implementation generates a menu with the checkable actions from the dock widgets and toolbars. You can reimplement createPopupMenu() for a custom menu.

**Creating Toolbars**

Toolbars are implemented in the QToolBar class. You add a toolbar to a main window with addToolBar(). You control the initial position of toolbars by assigning them to a specific Qt.ToolBarArea. You can split an area by inserting a toolbar break - think of this as a line break in text editing - with addToolBarBreak() or insertToolBarBreak(). You can also restrict placement by the user with QToolBar.setAllowedAreas() and QToolBar.setMovable().

The size of toolbar icons can be retrieved with iconSize(). The sizes are platform dependent; you can set a fixed size with setIconSize(). You can alter the appearance of all tool buttons in the toolbars with setToolButtonStyle().

An example of toolbar creation follows:

void MainWindow.createToolBars()

{

fileToolBar = addToolBar(tr("File"));

fileToolBar->addAction(newAct);

unction isWidgetType() returns whether an object is actually a widget. It is much faster than inherits(“QWidget" ).

Some QObject functions, e.g. children(), objectTrees() and queryList() return a QObjectList. A QObjectList is a QPtrList of QObjects. QObjectLists support the same operations as QPtrLists and have an iterator class, QObjectListIt. To convert the design file to python code saved as design.py, use cd command to change to the directory holding the design.ui file and simply run:

$ pyuic4 design.ui -o design.py

If you want to specify full path for either input or output file you can do that like this:

**$ pyuic4 path/to/design.ui -o output/path/to/design.py**

Writing the code

Now that we have the design.py file with the necessary design part of the application we can create our main application code and logic.

Create a file main.py in the same folder as your design.py file.

Using the design

For the application we'll need the following python modules imported:

**from PyQt4 import QtGui**

**import sys**

We also need the design code we created in the previous steps so add this too:

**import design**

Since the design file will be completely overwritten each time we change something in the design and recreate it we will not be writing any code in it, instead we'll create a new class e.g. ExampleApp that we'll combine with the design code so that we can use all of its features, like this:

**class ExampleApp(QtGui.QMainWindow, design.Ui\_MainWindow):**

**def \_\_init\_\_(self, parent=None):**

**super(ExampleApp, self).\_\_init\_\_(parent)**

**self.setupUi(self)**

In that class we'll interact with the GUI elements, add connections and everything else we need. But first we'll need to initialize that class on our code startup, we'll handle the class instance creation and other stuff in our main() function:

**def main():**

**app = QtGui.QApplication(sys.argv)**

**form = ExampleApp()**

**form.show()**

**app.exec\_()**

And to execute that main function we'll use well known:

**if \_\_name\_\_ == '\_\_main\_\_':**

**main()**

In the end our whole main.py file looks like this (with short explanations of the code):

from PyQt4 import QtGui # Import the PyQt4 module we'll need

import sys # We need sys so that we can pass argv to QApplication

import design # This file holds our MainWindow and all design related things

# it also keeps events etc that we defined in Qt Designer

class ExampleApp(QtGui.QMainWindow, design.Ui\_MainWindow):

def \_\_init\_\_(self):

super(self.\_\_class\_\_, self).\_\_init\_\_()

self.setupUi(self) # This is defined in design.py file automatically

# It sets up layout and widgets that are defined

def main():

app = QtGui.QApplication(sys.argv) # A new instance of QApplication

form = ExampleApp() # We set the form to be our ExampleApp (design)

form.show() # Show the form

app.exec\_() # and execute the app

if \_\_name\_\_ == '\_\_main\_\_': # if we're running file directly and not importing it

main() # run the main function

Running that will bring up our app running completely from python code!

But clicking button isn't doing anything, so we need to implement those features ourselves.

Implementing functions

(All of the following code is written inside the ExampleApp class)

Let's start with the "Pick a folder" button.

To connect a button event, such as clicked, to a function we use the following code:

**self.btnBrowse.clicked.connect(self.browse\_folder)**

And add it to the \_\_ini\_\_ method of our ExampleApp class so that it's set up when the application starts.

**Code Explanation:**

self.btnBrowse - btnBrowse is the name of the object we defined in Qt Designer. self is self exaplanatory and means that it belongs to current class.

clicked - the event we want to connect. Various elements have various events, for example list widgets have itemSelectionChanged etc.

connect() - used to specify with what we want to connect it with. In our example:

self.browse\_folder - simply a function name that we'll write inside our ExampleApp class:

For getting the directory browser dialog we can use the built in QtGui.QFileDialog.getExistingDirectory method like this:

directory = QtGui.QFileDialog.getExistingDirectory(self,"Pick a folder")

If the user picks a directory the directory variable will be equal to absolute path of the selected directory, otherwise it's None. To avoid running our code any further if the user cancels the folder browse dialog we'll use if directory: statement.

To list the directory contents, we'll need to add os to our imports:

import os

and to get current file list we can use os.listdir(path).

For adding items to the listWidget we use addItem() method on it, and to clear all existing items simply use self.listWidget.clear()

In the end our browse\_folder function looks something like this:

def browse\_folder(self):

self.listWidget.clear()

directory = QtGui.QFileDialog.getExistingDirectory(self,"Pick a folder")

if directory:

for file\_name in os.listdir(directory):

self.listWidget.addItem(file\_name)

Now you can run your app by typing python main.py and you should get the layout you desgined and picking the folder will populate list with folder items.

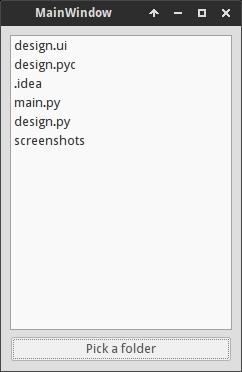
Finished main.py

from PyQt4 import QtGui # Import the PyQt4 module we'll need

import sys # We need sys so that we can pass argv to QApplication

import design # This file holds our Main Window and all design related things

# it also keeps events etc. that we defined in Qt Designer



import os # For listing directory methods

class ExampleApp(QtGui.QMainWindow, design.Ui\_MainWindow):

def \_\_init\_\_(self):

super(self.\_\_class\_\_, self).\_\_init\_\_()

self.setupUi(self) # This is defined in design.py file automatically

# It sets up layout and widgets that are defined

self.btnBrowse.clicked.connect(self.browse\_folder) # When the button is pressed

# Execute browse\_folder function

def browse\_folder(self):

self.listWidget.clear() # In case there are any existing elements in the list

directory = QtGui.QFileDialog.getExistingDirectory(self,

"Pick a folder")

# execute getExistingDirectory dialog and set the directory variable to be equal

# to the user selected directory

if directory: # if user didn't pick a directory don't continue

for file\_name in os.listdir(directory): # for all files, if any, in the directory

self.listWidget.addItem(file\_name) # add file to the listWidget

def main():

app = QtGui.QApplication(sys.argv) # A new instance of QApplication

form = ExampleApp() # We set the form to be our ExampleApp (design)

form.show() # Show the form

app.exec\_() # and execute the app

if \_\_name\_\_ == '\_\_main\_\_': # if we're running file directly and not importing it

main() # run the main function

That's the basic logic of using Qt Designer and PyQt to design and develop a GUI application.

**Image Template matching algorithm:**

**Template Matching is a method for searching and finding the location of a template image in a larger image. OpenCV comes with a function cv2.matchTemplate() for this purpose. It simply slides the template image over the input image (as in 2D convolution) and compares the template and patch of input image under the template image. Several comparison methods are implemented in OpenCV.**

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('messi5.jpg',0)

img2 = img.copy()

template = cv2.imread('template.jpg',0)

w, h = template.shape[::-1]

# All the 6 methods for comparison in a list

methods = ['cv2.TM\_CCOEFF', 'cv2.TM\_CCOEFF\_NORMED', 'cv2.TM\_CCORR',

'cv2.TM\_CCORR\_NORMED', 'cv2.TM\_SQDIFF', 'cv2.TM\_SQDIFF\_NORMED']

for meth in methods:

img = img2.copy()

method = eval(meth)

# Apply template Matching

res = cv2.matchTemplate(img,template,method)

min\_val, max\_val, min\_loc, max\_loc = cv2.minMaxLoc(res)

# If the method is TM\_SQDIFF or TM\_SQDIFF\_NORMED, take minimum

if method in [cv2.TM\_SQDIFF, cv2.TM\_SQDIFF\_NORMED]:

top\_left = min\_loc

else:

top\_left = max\_loc

bottom\_right = (top\_left[0] + w, top\_left[1] + h)

cv2.rectangle(img,top\_left, bottom\_right, 255, 2)

plt.subplot(121),plt.imshow(res,cmap = 'gray')

plt.title('Matching Result'), plt.xticks([]), plt.yticks([])

plt.subplot(122),plt.imshow(img,cmap = 'gray')

plt.title('Detected Point'), plt.xticks([]), plt.yticks([])

plt.suptitle(meth)

plt.show()

Methods used in template matching:

cv2.matchTemplate(image, templ, method[, result]) → result

|  |  |
| --- | --- |
| Parameters: | * image – Image where the search is running. It must be 8-bit or 32-bit floating-point. * templ – Searched template. It must be not greater than the source image and have the same data type. * result – Map of comparison results. It must be single-channel 32-bit floating-point. If image is  and templ is  , then result is  . * method – Parameter specifying the comparison method (see below). |

The function slides through image , compares the overlapped patches of size  against templ using the specified method and stores the comparison results in result . Here are the formulae for the available comparison methods (  denotes image,  template,  result ). The summation is done over template and/or the image patch: 

1. method=CV\_TM\_SQDIFF



1. method=CV\_TM\_SQDIFF\_NORMED



1. method=CV\_TM\_CCORR



1. method=CV\_TM\_CCORR\_NORMED



1. method=CV\_TM\_CCOEFF



where



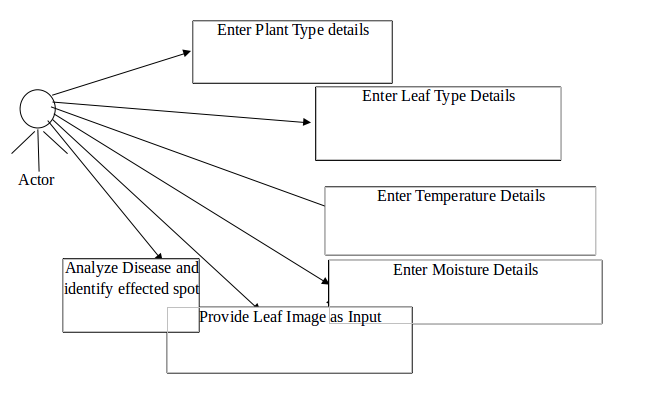
1. method=CV\_TM\_CCOEFF\_NORMED



**After the function finishes the comparison, the best matches can be found as global minimums (when CV\_TM\_SQDIFF was used) or maximums (when CV\_TM\_CCORR or CV\_TM\_CCOEFF was used) using the [minMaxLoc()](https://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html" \l "void minMaxLoc(InputArray src, double* minVal, double* maxVal, Point* minLoc, Point* maxLoc, InputArray mask)) function. In case of a color image, template summation in the numerator and each sum in the denominator is done over all of the channels and separate mean values are used for each channel. That is, the function can take a color template and a color image. The result will still be a single-channel image, which is easier to analyze.**

**3.3 UML DIAGRAMS**

**USE CASE DIAGRAM**

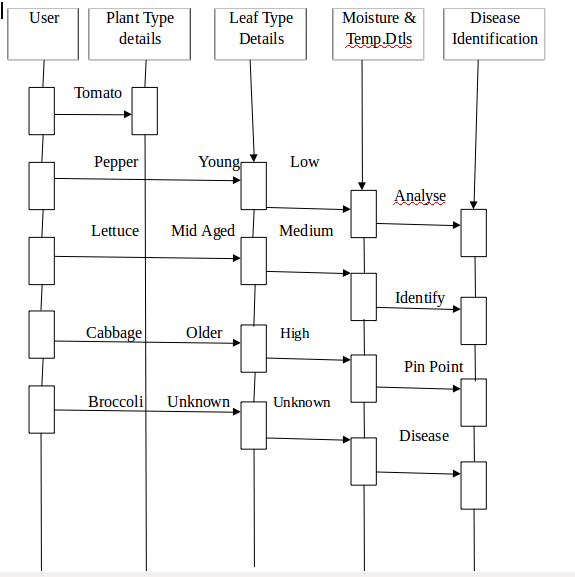
****

**Fig 1: USE CASE DIAGRAM**

**Description of Use case diagram**

Use case diagrams are usually referred to as behaviour diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

As we can see the user is interacting with system by a UI through which the customer can perform above mentioned operations like entering the plant type details, leaf type details, moisture details & Temperature details.

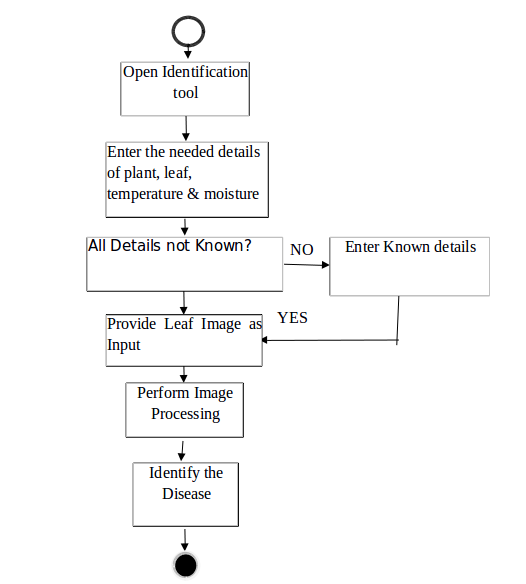


**SEQUENCE DIAGRAM**

**Description of sequence diagram**

A sequence diagram is an interaction diagram that shows how objects 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.

From above mentioned sequence diagram we have to go in sequence: Enter the needed details as shown in the above figure, Provide the leaf image as Input, Analyze the image & Identify the disease.



**Fig 3: ACTIVITY DIAGRAM**

**Description of Activity diagram**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow from 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. In activity diagram we can see that first we have to open the disease identification tool and then we will check whether all the needed details are there or not. If Yes go to next step else enter the needed details. After successfully entering the details the customer needs to provide the leaf image as input, and then click on ‘Analyze Leaf Imaget button which will lead to the analysis and identification of the disease.

**CHAPTER 4**

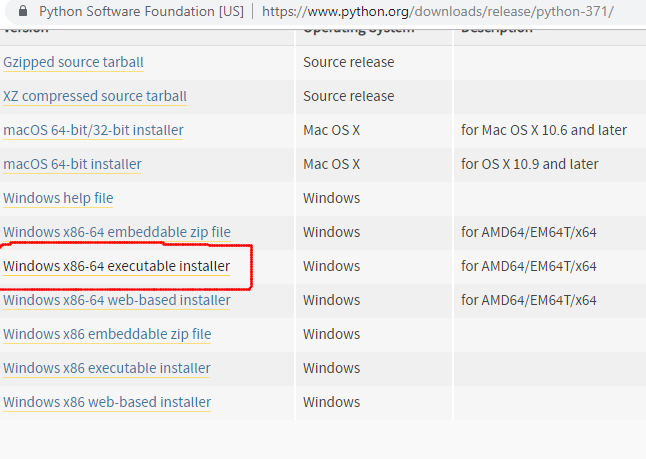
**IMPLEMENTATION**

**4.1 SOFTWARE USED**

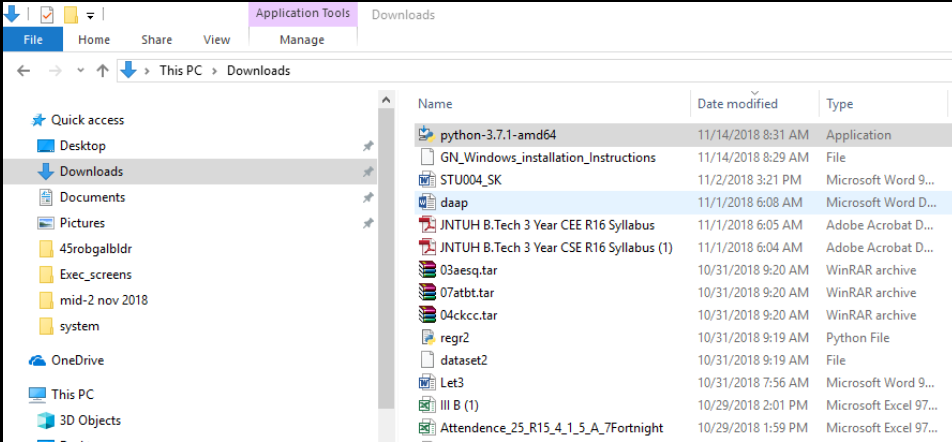
**Installation Process in Windows:**

**1) If the system don't have Python, then install Python by down loading it from the following site.**

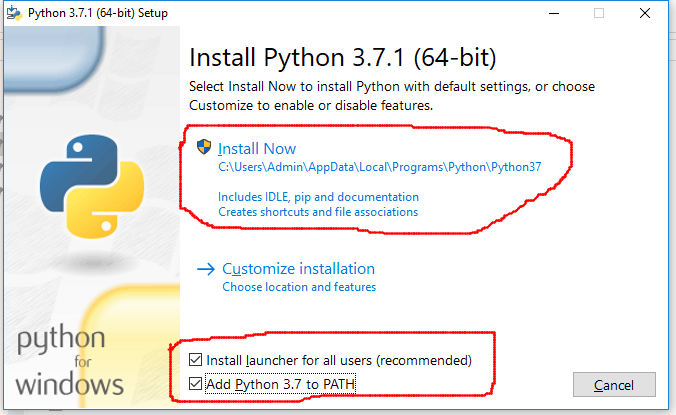
[**https://www.python.org/downloads/release/python-374/**](https://www.python.org/downloads/release/python-370/)

**1.a) Go to the above site, come to the bottom of the page, and click on ‘ Windows x86-64 executable Installer ‘, as shown in the above figure.**

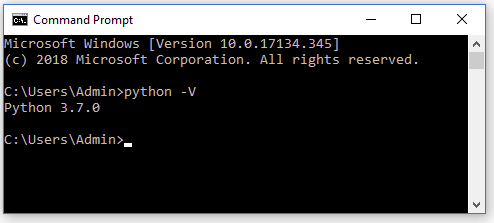
**1.b) Python Application will be downloaded into the ‘Downloads’ folder as shown in the following figure.**

****

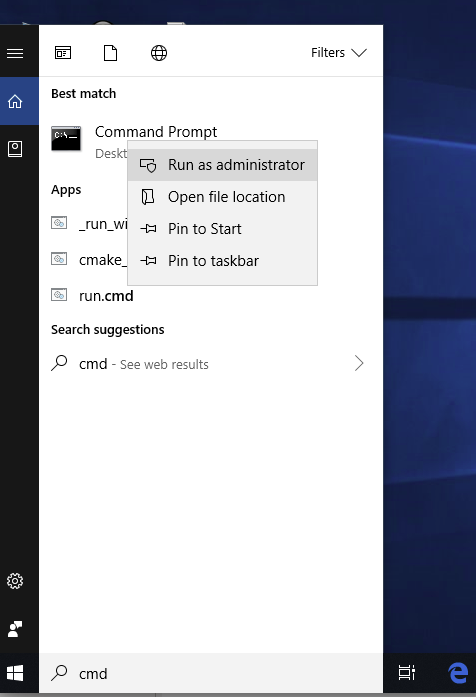
**1.c) Click on python ‘ python-3.7.1-amd64’ to get the following screen. Select both the options at the bottom of the screen, and click on “ Install Now“.**

****

**1.d) Follow the installation process, and, finish it. After completion, open command prompt, and give the command python -V, as shown below.**

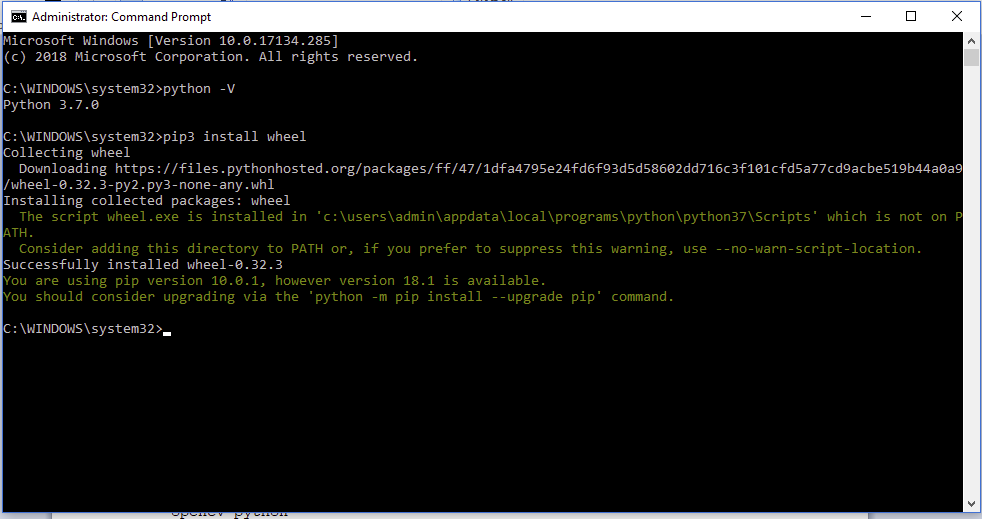
****

**2) Close the above command prompt, and re-open it as an administrator, as shown in the following figure.**

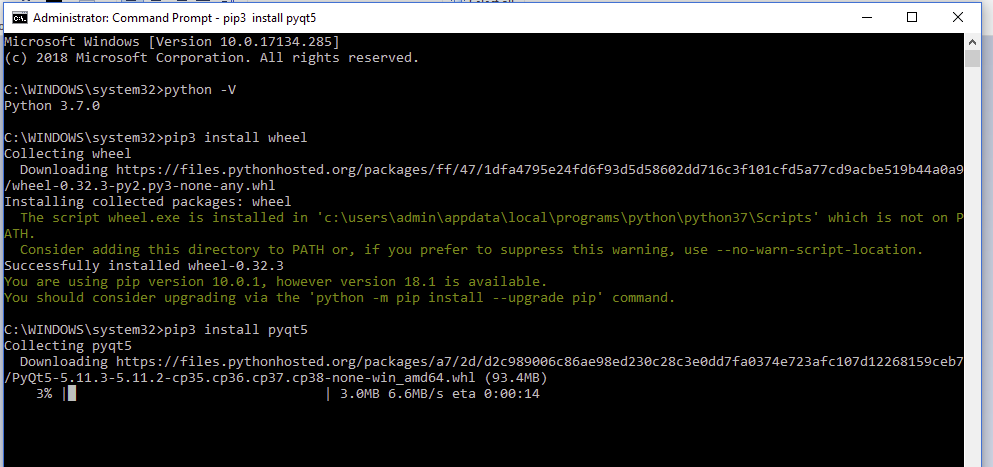
****

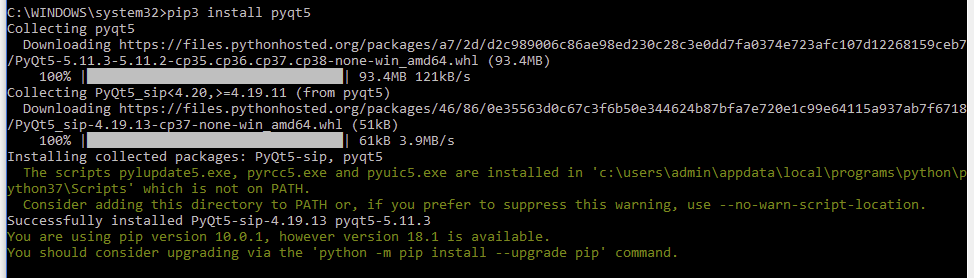
**(Note: When you RIGHT-CLICK on command prompt, the submenu will be opened. From there, you can run it as administrator.)**

**3.a)Execute the command from command prompt: pip3 install wheel. You should get the wheel successfully, installed as shown in the following figure. Otherwise, discuss with us, when we meet next time.**

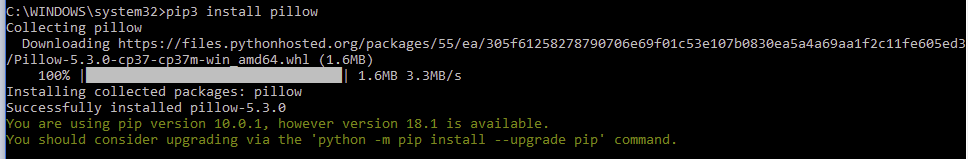
****

**4)Execute the command from command prompt: pip3 install pyqt5. You should get the pyqt5 successfully, installed as shown in the following two figures. Otherwise, discuss with us, when we meet next time.**

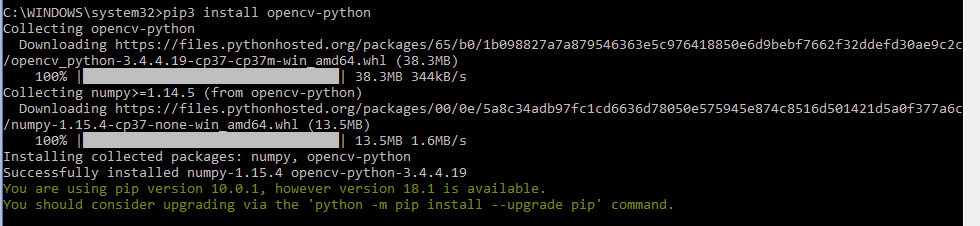
****

****

**5)Execute the command from command prompt: pip3 install pillow. You should get the pillow successfully, installed as shown in the following figure. Otherwise, discuss with us, when we meet next time.**

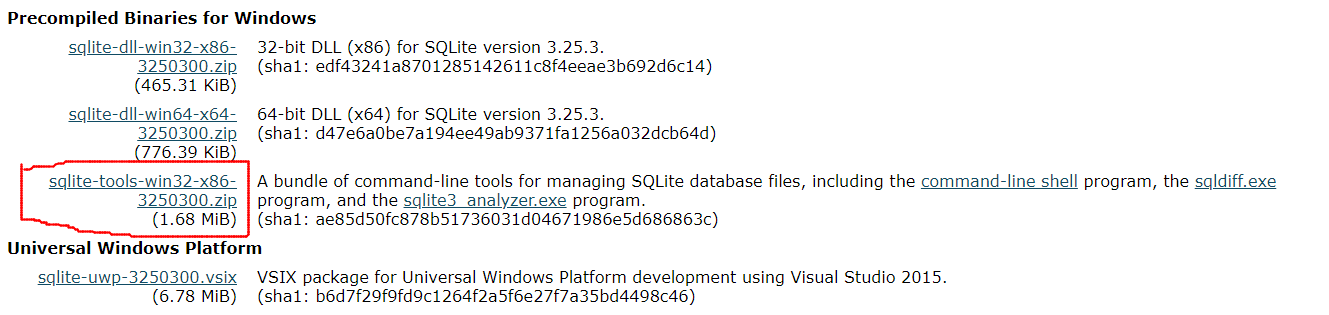
****

**6)Execute the command from command prompt: pip3 install opencv-python. You should get the opencv-python successfully, installed as shown in the following figure. Otherwise, discuss with us, when we meet next time.**

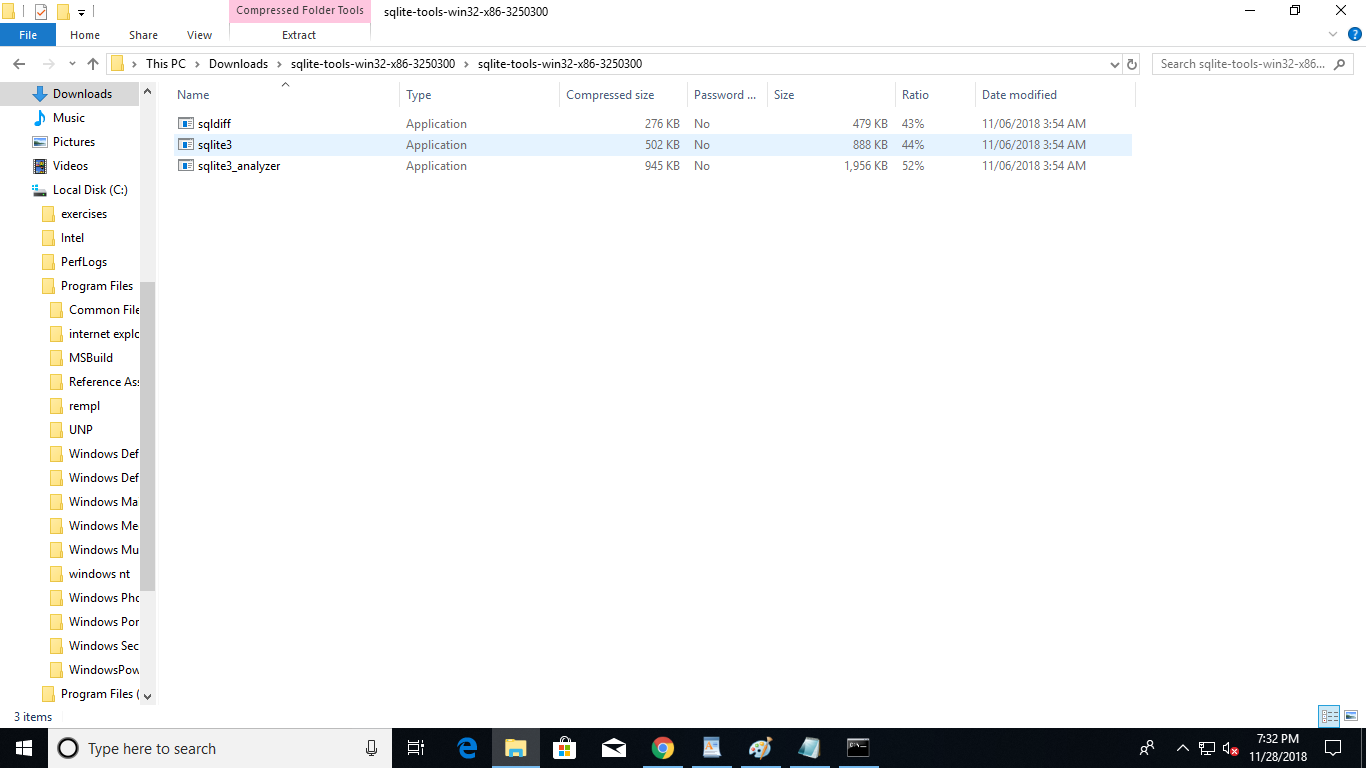
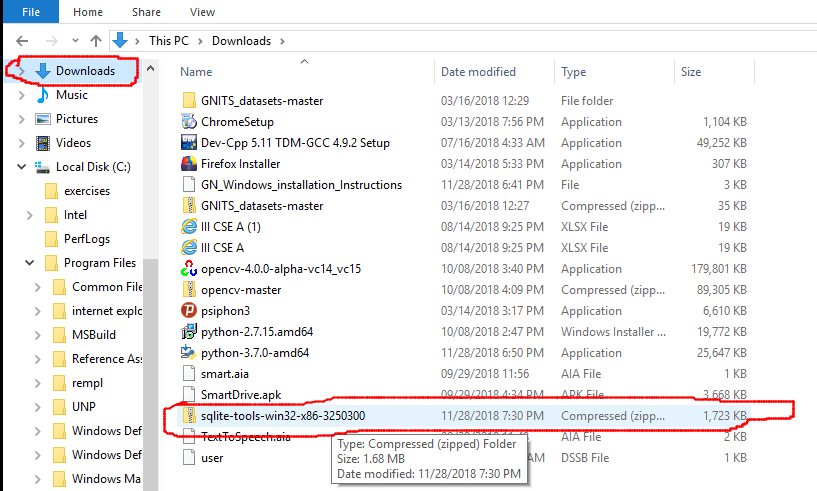
****

**6.a)Execute the command from command prompt(CMD): pip3 install pyqt5-dev-tools**

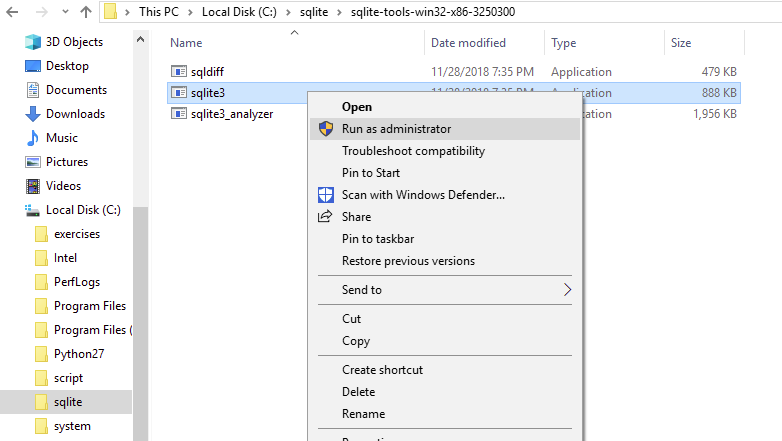
**7) Install sqlite by downloading it from the following site:** [**https://www.sqlite.org/download.html**](https://www.sqlite.org/download.html)**. Open this site, go to the bottom, and click on “sqlite-tools-win32-x86-32--------.zip”, as shown in the following figure.**

****

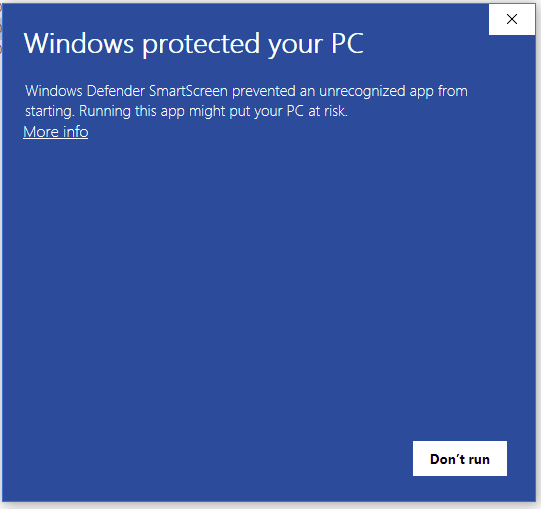
**7.a). The sqlite tools, will be downloaded into your downloads folder, as shown in the following figure.**

****

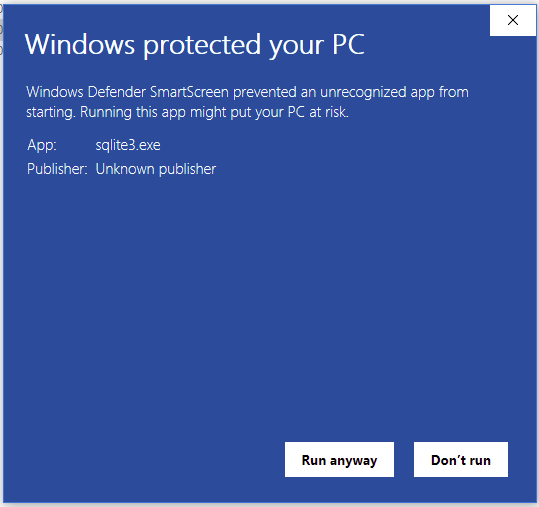
**7.b) Extract the three files in this compressed file, into a separate folder, and RIGHT CLICK on ‘ sqlite3 ‘ as shown in the following figure.**

****

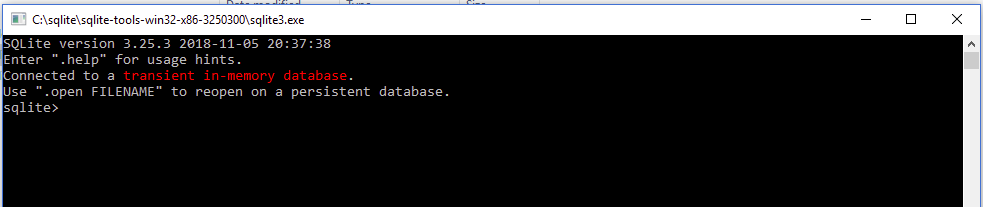
**7.c) You are likley to get a message, as shown in the following figure.**

****

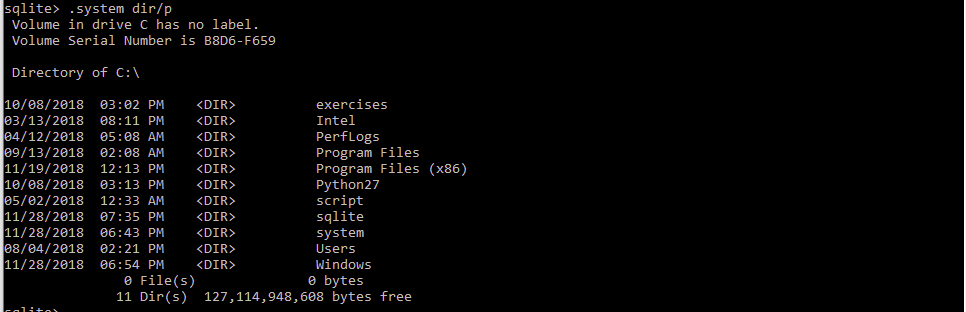
**7.d) Click on ‘More info’, if you get the above message. Click on ‘ Run anyway’ as shown in the following figure.**

****

**7.e) You should get a screen as shown below.**

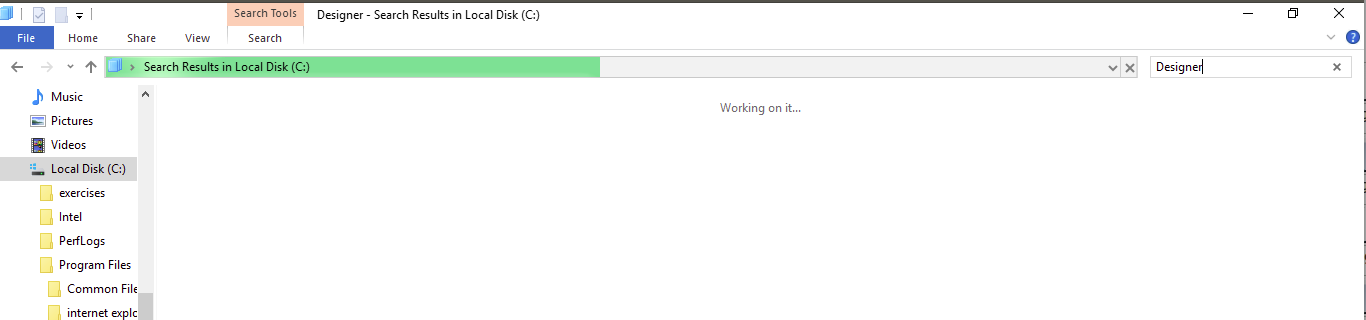
****

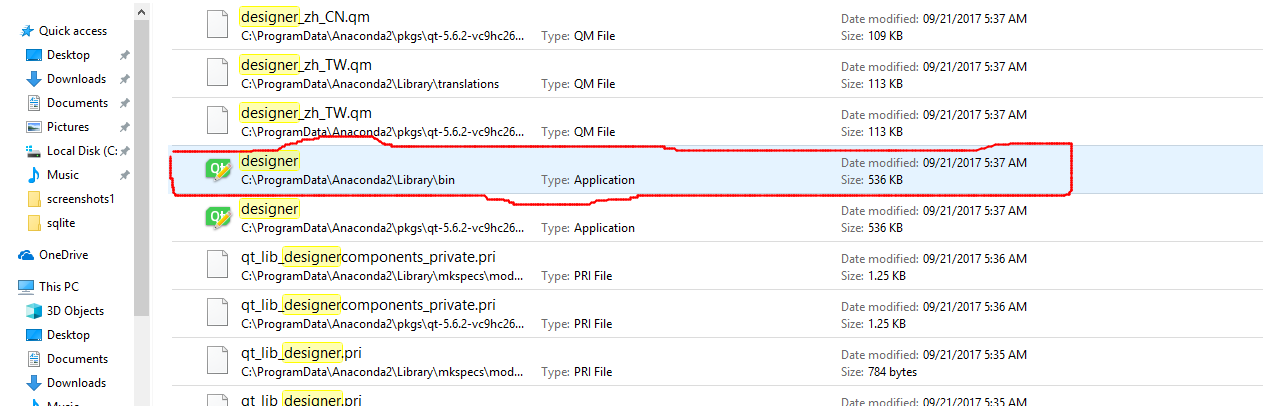
**7.f) Give the command .system dir/p as shown in the below figure. It should display the current directory structure.**

****

**7. g) You can give the command .cd .. to move one level up in the directory. Like wise, .cd ../.. can move you two levels up.**

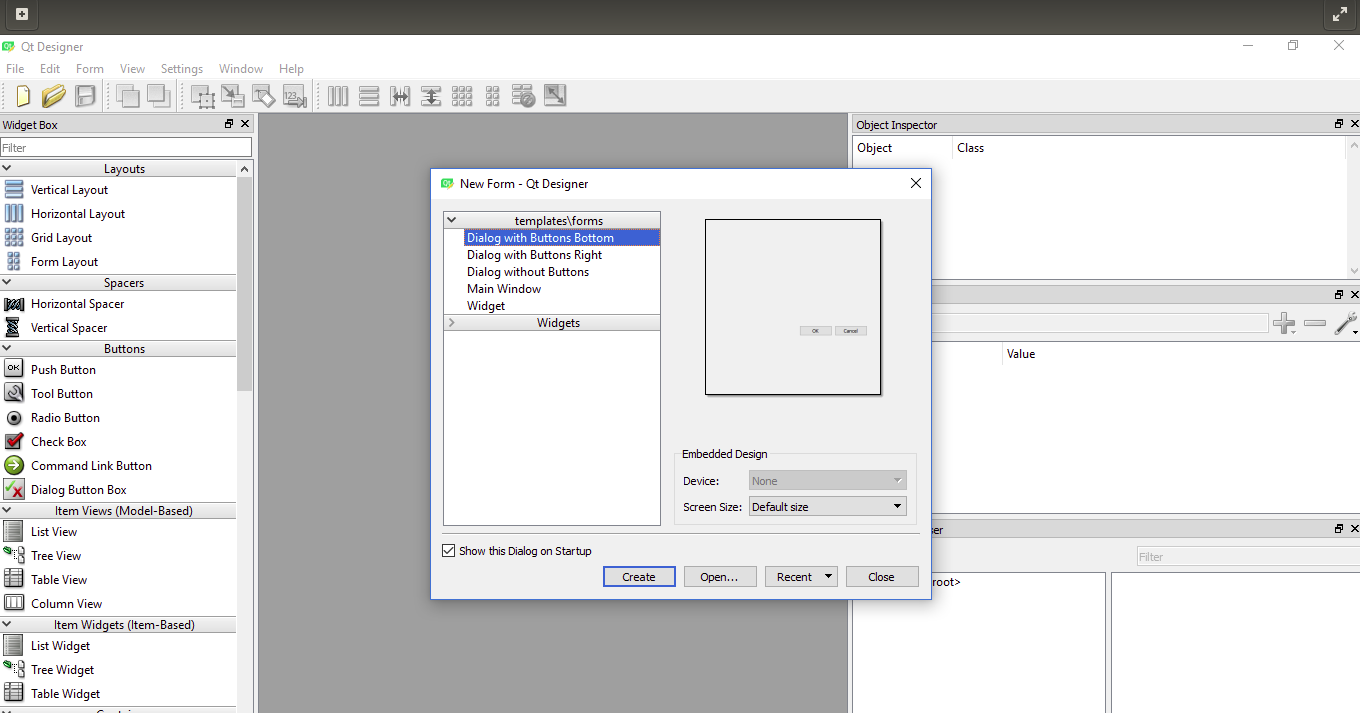
**8). Open windows explorer, and search for ‘Designer’. This search will take a few minutes. Find out whether you have Anaconda’s designer, as shown in below figures.**

****

****

**8. a) If you have the designer, as shown in the above figure. Remember it’s path(Take a screen shot on your mobile), as you have to use this designer many times in your project.**

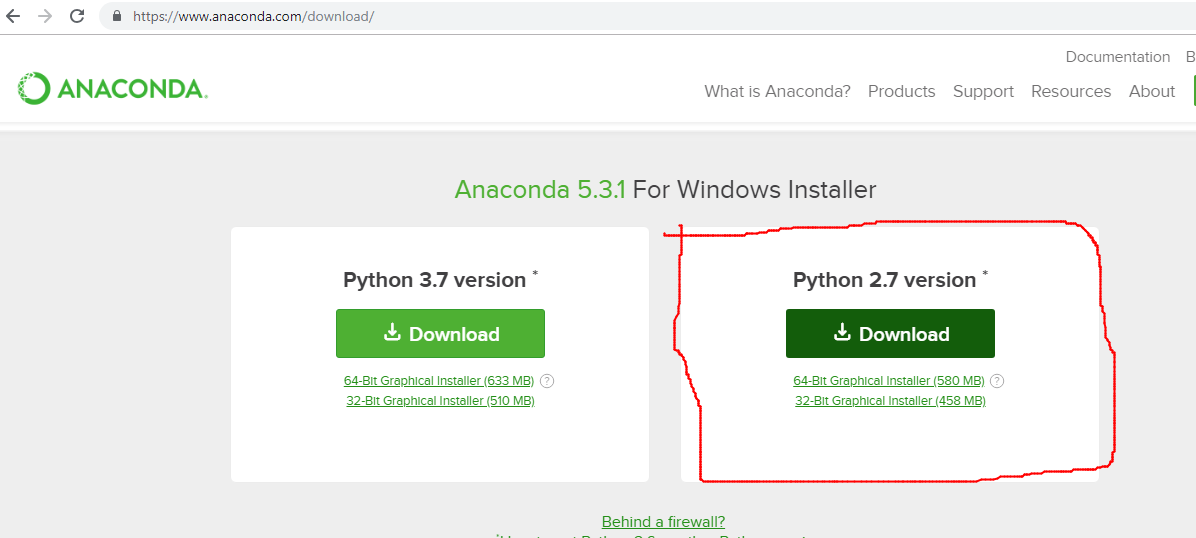
**8.b). Click on the above designer. You should get the following.**

****

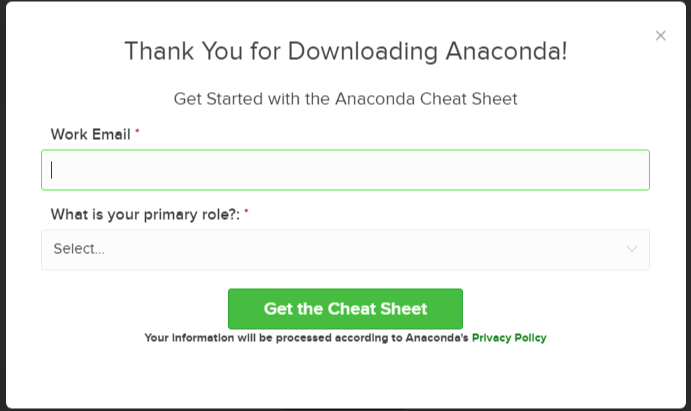
**8.c) If you are able to get the above, then you can ignore the remaining steps in this document.**

**8.d) If you are not able to get the above, then the Anacondas Designer is not successfully installed on your system. If you have Anaconda, on your system, remove it, and reinstall it as explained below.**

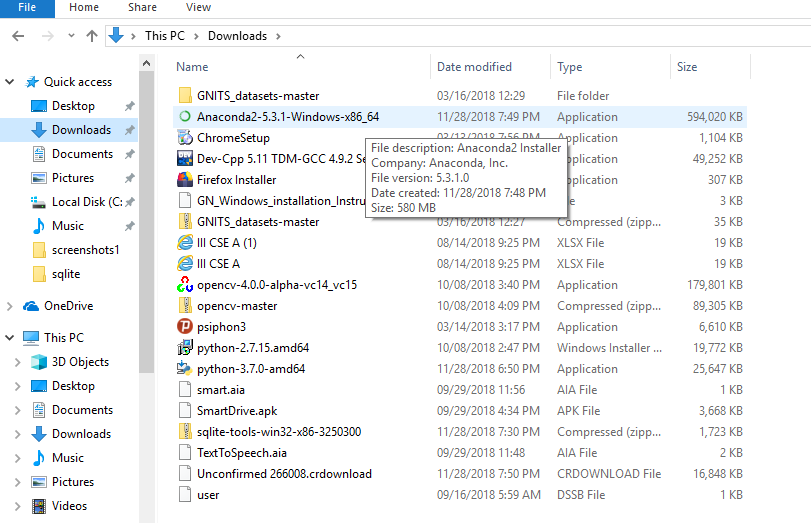
**8.e) install Anaconda by downloading it from:** [**https://www.anaconda.com/download/**](https://www.anaconda.com/download/)**. Click the Download button, under Python 2.7 version as shown below.**

****

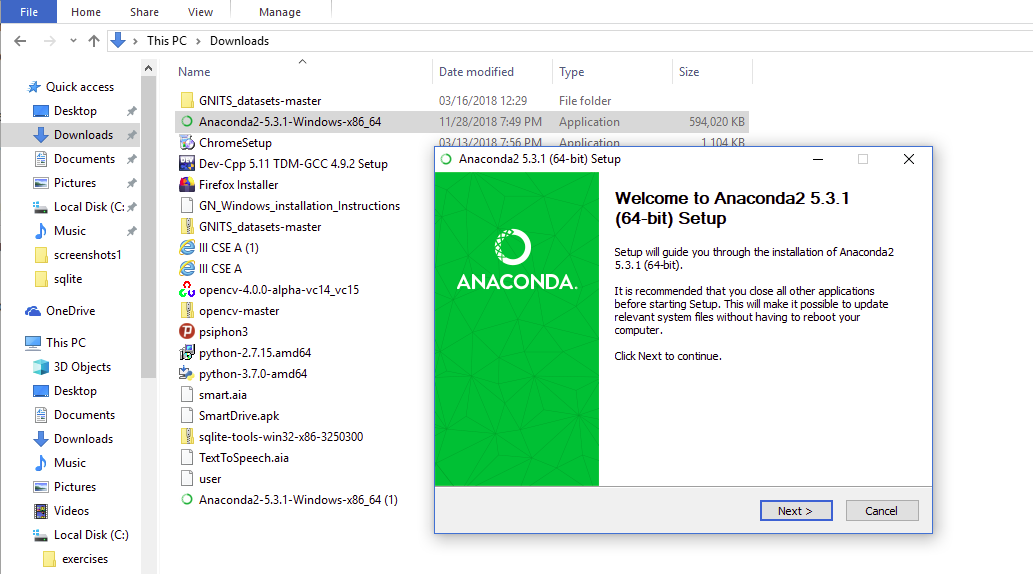
**8.f) If you get any cheat sheet, as shown below, fill in the details with your e-mail ID etc.**

****

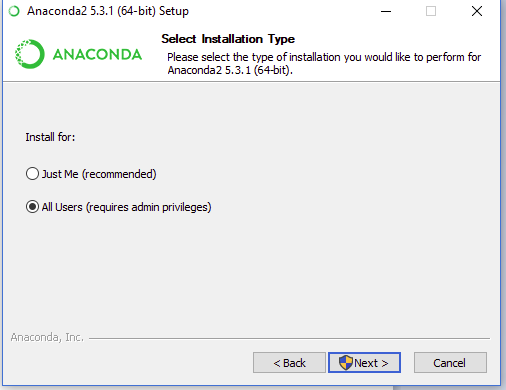
**8.g) Anaconda application will be downloaded into your Downloads, as shown below.**

****

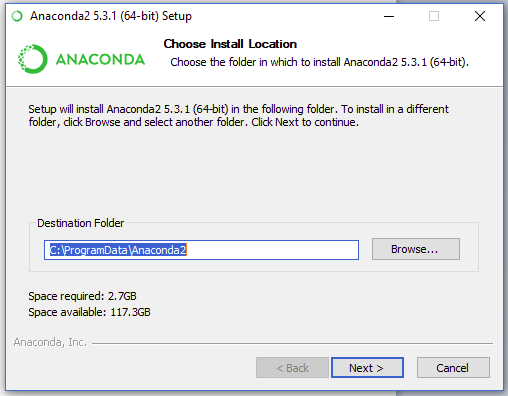
**8.h) By clicking on this Anaconda application, it will start installing as shown in the below figure.**

****

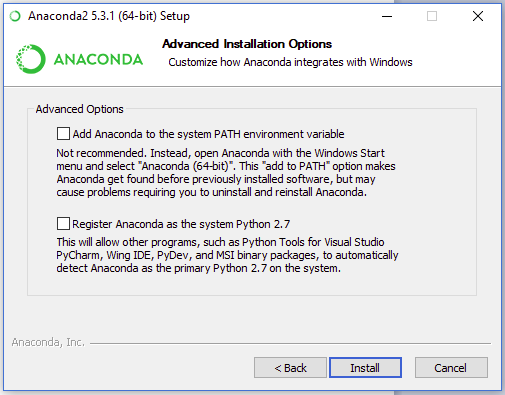
**8.i) Proceed with Anaconda’s installation by clicking on the ‘Next’ and selecting ‘All users’ as shown below.**

****

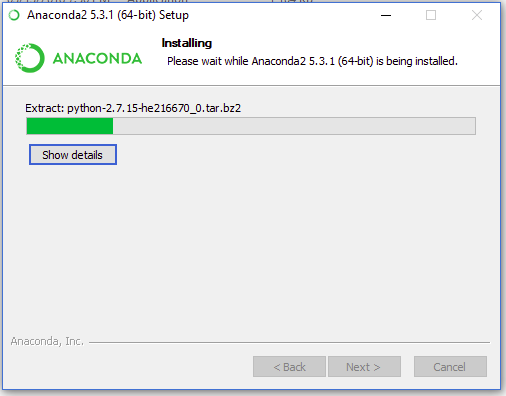
**8.j) Proceed with the installation, by choosing a folder for installation, as shown below.**

****

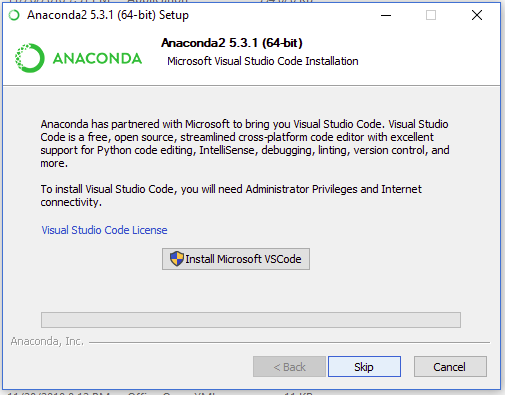
**8.k) You need not choose any Advanced options, and, leave them blank, as shown below.**

****

**8.l) Following screen appears, while installation being progressed.**

****

**8.m) You can choose ‘skip’ option, when the following screen is displayed.**

****

**8.n) Anaconda’s installation can be finished, by clicking the ‘Finish’ Button.**

**8.o) After Installation is finished, open windows explorer, and, repeat steps, 8, 8.a,8.b & 8.c**

**4.1.2 Languages**

**PYTHON**

Python was conceived in the late 1980s, and its implementation began in December 1989 by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL) capable of exception handling and interfacing.

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. It provides constructs that enable clear programming on both small and large scales. 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 interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations.

**FEATURES AND PHILOSOPHY**

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by meta programming and meta objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing, and a combination of reference counting and a cycle detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

**4.2 Sample Code**

**4.2.21 Python Code(To be filled in after coding)**

**idplf1.py**

**disease1.py**

**ima\_match1.py**

**limage1.py**

**ltype1.py**

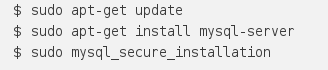
**moist1.py**

**ptype1.py**

**temprt1.py**

**4.2.3 Database Connectivity**

* MySQL is an open-source database management system, commonly installed as part of the popular LAMP (Linux, Apache, MySQL, PHP/Python/Perl) stack. It uses a relational database and SQL (Structured Query Language) to manage its data.
* The short version of the installation is simple: update your package index, install the mysql-server package, and then run the included security script



## Step 1 — Installing MySQL

On Ubuntu 16.04, only the latest version of MySQL is included in the APT package repository by default. At the time of writing, that's MySQL 5.7

To install it, simply update the package index on your server and install the default package with apt-get.

You'll be prompted to create a root password during the installation. Choose a secure one and make sure you remember it, because you'll need it later. Next, we'll finish configuring MySQL.

## Step 2 — Configuring MySQL

For fresh installations, you'll want to run the included security script. This changes some of the less secure default options for things like remote root logins and sample users. On older versions of MySQL, you needed to initialize the data directory manually as well, but this is done automatically now.

Run the security script.

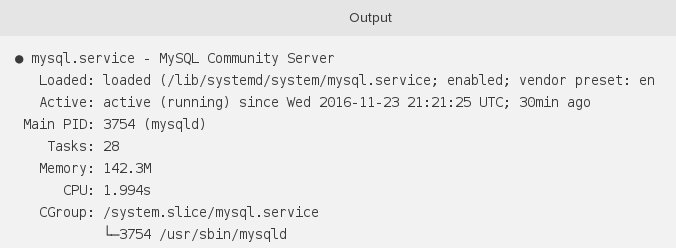


This will prompt you for the root password you created in Step 1. You can press Y and then ENTER to accept the defaults for all the subsequent questions, with the exception of the one that asks if you'd like to change the root password. You just set it in Step 1, so you don't have to change it now. For a more detailed walkt hrough of these options, you can see this step of the LAMP installation tutorial.

## Step 3 — Testing MySQL

Regardless of how you installed it, MySQL should have started running automatically. To test this, check its status.



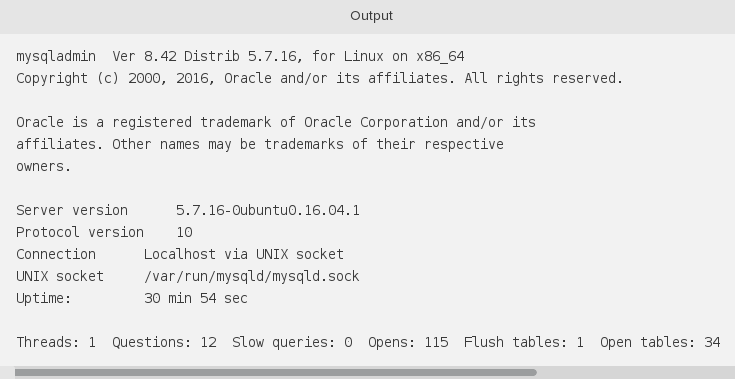
You'll see output similar to the following:

If MySQL isn't running, you can start it with sudo systemctl mysql start.

For an additional check, you can try connecting to the database using the mysqladmin tool, which is a client that lets you run administrative commands. For example, this command says to connect to MySQL as root (-u root), prompt for a password (-p), and return the version.



This means MySQL is up and running.



**CHAPTER 5**

**TESTING**

**5.1 Software Testing**

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words, software testing is a verification and validation process.

**Verification**

Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to.

**Validation**

Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements.

**Basics of software testing**

There are two basics of software testing: Black box testing and white box testing.

**Black box Testing**

Black box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing.

**White box Testing**

White box testing is a testing technique that takes into account the internal mechanism of a system. It is also called structural testing and glass box testing.

Black box testing is often used for validation and white box testing is often used for verification.

**5.1.1 Types of testing**

There are many types of testing like

 Unit Testing  Integration Testing  Functional Testing  System Testing  Regression Testing etc.

**Unit Testing**

Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.

**Integration Testing**

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing.

**Functional Testing**

Functional testing is the testing to ensure that the specified functionality required in the system requirements works. It falls under the class of black box testing.

**System Testing**

System testing is the testing to ensure that by putting the software in different environments (e.g., Operating Systems) it still works. System testing is done with full system implementation and environment. It falls under the class of black box testing.

**Regression Testing**

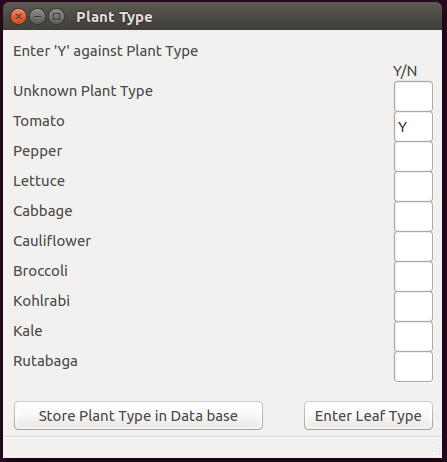
Regression testing is the testing after modification of a system, component, or a group of related units to ensure that the modification is working correctly and is not damaging or imposing other modules to produce unexpected results. It falls under the class of black box testing.

**CHAPTER 6**

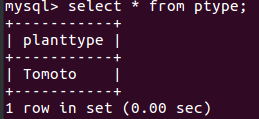
**System Testing Results**

The project is thoroughly tested by testing the each and every text box and push buttons of the GUI Screens, and verifying the corresponding results in the Data Base.

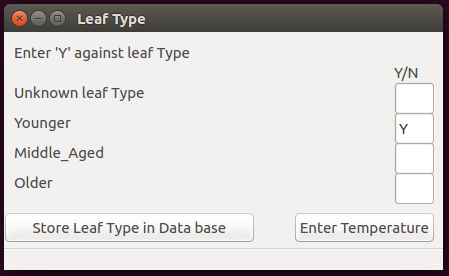
Following is the Plant type screen along with Data.



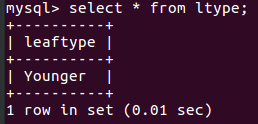
Following screen shot of the mysql db confirms that the above data is stored in the Database.



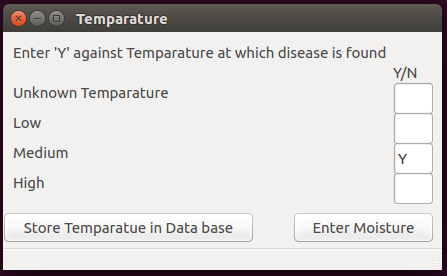
Following is the leaf type screen along with Data.



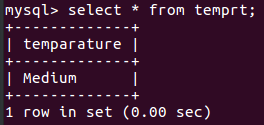
Following screen shot of the mysql db confirms that the above data is stored in the Database.



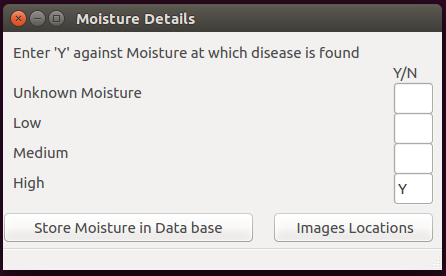
Following is the temperature screen along with Data.



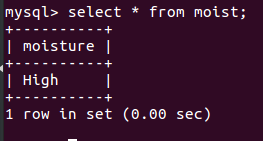
Following screen shot of the mysql db confirms that the above data is stored in the Database.



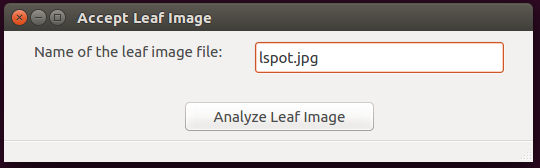
Following is the Moisture type screen along with Data.



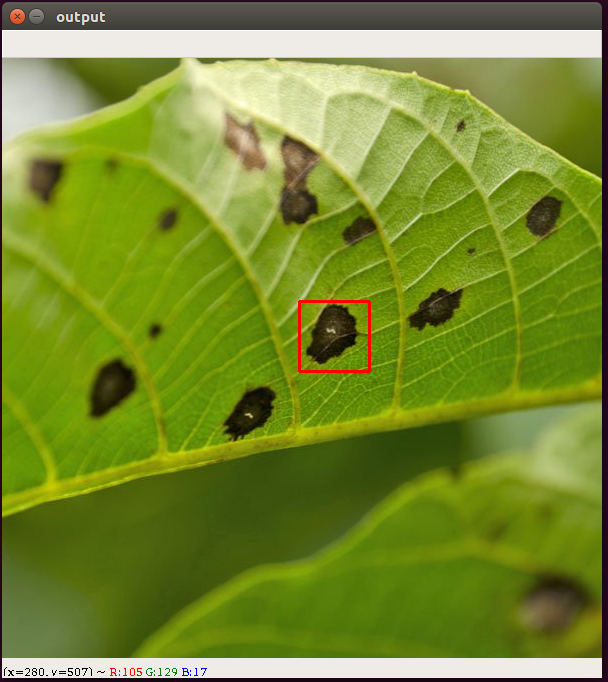
Following screen shot of the mysql db confirms that the above data is stored in the Database.



The leaf image is provided as input to the system, as shown in the following figure.



The ‘Analyze Leaf image’ push button is used to call the python routine to analyze the image, and to pinpoint the disease affected area, as shown in the following figure.



**CHAPTER 7**

**CONCLUSION AND FUTURE SCOPE**

**CONCLUSION**

This project entitled **“Detection of Diseases in Blueberry Leaves using Machine Learning.”** has presented an approach to identify the disease of a plant by processing the image of it’s effected leaf. This project is very useful to the farmers, as they need not go for the plant specialist consultation towards the identification of the disease. This project is also useful to the plant specialists, as they need not remember the details about each and every plant disease. This project finally leads to the enhancement average plant life.

**FUTURE SCOPE**

This project can be enhanced further by implementing the measures to be taken for the disease effected plant.

**CHAPTER 8**

**BIBLIOGRAPHY**

* 1. [Cecilia Sullca, Carlos Molina, Carlos Rodríguez, and Thais Fernández: Diseases Detection in Blueberry Leaves using Computer](https://www.planetnatural.com/pest-problem-solver/plant-disease/)

Vision and Machine Learning Techniques,International Journal of Machine Learning and Computing

* 1. [https://pdfs.semanticscholar.org/433d/4e68069add079ade95a0eedd87c284bae746.pdf](https://www.planetnatural.com/pest-problem-solver/plant-disease/)
  2. https://www.python.org/
  3. https://github.com/baoboa/pyqt5/blob/master/pyuic/uic/pyuic.py
  4. https://www.numpy.org/
  5. https://riverbankcomputing.com/software/pyqt/intro
  6. <https://www.ubuntu.com/>
  7. <https://www.planetnatural.com/pest-problem-solver/plant-disease/>