----College Emblem----

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the project entitled **“Activity Analysis for Diabetic Patients”** has been submitted by **----,----,----,----,** to the partial fulfillment of the requirements for the award of degree of –-------- 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 **"Activity Analysis for Diabetic Patients"** 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 –-------- 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)

CHAPTER 1

INTRODUCTION

It's good for diabetic patients to continuously do some activity, so that enough calories are consumed. But, Diabetic patients have a tendency to do some activity for some time and claim a lot about it. This project aims at identifying and analyzing the activities of such patients. The sensors data set, provided by California University, is used in this project.

The GUI screens in this project are created by a python tool called PyQt. PyQt is a Python binding of the cross-platform GUI toolkit Qt, implemented as a Python plug-in. PyQt is developed by the British firm Riverbank Computing. PyQt supports Microsoft Windows as well as various flavors of UNIX, including Linux and MacOS.

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.

The algorithm of the system involves in identifying and analyzing the activities of diabetic patients, and comparing them one by one using an iterative structure.

This project has three modules. The MSFE (Multi sensor Front end) module is used to create the Graphical User Interfaces. The MSPD (Multi sensor & Patient Details) data is used to store the patient details, as well as multi sensor data into the Data base. The AIA (Activity Identification & Analysis) module is used to identify and analyze the activities.

The project uses report lab tool generate the output report, in .pdf form. This Report contains the major reasons for absenteeism, along with the corresponding statistics. Report lab is a package for generating high-quality dynamic PDF documents/charts in real-time.

ADVANTAGES

Project Advantages:

The project is very useful to the diabetic patients, as they know the calories burned by them for their activities.

The project is also useful to the diabetic doctors in advising their patients regarding the food to be taken, and the calories to be burnt.

This project finally leads to the improvement of diabetic patients life.

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.

PROBLEM DEFINITION

Vision:

- The project aims at developing a tool for the Activity Analysis for Diabetic Patients, with all the above mentioned advantages.

Mission:

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

Literature Survey

Diabetes:

Diabetes mellitus, commonly known as diabetes, is a metabolic disease that causes high blood sugar. The hormone insulin moves sugar from the blood into your cells to be stored or used for energy. With diabetes, your body either doesn’t make enough insulin or can’t effectively use the insulin it does make.
Untreated high blood sugar from diabetes can damage your nerves, eyes, kidneys, and other organs.

There are a few different types of diabetes:

Type 1 diabetes is an autoimmune disease. The immune system attacks and destroys cells in the pancreas, where insulin is made. It’s unclear what causes this attack. About 10 percent of people with diabetes have this type.

Type 2 diabetes occurs when your body becomes resistant to insulin, and sugar builds up in your blood.

Prediabetes occurs when your blood sugar is higher than normal, but it’s not high enough for a diagnosis of type 2 diabetes

.

Gestational diabetes is high blood sugar during pregnancy. Insulin-blocking hormones produced by the placenta cause this type of diabetes.

A rare condition called diabetes insipidus is not related to diabetes mellitus, although it has a similar name. It’s a different condition in which your kidneys remove too much fluid from your body.

Each type of diabetes has unique symptoms, causes, and treatments. Learn more about how these types differ from one another.

Symptoms of diabetes

Diabetes symptoms are caused by rising blood sugar.

General symptoms

The general symptoms of diabetes include:

- increased hunger
- increased thirst
- weight loss
- frequent urination
- blurry vision
- extreme fatigue
- sores that don’t heal

Symptoms in men

In addition to the general symptoms of diabetes, men with diabetes may have a decreased sex drive, erectile dysfunction (ED), and poor muscle strength.

Symptoms in women

Women with diabetes can also have symptoms such as urinary tract infections, yeast infections, and dry, itchy skin.

Type 1 diabetes

Symptoms of type 1 diabetes can include:

- extreme hunger
- increased thirst
- unintentional weight loss
- frequent urination
- blurry vision
- tiredness

It may also result in mood changes.

Type 2 diabetes

Symptoms of type 2 diabetes can include:

- increased hunger
- increased thirst
- increased urination
- blurry vision
- tiredness
- sores that are slow to heal

It may also cause recurring infections. This is because elevated glucose levels make it harder for the body to heal.

Gestational diabetes

Most women with gestational diabetes don’t have any symptoms. The condition is often detected during a routine blood sugar test or oral glucose tolerance test that is usually performed between the 24th and 28th weeks of gestation.

In rare cases, a woman with gestational diabetes will also experience increased thirst or urination.

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

System Description

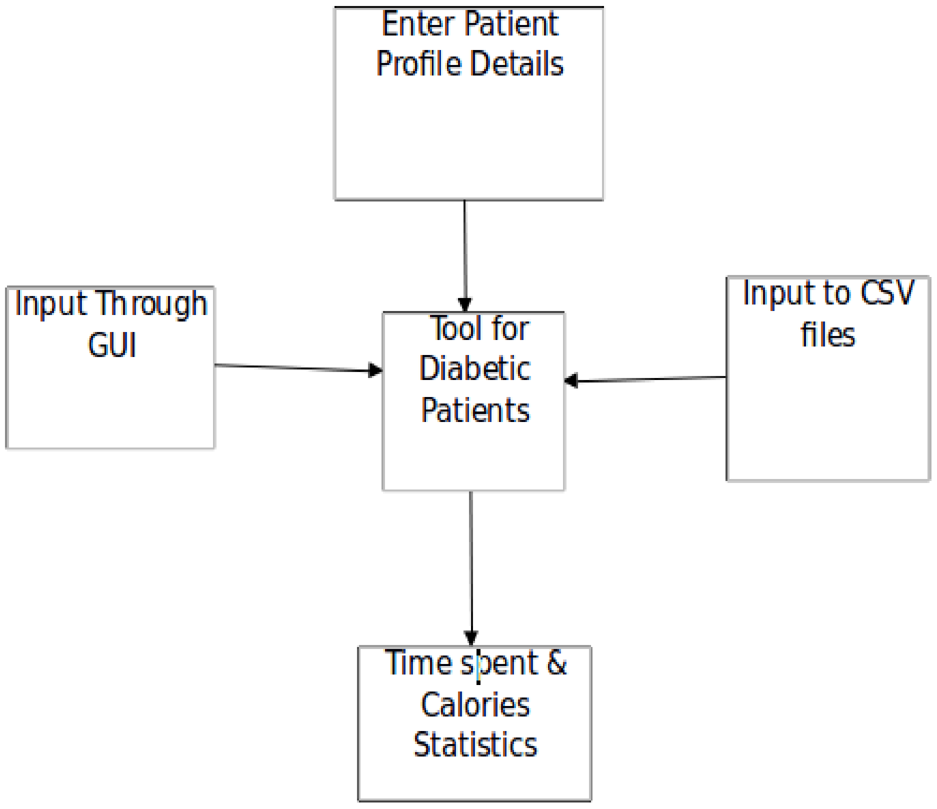
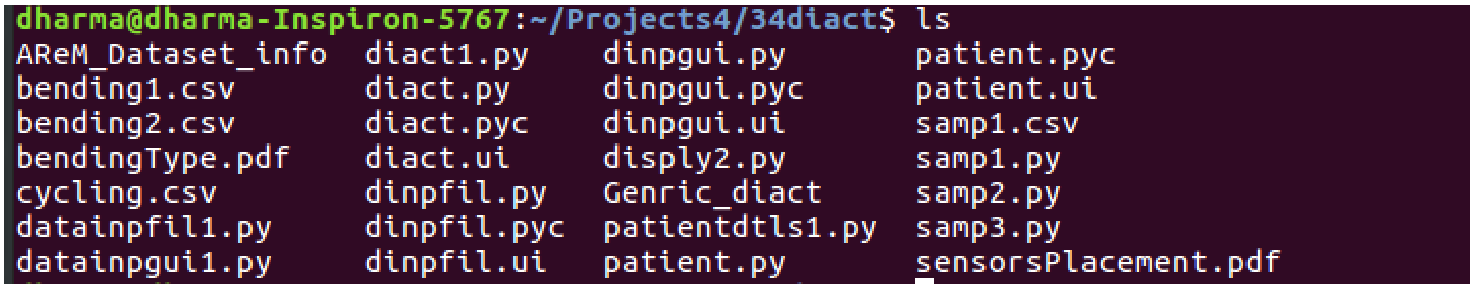

Enter Patient Profile Details
Input Through GUI
Tool for Diabetic Patients
Input to CSV files
Time spent & Calories Statistics

Fig 1: Data Flow Diagram

Details like Patient details, Time segments & coordinates are to be provided as Input to the system, using the corresponding user interfaces/files. The following section describe the screen shots of these user interfaces.

<u>Screen shots and descriptions:</u>



```
dharma@dharma-Inspiron-5767:~/Projects4/34diact$ ls
AReM_Dataset_info  diact1.py    dinpgui.py       patient.pyc
bending1.csv       diact.py     dinpgui.pyc      patient.ui
bending2.csv       diact.pyc    dinpgui.ui       samp1.csv
bendingType.pdf    diact.ui     disply2.py       samp1.py
cycling.csv        dinpfil.py   Genric_diact     samp2.py
datainpfil1.py     dinpfil.pyc  patientdtls1.py  samp3.py
datainpgui1.py     dinpfil.ui   patient.py       sensorsPlacement.pdf
```

ic1: Screen shot showing the files created during the project

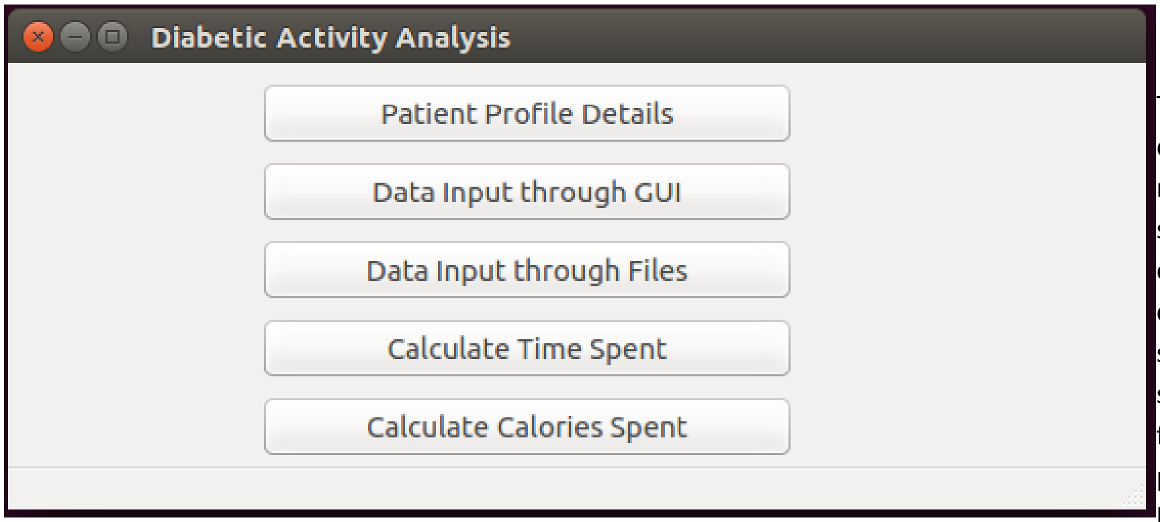
The above screen shot shows different files created during this projects. There are four different types of files: (1) .csv files (2) .ui files (3).py files and (4).txt files

.csv files are the comma separated value files, containing the data sets needed for 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.

diact1.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, patientdtls1.py program is instantiated resulting in a screen, by using which the user can enter the details of patients.

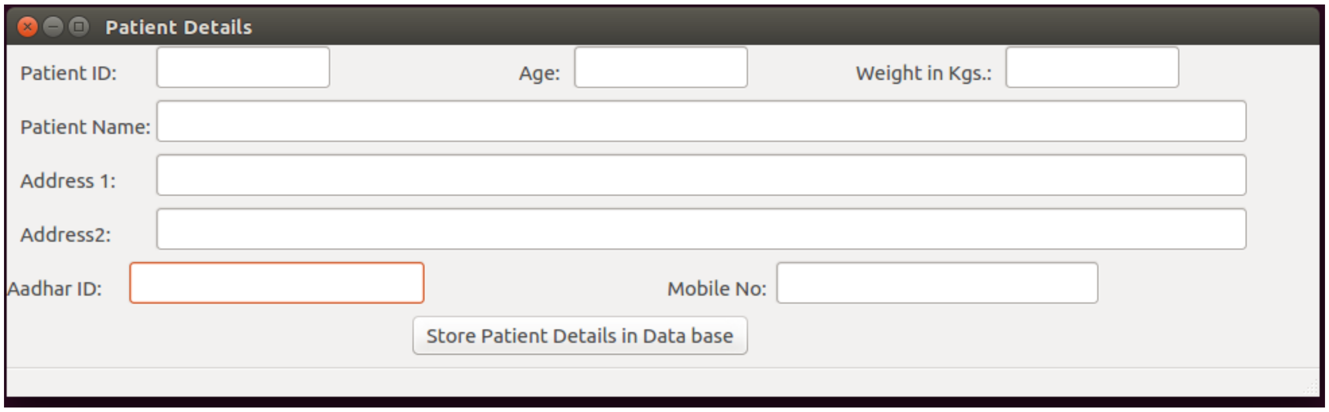
The second button leads to the invoking of datainpgui1.py program, which results in a screen, where the user can enter the data of diabetic activities.

Upon clicking the third button, datainpfil1.py program is instantiated resulting in a screen, by using which the user can enter the input through data files.

Upon clicking the fourth button, samp2.py program is instantiated which calculates the time spent in the activities.

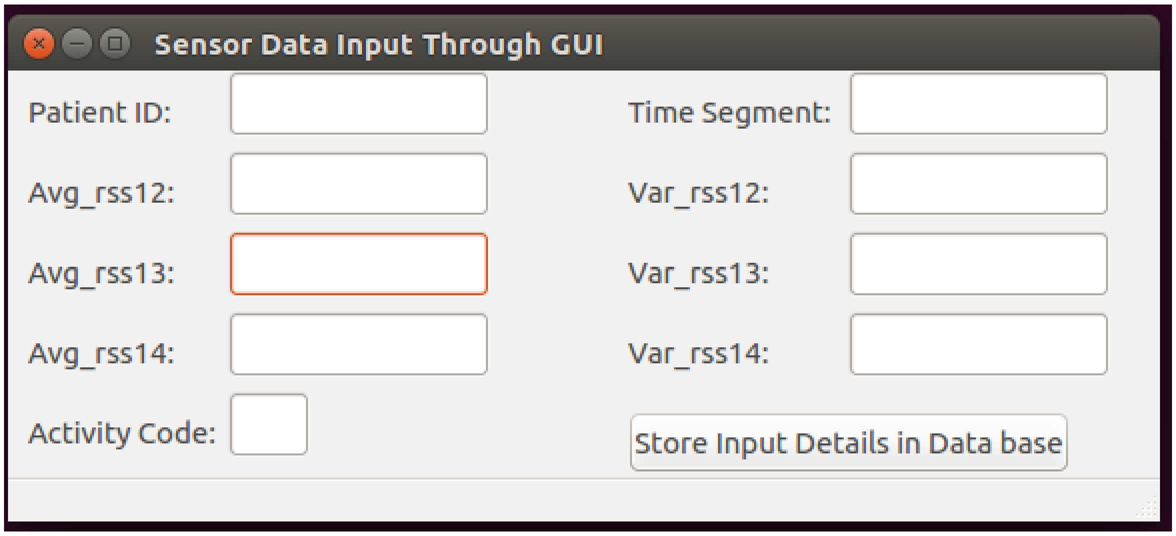
Upon clicking the fourth button, samp21py program is instantiated which calculates the Calories spent in the activities

patientdtls1.py program leads to the following screen.

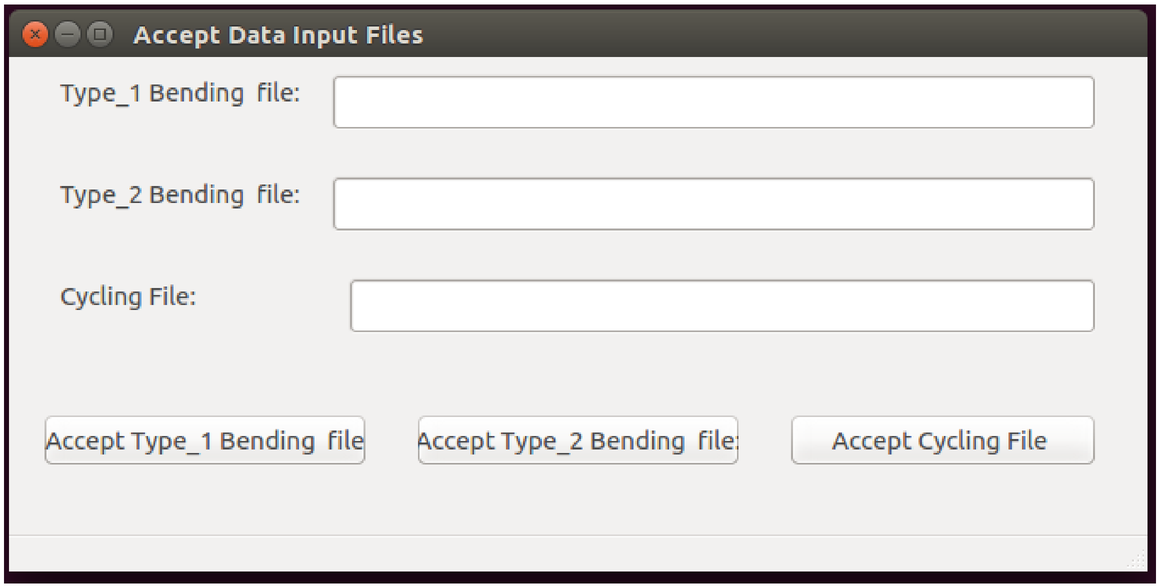


The user can enter the type of the diabetic patients, and store them in the DB, by using the above screen.

The user can enter the activity sensor readings, by using the following screen.



The user provide the data sets input through files, by using the following 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.

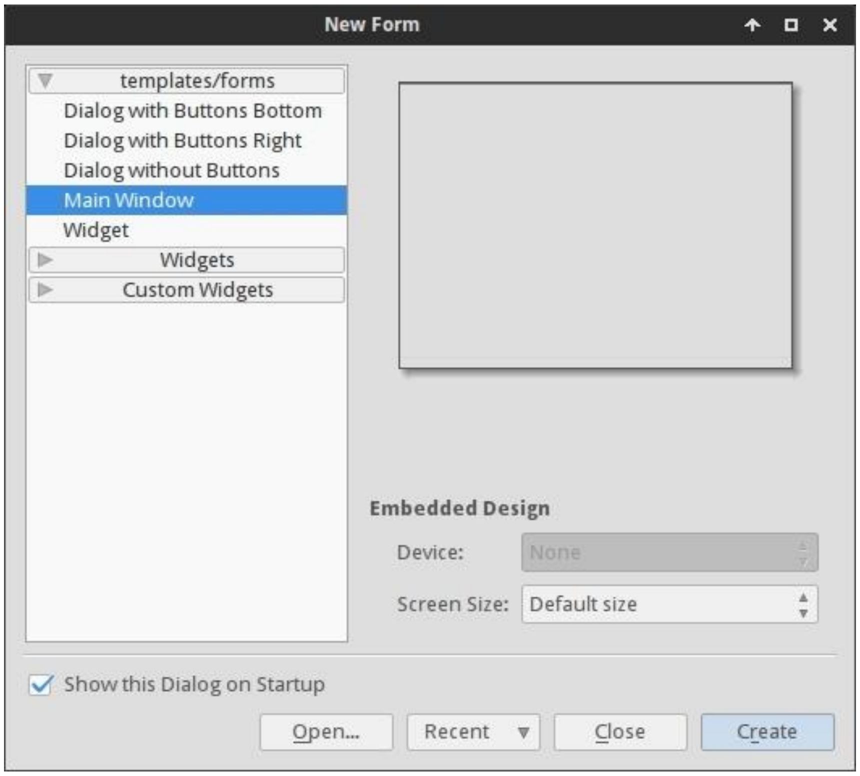
The easiest way to start application development with Qt is to download and install Qt 5. It contains Qt libraries, examples, documentation, and the necessary development tools, such as the Qt Creator integrated development environment (IDE).

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.

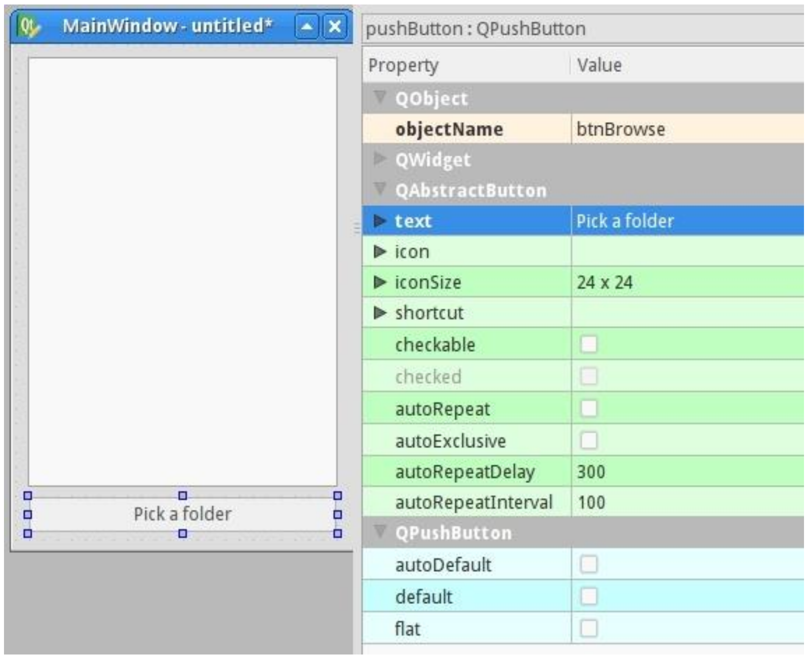
Python Qt Designer

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

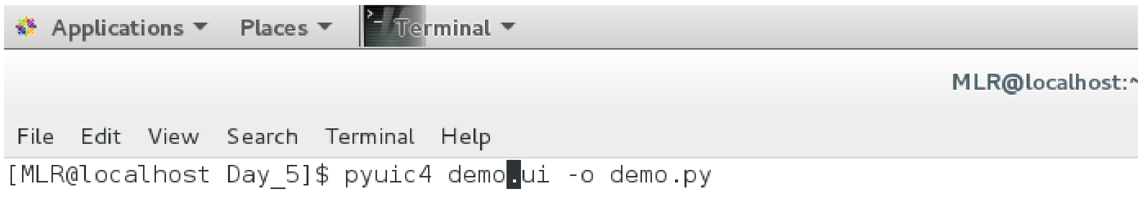


You can then drag and drop required widgets from the widget box on the left pane. You can also assign value to properties of widget laid on the form.



Property	Value
QObject	
objectName	btnBrowse
QWidget	
QAbstractButton	
text	Pick a folder
icon	
iconSize	24 x 24
shortcut	
checkable	☐
checked	☐
autoRepeat	☐
autoExclusive	☐
autoRepeatDelay	300
autoRepeatInterval	100
QPushButton	
autoDefault	☐
default	☐
flat	☐

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



```
[MLR@localhost Day_5]$ pyuic4 demo.ui -o demo.py
```

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.

QPushButton

3 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).

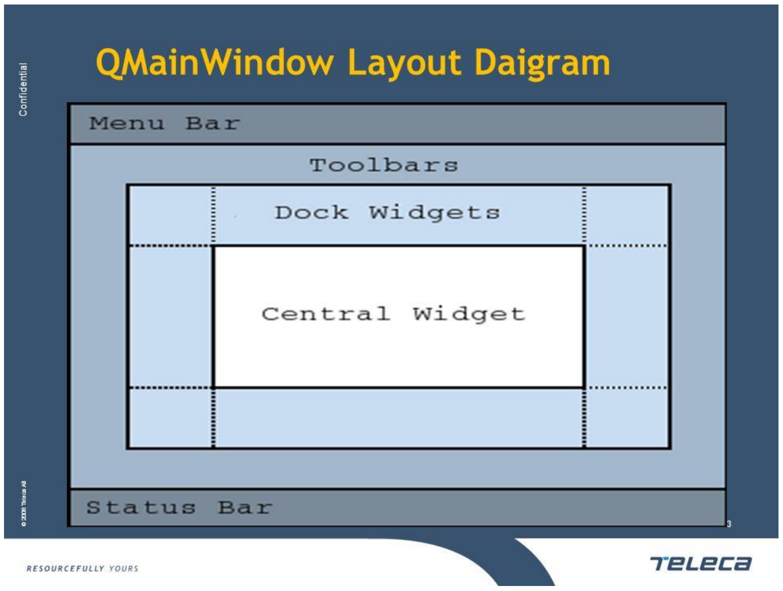
The default key bindings are described below. The line edit also provides a context menu (usually invoked by a right mouse click) that presents some of these editing options.

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 add Toolbar(). 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 = add Toolbar(tr("File"));

fileToolBar->addAction(newAct);
```

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()

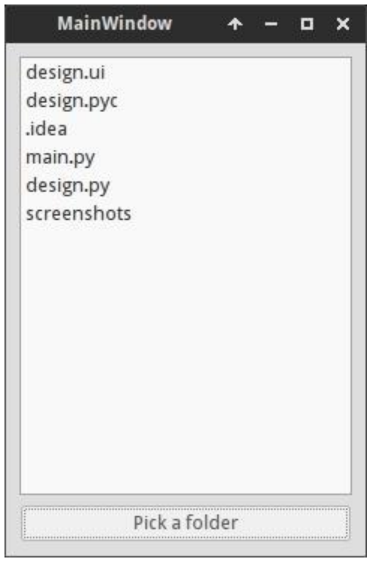
In the end our browse_folder function looks something like this:

```
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 designed 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
```



```
design.ui
design.pyc
.idea
main.py
design.py
screenshots
```

```
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.

UML DIAGRAMS

USE CASE DIAGRAM

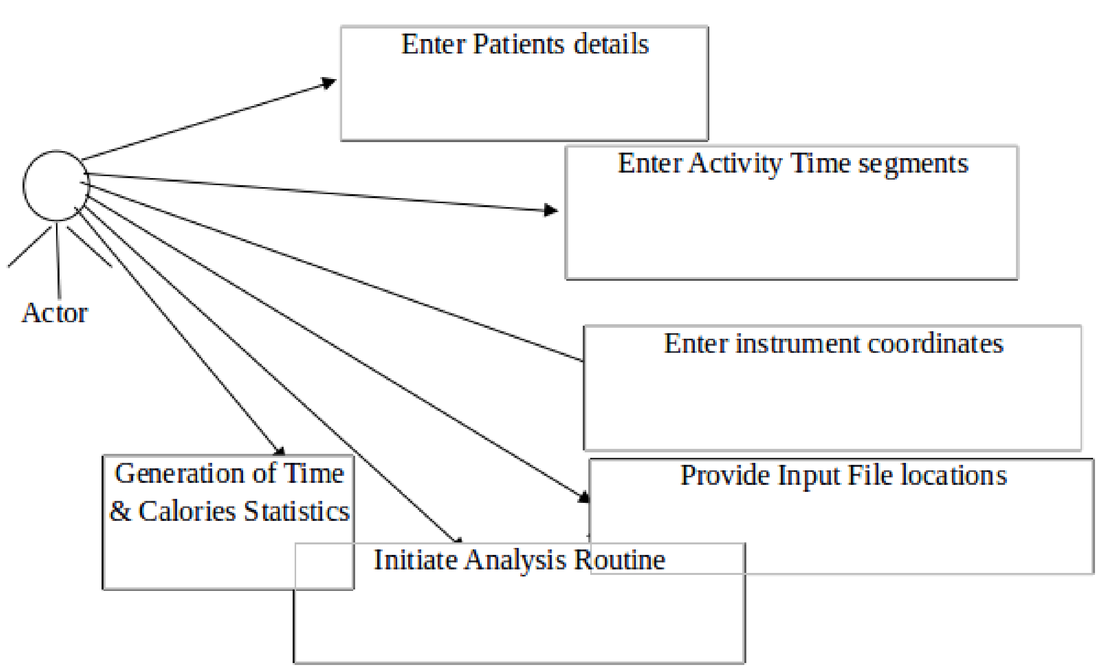

Enter Patients details
Enter Activity Time segments
Actor
Enter instrument coordinates
Generation of Time & Calories Statistics
Provide Input File locations
Initiate Analysis Routine

Fig 1: USE CASE DIAGRAM

Description of Use case diagram

Use case diagrams are usually referred to as behavior 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 Patients Details, Time & coordinates details and input file locations.

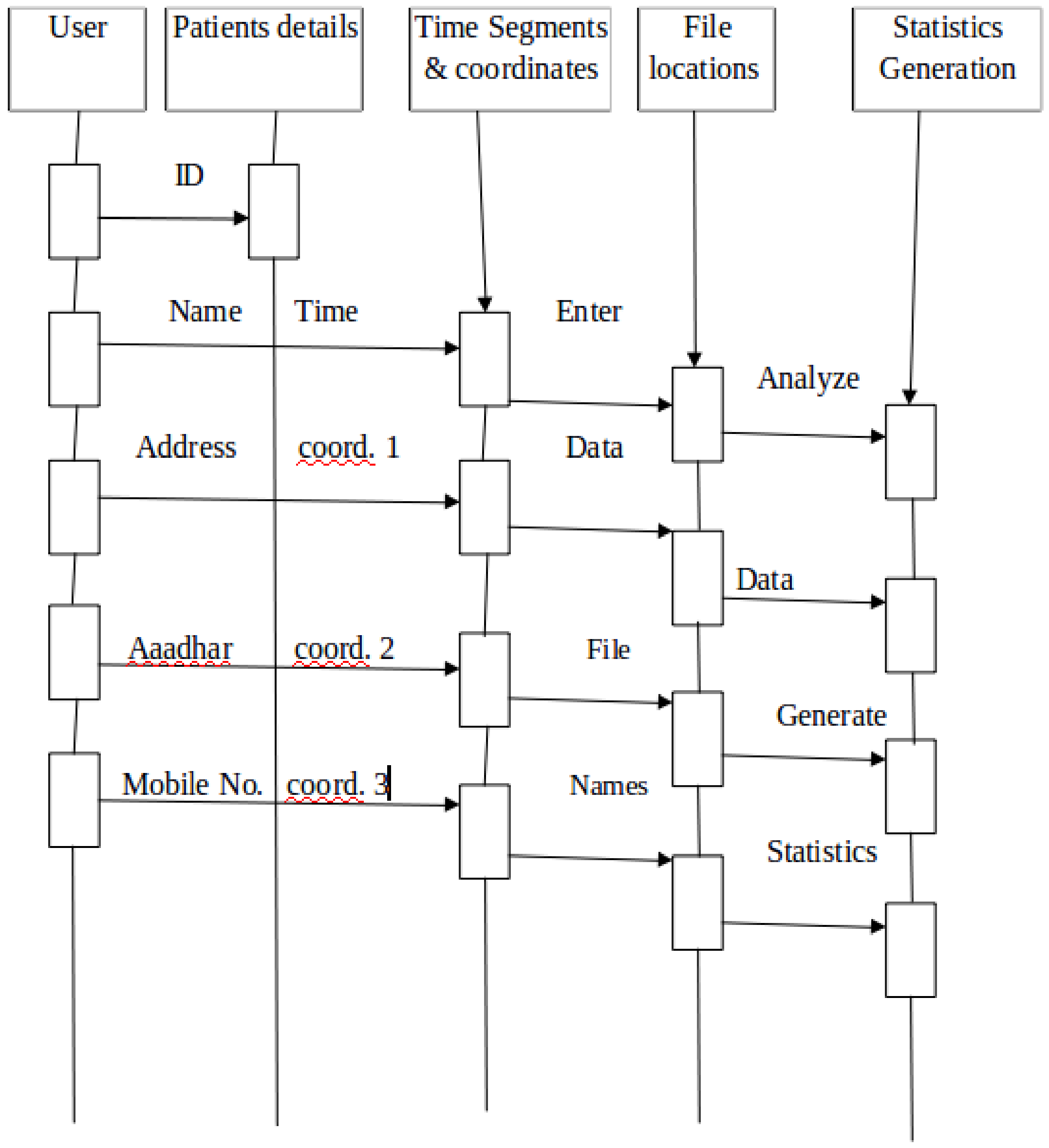
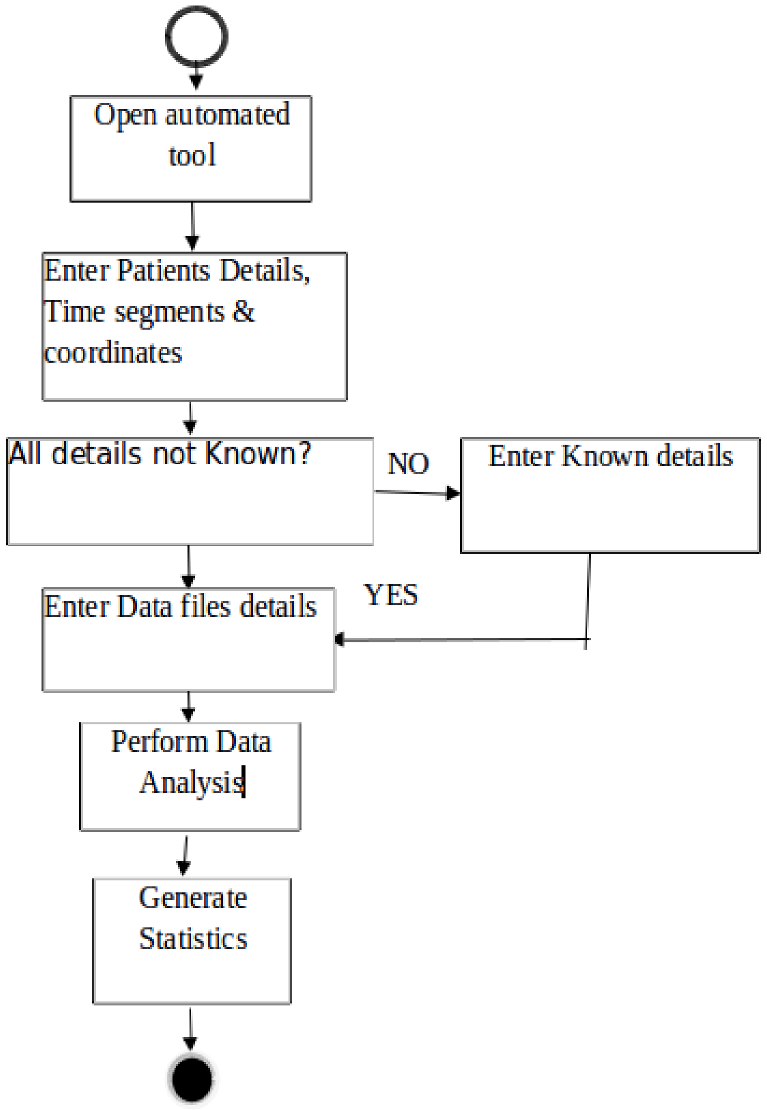


Fig : SEQUENCE DIAGRAM

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 Time segments & coordinates details as Input, Analyze and calculate the time & calories spent.


Open automated tool
Enter Patients Details, Time segments & coordinates
All details not Known?
NO
Enter Known details
YES
Enter Data files details
Perform Data Analysis
Generate Statistics

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 Automated 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 Patients Profile details, the customer needs to provide the time segments & coordinates details as input, and then click on the corresponding buttons to calculate the time & calories spent.

CHAPTER 4

IMPLEMENTATION

4.1 SOFTWARE USED

4.1.1 UBUNTU INSTALLATION

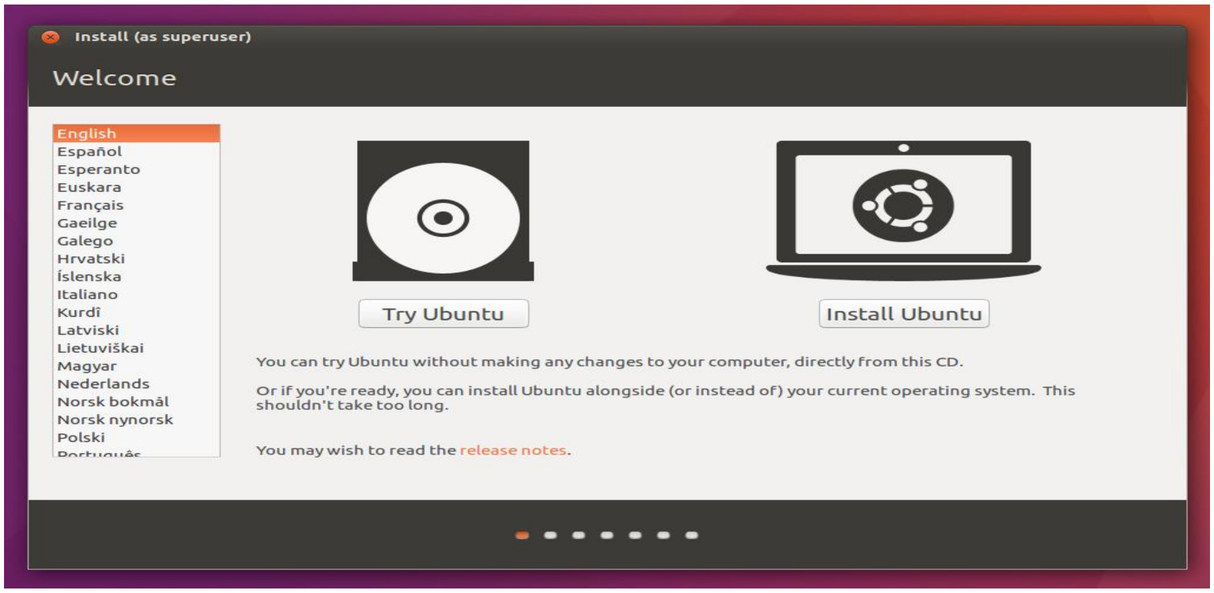
Using a DVD

It’s easy to install Ubuntu from a DVD. Here’s what you need to do:
Put the Ubuntu DVD into the DVD-drive
Restart your computer. You should see a welcome screen prompting you to choose your language and giving you the option to install Ubuntu or try it from the DVD.
If you don’t get this menu, read the booting from the DVD guide for more information.

Using a USB drive

Most new computers can boot from USB. You should see a welcome screen prompting you to choose your language and giving you the option to install Ubuntu or try it from the USB.

If your computer doesn’t automatically do so, you might need to press the **F12 key** to bring up the boot menu but be careful not to hold it down - that can cause an error message.



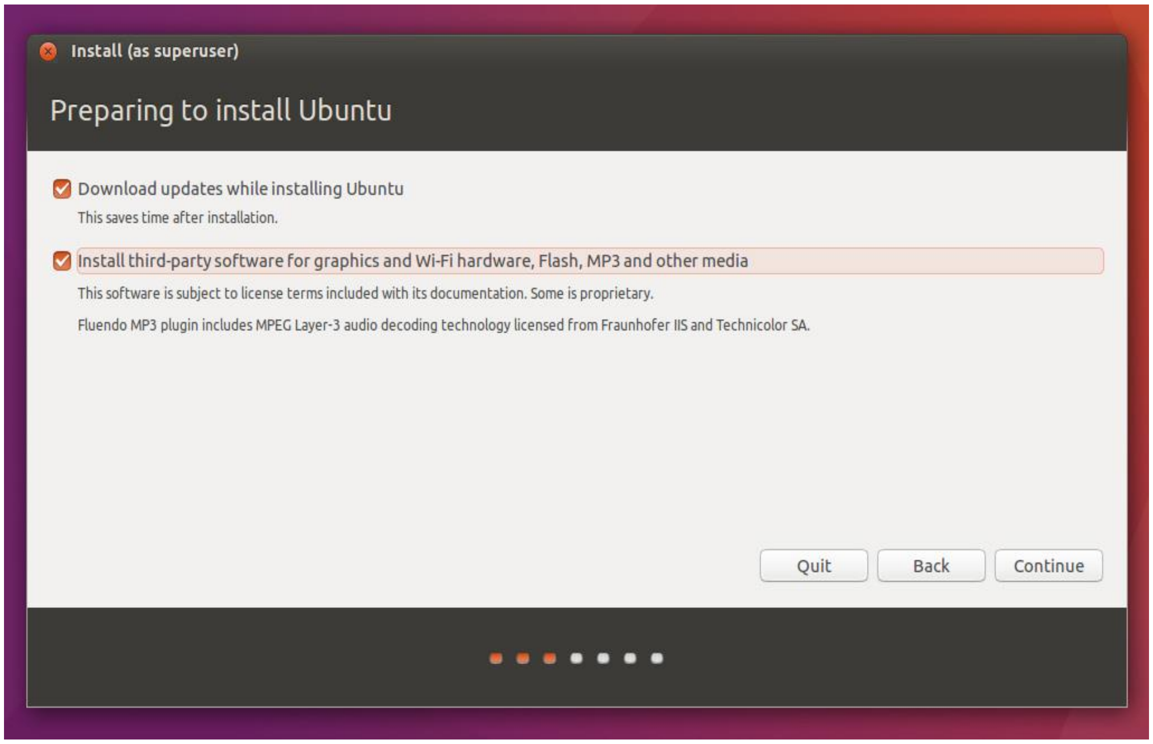
You can try Ubuntu without making any changes to your computer, directly from this CD.

Or if you're ready, you can install Ubuntu alongside (or instead of) your current operating system. This shouldn't take too long.

You may wish to read the release notes.

Prepare to install Ubuntu

- We recommend you plug your computer into a power source
- You should also make sure you have enough space on your computer to install Ubuntu
- We advise you to select Download updates while installing and Install this third-party software now.You should also stay connected to the internet so you can get the latest updates while you install Ubuntu If you are not connected to the internet, you will be asked to select a wireless network, if available. We advise you to connect during the installation so we can ensure your machine is up to date.


Install (as superuser)

Preparing to install Ubuntu

[✓] Download updates while installing Ubuntu

This saves time after installation.

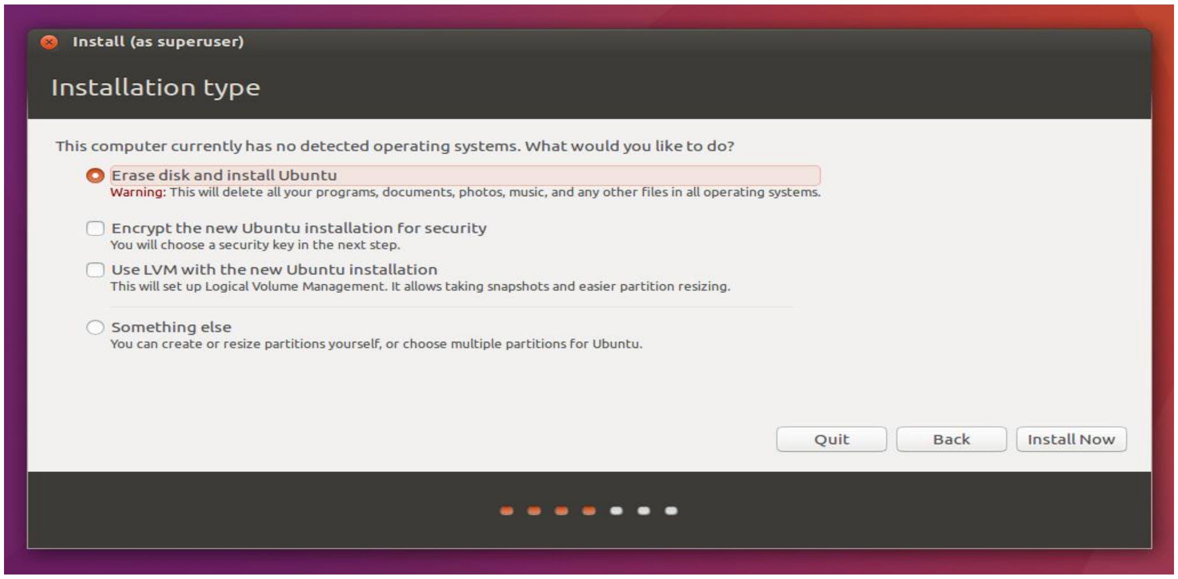
[✓] Install third-party software for graphics and Wi-Fi hardware, Flash, MP3 and other media

This software is subject to license terms included with its documentation. Some is proprietary.

Fluendo MP3 plugin includes MPEG Layer-3 audio decoding technology licensed from Fraunhofer IIS and Technicolor SA.

Allocate drive space

- Use the check boxes to choose whether you'd like to Install Ubuntu alongside another operating system, delete your existing operating system and replace it with Ubuntu, or — if you're an advanced user — choose the **'Something else'** option


Install (as superuser)

Installation type

This computer currently has no detected operating systems. What would you like to do?

[✓ Erase disk and install Ubuntu]
Warning: This will delete all your programs, documents, photos, music, and any other files in all operating systems.

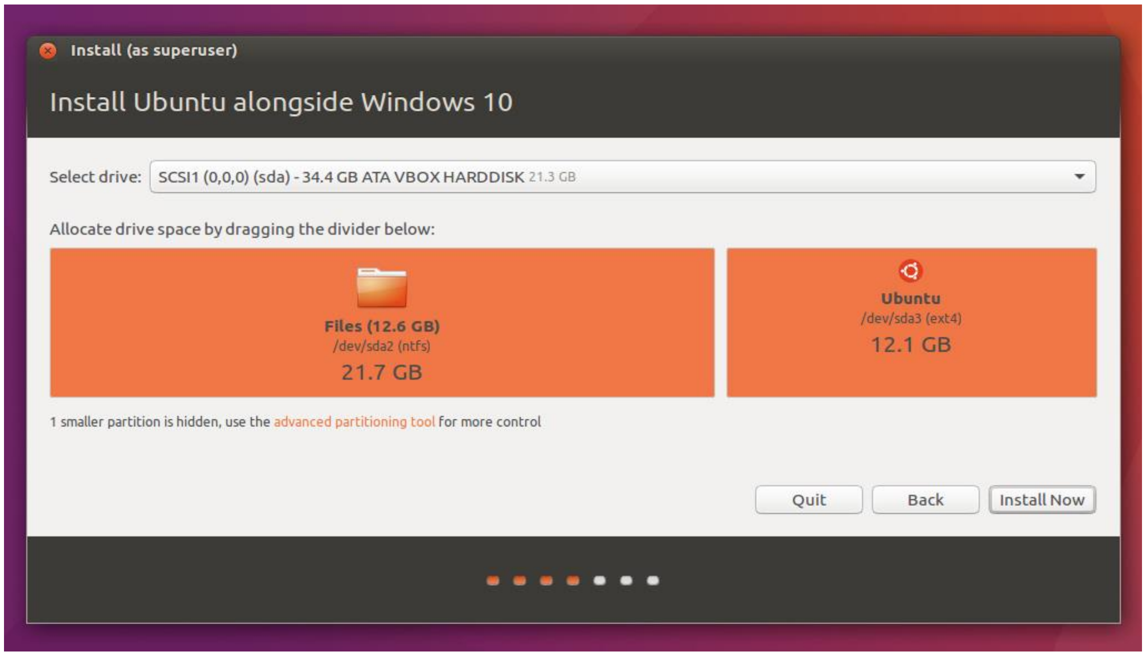
Encrypt the new Ubuntu installation for security
You will choose a security key in the next step.

Use LVM with the new Ubuntu installation
This will set up Logical Volume Management. It allows taking snapshots and easier partition resizing.

Something else
You can create or resize partitions yourself, or choose multiple partitions for Ubuntu.

Begin the installation

- Depending on your previous selections, you can now verify that you have chosen the way in which you would like to install Ubuntu. The installation process will begin when you click the Install Now button.
- Ubuntu needs about 4 GB to install, so add a



Install (as superuser)

Install Ubuntu alongside Windows 10

Select drive: SCSI1 (0,0,0) (sda) - 34.4 GB ATA VBOX HARDDISK 21.3 GB

Allocate drive space by dragging the divider below:

Files (12.6 GB) /dev/sda2 (ntfs) 21.7 GB

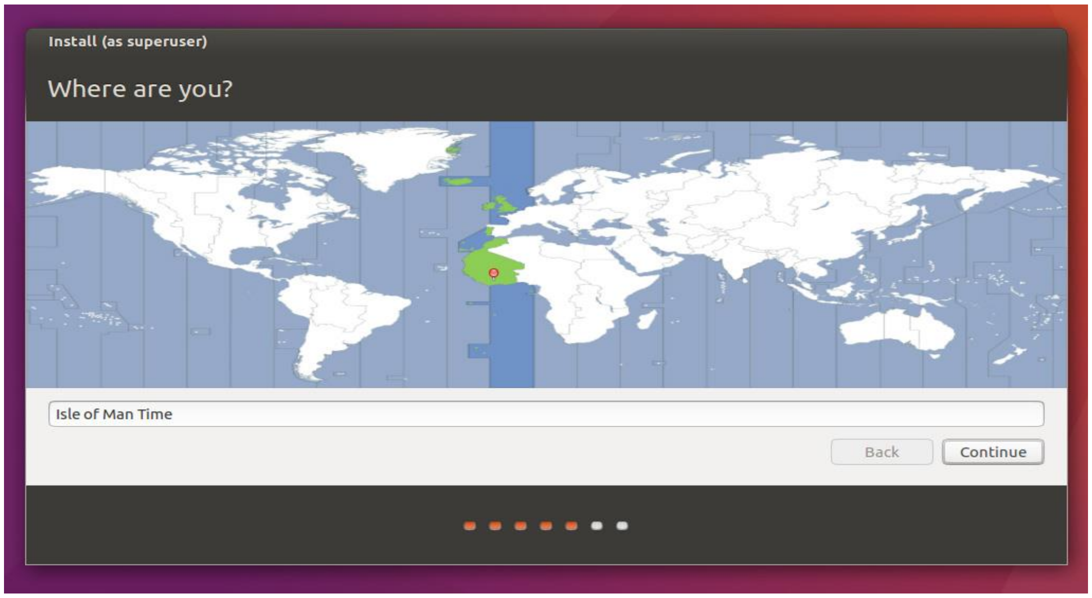
Ubuntu /dev/sda3 (ext4) 12.1 GB

1 smaller partition is hidden, use the advanced partitioning tool for more control

few extra GB to allow for your files.

Select your location

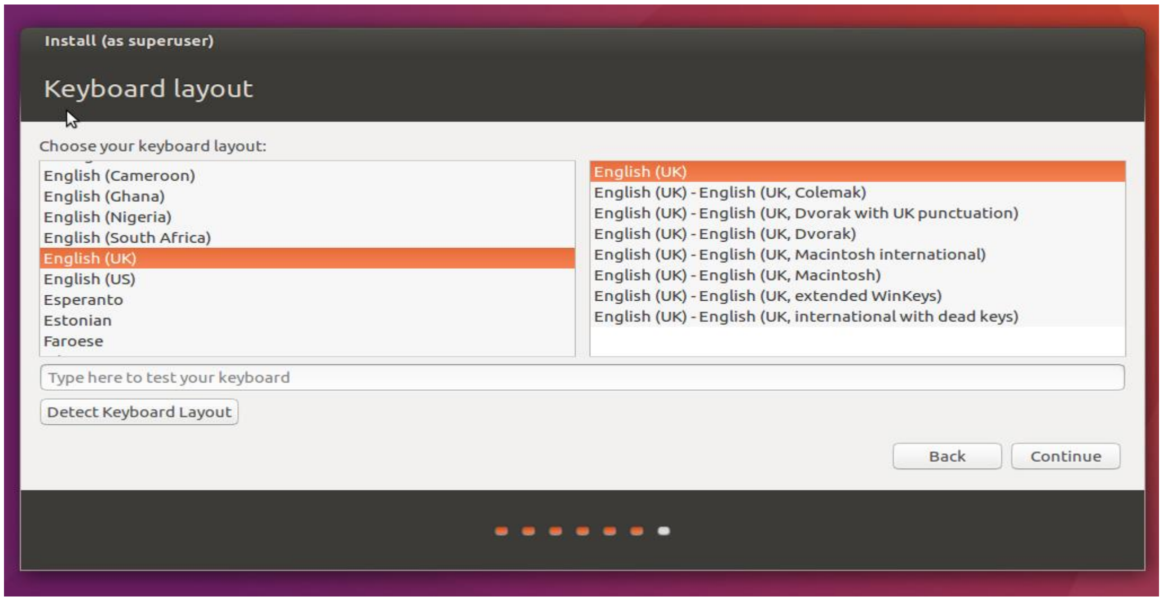
- If you are connected to the internet, this should be done automatically. Check your location is correct and click **’Forward’** to proceed. If you’re unsure of your time zone, type the name of the town you’re in or click on the map and we’ll help you find it.



Isle of Man Time

Select your preferred keyboard layout

- Click on the language option you need. If you're not sure, click the **'Detect Keyboard Layout'** button for help



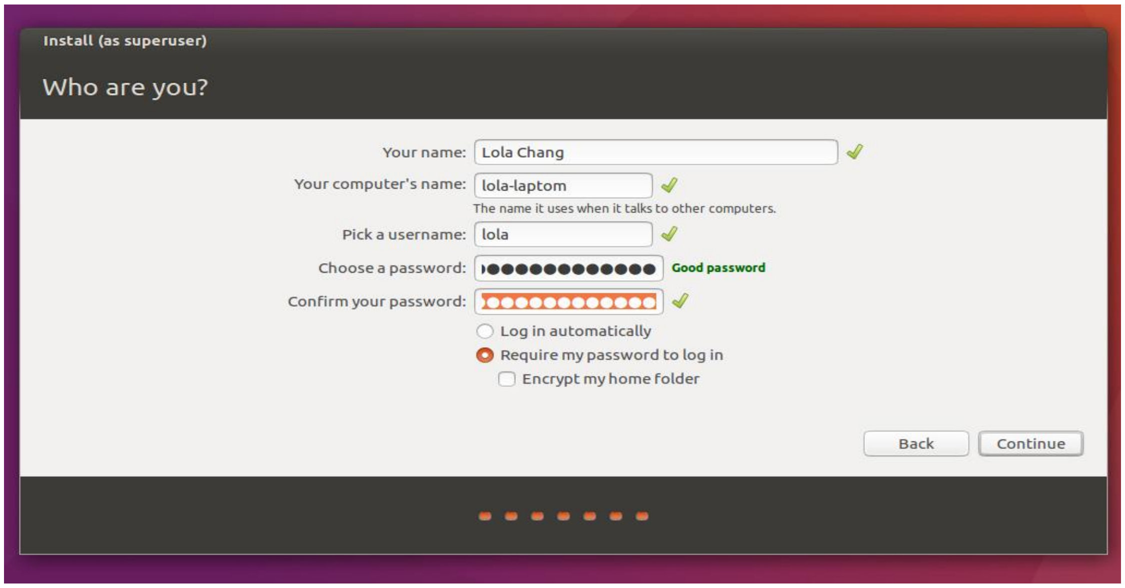
Choose your keyboard layout:

English (Cameroon)
English (Ghana)
English (Nigeria)
English (South Africa)
English (UK)
English (US)
Esperanto
Estonian
Faroese

English (UK)
English (UK) - English (UK, Colemak)
English (UK) - English (UK, Dvorak with UK punctuation)
English (UK) - English (UK, Dvorak)
English (UK) - English (UK, Macintosh international)
English (UK) - English (UK, Macintosh)
English (UK) - English (UK, extended WinKeys)
English (UK) - English (UK, international with dead keys)

Type here to test your keyboard

Enter your login and password details


Install (as superuser)

Who are you?

Your name: Lola Chang ✓

Your computer's name: lola-laptom ✓

The name it uses when it talks to other computers.

Pick a username: lola ✓

Choose a password: ●●●●●●●●●●●●● Good password

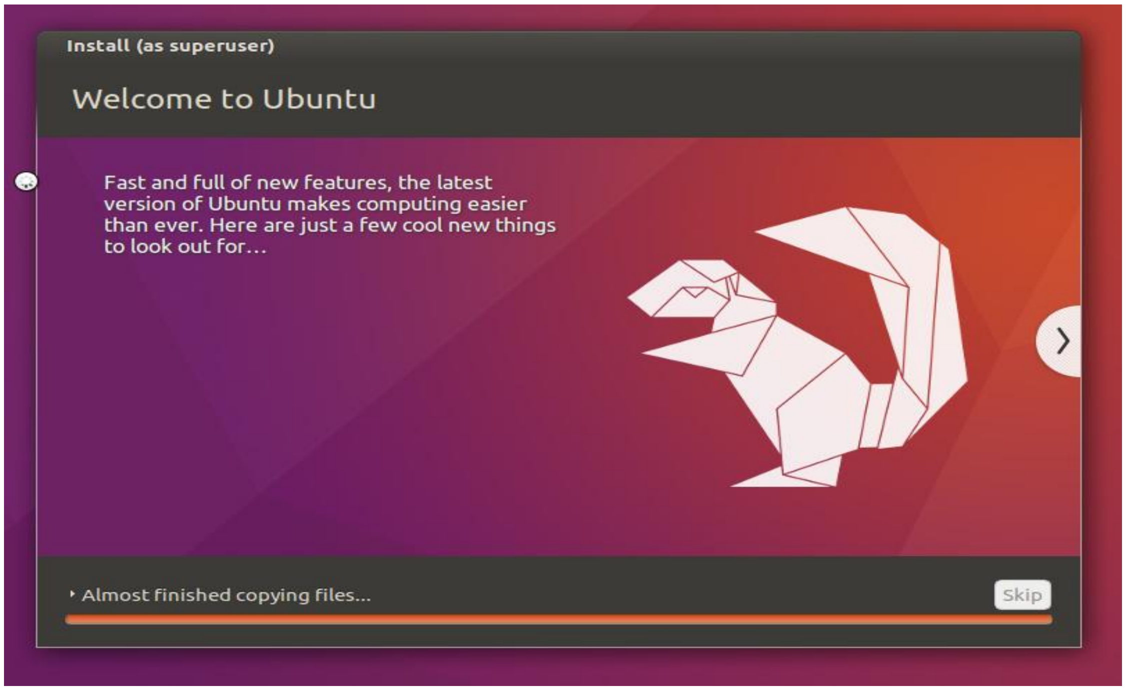
Confirm your password: ●●●●●●●●●●●●● ✓

○ Log in automatically

● Require my password to log in

☐ Encrypt my home folder

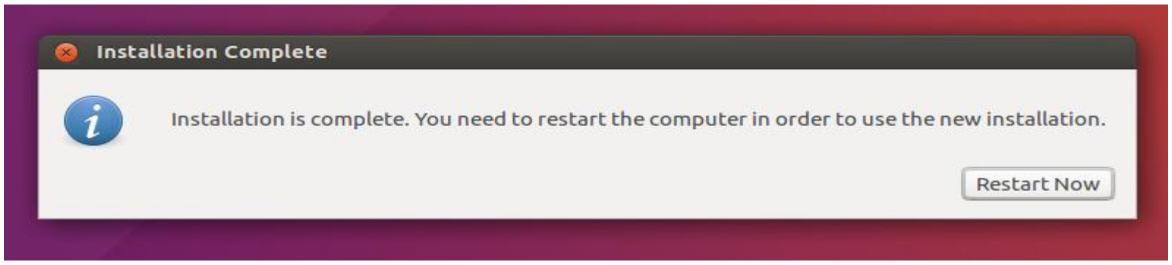
Learn more about Ubuntu while the system installs


Install (as superuser)

Welcome to Ubuntu

Fast and full of new features, the latest version of Ubuntu makes computing easier than ever. Here are just a few cool new things to look out for…

Almost finished copying files...


Installation Complete

Installation is complete. You need to restart the computer in order to use the new installation.

All that’s left is to restart your computer and start using Ubuntu

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.

Sample Code

Python Code

datainpfil1.py

```
import sys
import os
from dinpfil import *

class MyForm(QtGui.QMainWindow):
  def __init__(self,parent=None):
     QtGui.QWidget.__init__(self,parent)
     self.ui = Ui_MainWindow()
     self.ui.setupUi(self)
     QtCore.QObject.connect(self.ui.pushButton,QtCore.SIGNAL('clicked()'),self.checkfile1)
     QtCore.QObject.connect(self.ui.pushButton_2,QtCore.SIGNAL('clicked()'),self.checkfile2)
     QtCore.QObject.connect(self.ui.pushButton_3,QtCore.SIGNAL('clicked()'),self.checkfile3)


  def checkfile1(self):

        fname = str(self.ui.lineEdit.text())
        if (os.path.isfile(fname)):
                print ' file exists'
               else:
                print ' file does not exists'

  def checkfile2(self):

        fname = str(self.ui.lineEdit_2.text())
        if (os.path.isfile(fname)):
                print ' file exists'
               else:
                print ' file does not exists'

  def checkfile3(self):

        fname = str(self.ui.lineEdit_3.text())
        if (os.path.isfile(fname)):
                print ' file exists'
               else:
                print ' file does not exists'

if __name__ == "__main__":
    app = QtGui.QApplication(sys.argv)
    myapp = MyForm()
    myapp.show()
    sys.exit(app.exec_())
```

datainpgui1.py

```
import sys
from dinpgui import *

import MySQLdb as mdb

con = mdb.connect('localhost', 'team1', 'test623', 'diact1');

class MyForm(QtGui.QMainWindow):
  def __init__(self,parent=None):
     QtGui.QWidget.__init__(self,parent)
     self.ui = Ui_MainWindow()
     self.ui.setupUi(self)
     QtCore.QObject.connect(self.ui.pushButton,QtCore.SIGNAL('clicked()'),self.insertvalues)


  def insertvalues(self):

     with con:

        cur = con.cursor()
        arss13 = str(self.ui.lineEdit.text())
        pid = str(self.ui.lineEdit_3.text())
               tseg = str(self.ui.lineEdit_4.text())
               vrss14 = str(self.ui.lineEdit_7.text())
               arss14 = str(self.ui.lineEdit_8.text())
               arss12 = str(self.ui.lineEdit_5.text())
               vrss12 = str(self.ui.lineEdit_6.text())
        vrss13 = str(self.ui.lineEdit_2.text())
               actcode = str(self.ui.lineEdit_9.text())
        cur.execute('INSERT INTO dingui
VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s)',(pid,tseg,arss12,vrss12,arss13,vrss13,arss14,vrss14,
actcode))
        con.commit()


if __name__ == "__main__":
    app = QtGui.QApplication(sys.argv)
    myapp = MyForm()
    myapp.show()
    sys.exit(app.exec_())
```

diact1.py

```
import sys
import os
from diact import *

class MyForm(QtGui.QMainWindow):
  def __init__(self,parent=None):
     QtGui.QWidget.__init__(self,parent)
     self.ui = Ui_MainWindow()
     self.ui.setupUi(self)
     QtCore.QObject.connect(self.ui.pushButton_5,QtCore.SIGNAL('clicked()'),self.pdetails)
     QtCore.QObject.connect(self.ui.pushButton,QtCore.SIGNAL('clicked()'),self.dinpgui)
     QtCore.QObject.connect(self.ui.pushButton_2,QtCore.SIGNAL('clicked()'),self.dinpfil)
     QtCore.QObject.connect(self.ui.pushButton_3,QtCore.SIGNAL('clicked()'),self.timespe)
     QtCore.QObject.connect(self.ui.pushButton_4,QtCore.SIGNAL('clicked()'),self.calspe)
     #QtCore.QObject.connect(self.ui.pushButton_6,QtCore.SIGNAL('clicked()'),self.gnrep)


  def dinpgui(self):
		os.system("python datainpgui1.py")

  def pdetails(self):
		os.system("python patientdtls1.py")

  def dinpfil(self):
		os.system("python datainpfil1.py")

  def timespe(self):
		os.system("python samp2.py")

  def calspe(self):
		os.system("python samp3.py")

#  def gnrep(self):
#		os.system("python genrep1.py")


if __name__ == "__main__":
    app = QtGui.QApplication(sys.argv)
    myapp = MyForm()
    myapp.show()
    sys.exit(app.exec_())
```

patientdtls1.py

```
import sys
from patient import *

import MySQLdb as mdb

con = mdb.connect('localhost', 'team1', 'test623', 'diact1');

class MyForm(QtGui.QMainWindow):
  def __init__(self,parent=None):
     QtGui.QWidget.__init__(self,parent)
     self.ui = Ui_MainWindow()
     self.ui.setupUi(self)
     QtCore.QObject.connect(self.ui.pushButton,QtCore.SIGNAL('clicked()'),self.insertvalues)


  def insertvalues(self):

     with con:

        cur = con.cursor()
        aadhar = str(self.ui.lineEdit.text())
        pid = str(self.ui.lineEdit_3.text())
               pname = str(self.ui.lineEdit_4.text())
               page = str(self.ui.lineEdit_7.text())
               pweight = str(self.ui.lineEdit_8.text())
               addr1 = str(self.ui.lineEdit_5.text())
               addr2 = str(self.ui.lineEdit_6.text())
        mobile = str(self.ui.lineEdit_2.text())
        cur.execute('INSERT INTO patient
VALUES(%s,%s,%s,%s,%s,%s,%s,%s)',(pid,pname,page,pweight,addr1,addr2,aadhar,mobile))
        con.commit()


if __name__ == "__main__":
    app = QtGui.QApplication(sys.argv)
    myapp = MyForm()
    myapp.show()
    sys.exit(app.exec_())
```

samp1.py

```
from collections import Counter
lis1 = []
lis2 = []
with open("samp1.csv") as f:
    c = Counter(line.split()[0] for line in f)

for key,val in c.items():
    if val == 1:
        lis1.append(key)
    else:
        lis2.extend([key]*val)
print lis1
print lis2
```

samp2.py

```
from collections import Counter

cnt1 = 0
with open("bending1.csv") as f:
  c = Counter(line.split()[0] for line in f)
for key,val in c.items():
  if val == 1:
    cnt1 = cnt1 + 1
b1time_mins  = ((float(cnt1) * 250.0)/1000.0)/60.0
print 'Effective Minutes spent in bending1 is:' + str(b1time_mins)

cnt1 = 0
with open("bending2.csv") as f:
  c = Counter(line.split()[0] for line in f)
for key,val in c.items():
  if val == 1:
    cnt1 = cnt1 + 1
b2time_mins  = ((float(cnt1) * 250.0)/1000.0)/60.0
print 'Effective Minutes spent in bending2 is:' + str(b2time_mins)

cnt1 = 0
with open("cycling.csv") as f:
  c = Counter(line.split()[0] for line in f)
for key,val in c.items():
  if val == 1:
    cnt1 = cnt1 + 1
cytime_mins  = ((float(cnt1) * 250.0)/1000.0)/60.0
```

```
print 'Effective Minutes spent in cycling is:' + str(cytime_mins)
```

samp3.py

```
from collections import Counter

import MySQLdb as mdb
con = mdb.connect('localhost', 'team1', 'test623', 'diact1');

with con:
  cur = con.cursor()
  cur.execute('SELECT pweight,page FROM patient where pid = 001;');
  result = cur.fetchall()
  for row in result:
    pweight = row[0]
    page = row[1]

cnt1 = 0
with open("bending1.csv") as f:
  c = Counter(line.split()[0] for line in f)
for key,val in c.items():
  if val == 1:
    cnt1 = cnt1 + 1
b1time_mins  = ((float(cnt1) * 250.0)/1000.0)/60.0
calories = (((float(page)*0.217)-(float(pweight)*0.09036)+(float(220.0-float(page))*0.65*0.6309) -
55.0969)*float(b1time_mins)) / 41.84
print 'Calories spent in bending1 is:' + str(calories)

cnt1 = 0
with open("bending2.csv") as f:
  c = Counter(line.split()[0] for line in f)
for key,val in c.items():
  if val == 1:
    cnt1 = cnt1 + 1
b2time_mins  = ((float(cnt1) * 250.0)/1000.0)/60.0
calories = (((float(page)*0.217)-(float(pweight)*0.09036)+(float(220.0-float(page))*0.72*0.6309) -
55.0969)*float(b2time_mins)) / 41.84
print 'Calories spent in bending2 is:' + str(calories)

cnt1 = 0
with open("cycling.csv") as f:
  c = Counter(line.split()[0] for line in f)
for key,val in c.items():
  if val == 1:
    cnt1 = cnt1 + 1
cytime_mins  = ((float(cnt1) * 250.0)/1000.0)/60.0
calories = (((float(page)*0.217)-(float(pweight)*0.09036)+(float(220.0-float(page))*0.80*0.6309) -
55.0969)*float(cytime_mins)) / 41.84
```

```
print 'Calories spent in cycling is:' + str(calories)
```

Modules Imported in Python routines:

OS Module:

The OS module in Python provides a way of using operating system dependent functionality.

The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on – be that Windows, Mac or Linux.

OS functions:

Executing a shell command
os.system()

Get the users environment
os.environ()

Returns the current working directory.
os.getcwd()

Return the real group id of the current process.
os.getgid()

Return the current process's user id.
os.getuid()

Returns the real process ID of the current process.
os.getpid()

Set the current numeric umask and return the previous umask.
os.umask(mask)

Return information identifying the current operating system.
os.uname()

Change the root directory of the current process to path.
os.chroot(path)

Return a list of the entries in the directory given by path.
os.listdir(path)

Create a directory named path with numeric mode mode.
os.mkdir(path)

Recursive directory creation function.
os.makedirs(path)

Remove (delete) the file path.
os.remove(path)

Remove directories recursively.
os.removedirs(path)

Rename the file or directory src to dst.
os.rename(src, dst)

Remove (delete) the directory path.
os.rmdir(path)

Sys Module:

The sys module provides information about constants, functions and methods of the Python interpreter. dir(system) gives a summary of the available constants, functions and methods. Another possibility is the help() function. Using help(sys) provides valuable detail information.

The module sys informs e.g. about the maximal recursion depth (sys.getrecursionlimit()) and provides the possibility to change (sys.setrecursionlimit()).
The current version number of Python can be accessed as well by using this module.

Lots of scripts need access to the arguments passed to the script, when the script was started. argvargv (or to be precise sys.argv) is a list, which contains the command-line arguments passed to the script. The first item of this list contains the name of the script itself. The arguments follow the script name.

Every serious user of a UNIX or Linux operating system knows standard streams, i.e. input, standard output and standard error. They are known as pipes. They are commonly abbreviated as stdin, stdout, stderr.

The standard input (stdin) is normally connected to the keyboard, while the standard error and standard output go to the terminal (or window) in which you are working.

These data streams can be accessed from Python via the objects of the sys module with the same names, i.e. sys.stdin, sys.stdout and sys.stderr.

The standard output (stdout) can be redirected e.g. into a file, so that we can process this file later with another program. The same is possible with the standard error stream, we can redirect it into a file as well. We can redirect both stderr and stdout into the same file or into separate files.

Numpy:

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called axes.

NumPy is module for Python. The name is an acronym for "Numeric Python" or "Numerical Python". It is an extension module for Python, mostly written in C. This makes sure that the precompiled mathematical and numerical functions and functionalities of Numpy guarantee great execution speed.

Furthermore, NumPy enriches the programming language Python with powerful data structures, implementing multi-dimensional arrays and matrices. These data structures guarantee efficient calculations with matrices and arrays. The implementation is even aiming at huge matrices and arrays, better known under the heading of "big data". Besides that the module supplies a large library of high-level mathematical functions to operate on these matrices and arrays.

SciPy (Scientific Python) is often mentioned in the same breath with NumPy. SciPy needs Numpy, as it is based on the data structures of Numpy and furthermore its basic creation and manipulation functions. It extends the capabilities of NumPy with further useful functions for minimization, regression, Fourier-transformation and many others.

Both NumPy and SciPy are not part of a basic Python installation. They have to be installed after the Python installation. NumPy has to be installed before installing SciPy.

NumPy is based on two earlier Python modules dealing with arrays. One of these is Numeric. Numeric is like NumPy a Python module for high-performance, numeric computing, but it is obsolete nowadays. Another predecessor of NumPy is Numarray, which is a complete rewrite of Numeric but is deprecated as well. NumPy is a merger of those two, i.e. it is built on the code of Numeric and the features of Numarray.

NumPy's array class is called ndarray. It is also known by the alias array. Note that numpy.array is not the same as the Standard Python Library class array.array, which only handles one-dimensional arrays and offers less functionality. The more important attributes of an ndarray object are:

ndarray.ndim

The number of axes (dimensions) of the array.

ndarray.shape

The dimensions of the array. This is a tuple of integers indicating the size of the array in each dimension. For a matrix with n rows and m columns, shape will be (n,m). The length of the shape tuple is therefore the number of axes, ndim.

ndarray.size

The total number of elements of the array. This is equal to the product of the elements of shape.

ndarray.dtype

An object describing the type of the elements in the array. One can create or specify dtype's using standard Python types. Additionally NumPy provides types of its own. numpy.int32, numpy.int16, and numpy.float64 are some examples.

ndarray.itemsize

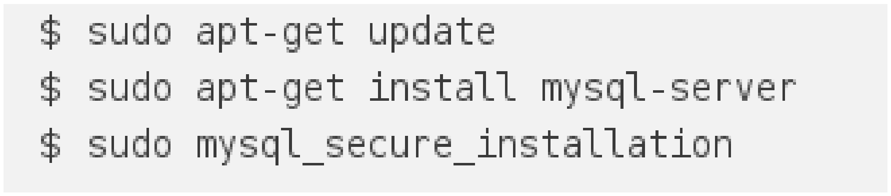
The size in bytes of each element of the array. For example, an array of elements of type float64 has itemsize 8 (=64/8), while one of type complex32 has itemsize 4 (=32/8). It is equivalent to ndarray.dtype.itemsize.

ndarray.data

The buffer containing the actual elements of the array. Normally, we won't need to use this attribute because we will access the elements in an array using indexing facilities.

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



```
$ sudo apt-get update
$ sudo apt-get install mysql-server
$ sudo mysql_secure_installation
```

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.

```
$ sudo mysql_secure_installation
```

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 walk through 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.

```
$ systemctl status mysql.service
```

You'll see output similar to the following:

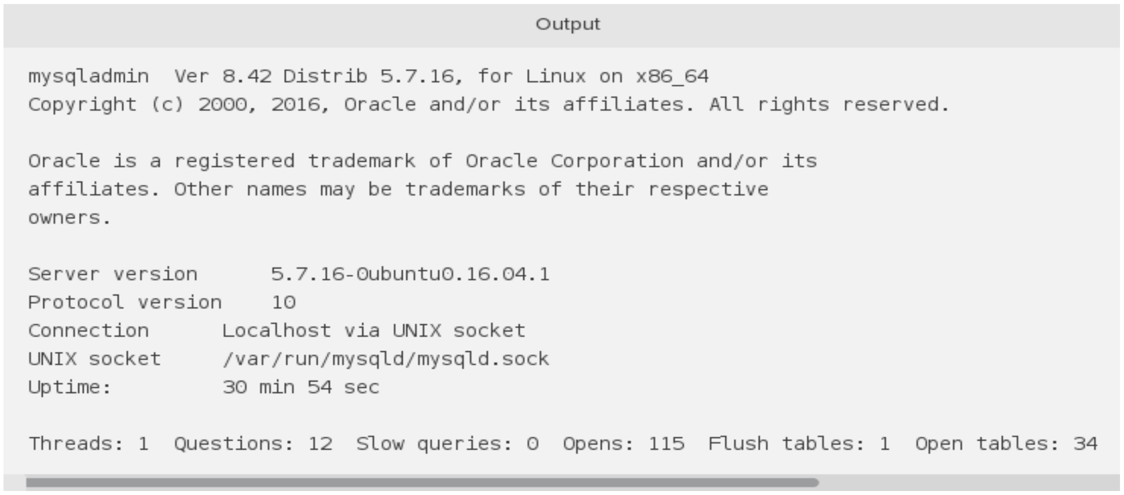
```
Output

● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: en
   Active: active (running) since Wed 2016-11-23 21:21:25 UTC; 30min ago
 Main PID: 3754 (mysqld)
    Tasks: 28
   Memory: 142.3M
      CPU: 1.994s
   CGroup: /system.slice/mysql.service
           └─3754 /usr/sbin/mysqld
```

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.

```
$ mysqladmin -p -u root version
```

This means MySQL is up and running.


Output

```
mysqladmin  Ver 8.42 Distrib 5.7.16, for Linux on x86_64
Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Server version      5.7.16-0ubuntu0.16.04.1
Protocol version    10
Connection        Localhost via UNIX socket
UNIX socket       /var/run/mysqld/mysqld.sock
Uptime:           30 min 54 sec

Threads: 1  Questions: 12  Slow queries: 0  Opens: 115  Flush tables: 1  Open tables: 34
```

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.

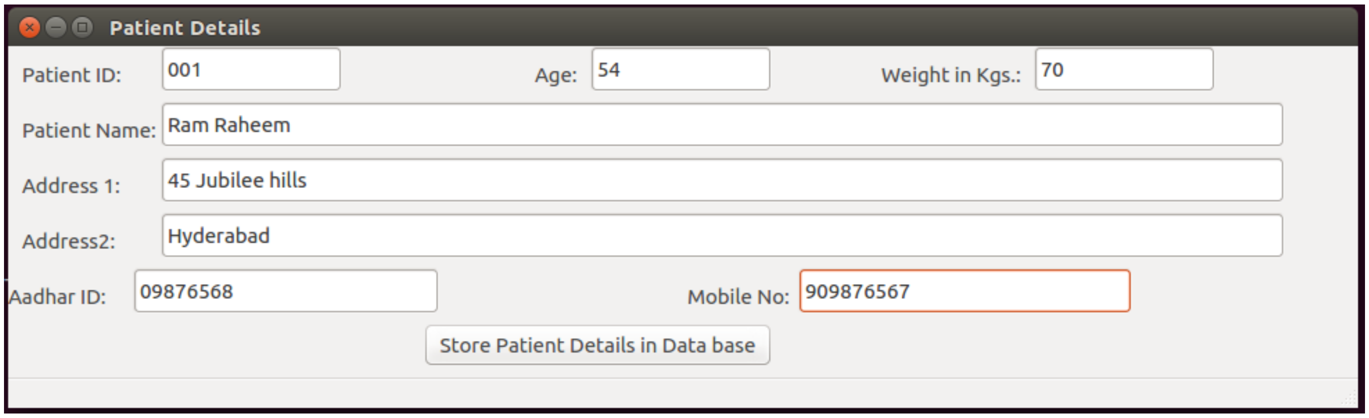
Regression Testing

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

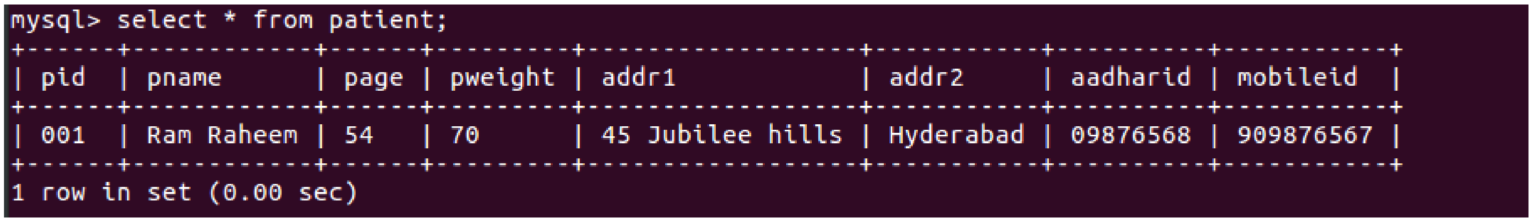
System Testing Results

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 Patient Details screen along with Data.



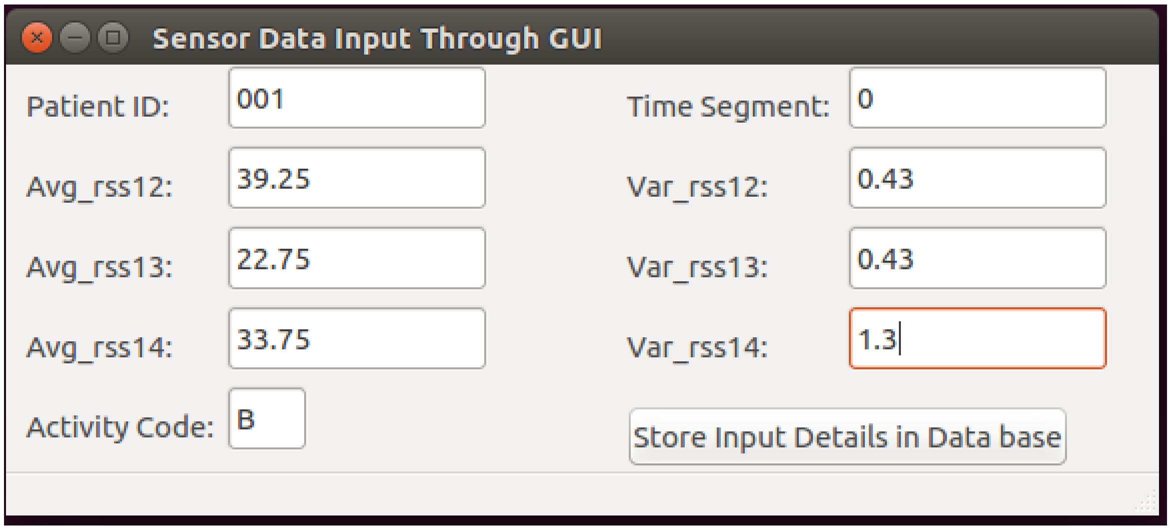
Patient ID:	001	Age:	54	Weight in Kgs.:	70
Patient Name:	Ram Raheem				
Address 1:	45 Jubilee hills				
Address2:	Hyderabad				
Aadhar ID:	09876568	Mobile No:	909876567		

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



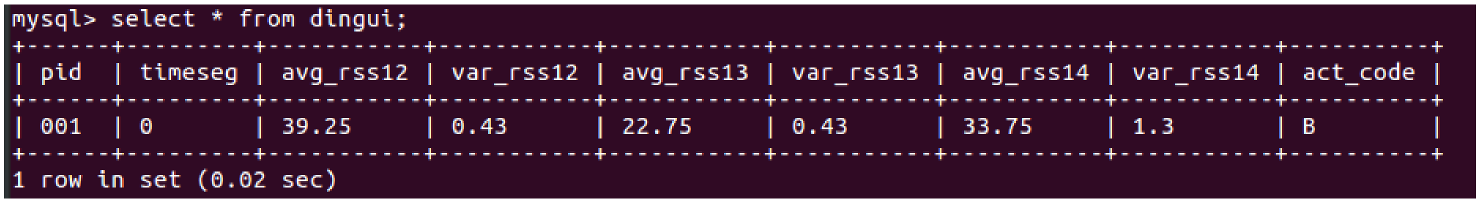
```
mysql> select * from patient;
+------+------------+------+---------+------------------+-----------+----------+-----------+
| pid  | pname      | page | pweight | addr1            | addr2     | aadharid | mobileid  |
+------+------------+------+---------+------------------+-----------+----------+-----------+
| 001  | Ram Raheem | 54   | 70      | 45 Jubilee hills | Hyderabad | 09876568 | 909876567 |
+------+------------+------+---------+------------------+-----------+----------+-----------+
1 row in set (0.00 sec)
```

Following is the 'Sensor Data Input Through GUI' screen along with Data.


Sensor Data Input Through GUI
Store Input Details in Data base

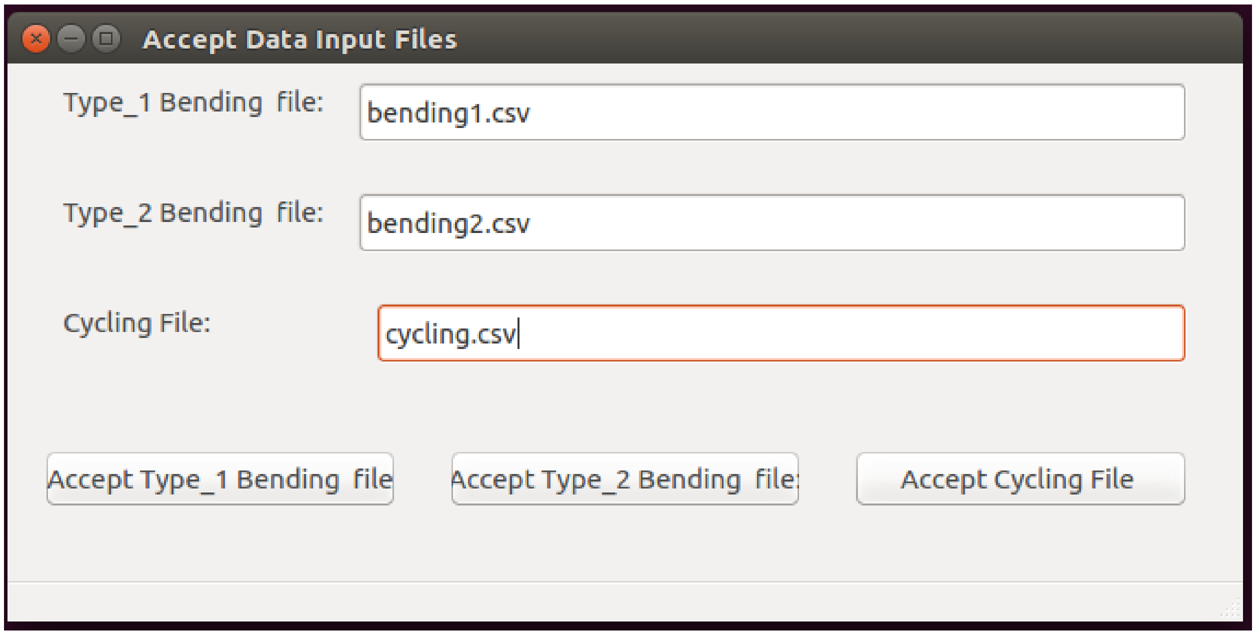
Patient ID:	001	Time Segment:	0
Avg_rss12:	39.25	Var_rss12:	0.43
Avg_rss13:	22.75	Var_rss13:	0.43
Avg_rss14:	33.75	Var_rss14:	1.3
Activity Code:	B		

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

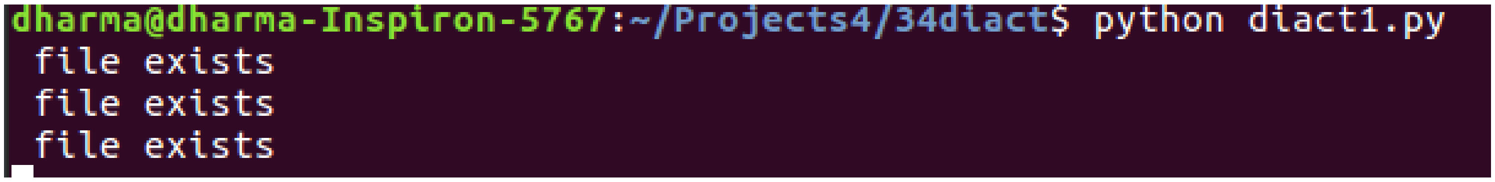


```
mysql> select * from dingui;
+------+---------+-----------+-----------+-----------+-----------+-----------+-----------+----------+
| pid  | timeseg | avg_rss12 | var_rss12 | avg_rss13 | var_rss13 | avg_rss14 | var_rss14 | act_code |
+------+---------+-----------+-----------+-----------+-----------+-----------+-----------+----------+
| 001  | 0       | 39.25     | 0.43      | 22.75     | 0.43      | 33.75     | 1.3       | B        |
+------+---------+-----------+-----------+-----------+-----------+-----------+-----------+----------+
1 row in set (0.02 sec)
```

Following is the 'Accept Data Input Files' screen along with Data.

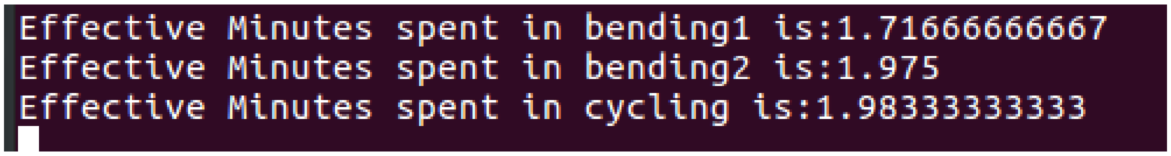


Following is the result of clicking the three push buttons, in the above screen.



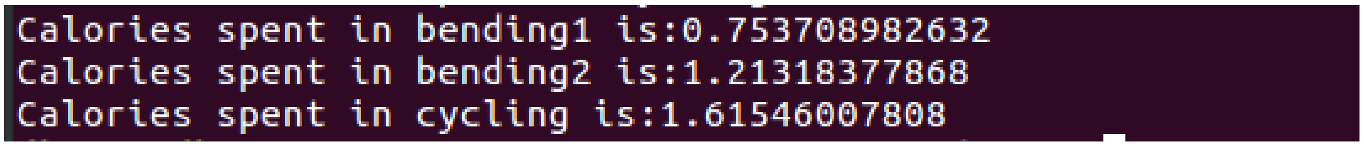
```
dharma@dharma-Inspiron-5767:~/Projects4/34diact$ python diact1.py
 file exists
 file exists
 file exists
```

Following is the result of running the time spent calculations python program.



```
Effective Minutes spent in bending1 is:1.71666666667
Effective Minutes spent in bending2 is:1.975
Effective Minutes spent in cycling is:1.98333333333
```

Following is the result of running the Calories calculations python program.



```
Calories spent in bending1 is:0.753708982632
Calories spent in bending2 is:1.21318377868
Calories spent in cycling is:1.61546007808
```

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

CONCLUSION

This project entitled **“Activity Analysis for Diabetic Patients”** is useful to the diabetic patients in measuring the calories spent by them during their activities. The project is very useful to the diabetic patients, as they know the calories burned by them for their activities. The project is also useful to the diabetic doctors in advising their patients regarding the food to be taken, and the calories to be burnt. This project finally leads to the improvement of diabetic patients life.

FUTURE SCOPE

As of now, the system is implementing only the software aspects of the activity recognition application. The hardware aspects can be treated in future.

CHAPTER 8

BIBLIOGRAPHY

1. https://www.healthline.com/health/diabetes
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/