

# **DJANGO WEB FRAMEWORK SOFTWARE METRICS MEASUREMENT USING RADON AND PYLINT**

*Major Project submitted to faculty of  
Computer Science and Engineering,  
JNTUH, Hyderabad  
For the Award of the Degree of*

## **BACHELOR OF TECHNOLOGY**

In

## **COMPUTER SCIENCE AND ENGINEERING**

*Submitted by*

<b>BOREDA ANAND</b>	<b>–</b>	<b>16L01A0507</b>
<b>A.ANIL KUMAR</b>	<b>–</b>	<b>16L01A0501</b>
<b>SYED MOIZUDDIN</b>	<b>–</b>	<b>16L01A0546</b>
<b>M.YASHWANTH</b>	<b>–</b>	<b>16L01A0525</b>

*Under the guidance of*

**R.Vinod Kumar**

**B.Tech,M.Tech**

**Assistant Professor,CSE**



**Department of Computer Science and Engineering  
TRR COLLEGE OF ENGINEERING**

*(Accredited by NBA, Affiliated to JNTUH)  
Inole(V), Patancheru (M), Sangareddy (Dt), T.S  
MAY 2020*

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  
**TRR COLLEGE OF ENGINEERING**

(Accredited by NBA, Affiliated to JNTUH)  
Inole(V), Patancheru (M), Sangareddy (Dt), T.S  
MAY 2020



**CERTIFICATE**

This is to certify that, the Major Project entitled “ **DJANGO WEB FRAMEWORK SOFTWARE METRICS MEASUREMENT USING RADON AND PYLINT**” has been submitted by **BOREDA ANAND, A.ANIL KUMAR, M.YASHWANTH, SYED MOIZUDDIN** in partial fulfillment of the requirements for the award of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING**. This record of bonafide work carried out by them under my guidance and supervision. The result embodied in this project report has not been submitted to any other University or Institute for the award of any degree.

**R.Vinod Kumar**  
B. Tech, M. Tech  
Internal Guide

**L. Praveen Kumar**  
B. Tech., M. Tech., M.B.A  
Head of the Department

***External Examiner***

## ACKNOWLEDGEMENT

First and foremost, I offer my sincere gratitude to my project guide **R. Vinod Kumar** Assistant Professor of Computer Science & Technology, for her/his guidance and encouragement to do this project work.

I am also grateful to **Mr.L.Praveen Kumar** Head of the Department of Computer Science & Engineering for his support and valuable suggestions during the project work. I thank him for his valuable suggestions at the time of seminars which encouraged me to give my best in the project.

I express my profound gratitude to members of TRR COLLEGE OF ENGINEERING for the successful development of this project and complying with our time schedules.

I convey my thanks to all the Faculty members of CSE Department, without their co-operation this project would not have been successful. Also, I convey my thanks to all the teaching and non-teaching staff members of CSE Department.

I consider it as my privilege to express my gratitude and respect to all who guided, inspired and helped me in completion of the project work.

Name of the student	Roll No
BOREDA ANAND	16L01A0507
A. ANIL KUMAR	16L01A0501
SYED MOIZUDDIN	16L01A0546
M.YASHWANTH	16L01A0525

## **ABSTRACT**

Nowadays, sites are complex applications that carry out transactions, render real-time information and offer interaction. Creation of web applications that allow many designers, advanced tools, and many more options. Web frameworks deliver a brilliant way between creating an application from the ground and using a content management system. This article centers around an open source web development framework, to be specific to Django. The methodology that used in the study is measuring Django Web Frameworks code quality metrics using Radon and Pylint. Django option in the main directory has 2,200 lines of code, Cyclomatic Complexity score is 16.375 considered as average complexity, and 6.69/10 by the Pylint score.

# INDEX

<b>SNO</b>	<b>CHAPTER NAME</b>	<b>PAGE NO</b>
	ABSTRACT	
1.	<b>INTRODUCTION</b>	<b>1</b>
2.	<b>FEASIBILITY STUDY</b>	<b>2</b>
	2.1 ECOMOIC STUDY	2
	2.2 TECHNICAL STUDY	2
	2.3 SOCIAL FEASIBILITY	3
3.	<b>SYSTEM ANALYSIS</b>	<b>4</b>
	3.1 EXISTING SYSTEM	4
	3.2 PROPOSED SYSTEM	4
	3.3 MODULES	5
	3.4 SYSTEM CONFIGURATION	7
4.	<b>SYSTEM DESIGN</b>	<b>9</b>
	4.1 INPUT AND OUTPUT DESIGN	9
	4.2 DATA FLOW DIAGRAM	12
	4.3 UML DIAGARM	14
	4.4 USE CASE DIAGRAM	15
	4.5 CLASS DIAGRAM	17
	4.6 SEQUENCE DIAGRAM	17
	4.7ACTIVITY DIAGRAM	18

5.	<b>IMPLEMENTATION</b>	<b>20</b>
	4.5 SYSTEM ARCHITECTURE	20
	4.5 PYTHON	21
	4.5 DJANGO	45
6.	<b>SAMPLE CODE</b>	<b>63</b>
7.	<b>TESTING</b>	<b>67</b>
	7.1 UNIT TESTING	67
	7.2 INTEGRATION TESTING	67
	7.3 FUNCTIONAL TESTING	68
	7.4 SYSTEM TEST	68
	7.5 WHITE BOX TESTING	69
	7.6 BLACK BOX TESTING	69
	7.7 ACCEPTANCE TESTING	69
	7.8 TEST CASES	70
8.	<b>OUTPUT SCREENS</b>	<b>71</b>
	8.1 HOME PAGE	71
	8.2 USER LOGIN PAGE	72
	8.3 NEW USER REGISTRATION	73
	8.4 ADMIN LOGIN PAGE	74
	8.5 USER DATE PAGE	75
	8.6 FILE DATA IN USER SESSION	76

	8.7 NEW FILE UPLOAD PAGE	77
	8.8 FILE DATA IN ADMIN SESSION	78
	8.9 ANALYTICS IN USER SESSION	79
	8.10 ANALYTICS IN ADMIN SESSION	80
9.	<b>CONCLUSION</b>	<b>81</b>
10.	<b>FUTURE ENHANCEMENTS</b>	<b>82</b>
	<b>REFERENCES</b>	<b>83</b>

# 1. INTRODUCTION

Nowadays, in a fast-paced, very demanding and competitive era people tend to use a web application framework to build their web-enabled applications. Software frameworks used to establish web application, and website development, web services, and web resources are called Web application frameworks. A favorite type of web app framework is the Model-View-Controller (MVC) design separates the code for each application component into modules. Python gains its popularity since 2016 and become number. Together with Python, C, Java, and C++ remain in the top four in popularity; this paper focuses on Python web frameworks. The Python web frameworks are full of options, such as Django framework is a compact Python framework intended for quick development in quickly paced environments that function well with relational databases. Flask framework is a Python micro-framework with a moderate approach and powerful by itself. Flask perfect for stand-alone applications and rapid prototyping. Pyramid known as Pylons is a framework that gives reproducibility with NoSQL integration. Pyramid best for APIs, prototyping, and extensive web apps, like content management systems development. Tornado is an event-based, nonblocking Python web server and web application framework serve high-traffic volume. The Bottle is a light and simple microframework. Django is chosen because of its popularity and widely used. In this article, we will find out software metrics for Django using Radon that provides cyclomatic complexity raw metrics that consists SLOC, comment lines, number blank lines, Maintainability Index and Halstead metrics, also use pylint. Pylint is a source code analyzer that finds for errors in programming, assists to use a coding standard strictly.



## **2. FEASIBILITY STUDY**

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

Three key considerations involved in the feasibility analysis are

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

### **2.1 ECONOMIC FEASIBILITY**

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

### **2.2 TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high

demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

## **2.3 SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

## **3.SYSTEM ANALYSIS**

### **3.1 EXISTING SYSTEM:**

Flask framework is a Python micro-framework with a moderate approach and powerful by itself. Flask perfect for stand-alone applications and rapid prototyping. Pyramid known as Pylons is a framework that gives reproducibility with NoSQL integration. Pyramid best for APIs, prototyping, and extensive web apps, like content management systems development. Tornado is an event-based, non-blocking Python web server and web application framework serve high-traffic volume. The Bottle is a light and simple microframework.

### **DISADVANTAGES OF EXISTING SYSTEM:**

- There will be times as a software developer when a programmer works on projects that do not have unit tests, a daily build, a mess of legacy spaghetti code
- Lines of code (LOC) are the simplest indicators and warnings. A massive, complicated project will have hundreds of thousands of LOC and that normal if the project is well-managed.

### **3.2 PROPOSED SYSTEM:**

Django Reinhardt is used as the framework name Django is a framework for web development using python and created for the quick development of database driven sites. Inside Django is MVC-style, a high-level, open source bunch of libraries programmed in Python. Django is the most popular server-side web frameworks.

Don't repeat yourself is Django's motto. As much as Python, it focuses inefficiency, giving developer to do almost all tasks with as little coding effort as possible.

### **ADVANTAGES OF PROPOSED SYSTEM:**

- LLOC: The variety of logical strains of code. Each logical line of code contains precisely one statement.
- SLOC: The quantity of source lines of code. It does not relate to the LLOC.
- LOC: The aggregate number of lines of code. It doesn't relate to the number of lines in the file.
- Multi: The number of lines that represent multiline strings.
- Comments The number of comment lines. Multiline strings are not considered comment, to the Python, they are just strings.

### **3.3 MODULES**

- User
- Admin
- Django

#### **User:**

in a fast-paced, very demanding and competitive era people tend to use a web application framework to build their web-enabled applications. Software frameworks used to establish web application, and website development, web services, and web resources are called Web application frameworks. A favorite type of

web app framework is the Model-View-Controller (MVC) design separates the code for each application component into modules shows the Maintainability Index, so software developer can measure and aware how their code complexity and its maintainability then he/ she can charge accordingly to the user.

### **Admin:**

we will find out software metrics for Django using Radon that provides cyclomatic complexity raw metrics that consists SLOC, comment lines, number blank lines, Maintainability Index and Halstead metrics, also use Pylint. Pylint is a source code analyzer that finds for errors in programming, assists to use a coding standard strictly.

### **Django:**

Django was created at the Lawrence Journal-World newspaper. Adrian Holovaty and Simon Willison started using Python to create web applications. In July 2005, Django was released under a BSD license for the public. Django is used by The National Aeronautics and Space Administration (NASA), Google, Public Broadcasting Service (PBS), Bit-Bucket, and other newspapers. Django Reinhardt is used as the framework name. Django is a framework for web development using python and created for the quick development of database driven sites. In side Django is MVC-style, a high-level, opensource bunch of libraries programmed in Python. Django is the most popular server-side web frameworks. Don't repeat yourself is Django's motto. As much as Python, it focuses inefficiency, giving

developer to do almost all tasks with as little coding effort as possible. Furthermore, Django is also scalable, mature, and fast with a numerous community of developer and a robust set of built-in components. Django can access or create JSON or XML data instances and deal with out-of-the-box with relational database management systems such as Oracle, MySQL, SQLite, and PostgreSQL. At deployment, Django is fully backed up by the cloud platform AWS Elastic Beanstalk and Heroku. Three items make Django excel are its maturity, structure, and community. In contrast to Flask and Pyramid, Django stresses on getting started right out of the box using particular modules. In other words, Django's bundled applications and modular components can reduce developer time when he/she was trying to make a web application ready and functioning. Data science is a buzzword nowadays, and multitalented developers are needed more than ever before. A simple preparation in building apps with a framework as powerful, minimalistic, and easy-to-learn as Django will be an essential skill when launching their career as the web developer or an entrepreneur.

### **3.4 SYSTEM CONFIGURATION**

#### **HARDWARE REQUIREMENTS:**

- ❖ System : Pentium IV 2.4 GHz.
- ❖ Hard Disk : 40 GB.
- ❖ Floppy Drive : 1.44 Mb.
- ❖ Monitor : 14' Colour Monitor.
- ❖ Mouse : Optical Mouse.
- ❖ Ram : 512 Mb.

## **SOFTWARE REQUIREMENTS:**

- ❖ Operating system : Windows 7 Ultimate.
- ❖ Coding Language : Python.
- ❖ Front-End : Python.
- ❖ Designing : Html, CSS, JavaScript.
- ❖ Data Base : MySQL.

## **4. SYSTEM DESIGN**

### **4.1 INPUT AND OUTPUT DESIGN**

#### **INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

#### **OBJECTIVES**

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data



entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus, the objective of input design is to create an input layout that is easy to follow

## **OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

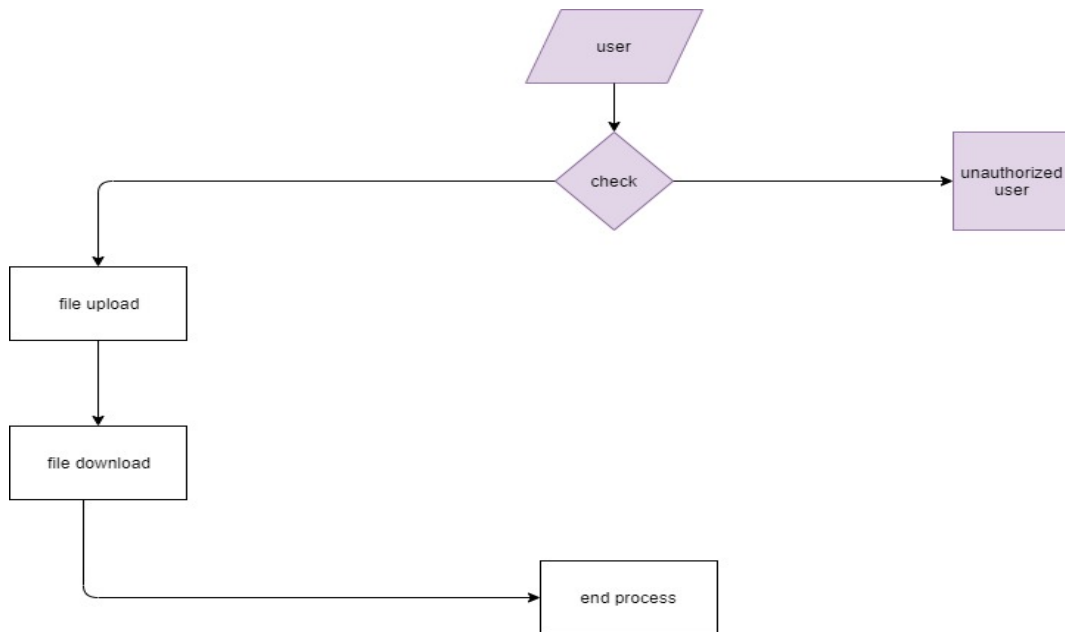
3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

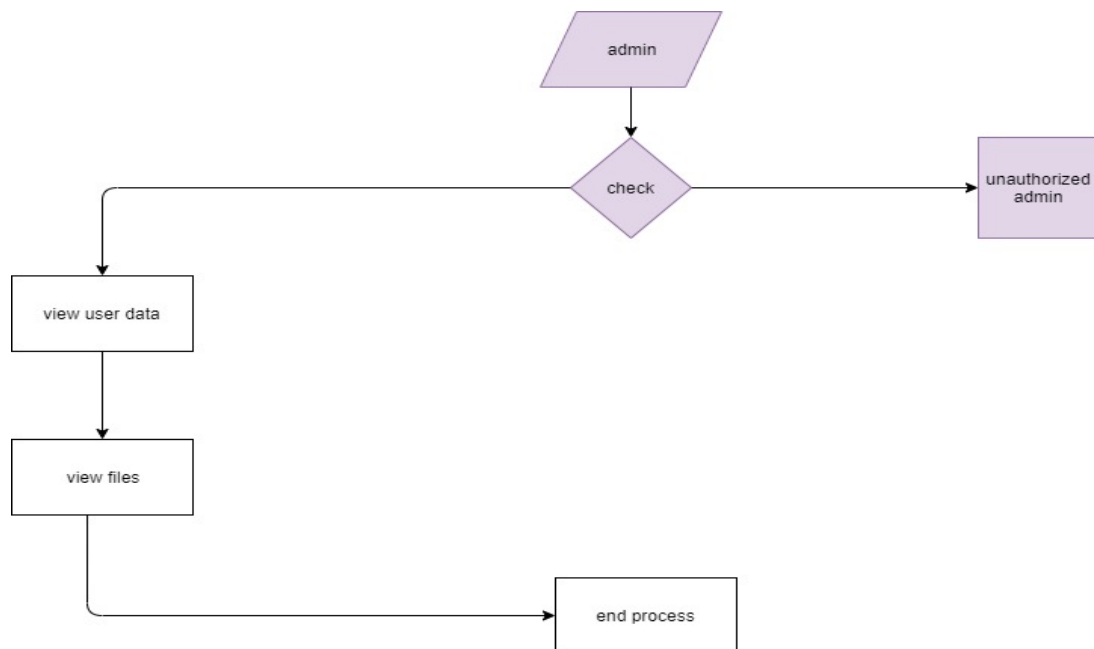
## 4.2 DATA FLOW DIAGRAM:

- ❖ The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
- ❖ The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
- ❖ DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
- ❖ DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



**fig 4.2.1 User DFD**

The above figure shows the data flow diagram of how user logs in and how he gets access to database



**fig 4.2.2 Admin DFD**

The above figure shows the data flow diagram of how admin logs in and how he gets access to database

### 4.3 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

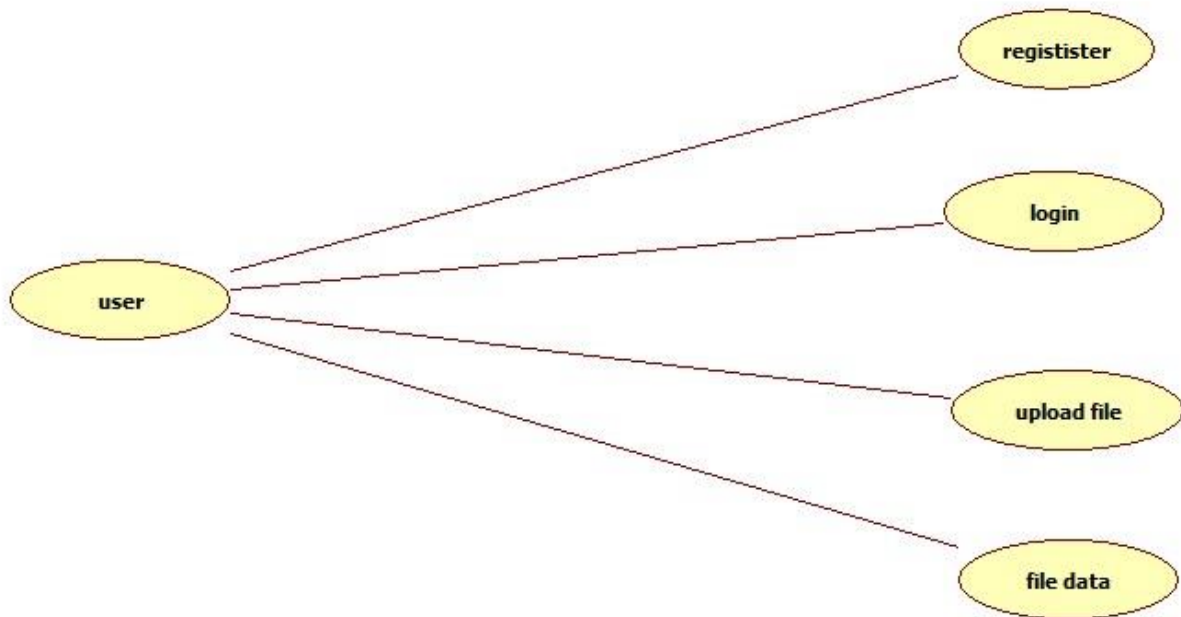
#### **GOALS:**

The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

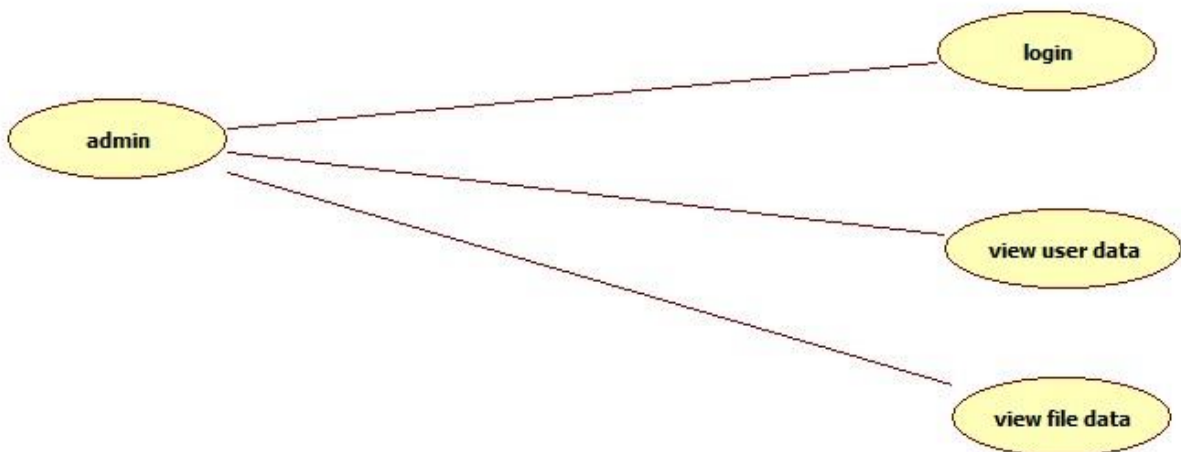
#### **4.4 USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



***fig 4.4.1 Use Case diagram***

The above figure shows the use case diagram of the accessibilities of the user



***fig 4.4.2 Use Case diagram***

The above figure shows the use case diagram of the accessibilities of the admin

#### 4.5 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



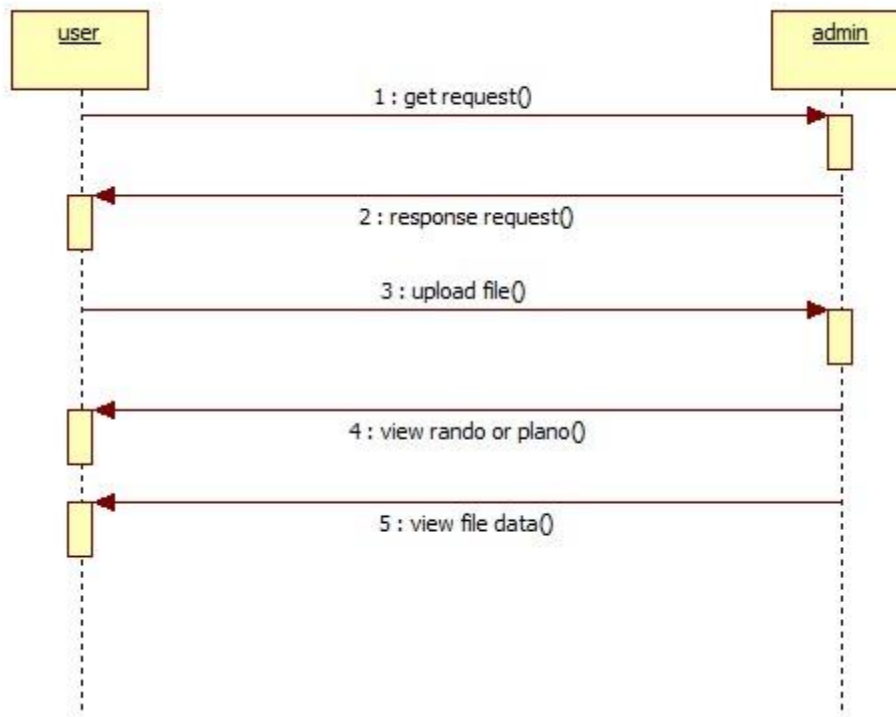
*fig 4.5 Class diagram*

The above figure shows the class diagram of operations and relations of the user and admin

#### 4.6 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



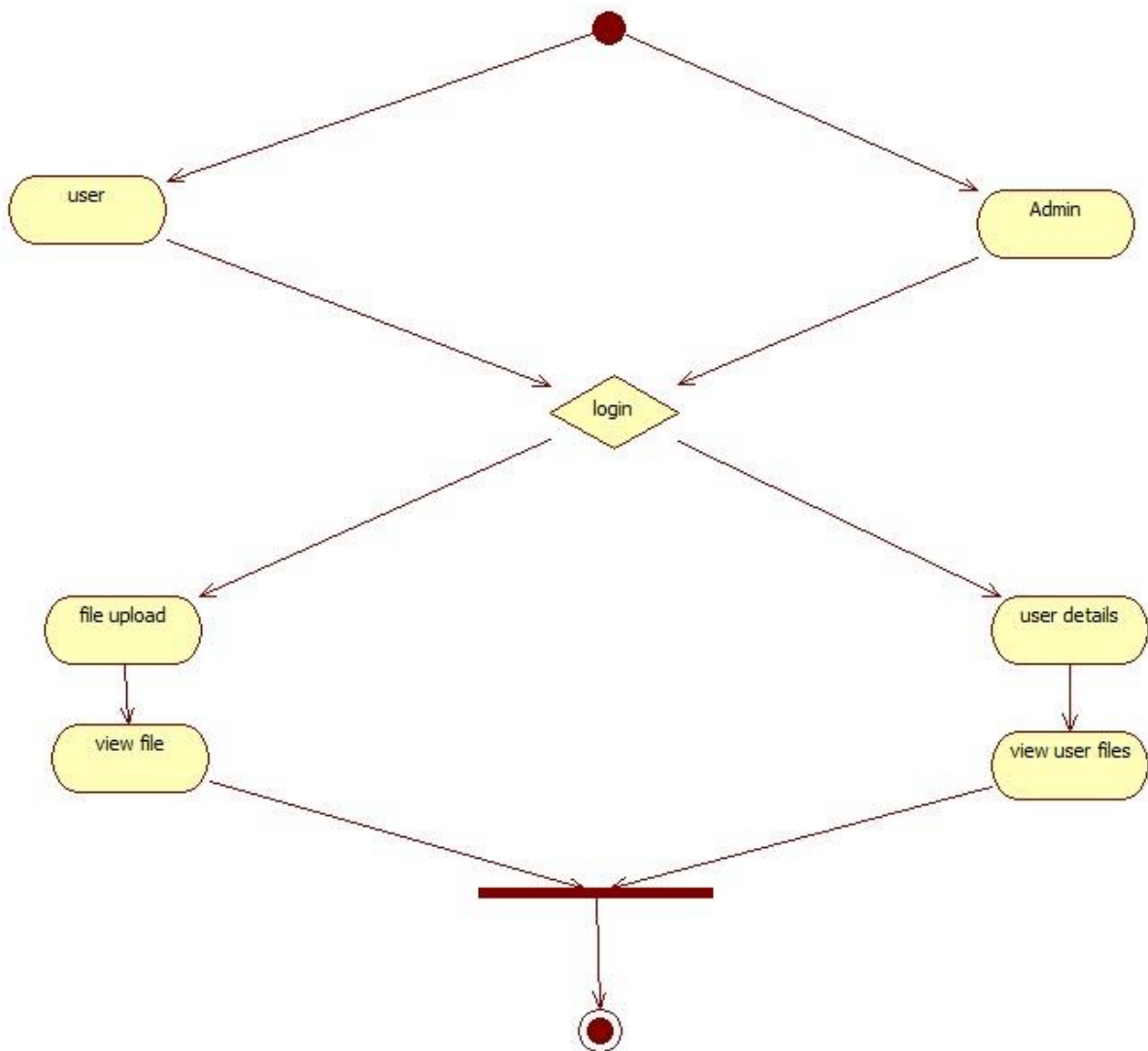


***fig 4.6 Sequence diagram***

The above figure shows the sequence diagram of request and response between user and admin

#### **4.7 ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control



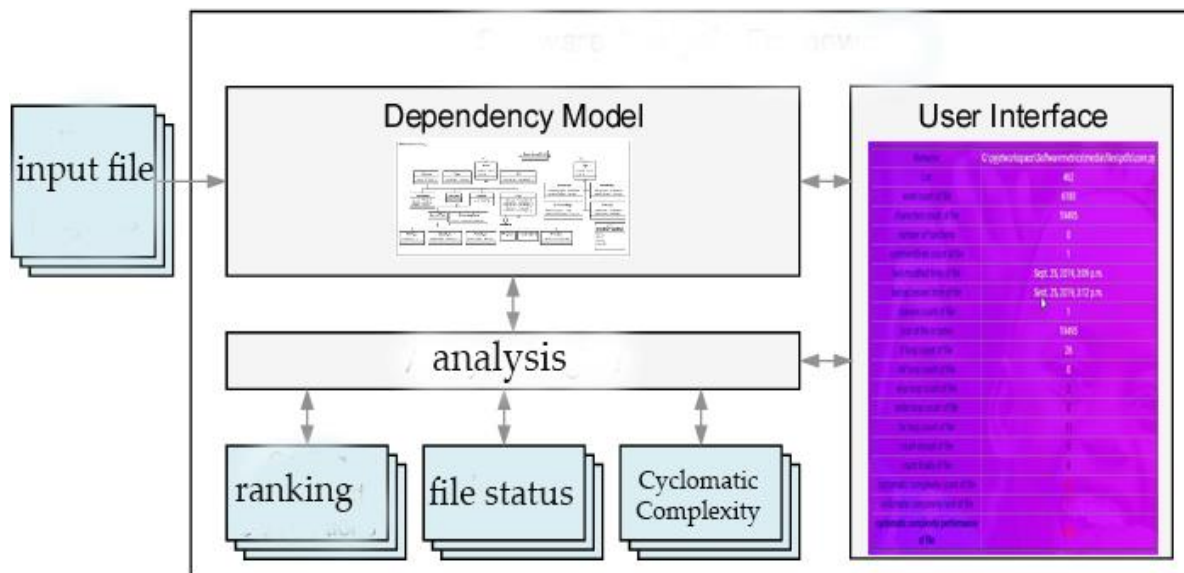
***fig 4.7 Activity diagram***

The above figure shows the activity diagram containing activities of user and admin

## 5. IMPLEMENTATION

### 5.1 SYSTEM ARCHITECTURE:

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well-ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.



**fig 5.1 Architecture of proposed system**

The above figure shows the architecture of the system of how the input file is taken as input, analyzed and result is given as output

## 5.2 PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. 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. It provides constructs that enable clear programming on both small and large scales. 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. CPython is managed by the non-profit Python Software Foundation. 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.

### **Interactive Mode Programming**

Invoking the interpreter without passing a script file as a parameter brings up the following prompt –

```
$ python
```

Python 2.4.3 (#1, Nov 11 2010, 13:34:43)

[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2

Type "help", "copyright", "credits" or "license" for more information.

```
>>>
```

Type the following text at the Python prompt and press the Enter –

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, Python!");. However, in Python version 2.4.3, this produces the following result –

Hello, Python!

## **Script Mode Programming**

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file –

Live Demo

```
print "Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable.  
Now, try to run this program as follows –

```
$ python test.py
```

This produces the following result –

```
Hello, Python!
```

Let us try another way to execute a Python script. Here is the modified test.py file –

Live Demo

```
#!/usr/bin/python
```

```
print "Hello, Python!"
```

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows –

```
$ chmod +x test.py    # This is to make file executable
```

```
$ ./test.py
```

This produces the following result –

Hello, Python!

## **Python Identifiers**

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (\_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, \$, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers –

Class names start with an uppercase letter. All other identifiers start with a lowercase letter.

Starting an identifier with a single leading underscore indicates that the identifier is private.

Starting an identifier with two leading underscores indicates a strongly private identifier.

If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

## **Reserved Words**

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and exec not

assert finally or

break for pass

class from print

continue global raise



```
def if return
del import try
elif in while
else is with
except lambda yield
```

## **Lines and Indentation**

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

```
if True:
    print "True"
else:
    print "False"
```

However, the following block generates an error –

```
if True:

    print "Answer"

    print "True"

else:

    print "Answer"

    print "False"
```

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks –

Note – Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces.

```
#!/usr/bin/python
```

```
import sys
```

```
try:
```

```
    # open file stream
```

```
file = open(file_name, "w")

except IOError:

    print "There was an error writing to", file_name

    sys.exit()

print "Enter ", file_finish,

print "" When finished"

while file_text != file_finish:

    file_text = raw_input("Enter text: ")

    if file_text == file_finish:

        # close the file

        file.close

        break

    file.write(file_text)

    file.write("\n")

file.close()

file_name = raw_input("Enter filename: ")

if len(file_name) == 0:

    print "Next time please enter something"

    sys.exit()
```

```
try:

    file = open(file_name, "r")

except IOError:

    print "There was an error reading file"

    sys.exit()

file_text = file.read()

file.close()

print file_text
```

## Multi-Line Statements

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example –

```
total = item_one + \

    item_two + \

    item_three
```

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example –

```
days = ['Monday', 'Tuesday', 'Wednesday',
```

```
'Thursday', 'Friday']
```

## Quotation in Python

Python accepts single (`'`), double (`"`) and triple (`'''` or `"""`) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal –

```
word = 'word'
```

```
sentence = "This is a sentence."
```

```
paragraph = """This is a paragraph. It is  
made up of multiple lines and sentences."""
```

## Comments in Python

A hash sign (`#`) that is not inside a string literal begins a comment. All characters after the `#` and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

## Live Demo

```
#!/usr/bin/python
```

```
# First comment
```

```
print "Hello, Python!" # second comment
```

This produces the following result –

Hello, Python!

You can type a comment on the same line after a statement or expression –

```
name = "Madisetti" # This is again comment
```

You can comment multiple lines as follows –

```
# This is a comment.
```

```
# This is a comment, too.
```

```
# This is a comment, too.
```

```
# I said that already.
```

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comments:

```
'''
```

This is a multiline  
comment.

'''

## Using Blank Lines

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

In an interactive interpreter session, you must enter an empty physical line to terminate a multiline statement.

## Waiting for the User

The following line of the program displays the prompt, the statement saying “Press the enter key to exit”, and waits for the user to take action –

```
#!/usr/bin/python
```

```
raw_input("\n\nPress the enter key to exit.")
```

Here, "\n\n" is used to create two new lines before displaying the actual line. Once the user presses the key, the program ends. This

is a nice trick to keep a console window open until the user is done with an application.

### Multiple Statements on a Single Line

The semicolon ( ; ) allows multiple statements on the single line given that neither statement starts a new code block. Here is a sample snip using the semicolon.

```
import sys; x = 'foo'; sys.stdout.write(x + '\n')
```

### Multiple Statement Groups as Suites

A group of individual statements, which make a single code block are called suites in Python. Compound or complex statements, such as if, while, def, and class require a header line and a suite.

Header lines begin the statement (with the keyword) and terminate with a colon ( : ) and are followed by one or more lines which make up the suite. For example –

```
if expression :
```

```
    suite
```

```
elif expression :
```

```
    suite
```

```
else :
```

```
    suite
```



## Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with `-h` –

```
$ python -h
```

```
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
```

Options and arguments (and corresponding environment variables):

`-c cmd` : program passed in as string (terminates option list)

`-d` : debug output from parser (also `PYTHONDEBUG=x`)

`-E` : ignore environment variables (such as `PYTHONPATH`)

`-h` : print this help message and exit

You can also program your script in such a way that it should accept various options. Command Line Arguments is an advanced topic and should be studied a bit later once you have gone through rest of the Python concepts.

## Python Lists

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example –

```
list1 = ['physics', 'chemistry', 1997, 2000];
```

```
list2 = [1, 2, 3, 4, 5 ];
```

```
list3 = ["a", "b", "c", "d"]
```

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example –

```
tup1 = ('physics', 'chemistry', 1997, 2000);
```

```
tup2 = (1, 2, 3, 4, 5 );
```

```
tup3 = "a", "b", "c", "d";
```

The empty tuple is written as two parentheses containing nothing –

```
tup1 = ();
```

To write a tuple containing a single value you have to include a comma, even though there is only one value –

```
tup1 = (50,);
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

### Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example –

#### Live Demo

```
#!/usr/bin/python
```

```
tup1 = ('physics', 'chemistry', 1997, 2000);
```

```
tup2 = (1, 2, 3, 4, 5, 6, 7 );
```

```
print "tup1[0]: ", tup1[0];  
  
print "tup2[1:5]: ", tup2[1:5];
```

When the above code is executed, it produces the following result –

```
tup1[0]: physics  
  
tup2[1:5]: [2, 3, 4, 5]
```

## Updating Tuples

## Accessing Values in Dictionary

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example –

### Live Demo

```
#!/usr/bin/python
```

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}  
  
print "dict['Name']: ", dict['Name']  
  
print "dict['Age']: ", dict['Age']
```

When the above code is executed, it produces the following result –

```
dict['Name']: Zara
```

```
dict['Age']: 7
```

If we attempt to access a data item with a key, which is not part of the dictionary, we get an error as follows –

Live Demo

```
#!/usr/bin/python
```

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
```

```
print "dict['Alice']: ", dict['Alice']
```

When the above code is executed, it produces the following result –

```
dict['Alice']:
```

Traceback (most recent call last):

File "test.py", line 4, in <module>

```
print "dict['Alice']: ", dict['Alice'];
```

KeyError: 'Alice'

## Updating Dictionary

You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example –

### Live Demo

```
#!/usr/bin/python
```

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
```

```
dict['Age'] = 8; # update existing entry
```

```
dict['School'] = "DPS School"; # Add new entry
```

```
print "dict['Age']: ", dict['Age']
```

```
print "dict['School']: ", dict['School']
```

When the above code is executed, it produces the following result –

```
dict['Age']: 8
```

```
dict['School']: DPS School
```

### Delete Dictionary Elements

You can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the del statement. Following is a simple example –

Live Demo

```
#!/usr/bin/python
```

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}  
  
del dict['Name']; # remove entry with key 'Name'  
  
dict.clear();    # remove all entries in dict  
  
del dict ;      # delete entire dictionary  
  
print "dict['Age']: ", dict['Age']  
  
print "dict['School']: ", dict['School']
```

This produces the following result. Note that an exception is raised because after del dict dictionary does not exist any more –

```
dict['Age']:
```

Traceback (most recent call last):

File "test.py", line 8, in <module>

```
print "dict['Age']: ", dict['Age'];
```

TypeError: 'type' object is unsubscriptable

Note – del() method is discussed in subsequent section.

## **Properties of Dictionary Keys**

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys –

(a) More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins. For example –

Live Demo



```
#!/usr/bin/python
```

```
dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'}  
print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result –

```
dict['Name']: Manni
```

(b) Keys must be immutable. Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed. Following is a simple example –

Live Demo

```
#!/usr/bin/python
```

```
dict = {'Name': 'Zara', 'Age': 7}  
print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result –

Traceback (most recent call last):

File "test.py", line 3, in <module>

```
dict = {'Name': 'Zara', 'Age': 7};
```

TypeError: unhashable type: 'list'

Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates –

Live Demo

```
#!/usr/bin/python
```

```
tup1 = (12, 34.56);
```

```
tup2 = ('abc', 'xyz');
```

```
# Following action is not valid for tuples
```

```
# tup1[0] = 100;
```

```
# So let's create a new tuple as follows
```

```
tup3 = tup1 + tup2;
```

```
print tup3;
```

When the above code is executed, it produces the following result –

```
(12, 34.56, 'abc', 'xyz')
```

### Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the del statement. For example –

### Live Demo

```
#!/usr/bin/python
```

```
tup = ('physics', 'chemistry', 1997, 2000);
```

```
print tup;
```

```
del tup;
```

```
print "After deleting tup : ";
```

```
print tup;
```

This produces the following result. Note an exception raised, this is because after `del tup` tuple does not exist any more –

```
('physics', 'chemistry', 1997, 2000)
```

After deleting `tup` :

Traceback (most recent call last):

```
File "test.py", line 9, in <module>
```

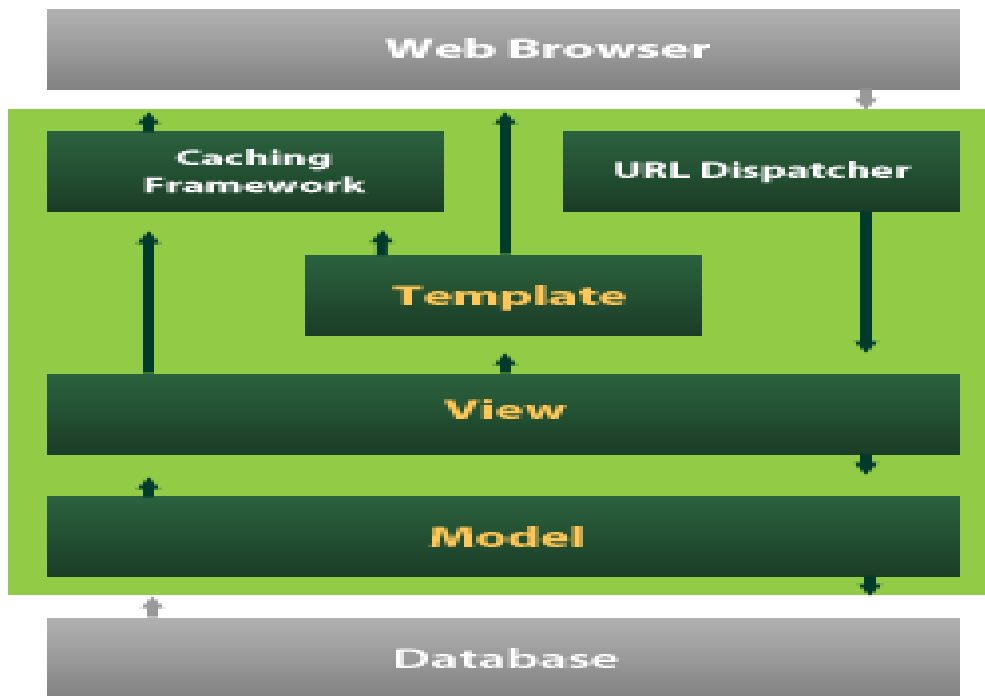
```
    print tup;
```

NameError: name 'tup' is not defined

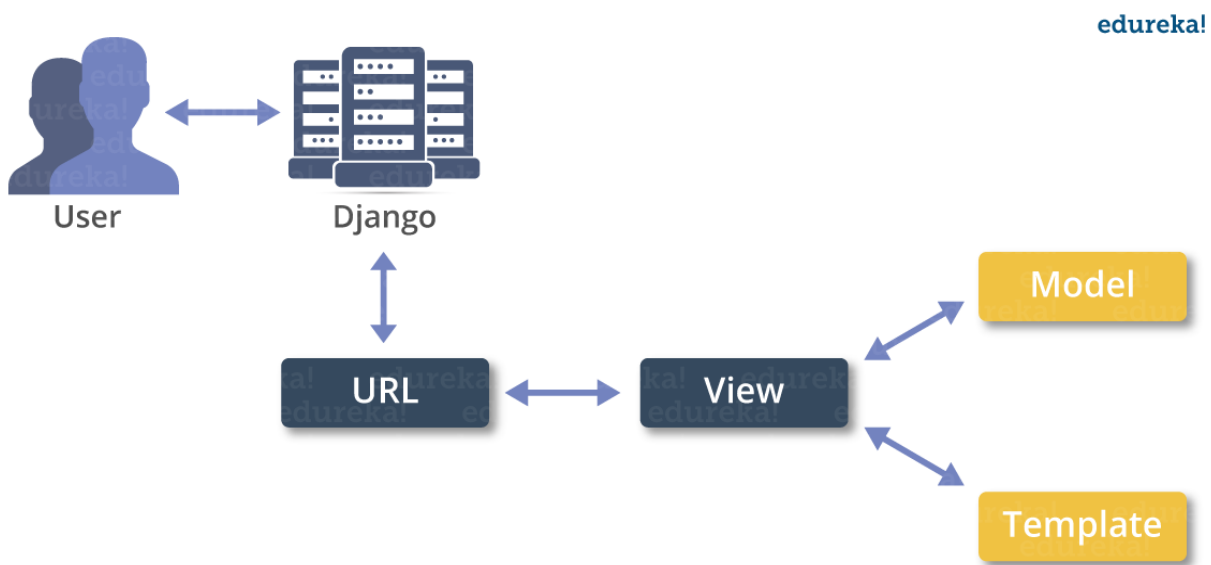
## 5.3 DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.



Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models



## Create a Project

Whether you are on Windows or Linux, just get a terminal or a cmd prompt and navigate to the place you want your project to be created, then use this code –

```
$ django-admin startproject myproject
```

This will create a "myproject" folder with the following structure –

```
myproject/
```

```
    manage.py
```

```
myproject/
```

```
    __init__.py
```

```
    settings.py
```

```
    urls.py
```

```
    wsgi.py
```

### The Project Structure

The “myproject” folder is just your project container, it actually contains two elements –

manage.py – This file is kind of your project local django-admin for interacting with your project via command line (start the development server, sync db...). To get a full list of command accessible via manage.py you can use the code –

```
$ python manage.py help
```

The “myproject” subfolder – This folder is the actual python package of your project. It contains four files –

\_\_init\_\_.py – Just for python, treat this folder as package.

settings.py – As the name indicates, your project settings.

urls.py – All links of your project and the function to call. A kind of ToC of your project.

wsgi.py – If you need to deploy your project over WSGI.

## Setting Up Your Project

Your project is set up in the subfolder myproject/settings.py. Following are some important options you might need to set –

```
DEBUG = True
```

This option lets you set if your project is in debug mode or not. Debug mode lets you get more information about your project's error. Never set it to 'True' for a live project. However, this has to be set to 'True' if you want the Django light server to serve static files. Do it only in the development mode.

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.sqlite3',  
        'NAME': 'database.sql',  
        'USER': '',  
        'PASSWORD': '',  
        'HOST': '',  
        'PORT': '',  
    }  
}
```



Database is set in the 'Database' dictionary. The example above is for SQLite engine. As stated earlier, Django also supports –

MySQL (django.db.backends.mysql)

PostgreSQL (django.db.backends.postgresql\_psycopg2)

Oracle (django.db.backends.oracle) and NoSQL DB

MongoDB (django\_mongodb\_engine)

Before setting any new engine, make sure you have the correct db driver installed.

You can also set others options like: TIME\_ZONE, LANGUAGE\_CODE, TEMPLATE...

Now that your project is created and configured make sure it's working –

```
$ python manage.py runserver
```

You will get something like the following on running the above code –

Validating models...

0 errors found

September 03, 2015 - 11:41:50

Django version 1.6.11, using settings 'myproject.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CONTROL-C.

A project is a sum of many applications. Every application has an objective and can be reused into another project, like the contact form on a website can be an application, and can be reused for others. See it as a module of your project.

## **Create an Application**

We assume you are in your project folder. In our main “myproject” folder, the same folder then manage.py –

```
$ python manage.py startapp myapp
```

You just created myapp application and like project, Django create a “myapp” folder with the application structure –

myapp/

\_\_init\_\_.py

admin.py

models.py

tests.py

views.py

\_\_init\_\_.py – Just to make sure python handles this folder as a package.

admin.py – This file helps you make the app modifiable in the admin interface.

models.py – This is where all the application models are stored.

tests.py – This is where your unit tests are.

views.py – This is where your application views are.

## Get the Project to Know About Your Application

At this stage we have our "myapp" application, now we need to register it with our Django project "myproject". To do so, update `INSTALLED_APPS` tuple in the `settings.py` file of your project (add your app name) –

```
INSTALLED_APPS = (  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'myapp',  
)
```

Creating forms in Django, is really similar to creating a model. Here again, we just need to inherit from Django class and the class attributes will be the form fields. Let's add a `forms.py` file in `myapp` folder to contain our app forms. We will create a login form.

myapp/forms.py

```
#-*- coding: utf-8 -*-
```

```
from django import forms
```

```
class LoginForm(forms.Form):
```

```
    user = forms.CharField(max_length = 100)
```

```
    password = forms.CharField(widget = forms.PasswordInput())
```

As seen above, the field type can take "widget" argument for html rendering; in our case, we want the password to be hidden, not displayed. Many others widget are present in Django: DateInput for dates, CheckboxInput for checkboxes, etc.

## Using Form in a View

There are two kinds of HTTP requests, GET and POST. In Django, the request object passed as parameter to your view has an attribute called "method" where the type of the request is set, and all data passed via POST can be accessed via the request.POST dictionary.

Let's create a login view in our myapp/views.py –

```
#-*- coding: utf-8 -*-

from myapp.forms import LoginForm

def login(request):

    username = "not logged in"

    if request.method == "POST":

        #Get the posted form

        MyLoginForm = LoginForm(request.POST)

        if MyLoginForm.is_valid():

            username = MyLoginForm.cleaned_data['username']

        else:

            MyLoginForm = LoginForm()

    return render(request, 'loggedin.html', {"username" : username})
```

The view will display the result of the login form posted through the loggedin.html. To test it, we will first need the login form template. Let's call it login.html.

```
<html>
```

```
<body>
```

```
<form name = "form" action = "{% url 'myapp.views.login' %}"  
      method = "POST" >{% csrf_token %}
```

```
<div style = "max-width:470px;">
```

```
<center>
```

```
<input type = "text" style = "margin-left:20%;"
```

```
      placeholder = "Identifiant" name = "username" />
```

```
</center>
```

```
</div>
```

```
<br>
```

```
<div style = "max-width:470px;">
```

```
<center>
```

```
<input type = "password" style = "margin-left:20%;"
```

```
placeholder = "password" name = "password" />
```

```
</center>
```

```
</div>
```

```
<br>
```

```
<div style = "max-width:470px;">
```

```
<center>
```

```
<button style = "border:0px; background-color:#4285F4;  
margin-top:8%;
```

```
height:35px; width:80%;margin-left:19%;" type =  
"submit"
```

```
value = "Login" >
```

```
<strong>Login</strong>
```

```
</button>
```



```
</center>

</div>

</form>

</body>

</html>
```

The template will display a login form and post the result to our login view above. You have probably noticed the tag in the template, which is just to prevent Cross-site Request Forgery (CSRF) attack on your site.

```
{% csrf_token %}
```

Once we have the login template, we need the loggedin.html template that will be rendered after form treatment.

```
<html>

<body>
```

```
        You are : <strong>{{username}}</strong>

</body>

</html>
```

Now, we just need our pair of URLs to get started: `myapp/urls.py`

```
from django.conf.urls import patterns, url

from django.views.generic import TemplateView

urlpatterns = patterns('myapp.views',

    url(r'^connection/', TemplateView.as_view(template_name =
'login.html')),

    url(r'^login/', 'login', name = 'login'))
```

When accessing `"/myapp/connection"`, we will get the following `login.html` template rendered –

## Setting Up Sessions

In Django, enabling session is done in your project `settings.py`, by adding some lines to the `MIDDLEWARE_CLASSES` and the `INSTALLED_APPS` options. This should be done while creating the

project, but it's always good to know, so `MIDDLEWARE_CLASSES` should have –

```
'django.contrib.sessions.middleware.SessionMiddleware'
```

And `INSTALLED_APPS` should have –

```
'django.contrib.sessions'
```

By default, Django saves session information in database (`django_session` table or collection), but you can configure the engine to store information using other ways like: in file or in cache.

When session is enabled, every request (first argument of any view in Django) has a `session` (dict) attribute.

Let's create a simple sample to see how to create and save sessions. We have built a simple login system before (see Django form processing chapter and Django Cookies Handling chapter). Let us save the username in a cookie so, if not signed out, when accessing our login page you won't see the login form. Basically, let's make our login system we used in Django Cookies handling more secure, by saving cookies server side.

For this, first lets change our login view to save our username cookie server side –

```
def login(request):  
  
    username = 'not logged in'  
  
  
    if request.method == 'POST':  
  
        MyLoginForm = LoginForm(request.POST)  
  
  
        if MyLoginForm.is_valid():  
  
            username = MyLoginForm.cleaned_data['username']  
  
            request.session['username'] = username  
  
        else:  
  
            MyLoginForm = LoginForm()  
  
  
    return render(request, 'loggedin.html', {"username" : username})
```

Then let us create formView view for the login form, where we won't display the form if cookie is set –

```
def formView(request):

    if request.session.has_key('username'):

        username = request.session['username']

        return render(request, 'loggedin.html', {"username" : username})

    else:

        return render(request, 'login.html', {})
```

Now let us change the url.py file to change the url so it pairs with our new view –

```
from django.conf.urls import patterns, url

from django.views.generic import TemplateView
```

```
urlpatterns = patterns('myapp.views',

    url(r'^connection/', 'formView', name = 'loginform'),

    url(r'^login/', 'login', name = 'login'))
```

When accessing /myapp/connection, you will get to see the following page

## 6. SAMPLE CODE

```
def filedata(request):  
    if request.method == "GET":  
        file = request.GET.get('id')  
        try:  
            print("file", file)  
            head, fileName = os.path.split(file)  
            fPath = settings.MEDIA_ROOT + '/' + 'files/pdfs' + '/' +  
fileName  
            f = open(fPath)  
            loc = 0  
            wordcount = 0  
            chrcount = 0  
            cmntcount = 0  
            classcount = 0  
            for line in f:  
                loc = loc + 1  
                chrcount = chrcount + len(line)  
                word = line.split(' ')  
                wordcount = wordcount + len(word)  
                if line.startswith('#'):  
                    cmntcount = cmntcount + 1  
                if line.startswith('class'):  
                    classcount = classcount + 1  
            f = open(fPath)  
            fd = f.read()  
            stats = os.stat(fPath)
```

```

s = datetime.fromtimestamp(stats.st_atime)
s1 = datetime.fromtimestamp(stats.st_mtime)

size1 = stats.st_size

print("file size in bytes:", size1)

print('class-count', classcount)

with open(fPath) as f:

    tree = ast.parse(f.read())

    func = sum(isinstance(exp, ast.FunctionDef) for exp in
tree.body)

f = open(fPath)

lpcountif = lpcountelif = lpcountelse = lpcountfor
=lpcountwhile=countwith=countexcept=countfinally=0

line = f.readlines()

for x in line:

    if re.search('if ', x):

        lpcountif = lpcountif + 1

    if re.search('elif ', x):

        lpcountelif = lpcountelif + 1

    if re.search('else ', x):

        lpcountelse = lpcountelse + 1

    if re.search('for ', x):

        lpcountfor = lpcountfor + 1

    if re.search('while ', x):

        lpcountwhile = lpcountwhile + 1

    if re.search('with ', x):

        countwith=countwith+1

    if re.search('except:', x):

```

```

        countexcept=countexcept+1

    if re.search('finally:', x):

        countfinally=countfinally+1


cmlx=lpcountif+lpcountelif+lpcountelse+lpcountwhile+lpcountfor+1+countexcept+countfinally

    if cmlx<=5 and cmlx>=1:

        perfrmance="Low-simple block"

        rank="A"

    elif cmlx>=6 and cmlx<=10:

        perfrmance="low"

        rank="B"

    elif cmlx>=11 and cmlx<=20:

        perfrmance = "Moderate"

        rank = "C"

    elif cmlx>=21 and cmlx<=30:

        perfrmance = "More than Moderate"

        rank = "D"


    elif cmlx>=31 and cmlx<=40:

        perfrmance = "high"

        rank = "E"

    else:

        perfrmance="very high"

        rank='F'

```



```

        message = {"filename": f.name, "lines": loc, "words":
wordcount, "charecters": chrcount, "content": fd,

                    "functionscount": func, "commentlinescount":
cmntcount, "lastmodifiedtime": sl,

                    "lastaccessedtime": s, "classescount":
classcount, "ifloop": lpcountif, "elifloop": lpcountelif,

                    "elseloop":
lpcountelse, "filesize": size1, "countexcept": countexcept, "countfinally":
countfinally, "forloop":
lpcountfor, "whileloop": lpcountwhile, "cmplx": cmplx, "rank": rank, "perf": p
erfrmance}

        return render(request, "user/userfiledata.html",
{"message": message})

    except Exception as e:

        print('Exception is ', str(e))

        messages.success(request, 'Invalid Details')

    return render(request, 'user/userfiledata.html')

```

## **7. TESTING**

### **7.1 Unit testing:**

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

### **7.2 Integration testing:**

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

### **7.3 Functional test:**

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

Functional testing is centered on the following items:

Valid Input: identified classes of valid input must be accepted.

Invalid Input: identified classes of invalid input must be rejected.

Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

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

### **7.4 System Test:**

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

### **7.5 White Box Testing:**

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

### **7.6 Black Box Testing:**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

### **7.7 Acceptance Testing:**

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

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

## 7.8 TEST CASES

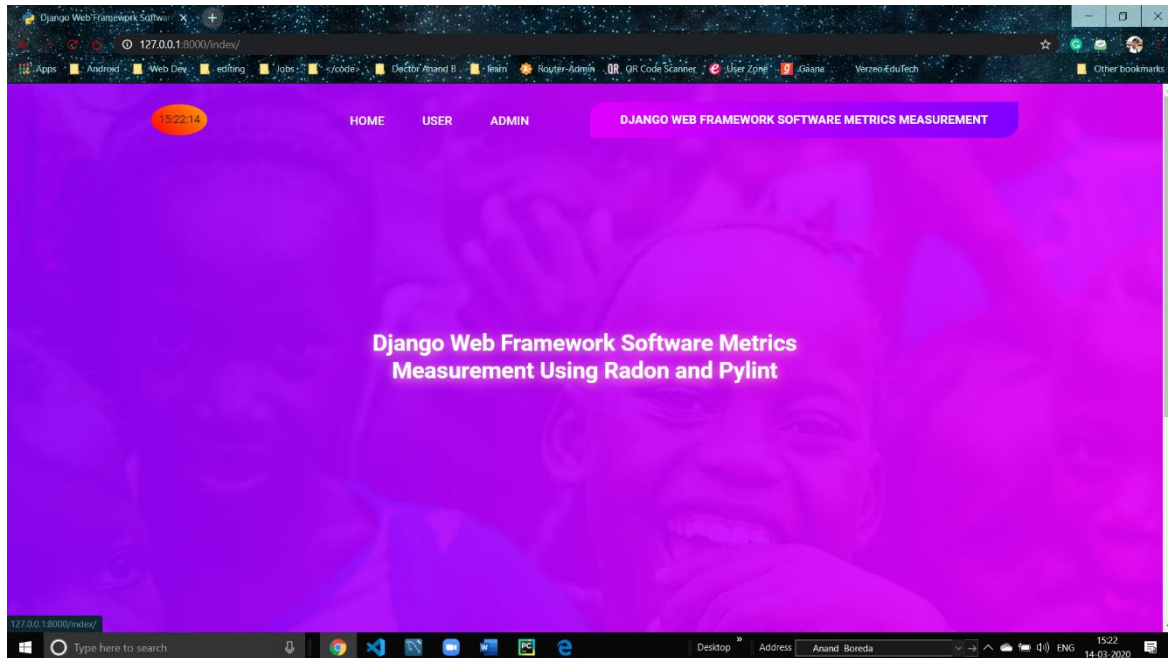
S.no	Test Case	Excepted Result	Result	Remarks (IF Fails)
1	user registration	If user filled all form fields should be registered successfully.	Pass	If user is filled the form fields.
2	ADMIN	user rights will be approved here.	Pass	If user is not registered.
3	user LOGIN	If user name and password is correct then it will be getting valid page.	Pass	If user name or password is not correct.
4	user upload code file	If user is correct then it will be getting valid page.	Pass	If user is not correct.
5	Admin view	Admin can view user data. if admin id and password is correct.it will show valid page.	Pass	If admin is not correct.
6	User view	If user is logged in then show analysis	Pass	If user is not logged in.

***fig 7.8 Table showing test cases***

The above table shows all the tests that have been conducted and results of the tests is indicated as pass if it qualifies and fail if case not

## 8. OUTPUT SCREEN

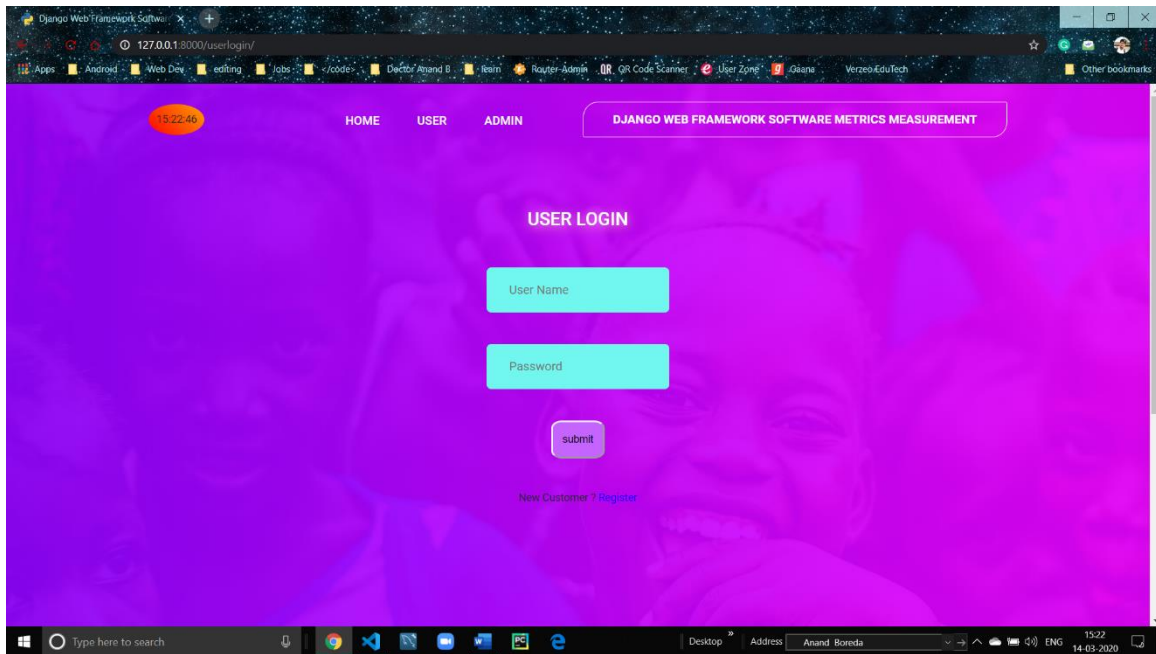
### 8.1 Home Page:



***fig 8.1 Home Page***

*Home Page for all users. Whenever user enters the website, this page will show up*

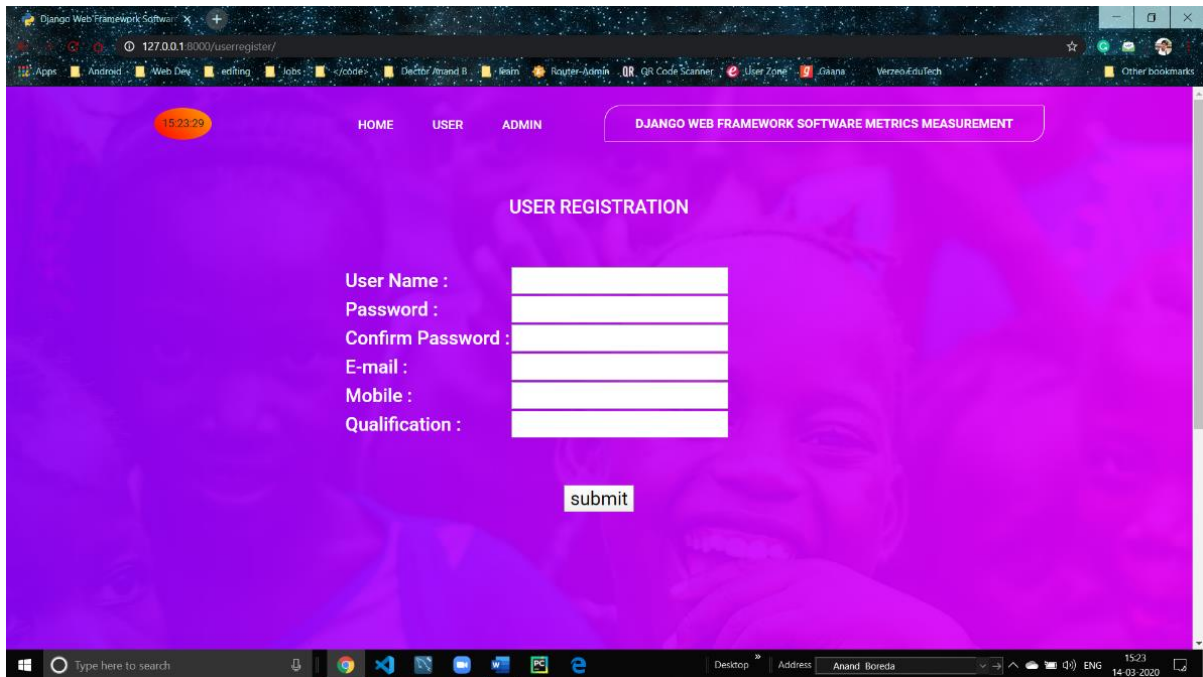
## 8.2 User Login Page:



**fig 8.2 user login Page**

*An already existing authorized user who is approved by the admin can login and access database by entering his credentials in this page. For new users, there's a register button below.*

### 8.3 New User Registration Page:



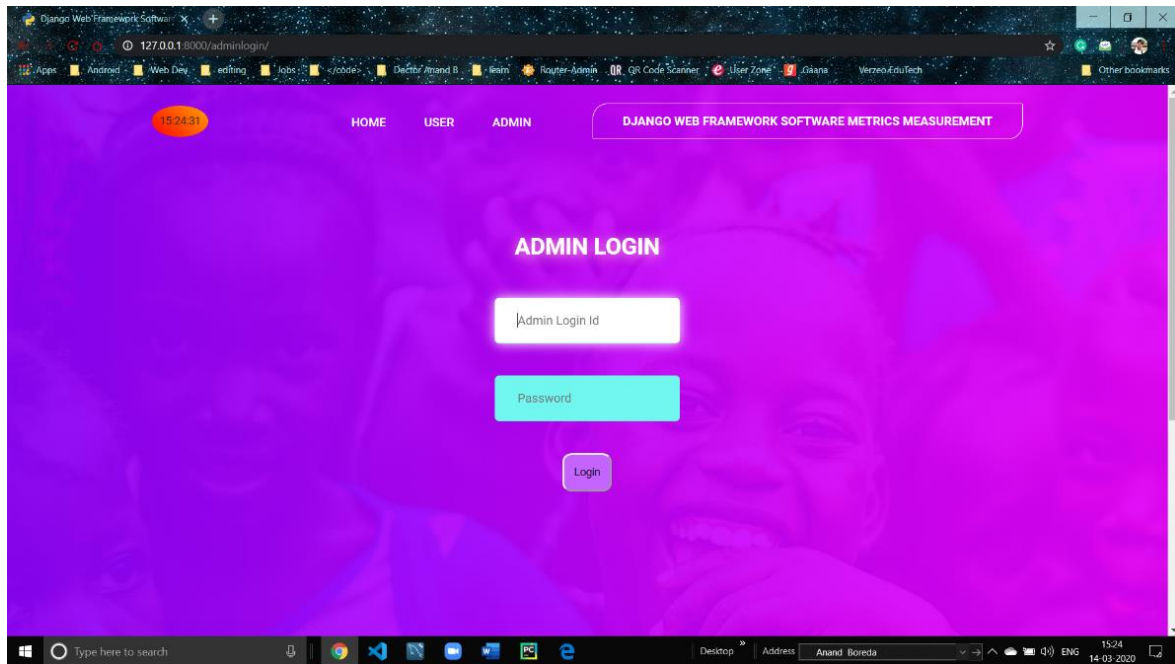
The screenshot shows a web browser window with the URL `127.0.0.1:8000/userregister/`. The page has a purple background with a faint image of a person. At the top, there is a navigation bar with a clock showing 15:23:29, links for HOME, USER, and ADMIN, and a button labeled DJANGO WEB FRAMEWORK SOFTWARE METRICS MEASUREMENT. The main heading is USER REGISTRATION. Below it, there are input fields for User Name, Password, Confirm Password, E-mail, Mobile, and Qualification. A submit button is located at the bottom of the form. The browser's taskbar at the bottom shows various application icons and the system clock indicating 15:23 on 14-03-2020.

**fig 8.3 New User Registration Page**

*Any new member visiting the website can register himself as a user. Once registered, he must wait for approval by admin before he can access the database.*



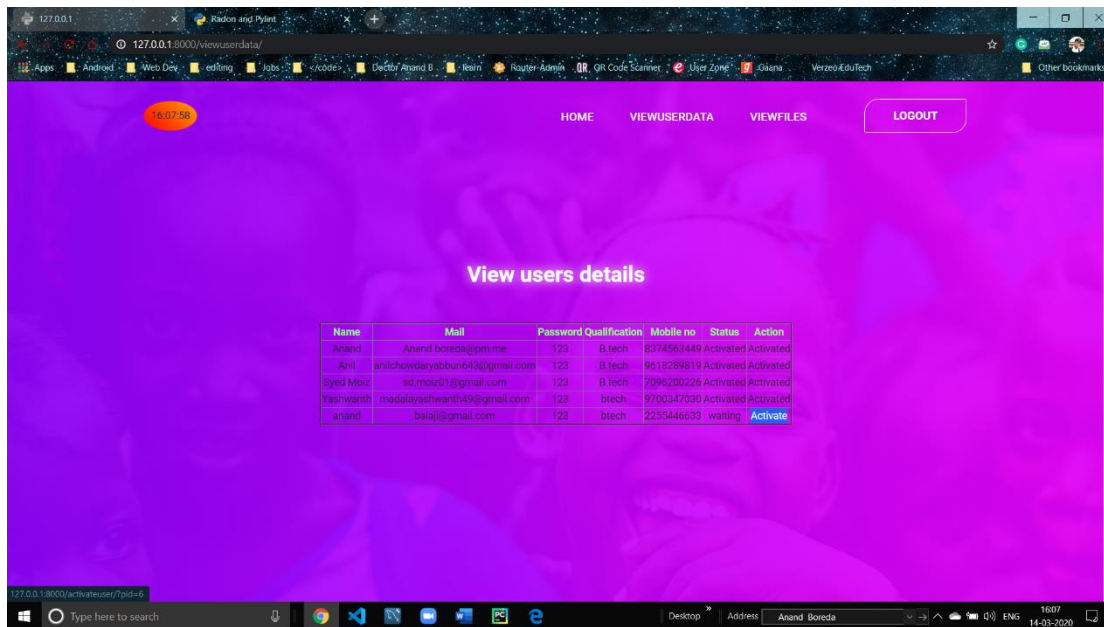
## 8.4 Admin Login Page:



**fig 8.4 Admin Login Page**

*The Admin can log in to the web application with his credentials, id and a secure password*

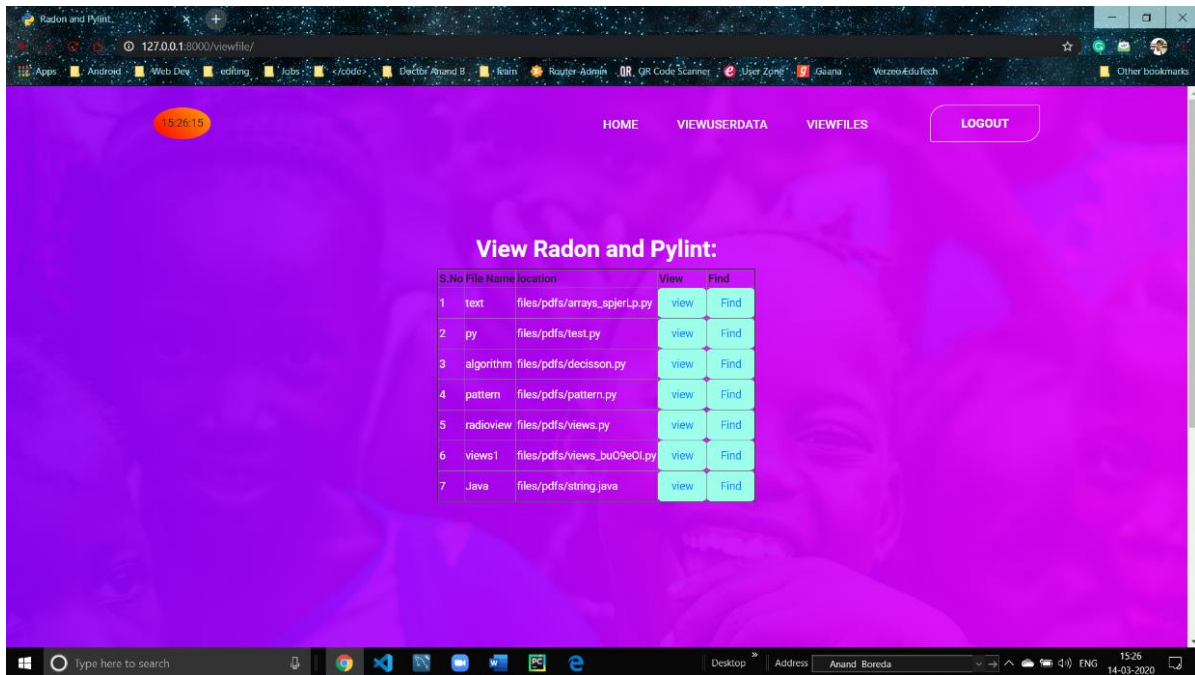
## 8.5 User Data Page:



**fig 8.5 User Details Page**

The Admin can look user's credentials and authorize new users to gain access to database

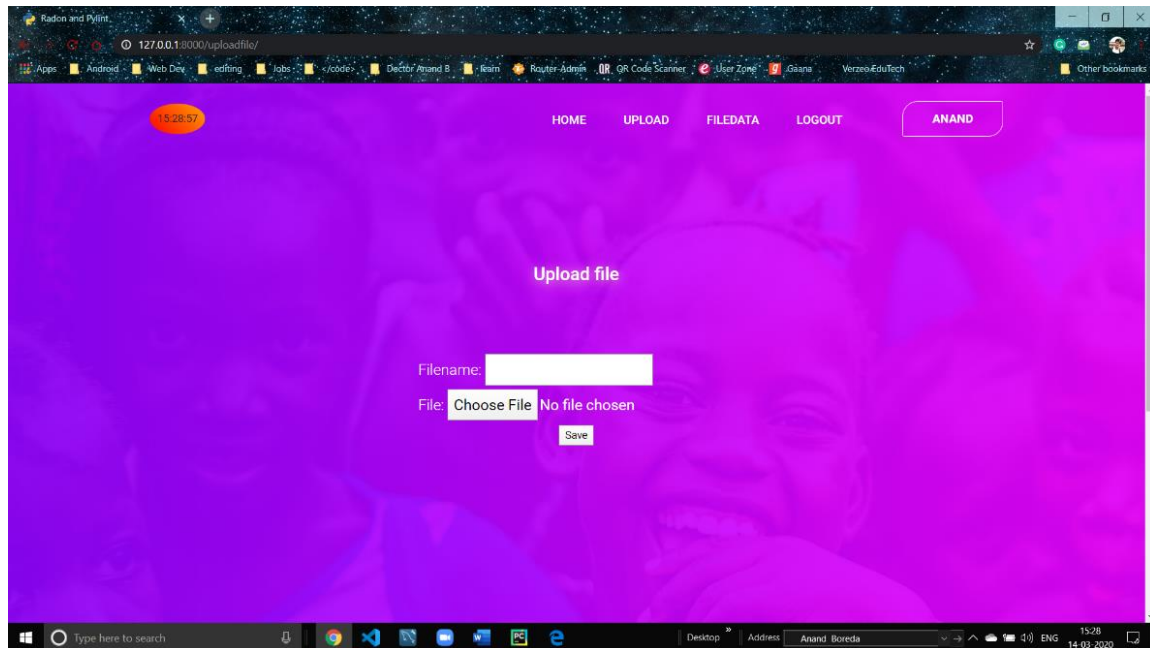
## 8.6 File data in user session:



**fig 8.6 Code Files Page in admin session**

The Admin can look up the files or documents uploaded by the users to database.

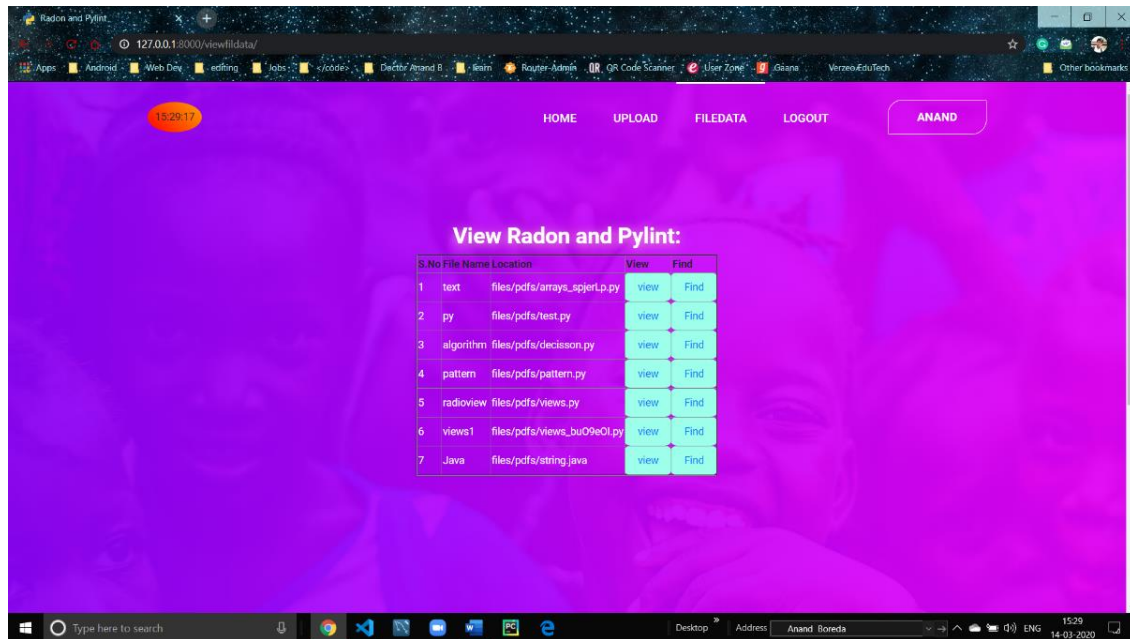
## 8.7 New File Upload Page:



***fig 8.7 File Upload Page***

*Authorized users can upload their code file or document to the database with a proper name attached to it.*

## 8.8 File data in admin session:



**fig 8.8 Code files page in user session**

All the authorized users can access files on the database. He can view the source code by clicking “view” and see the metrics of the file by clicking “Find”

## 8.9 Analytics Page in user session:



File Name	C:\Users\anand\OneDrive\Documents\django-projects\new git merged file\Django-web-framework-software-metrics-measurement-using-Radon-and-Pylint\Code\Software\metrics\media\files\pdfs\arrays_sjperLp.py
LOC	4
Word count of file	13
Characters count of file	68
Number of functions	0
Commentlines count of file	1
Last-modified time of file	March 14, 2020, 3:13 p.m.
Last-accessed time of file	March 14, 2020, 3:13 p.m.
Classes count of file	0
Size of file in bytes	71
If loop count of file	0
elif loop count of file	0
else loop count of file	0
While loop count of file	0
for loop count of file	1
Count except of file	0
Count finally of file	0
Cyclomatic complexity count of file	2
Cyclomatic Complexity rank of file	A
Cyclomatic Complexity performance of file	Low-simple block

**fig 8.9 Analytics Page in user session**

Whenever a user requests to see the metrics of a specific file, he will be redirected to new page containing all the details and metrics of that specific document

## 8.10 Analytics in Admin session:

File Name	C:\Users\anand\OneDrive\Documents\django-projects\new git merged file\ Django-web-framework-software-metrics-measurement-using-Radon-and-Pylint\Code\Software\metrics\media\files\pdfs\views_AoiLtaM.py
LOC	317
Word count of file	3834
Characters count of file	12056
Number of functions	18
Commentlines count of file	2
Last-modified time of file	March 14, 2020, 3:31 p.m.
Last-accessed time of file	March 14, 2020, 5:03 p.m.
Classes count of file	0
Size of file in bytes	12373
If loop count of file	44
elif loop count of file	12
else loop count of file	7
While loop count of file	2
for loop count of file	12
Count except of file	1
Count finally of file	1
Cyclomatic complexity count of file	80
Cyclomatic Complexity rank of file	F
Cyclomatic Complexity performance of file	very high

**fig 8.10 Analytics Page in Admin session**

Whenever the Admin requests to see the metrics of a specific file, he will be redirected to new page containing all the details and metrics of that specific document

## 9. CONCLUSION

In conclusion, this paper has measure Django Web Framework code quality metrics. Django option in the main directory has 2,200 lines of code, Cyclomatic Complexity score is 16.375 considered as moderate complexity, and 6.69/10 by the Pylint score. With the same method, other web frameworks can be measured too.



## 10. FUTURE ENHANCEMENTS

Currently in this version of the project, users can register and only those who are approved and authorized by the admin can view file data and upload their code file to analyze the metrics and complexity of the code. In future we are planning to provide an API (application programming interface). There is an idea to make this project ubiquitous by launching an API. We can actually do that using the Django REST Framework. Using the API, other authorized organizations can get access to our functionality what we are having now.

## REFERENCES

- [1] D. P. Pop and A. Altar. (2014). Designing an MVC model for rapid web application development. *Procedia Engineering*, 69, 1172-1179.
- [2] M. Aniche, G. Bavota, C. Treude, M. A. Gerosa, and A. van Deursen, "Code smells for Model-View-Controller architectures," *Empir. Softw. Eng.*, vol. 23, no. 4, pp. 2121-2157, 2018.
- [3] S. Cass. (2017). The 2017 top programming languages. <https://spectrum.ieee.org/computing/software/the-2017-topprogramming-languages>. Accessed 18 October 2017.
- [4] A. Pelme. (2014). Evaluation of a web application architecture. (Dissertation). Retrieved from <http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-107011>
- [5] R. Brown. (2015). Django vs. flask vs. pyramid: Choosing a python web framework. *Recuperado el*, 31.
- [6] P. Vogel, T. Klooster, V. Andrikopoulos, and M. Lungu, "A LowEffort Analytics Platform for Visualizing Evolving Flask-Based Python Web Services," in *2017 IEEE Working Conference on Software Visualization (VISSOFT)*, 2017, pp. 109-113.
- [7] Anonym. Tornado Python framework. <http://www.tornadoweb.org/en/stable/>
- [8] Hellkamp, M. (2017). Bottle: Python Web Framework. Available: <https://bottlepy.org/docs/dev/>. Accessed 28 October 2017.
- [9] S. Dasgupta and S. Hooshangi, "Code Quality: Examining the Efficacy of Automated Tools," *AMCIS 2017 Proc.*, Aug. 2017.
- [10] G. Farah, J. S. Tejada, and D. Correal, "OpenHub: A Scalable Architecture for the Analysis of Software Quality Attributes," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, 2014, pp. 420-423.
- [11] Y. Saleh. (2017). How does the django framework in python overcome the frameworks in PHP? <https://www.egrovesys.com/blog/how-does-the-django-framework-inpython-overcome-the-frameworks-in-php/> Accessed 24 November 2017.

- [12] M. P. Malhotra and M. K. Shah. Python Based Software for Calculating Cyclomatic Complexity. (2015). International Journal of Innovative Science, Engineering and Technology, 2(3):546–549, March 2015.
- [13] M. del Pilar Salas-Zárate, G. Alor-Hernández, R. Valencia-García, L. Rodríguez-Mazahua, A. Rodríguez-González, and J. L. L. Cuadrado (2015). Analyzing best practices on Web development frameworks: The lift approach. Science of Computer Programming, 102, 1-19.
- [14] N. Gift. (2017). Writing clean, testable, high-quality code in python. <https://www.ibm.com/developerworks/aix/library/au-cleancode/> Accessed 19 December 2017.
- [15] C. Thirumalai, R. R. Shridharshan, and L. R. Reynold, “An assessment of halstead and COCOMO model for effort estimation,” in 2017 Innovations in Power and Advanced Computing Technologies (i-PACT), 2017, pp. 1–4.
- [16] Anonymous. (2017). Radon. <https://pypi.python.org/pypi/radon> Accessed 30 November 2017.
- [17] A. Calleja, J. Tapiador and Caballero, J. (2016, September). A look into 30 years of malware development from a software metrics perspective. In International Symposium on Research in Attacks, Intrusions, and Defenses, pp. 325-345. Springer, Cham.
- [18] T. J. McCabe. (1976). A complexity measure. IEEE Transactions on Software Engineering, SE-2(4):308–320, Dec 1976
- [19] R. Smith, T. Tang, J. Warren and S. Rixner. (2017, June). An Automated System for Interactively Learning Software Testing. In Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education, pp. 98-103.
- [20] Anonymous. Pylint. <https://www.pylint.org> Accessed 12 December 2017.
- T. J. McCabe. A complexity measure. IEEE Transactions on Software Engineering, SE-2(4):308–320, Dec 1976.