MINI PROJECT REPORT

**On**

**CLOUD BASED BUSS PASS SYSTEM**

**SUBMITTED BY-**

**AAYUSHI RAI (171510003)**

**PRANJUL SINGHAL (171510040)**

**SAKSHAM JOHRI (171510046)**

Department of Computer Engineering & Applications

**Institute of Engineering & Technology**



**GLA UNIVERSITY**

**MATHURA-281406, INDIA**

**2019**

**SYNOPSIS**

**Student Information:**

|  |  |
| --- | --- |
| Name: Saksham Johri | University Roll. No: 171510046 |
| Mobile: +91 8979 808914 | Email: [sakshamjohri@gmail.com](mailto:sakshamjohri@gmail.com) |

**Project Information:**

|  |  |
| --- | --- |
| Title Of Project/Training/Task | Python |
| Role & Responsibility | - |
| Technical Details | Hardware Requirements: Computer System with minimum 4GB of RAM  Software Requirements: Python 3.7.3.Shell /  JetBrains PyCharm Community Edition 2019.2 x64 |
| Training Implementation  Details | Implemented YES  Not Implemented NO |
| Training Period | Start Date:04/06/2019  End Date:16/07/2019  Duration Of Training (In Weeks): 6 WEEKS |

**Summary of the Training Work:**

|  |
| --- |
| During Training, I learnt about Python language. Python is an interpreted, high-level general-purpose programming language.  Python is a simple and easy to learn because of its clear syntax and readability that’s why it reduces the cost of program maintenance  We can use Python for developing desktop GUI application, websites and web applications. |

**ACKNOWLEDGEMENT**

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

 I respect and thank **Mr. Dheeraj Singh** for providing me an opportunity to do the project work and giving me all support and guidance which made me complete the project duly. I am extremely thankful to him for providing such a nice support and guidance, although he had busy schedule managing the corporate affairs. I owe my deep gratitude to our project guide **Mr. Dheeraj Singh** who took keen interest on my project work and guided me all along, till the completion of our project work by providing all the necessary information for developing a good project. After doing this project I can confidently say that this experience has not only enriched me with technical knowledge but also has unparsed the maturity of thought and vision. The attributes required in being a successful professional

Saksham Johri

(171510046)

**Department of computer Engineering and Applications**

**GLA University, Mathura**

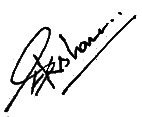
**17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,**

**Mathura – 281406**



**Declaration**

I hereby declare that the work which is being presented in the Summer Training “**Developing GUI Calculator using Tkinter in python”,** in partial fulfilment of the requirements for Summer Training viva voice, is an authentic record of my own work carried under the supervision of **Mr. Dheeraj Singh.**



Signature of Candidate:

Name of Candidate: Saksham Johri

Roll. No: 171510046

Course: B.Tech (CCV)

Year: III

Semester: V

**ABSTRACT**

Introduction Here by using GUI (graphical user interface) we have made calculator, for understanding the basic function of the GUI and programming we have made the calculator which will perform the basic function mathematics like addition, multiplication, subtraction, division and some more which helped lots us to understand tools in programming.

Contents

1. Introduction ……………………………………………………………………….. 1
   1. Purpose …..……………………………………………………………….....1
   2. Objective ….………………………………………………………………...1
   3. Problem Statement .………………………………………………………....1
2. Introduction to python …………………………………………………………….. 2
3. Python Data types .……………………………………………………………….... 3
4. Python Oops Concept .……………………………………………………………...8
5. Python Applications ……………………………………………………………….11
6. Introduction to Tkinter with Python ….…………………………………………...12
   1. Designing User Interface ..………………………………………………....12
   2. What is Tkinter .…………………………………………………………....12
   3. Why Tkinter ..………………………………………………………………13
   4. Fundamentals of Tkinter ..………………………………………………….14
7. Source Code …….………………………………………………………………….15
8. Use Case Diagram and Output ….………………………………………………....18
   1. Use Case Diagram of Calculator ...…………………………………………18
   2. Look of my Calulator ...…………………………………………………….19
9. Conclusion & Future Scope ……………………………………………………......20
10. Bibliography ……………………………………………………………………......21

**CHAPTER-1**

**INTRODUCTION**

This is a simple graphical user interfaced (GUI) Python application designed using tkinter package to perform basic arithmetic calculations such as addition, subtraction, multiplication, and division of two numbers through the mouse click event. Copy an icon image ‘favicon.ico’ to the working directory to change the default icon in the title bar of the application.

**1.1 Purpose**

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter outputs the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

**1.2 Objective**

The main objective is to make calculation easy and fast.

* 1. **Problem Statement**

To develop an algorithm for calculator using python programming and graphical user interface to make the GUI in such a way that user can give input and he will get the output in the graphical or algebraic form.

**CHAPTER-2**

**INTRODUCTION TO PYTHON**

* Python is an interpreted high-level programming language. It has advantages of both scripting and programming languages.
* It is very useful for rapid application development
* It is easy to learn when comparing with other programming languages and has a design philosophy that emphasizes code readability, mainly using indentation.
* The language was originally created by Guido Van Rossum and first released on 1991.
* Nowadays, Python is gaining more attention as it's great for data analysis, artificial intelligence and scientific computing
* It does not need a compiler to run the application. It’s basically an interpreter language.

**Chapter- 3**

**PYTHON DATA TYPES**

Variables can hold values of different data types. Python is a dynamically typed language hence we need not define the type of the variable while declaring it. The interpreter implicitly binds the value with its type.

Python provides us the type() function which returns the type of the variable passed.

**For** **Examples**:

a=10

b="Hi Python"

c = 10.5

print(type(a))

print(type(b))

print(type(c))

**Output:**

<type 'int'> <type ‘str’>

<type 'float'>

**Standard Data Type**

Python provides various standard data types that define the storage method on each of them. The data types defined in Python are given below.

1. Numbers

2. String

3. List

4. Tuple

5. Dictionary

**1. Numbers**

Number stores numeric values. For example;

a = 3, b = 5

Python supports 4 types of numeric data.

1. int (signed integers like 10, 2, 29, etc.)

2. long (long integers used for a higher range of values like 908090800L, -0x1929292L, etc.)

3. float (float is used to store floating point numbers like 1.9, 9.902, 15.2, etc.)

4. complex (complex numbers like 2.14j, 2.0 + 2.3j, etc.)

A complex number contains an ordered pair, i.e., x + iy where x and y denote the real and imaginary parts respectively.

**2. String**

The string can be defined as the sequence of characters represented in the quotation marks. In python, we can use single, double, or triple quotes to define a string.

String handling in python is a straightforward task since there are various inbuilt functions and operators provided.

In String, the operator + is used to concatenate and \* is used as repetition operator.

**For Example -**

str1 = 'hello'

str2 = ' how are you'

print (str1[0:2])

print (str1[4])

print (str1\*2)

print (str1 + str2)

**Output:**

he

o

hellohello

hello how are you

**3. List**

Lists can contain data of different types. The items stored in the list are separated with a comma (,) and enclosed within square brackets [].

We can use slice [:] operators to access the data of the list. The concatenation operator (+) and repetition operator (\*) works with the list in the same way as they were working with the strings.

Consider the following example.

la = [1, "hi", "python", 2]

print (la[3:]);

print (la[0:2]);

print (la);

print (la + l);

print (la \* 3);

**Output:**

[2]

[1, 'hi']

[1, 'hi', 'python', 2]

[1, 'hi', 'python', 2, 1, 'hi', 'python', 2]

[1, 'hi', 'python', 2, 1, 'hi', 'python', 2, 1, 'hi', 'python', 2]

**4. Tuple**

A tuple is similar to the list in many ways. Like lists, tuples also contain the collection of the items of different data types. The items of the tuple are separated with a comma (,) and enclosed in parentheses ().

A tuple is a read-only data structure as we can't modify the size and value of the items of a tuple.

**Example -**

t = ("hi", "python", 2)

print (t[1:]);

print (t[0:1]);

print (t);

print (t + t);

print (t \* 3);

t[2] = "hi";

Output:

('python', 2)

('hi',)

('hi', 'python', 2)

('hi', 'python', 2, 'hi', 'python', 2)

('hi', 'python', 2, 'hi', 'python', 2, 'hi', 'python', 2)

Traceback (most recent call last):

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

t[2] = "hi";

TypeError: 'tuple' object does not support item assignment

**5. Dictionary**

Dictionary is an ordered set of a key-value pair of items. Key can hold any primitive data type whereas value is an arbitrary Python object.

The items in the dictionary are separated with the comma and enclosed in the curly braces{}.

Consider the following example.

1. d = {1:'Jimmy', 2:'Alex', 3:'john', 4:'mike'};

2. print("1st name is "+d[1]);

3. print("2nd name is "+ d[4]);

4. print (d);

5. print (d.keys());

6. print (d.values());

**Output:**

1st name is Jimmy

2nd name is mike

{1: 'Jimmy', 2: 'Alex', 3: 'john', 4: 'mike'}

[1, 2, 3, 4]

['Jimmy', 'Alex', 'john', 'mike']

**Chapter-4**

**PYTHON OOP’s CONCEPT**

Like other general purpose languages, python is also an object-oriented language since its beginning. Python is an object-oriented programming language. It allows us to develop applications using an Object Oriented approach. In Python, we can easily create and use classes and objects.

Major principles of object-oriented programming system are given below.

o Object

o Class

o Method

o Inheritance

o Polymorphism

o Data Abstraction

o Encapsulation

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Object**

The object is an entity that has state and behaviour. It may be any real-world object like the mouse, keyboard, chair, table, pen, etc.

Everything in Python is an object, and almost everything has attributes and methods. All functions have a built-in attribute.

**Class**

The class can be defined as a collection of objects. It is a logical entity that has some specific attributes and methods. For example: if you have an employee class then it should contain an attribute and method, i.e. an email id, name, age, salary, etc

Syntax

1. class ClassName:

2. <statement-1>

3. .

4. .

5. <statement-N>

**Method**

The method is a function that is associated with an object. In Python, a method is not unique to class instances. Any object type can have methods.

**Inheritance**

Inheritance is the most important aspect of object-oriented programming which simulates the real world concept of inheritance. It specifies that the child object acquires all the properties and behaviours of the parent object.

By using inheritance, we can create a class which uses all the properties and behaviour of another class. The new class is known as a derived class or child class, and the one whose properties are acquired is known as a base class or parent class.

It provides re-usability of the code.

**Polymorphism**

Polymorphism contains two words "poly" and "morphs". Poly means many and Morphs means form, shape. By polymorphism, we understand that one task can be performed in different ways.

For example: You have a class animal, and all animals speak. But they speak differently. Here, the "speak" behaviour is polymorphic in the sense and depends on the animal. So, the abstract "animal" concept does not actually "speak", but specific animals (like dogs and cats) have a concrete implementation of the action "speak".

**Encapsulation**

Encapsulation is also an important aspect of object-oriented programming. It is used to restrict access to methods and variables. In encapsulation, code and data are wrapped together within a single unit from being modified by accident.

Data Abstraction

Data abstraction and encapsulation both are often used as synonyms. Both are nearly synonym because data abstraction is achieved through encapsulation.

Abstraction is used to hide internal details and show only functionalities. Abstracting something means to give names to things so that the name captures the core of what a function or a whole program does.

## Chapter-5

## PYTHON APPLICATIONS

#### 

#### 1) Web Applications

We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML, Email processing request, etc. It also provides Frameworks such as Django, Pyramid, Flask, etc to design and develop web based applications.

#### 2) Desktop GUI Applications

Python provides Tk GUI library to develop user interface in python based application. Some other useful toolkits like wx widgets, etc that are usable on several platforms.

#### 3) Software Development

Python is helpful for software development process. It works as a support language and can be used for build control and management, testing etc.

**Chapter-6**

**INTRODUCTION TO TKINTER WITH PYTHON**

**6.1 Designing User Interfaces**

User interfaces are what allows end users to interact with an application. An application can be excellent, but without a good user interface, it becomes more difficult to use, and less enjoyable. It is thus very important to design good user interfaces. Designing user interface takes place at two different levels: the graphical level and the event level. Graphical elements of a user interface are called widgets. Widgets are basic components like buttons, scrollbars, etc. But user interfaces involve more than a collection of widgets placed in a window. The application must be able to respond to mouse clicks, keyboard actions or system events such as minimizing the window. For this to happen, events must be associated to some pieces of code. This process is called binding. The next two chapters will cover each level in more details, but this chapter will present an overview of Tkinter and explain why it has become the leading GUI toolkit for the Python language.

**6.2 What is Tkinter?**

Tkinter is an open source, portable graphical user interface (GUI) library designed for use in Python scripts. Tkinter relies on the Tk library, the GUI library used by Tcl/Tk. Thus, Tkinter is implemented using multiple layers. Several competing GUI toolkits are available to use with the Python language, namely:

**wxPython :** A wrapper extension for wxWindows, a portable GUI library originally developed for the C++ language. It is the second most popular GUI toolkit for Python since it is considered excellent for complex interface design.

**JPython (Jython) :** Since it is implemented in java, JPython has access to Java GUI libraries, namely SWING and AWT. Recently, JTkinter has been implemented and provides a Tkinter port to JPython using the Java Native

**6.3 Why Tkinter?**

With all the competing GUI toolkits available for the Python language, what makes Tkinter stand out of the rest? Why is it the most popular toolkit for use interface design?

To find the answer, one must look at the advantages that it offers.

1. **Layered design** The layered approach used in designing Tkinter gives Tkinter all of the advantages of the TK library. Therefore, at the time of creation, Tkinter inherited from the benefits of a GUI toolkit that had been given time to mature. This makes early versions of Tkinter a lot more stable and reliable than if it had been rewritten from scratch. Moreover, the conversion from Tcl/Tk to Tkinter is really trivial, so that Tk programmers can learn to use Tkinter very easily.

2. **Accessibility Learning** Tkinter is very intuitive, and therefore quick and painless. The Tkinter implementation hides the detailed and complicated calls in simple, intuitive methods. This is a continuation of the Python way of thinking, since the language excels at quickly building prototypes.

**6.4 Fundamentals Of Tkinter**

Consider the following diagram, it shows how an application actually executes in Tkinter:



**CHAPTER-7**

**SOURCE CODE**







**CHAPTER-8**

**USE CASE DIAGRAM & OUTPUT**



Fig 8.1: Use Case Diagram for Calculator

****

**Fig 8.2: Look of my Calculator**

**CHAPTER-9**

**CONCLUSION**

**9.1 CONCLUSION & FUTURE SCOPE**

The project has been successfully completed by having established the user friendly interface with the help of python GUI tool. It consists of calculator. At the same time there is some scope for improvement in the future. It can be possible to make it more users friendly by adding more variety of functions then it will give user what he wants, that will be ultimate success of our attempts.

**Chapter-10**

**BIBLIOGRAPHY**

**10.1 BIBLIOGRAPHY**

* [**www.w3schools.com**](http://www.w3schools.com)
* [**https://www.geeksforgeeks.org**](https://www.geeksforgeeks.org/)
* [**https://www.tutorialspoint.com**](https://www.tutorialspoint.com/)
* [**www.quora.com**](http://www.quora.com)