

ABSTRACT

Internet is the most integral part of our daily life and the people who manages their work with internet like bank transaction, online shopping is also constantly growing. The websites which provide these services should be an authenticated one i.e., they should allow the user to create their own username and password with a reliable service. So only qualified people can access their account by password authentication. A technique such as Secure Socket Layer (SSL) is used for a secure transaction purpose. But some websites offers a poor authentication service which leads to password attacks.

Normally, users maintain the same username and password for the websites they use. This leads to the hackers to hack the password and can easily retrieve the users' personal information. Since most of the websites use low security suite, it stores the username and password in a clear text and this allows the hackers to hack it. These authentication problems can be resolved by using hardware tokens or client certificates. But these two methods are difficult to use because of high cost and problematic for users to use. Many cryptographic hash functions are used for the development of secure password system. This causes the user to enter into the burden for maintaining password and public keys to develop the Public Key Infrastructure. A poorly designed hash function can make attacks feasible even if a strong password is chosen. So these problems has drawn attention to the need for new hash methodology iv to improve the security which is directly related to syndrome decoding problem from the theory of error correcting codes.

To avoid the problems cited above, a new proposal technique Pseudo Random Key Generation has been designed to improve the efficiency of password authentication. The Pseudo Random Key Generator uses computational algorithm to generate a random key. The goal of the computational algorithm is to generate a sequence of numbers from the key that appear to be random. This technique provides a user interface with hash based quantum key password i.e., the generated random key generates a hash key value for each user password by using hash function. These key values are shared by the user and trusted center and it is difficult for the intruders to find because of the state of keys may change at a time. In other words, an outside observer cannot predict the next number to be generated from the list of numbers previously generated without expending a great deal of computational effort. This Pseudo Random Password authentication system is used mainly for strengthening the user password key.

CONTENTS

Cover Page

Certificate

Declaration

Acknowledgement

Abstract

Contents

List of figures

1. INTRODUCTION

1.1 About Project

2. LITERATURE SURVEY

2.1 Information security and the CIA triad

2.2 Authentication and passwords

2.3 Managing multiple passwords

2.4 Password managers

2.5 Synthesis

3. SYSTEM ANALYSIS

3.1 Proposed System and its Advantages

3.2 Software Requirement Analysis

3.2.1 Functional Requirements

3.2.2.1 Input

3.2.2.2 Output

3.2.3 Non-Functional Requirements

3.3 Software Requirements/Environment

3.4 Hardware Requirements/Environment

3.5 Technologies Used

3.5.1 Python

3.5.2 Python Features

3.5.3 Python Tkinter

3.5.4 Advantages and Disadvantages of Python Programming Language

3.5.4.1 Advantages of Python Programming Language

3.5.4.2 Disadvantages of Python Programming Language

4. SOFTWARE DESIGN

4.1 System Architecture

4.2 Data Flow Diagram

4.3 ER Diagram

4.4 UML Diagrams

4.4.1 Use Case Diagram

4.4.2 Class Diagram

4.4.3 Activity Diagram

4.4.4 Sequence Diagram

4.4.5 Collaboration Diagram

4.4.6 Component Diagram

4.4.7 Deployment Diagram

5. CODING

6. TESTING

6.1 Testing Methodologies

6.1.1 Unit Testing

6.1.2 Integration Testing

6.1.3 Acceptance Testing

6.1.4 Output Testing

6.1.6 Validation Testing

6.1.7 Testing Strategy

6.2 Test Cases For Unit Testing And Validation Testing

7. OUTPUT SCREENS

8. CONCLUSION

9. REFERENCE

LIST OF TABLES

S.No	Table Number	Table Name	Page.No
1	Table Number 1	Validation Test Case 1	
2	Table Number 2	Validation Test Case 2	
3	Table Number 3	Validation Test Case 3	
4	Table Number 4	Validation Test Case 4	
5	Table Number 5	Validation Test Case 5	
6	Table Number 6	Validation Test Case 6	
7	Table Number 7	Validation Test Case 7	
8	Table Number 8	Validation Test Case 8	
9	Table Number 9	Validation Test Case 9	
10	Table Number 10	Validation Test Case 10	

LIST OF DIAGRAMS

S.No	Diagram Number	Diagram Name	Page.No
1	Fig 4.1	System Architecture	
2	Fig 4.2	Data Flow Diagram	
3	Fig 4.3	ER Diagram	
4	Fig 4.4.1	Use Case Diagram	
5	Fig 4.4.2	Class Diagram	
6	Fig 4.4.3	Activity Diagram	
7	Fig 4.4.4	Sequence Diagram	
8	Fig 4.4.5	Collaboration Diagram	
9	Fig 4.4.6	Component Diagram	
10	Fig 4.4.7	Deployment Diagram	

INTRODUCTION

1.1 ABOUT PROJECT

Python | Random Password Generator using Tkinter

- With growing technology, everything has relied on data and securing these data is the main concern. Passwords are meant to keep the data safe that we upload on the Internet.

An easy password can be hacked easily and all the personal information can be misused. In order to prevent such things and keep the data safe, it is quite necessary to keep our passwords very strong.

- Let's create a simple application which can randomly generate strong passwords using Python **Tkinter** module.
- This application can generate random password, with the combination of letters, numeric's, and special characters. One can mention length of the password based on requirement and can also select the strength of the password.
- It's important to protect your personal information online, and in this project design it allows the user to generate unique passwords.
- The passwords will be random, so no one will be able to guess them!
- These Unique random passwords generated by the software are highly secure and cannot be cracked easily.
- Software generated passwords are useful for business purpose, personal use, to generate e-commerce coupon codes, to encrypt the data..... etc.

How secure is your password?

- A computer could try to guess your password by using 'brute force' – this means trying out lots of passwords until it guesses the right one.
- Let's find out how long it would take a computer to guess your password.

Activity Checklist:

- Go to howsecureismypassword.net, which is a website for finding out how secure your passwords are.



- Type in “**letmein**” (Let me in) as the password. You’ll see that a computer would guess this password instantly!

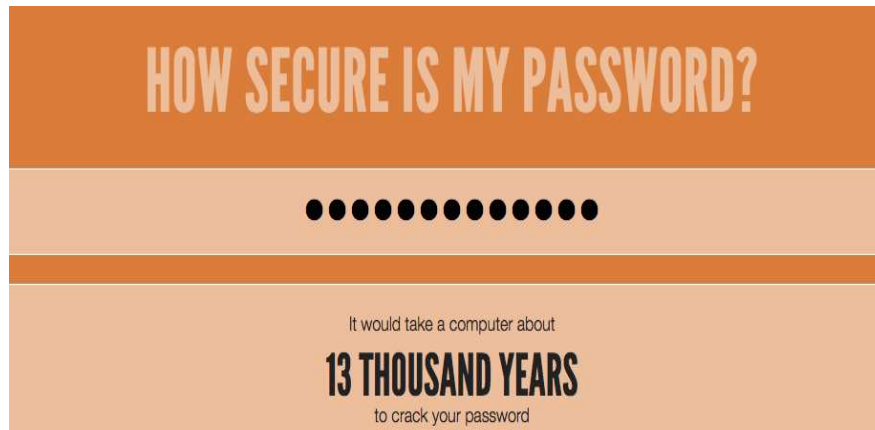


Here we see some reasons why “letmein” isn’t a good password to use:

- It’s a very common password (one of the 15 most used passwords). A computer would guess these first.
- It contains words from dictionary. A computer would also try these passwords first.
- It’s very short. It would take a computer more time to guess a longer password.
- It only contains letters. Passwords are more secure if they also contain numbers and punctuation

Creating a better password

- Can you enter a password that would take a computer more than 1,000 years to crack but isn’t too long to type?



- Remember that your password is harder to guess if it's:
 - Long
 - Not a word in the dictionary
 - Contains letters, numbers and punctuation
- You're going to generate passwords that are hard for a computer to crack. These are useful for protecting important accounts.
- Note that many companies use a password manager program to help them remember lots of tricky passwords.

Do password generators work?

How Do Random Password Generators Work?

A random password generator is a software program, hardware device, or online tool that automatically generates a password using parameters that a user sets, including mixed-case letters, numbers, symbols, pronounceability, length, and strength.

What does generate password mean?

A random password generator is software program or hardware device that takes input from a random or pseudo-random number generator and automatically generates a password. Random passwords can be generated manually, using simple sources of randomness such as dice or coins, or they can be generated using a computer.

What is a high strength password?

Password strength is a measure of the effectiveness of a password against guessing or brute-force attacks. In its usual form, it estimates how many trials an attacker who does not have direct access to the password would need, on average, to guess it correctly.

What is an example of a strong password?

Use a password that has at least 16 characters, use at least one number, one uppercase letter, one lowercase letter and one special symbol. ... Do not use any dictionary word in your passwords.

Are password generators safe?

It is not safe to generate passwords online. ... The average user has no way to verify that the password generation code is using good entropy (and JavaScript's `Math.random()`, which is the obvious thing to use for this purpose, is not a great pseudorandom number generator).

LITERATURE SURVEY

2. LITERATURE SURVEY

In this section of the chapter, the literature review is presented. Theory presented in this literature review is partly based on literature presented in courses in the Information Security master program at Lulea University of Technology (LTU). Google Scholar was then used to find more profound information about the topics. Keywords that have been used are presented under each topic.

2.1 Information security and the CIA triad

Keywords used on Google Scholar includes “information security” and “confidentiality and integrity and availability”.

Information security can be defined as protecting information with regard to three main requirements: confidentiality, integrity, and availability. Confidentiality meaning that unauthorized parties are not allowed to intercept the information. Integrity meaning that the information has not been corrupted or changed during storage or transmission by unauthorized parties. Availability meaning that information is available to the authorized parties when trying to access it. This model is often referred to as the CIA triad. Modern business needs have also increased the need for non-repudiation which means that an action, e.g. a business transaction, cannot be denied afterwards (**Siponen & Oinas-Kukkonen, 2007**). Some has criticized the model of not including enough principles to ensure the information security of an organization. Dhillon and Backhouse proposed already in 2000 that the principles of responsibility, integrity (integrity of a person and not only information), trust and ethicality could be added to the model (**Dhillon & Backhouse, 2000**). Another study proposes that authentication, access control, and non-repudiation should be added to the model (**Simmonds, Sandilands, & Ekert, 2004**). Even though the CIA triad may not present the whole picture when it comes to information security there is consensus in the scientific community that confidentiality, integrity, and availability are key concepts in ensuring the security of information (**Siponen & Oinas-Kukkonen, 2007; Simmonds, Sandilands, & Ekert, 2004; Hilton & Cherdantseva, 2013**).

2.2 Authentication and passwords

Keywords used on Google Scholar includes “authentication”, “password”, “password security”, and “information security and behavioural”.

Authentication can be divided into four categories; something the user knows, for example a password; something the user possesses, for example a smart card; something the user is, for example static biometrics like fingerprint sensor; something the user does, for example dynamic biometric like sound recognition (Stalling & Brown, 2015). Although there are many advanced and secure authentication methods – like biometrics and smart-cards – text-based passwords (often together with a username) that rely on what the user knows is still the most common type of authentication (Bonneau, Herley, van Oorschot, & Stajano, 2012). Because the user is involved in creating and managing these passwords, the biggest security issue with this type of authentication is the human factor (Furnell, Jushoh, & Katsabas, 2006).

2.3 Managing multiple passwords

Keyword used on Google Scholar includes “password reuse”, “password recovery”, “multiple passwords”, and “password sharing”.

Having multiple passwords can cause even more security issues with regard to security. Using the same or similar password on many different systems increases the risk of compromising the password on all systems if one is accidentally or intentionally shared (Duggan, Johnson, & Grawemeyer, 2012; Bang, Lee, Bae, & Ahn, 2012). If a password is forgotten both the lack of productivity and the cost of maintaining a service desk can be very costly to the organization (Duggan, Johnson, & Grawemeyer, 2012; Bonneau, Herley, van Oorschot, & Stajano, 2012). Another security issue is using weak passwords for easier memorability and other reasons (Hoonakker, Bornoe, & Carayon, 2009).

Regardless of the comprehensiveness of the information security policy mismanaging passwords with reusing passwords, writing them down, and sharing them is still a big part of the everyday process (Grawemeyer & Hilary, 2011). A strict security policy also poses the risks of backfiring when users circumvent security controls because they are simply too inconvenient (Adams & Sasse, 1999).

There are two primary factors that explains why users are not engaged in good password management; firstly because they do not see the immediate threat and consequences to themselves; and secondly because good password management often decreases the convenience factor (Tam, Glassman, & Vandenwauver, 2010).

In summary multiple password environments creates security issues regarding:

- **Weak passwords** – using weak passwords, for example using known words and terms.
- **Password reuse** – using the same or similar password on more than one system
- **Password recovery** – loss of productivity and cost of maintaining a service desk for password related issues
- **Password notes** – writing down passwords to remember them, both electronic and on paper
- **Password sharing** – sharing passwords with friends and/or colleagues

Using multiple passwords in an organizational context should (if possible) be avoided. Having users remember more than four to five passwords causes them to forget the password and therefore increasing the risk of engaging in poor password management behavior (Adams & Sasse, 1999).

2.4 Password managers

Keywords used on Google Scholar includes “password manager”, “password managers”, “password manager AND usability”, and “password manager AND user”.

Password managers can be one way to mitigate the vulnerabilities that appear when multiple passwords are required. These software programs are able to store multiple passwords with the help of one master password that encrypts the whole password file. These programs also have the ability to generate safe passwords which decreases the risk of password reuse (Silver, Jana, Boneh, Chen, & Jackson, 2014; Reichl, 2016). KeePass is an open source password manager, which is free of charge and works on the most common operating systems (Reichl, 2016). According to the security expert Bruce Schneier, security software should always be open source to be considered secure (Schneier, 1999).

Even though password managers could solve many of the security issues regarding multiple passwords they seem to be rarely used (Stobert & Biddle, 2014; Hoonakker, Bornoe, & Carayon, 2009; Ur, et al., 2015). The main area of research regarding password managers are the technical design and technical security (Bojinov, Bursztein, & Boyen, 2010; Al-Sinani & Mitchell, 2010; Gasti & Rasmussen, 2012). This aspect is important but not relevant for this study. Another popular area of research regarding password managers is the usability aspect of the software. When testing the usability of different password

managers it was found that users prefer an older portable solution that demands a software to be run on the computer rather than a modern online-based password manager (**Karole, Saxena, & Christin, 2010**). This supports the use of KeePass in this study. A ten-year-old study found that users acted reluctant in using the password management software because they were not comfortable with giving control of their password to a password manager, did not feel the need for it, and felt that the password manager did not provide greater security than before the implementation (**Chiasson & Oorschot, 2006**). A relevant question would be if this would be the result even ten years later.

2.5 Synthesis

When summarizing the data from the different studies it can be said that password managers can be used to solve many of the information security issues that arise when using one or multiple passwords as authentication. A ten-year-old study has found that users were reluctant in using the password manager. Is this still the case even after ten years, and when the password manager is implemented in a multiple-password environment where the password management has been diagnosed as a security issue?

SYSTEM ANALYSIS

3.1 PROPOSED SYSTEM AND ITS ADVANTAGES

Random Password Generator

Random password generator provides ability to create strong password of high security. There are three advantages of passwords created by random password generator in comparison with passwords created by human:

Reliability: Password generator also uses numerals, special characters (@ # \$ % ^ &) and upper case letters to make password really unbreakable, while using simple dictionary words for passwords is unsafe;

Uniqueness: Consist of random symbols combinations, while pet's nicks, dates of birth or other personal information used as password could be easily cracked by associations;

Password length: 10 or more characters, while short passwords are easy to remember, but also easy to hack.

Handy Password manager allows to set up all parameters of password generator including password length, enable or disable special characters, numerals and upper case letters usage.

3.2 SOFTWARE REQUIREMENT ANALYSIS

A *Software requirements specification* (SRS) is a detailed description of a software system to be developed with its functional and non-functional requirements. It may include the use cases of how user is going to interact with software system. The software requirement specification document consistent of all necessary requirements required for project development. To develop the software system we should have clear understanding of Software system.

A good SRS defines the how Software System will interact with all internal modules, hardware, communication with other programs and human user interactions with wide range of real life scenarios. Using the *Software requirements specification* (SRS) document on QA lead, managers creates test plan. It is very important that testers must be cleared with every detail specified in this document in order to avoid faults in test cases and its expected results.

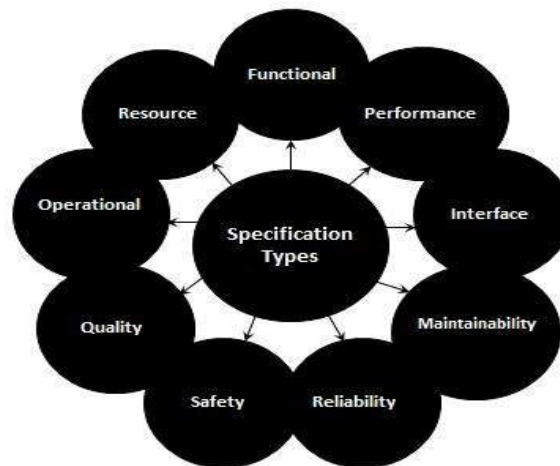
It is highly recommended to review or test SRS documents before start writing test cases and making any plan for testing.

QUALITIES OF SRS:

- Correct, Unambiguous, Complete, Consistent, Ranked for importance and/or stability, Verifiable, Modifiable, Traceable.

TYPES OF REQUIREMENTS

The below diagram depicts the various types of requirements that are captured during SRS.



3.2.1 FUNCTIONAL REQUIREMENTS

Functional requirements describe what the system should do, i.e. the services provided for the users and for other systems.

3.2.2.1 INPUT

- Allows user to select the required length from the ComboBox
- Allows user to select the multiple combination using CheckBoxes
- Allows user to select either single/multiple password generator using RadioButtons
- If user selects single, When user clicks on Generate Button password will be generated in the Entry Box
- If user selects Multiple, another Window will be opened in that user need to enter the required count of password. When user clicks on generate Button multiple passwords will be generated in the ListBox
- Every time when user click on Generate Button New passwords will be Generated
- Copy Button allows user to copy the password/passwords

- Reset Button allows user to reset all the values to default
- Exit Button allows user to exit the window

3.2.2.2 OUTPUT

- Generates Unique Single Random Password of required length for the user selection
- Generates Unique Multiple Random passwords of required length and count of passwords for the user selection

3.2.3 NON-FUNCTIONAL REQUIREMENTS

In non-functional requirements the following are the things that come under. They are as follows:

- 1) Portability: As the application is designed with python programming language, it can run on any operating system. Hence the application is portable to run on any operating system.
- 2) Extensibility: The application can be extended at any level if the user wish to extend that in future this is done because python is a open source medium which doesn't have any time limits for expiry or renewal

3.3 SOFTWARE REQUIREMENTS/ENVIRONMENT

- Operating System : Windows 7/8/8.1/10/Linux/Unix
- Platform : Python IDLE
- Back End : Python 3.7
- Front End : Python Tkinter GUI

3.4 HARDWARE REQUIREMENTS/ENVIRONMENT

- Processor : Pentium IV/Intel i3+
- Hard Disk : 2 GB Free Space
- Ram : 2GB

3.5 TECHNOLOGIES USED

3.5.1 Python

Python is a general purpose, dynamic , high level and interpreted programming language. It supports Object Oriented Programming approach to develop applications. It is simple and easy to learn and Provides lots of high level data structures.

Python is easy to learn yet powerful and versatile scripting language which makes it attractive for application Development. Python's syntax and dynamic typing with its interpreted nature, makes it an ideal language for scripting and rapid application development.

Python supports multiple programming pattern, including object oriented, imperative and functional or procedural programming styles. Python is not intended to work on special area such as web programming. That is why it is known as multipurpose because it can be used with web, enterprise, 3D CAD etc. We don't need to use data types to declare variable because it is dynamically typed so we can write `a=10` to assign an integer value in an integer variable.

Python makes the development and debugging fast because there is no compilation step included in python development and edit-test-debug cycle is very fast.

3.5.2 Python Features

Python provides lots of features that are listed below.

1) Easy to Learn and Use

Python is easy to learn and use. It is developer-friendly and high level programming language.

2) Expressive Language

Python language is more expressive means that it is more understandable and readable.

3) Interpreted Language

Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.

4) Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.

5) Free and Open Source

Python language is freely available at official web address. The source-code is also available. Therefore it is open source.

6) Object-Oriented Language

Python supports object oriented language and concepts of classes and objects come into existence.

7) Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.

8) Large Standard Library

Python has a large and broad library and provides rich set of module and functions for rapid application development.

9) GUI Programming Support

Graphical user interfaces can be developed using Python.

10) Integrated

It can be easily integrated with languages like C, C++, JAVA etc.

3.5.3 Python Tkinter



Python provides the standard library for creating the graphical user interface for desktop based applications.

Developing desktop based applications with python Tkinter is not a complex task. An empty Tkinter top-level window can be created by using the following steps.

1. import the Tkinter module.
2. Create the main application window.
3. Add the widgets like labels, buttons, frames, etc. to the window.
4. Call the main event loop so that the actions can take place on the user's computer screen.

Modules needed:

- import random
- from tkinter import *
- from tkinter.ttk import *
- import pyperclip

3.5.4 Advantages and Disadvantages of Python Programming Language

When we want to choose a language for a project, we want to be thorough with what we can do with it. We want to be aware of how it can help us be efficient at what we want to do, but we also want to be careful of the problems that can arise. So, we believe it is worthwhile to take out some time and find out more. In this advantages and disadvantages of the Python programming language tutorial, we will learn the advantages and disadvantages of

a python programming language that will help you in knowing the benefits of learning Python programming.

3.5.4.1 Advantages of Python Programming Language

Extensive Libraries:

It downloads with an extensive library. These contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

Extensible:

Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

Embeddable:

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

Improved Productivity:

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less lets more get done.

IOT Opportunities:

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for Internet Of Things. This is a way to connect the language with the real world.

Simple and Easy:

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code.

This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

Readable:

Because it is not such a verbose language, reading Python is much like reading English. This is also why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

Object-Oriented:

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

Free and Open-Source:

Python is freely available. But not only can you download python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

Portable:

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

Interpreted:

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

3.5.4.2 Disadvantages of Python Programming Language

Speed Limitations:

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

Weak in Mobile Computing and Browsers:

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle. The reason it is not so famous despite the existence of Brython is that it isn't that secure.

Design Restrictions:

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors. In Any query regarding the advantages and disadvantages of Python programming language, tutorial feel free to drop a comment.

Underdeveloped Database Access Layers:

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

Simple:

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

SOFTWARE DESIGN

SOFTWARE DESIGN

In the context of software, design is problem solving process whose objective is to find and describe a way to implement the functional requirements while respecting the constraints imposed by the non-functional requirements and by adhering to general principles of good quality. The goal of the design process is to produce a model or representative of a system which can be used later to build that system and use this model to build the overall system.

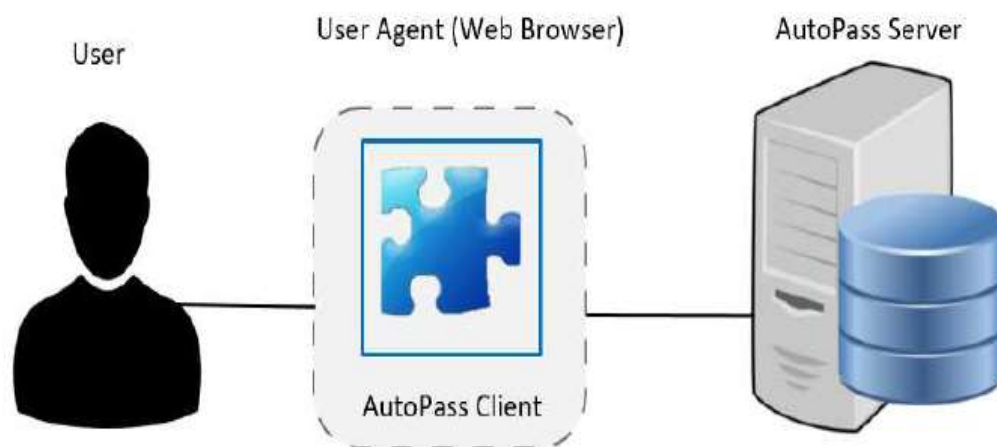


Fig 4.1 System Architecture

This architecture describes the whole functionality of the system as well as the work of every module.

AutoPass conforms to the general model given above. The three main elements, i.e. the input values, the password generation function, and the output method, are as follows. AutoPass uses the following input types (as used in previous schemes). (a) A master password: a long-term strong password selected by the user or generated by the system. It is stored in encrypted form on the AutoPass server as part of the user-specific configuration data. Since it does not need to be remembered by the user, it could, for example, be a 128-bit random value; the precise choice is implementation dependent. The user should make a written record of this value when it is initially chosen, and store it securely for backup/recovery purposes.

4.2 DATA FLOW DIAGRAM

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangle, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowchart can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled.

They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually say things that would be hard to explain in words, and they work for both technical and non technical audience , from developer to CEO. That's why DFDs remain so popular after all these years.

While they work well for data flow software and systems, they are less applicable now-a-days to visualizing interactive, real-time or database-oriented software or systems.

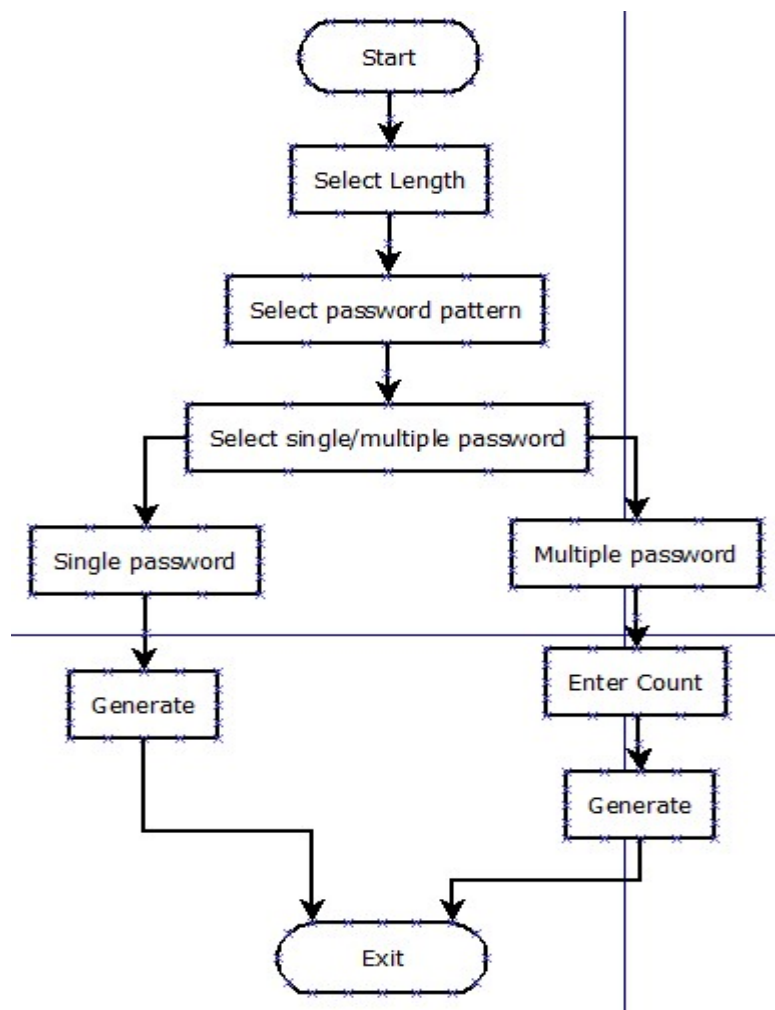


Fig 4.2 Data Flow Diagram

4.3 ER DIAGRAM

An entity relationship diagram (ERD) shows the relationship of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagram illustrate the logical structure of databases.

ER Diagram are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationship and their attributes. They mirror grammatical structure, with entities as nouns and relationship as verbs.

ER diagrams are related to data structure diagrams (DSDI), which focus on the relationships of elements within entities instead of relationships between entities themselves. ER diagrams also are often used in conjunction with data flow diagrams (DFDs), which map out the flow of information for processes or systems.

ER Diagrams are composed of entities, relationships and attributes. They also depict cardinality, which defines relationships in terms of numbers. Here's a glossary:

Entity

A definable thing - such as a person, object, concept or event - that can have data stored about it. Think of entities as nouns. Examples: a customer, student, car or product. Typically shown as a rectangle.

Entity Type : A group of definable things, such as students or athletes, whereas the entity would be the specific student or athlete. Other examples: customers, cars or products.

Entity Set : Same as an entity type, but defined at a particular point in time, such as students enrolled in a class on the first day. Other examples: Customers who purchased last month, cars currently registered in Florida. A related term is instance, in which the specific person or would be an instance of the entity set.

Entity Categories : Entities are categorized as strong, weak or associative. A strong entity can be defined solely by its own attributes, while a weak entity cannot. An associates entities (or elements) within an entity set.

Entity Keys: Refers to an attribute that uniquely defines an entity in an entity set. Entity keys can be super, candidate or primary.

Super Key: A set of attributes (one or more) that define an entity in an entity set.

Candidate Key: A minimal super key, meaning it has the least possible number of attributes to still be a super key. An entity set may have more than one candidate key.

Primary Key: A candidate key chosen by the database designer to uniquely identify the entity set.

Foreign Key: Identifies the relationship between entities.

Relationship

How entities act upon each other or are associated with each other. Think of relationships as verbs. For example, the named student might register for a course. The two entities would be the student and the course, and the relationship depicted is the act of enrolling, connecting the two entities in that way. Relationships are typically shown as diamonds or labels directly on the connecting lines.

Recursive Relationship : The same entity participates more than once in the relationship.

Attribute

A property or characteristic of an entity. Often shown as an oval or circle.

Descriptive Attribute : A property or characteristic of a relationship (versus of an entity.)

Attribute Categories : Attributes are categorized as simple, composite, derived, as well as single-value or multi-value.

Simple : Means the attribute value is atomic and can't be further divided, such as a phone number.

Composite : Sub-attributes spring from an attribute.

Derived : Attributed is calculated or otherwise derived from another attribute, such as age from a birth date.

Multi-Value : More than one attribute value is denoted, such as multiple phone numbers for a person.

Single-Value : Just one attribute value. The types can be combined, such as: simple single-value attributes or composite multi-value attributes.

Cardinality

Defines the numerical attributes of the relationship between two entities or entity sets. The three main cardinal relationships are one-to-one, one-to-many, and many-many.

Cardinality Views : Cardinality can be shown as look-across or same-side, depending on where the symbols are shown.

Cardinality Constraints : The minimum or maximum numbers that apply to a relationship.

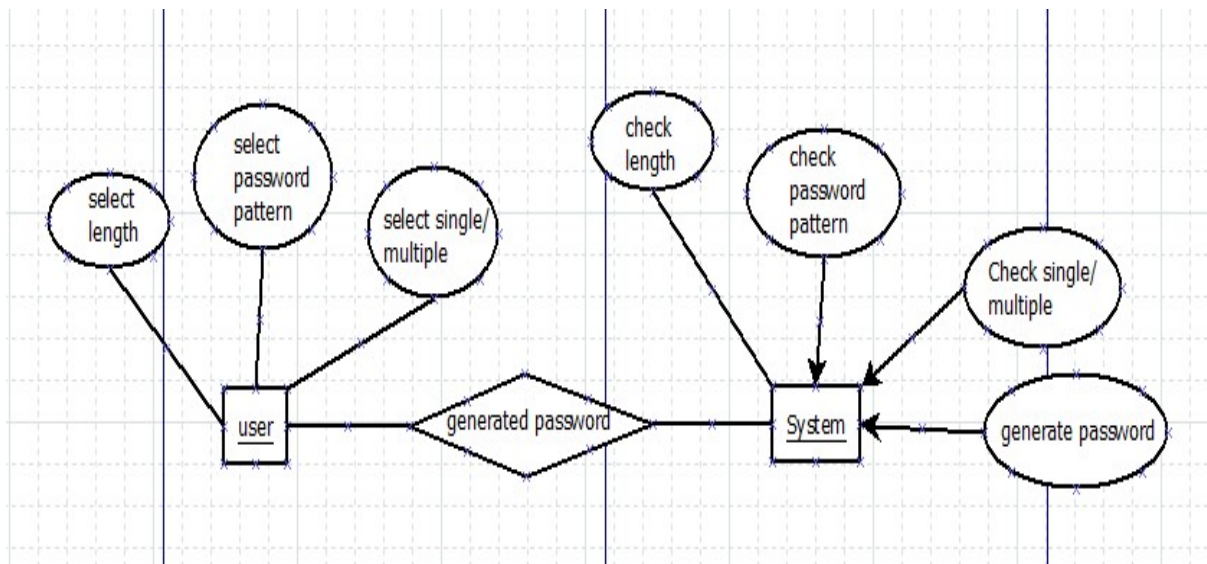


Fig 4.3 ER Diagram

4.4 UML DIAGRAMS

UML stands for Unified Modelling Language. UML is a standardized general-Purpose modelling language or 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 Modelling Language is a standard language for specifying, Visualization, constructing and documenting the artifacts of software system, as well as for business mode and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling 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:

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

UML is powerful enough to represent all the concepts that exist in object-oriented analysis and design. UML diagrams are representation of object-oriented concepts only. Thus, before learning UML. it becomes important to understand OO concept in detail.

Following are some fundamental concepts of the object-oriented world -

- **Objects** : Objects represent an entity and the basic building block.
- **Class** : Class is the blue print of an object.
- **Abstraction** : Abstraction represents the behavior of an real world entity
- **Encapsulation** : Encapsulation is the mechanism of binding the data together and hiding them from the outside world.
- **Inheritance** : Inheritance is the mechanism of making new classes from existing ones.
- **Polymorphism** : It defines the mechanism to exists in different forms.

As UML describes the real-time systems, it is very important to make a conceptual model and then proceed gradually. The conceptual model of UML can be mastered by learning the following three major elements-

- UML building blocks
- Rules to connect the building blocks
- Common mechanisms of UML

The building blocks of UML can be defined as -

- Things
- Relationships
- Diagrams

THINGS

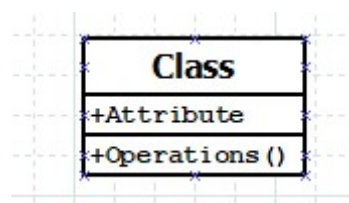
Things are the most important building blocks of UML. Things can be –

- Structural
- Behavioral
- Grouping
- Annotational

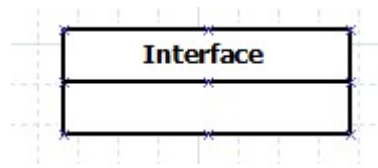
Structural Things

Structural things define the static part of the model. They represent the physical and conceptual elements. Following are the brief descriptions of the structural things.

Class : Class represents a of objects having similar responsibilities



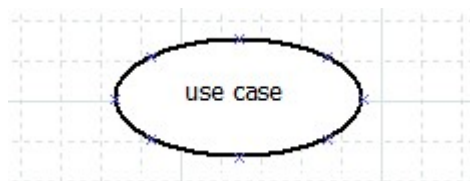
Interface : Interface defines a set of operations, which specify the responsibility of a class.



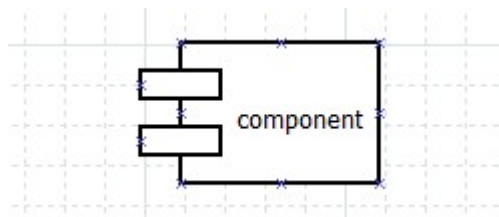
Collaboration : Collaboration defines an interaction between elements.



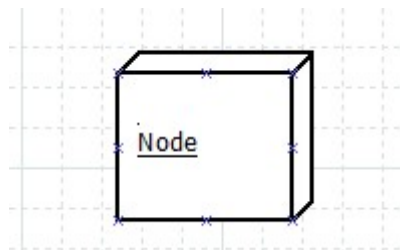
Use Case : Use case represents a set of actions performed by a system for a specific goal.



Component : Component describes the physical part of a system.



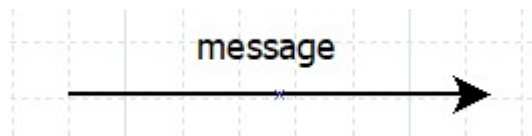
Node : A node can be defined as a physical element that exists at run time.



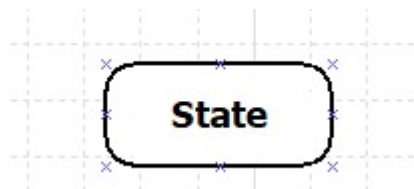
Behavioral Things

A behavioral thing consists of the dynamic parts of UML models. Following are the behavioral things -

Interaction: Interaction is defined as a behavioural that consists of a group of messages exchanged among elements to accomplish a specific task.



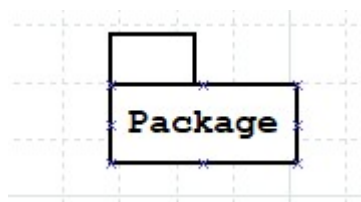
State Machine - state machine is useful when the state of an object in its life cycle is important. It defines the sequence of states an object goes through in response to events. Event are external factors responsible for state change.



Grouping Things

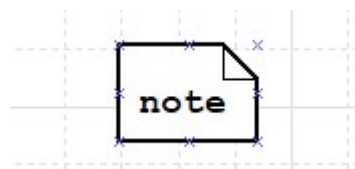
Grouping things can be defined as a mechanism to group elements of a UML model together. There is only one grouping thing available -

Package - Package is the only one grouping thing available for gathering structural and behavioral things.



Annotational Things

Annotational things can be defined as a mechanism to capture remarks, descriptions, an comments of UML model elements. Note - It is the only one Annotational thing available. A note is used to render comments, constraints, etc. of an UML element.



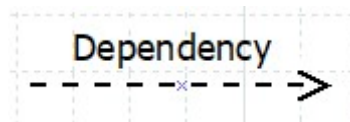
RELATIONSHIP

Relationship is another most important building block of UML. It shows how the elements are associated with each other and this association describes the functionality of an application.

There are four kinds of relationships available.

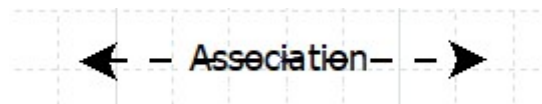
Dependency

Dependency is a relationship between two things in which change in one element also affects the other.



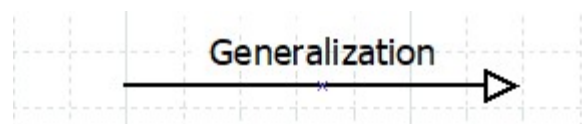
Association

Association is basically a set of links that connects the elements of a UML model. It describes how many objects are taking part in that relationship.



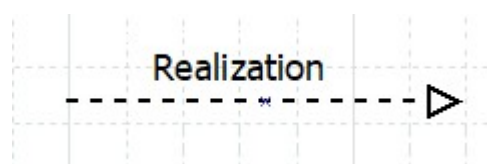
Generalization

Generalization can be defined as a relationship which connects a specialized element with a generalized element. It basically describes the inheritance relationship in the world of objects.



Realization

Realization can be defined as a relationship in which two elements are connected. One element describes some responsibility, which is not implemented and the other one implements them. This relationship exists in case of interfaces.



UML DIAGRAMS

UML diagrams are the ultimate output of the entire discussion. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system.

The visual effects of the UML diagram is the most important part of the entire process. All the other elements are used to make it complete.

UML includes the following nine diagrams, the details of which are described in the subsequent chapters.

- Class diagram
- Object diagram
- Use Case diagram
- Sequence diagram
- Collaboration diagram
- Activity diagram
- State Chart diagram
- Deployment diagram
- Component Diagram

Any real-world system is used by different users. The users can be developers, testers, business people, analysts, and many more. Hence, before designing a system, the architecture is made with different perspectives in mind. The better we understand the better we can build the system.

UML plays an important role in defining different perspectives of a system. These perspectives are-

- Design
- Implementation
- Process
- Deployment

4.4.1 USE CASE DIAGRAM

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural 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 case), and any dependencies between those use cases.

Use case diagram are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities use cases are prepared and actors are identified. When the initial task is complete, use case diagram are modelled to present the outside view.

In brief, the purpose of use case diagram can be said to be as follows-

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.
- Show the interaction among the requirements are actors.

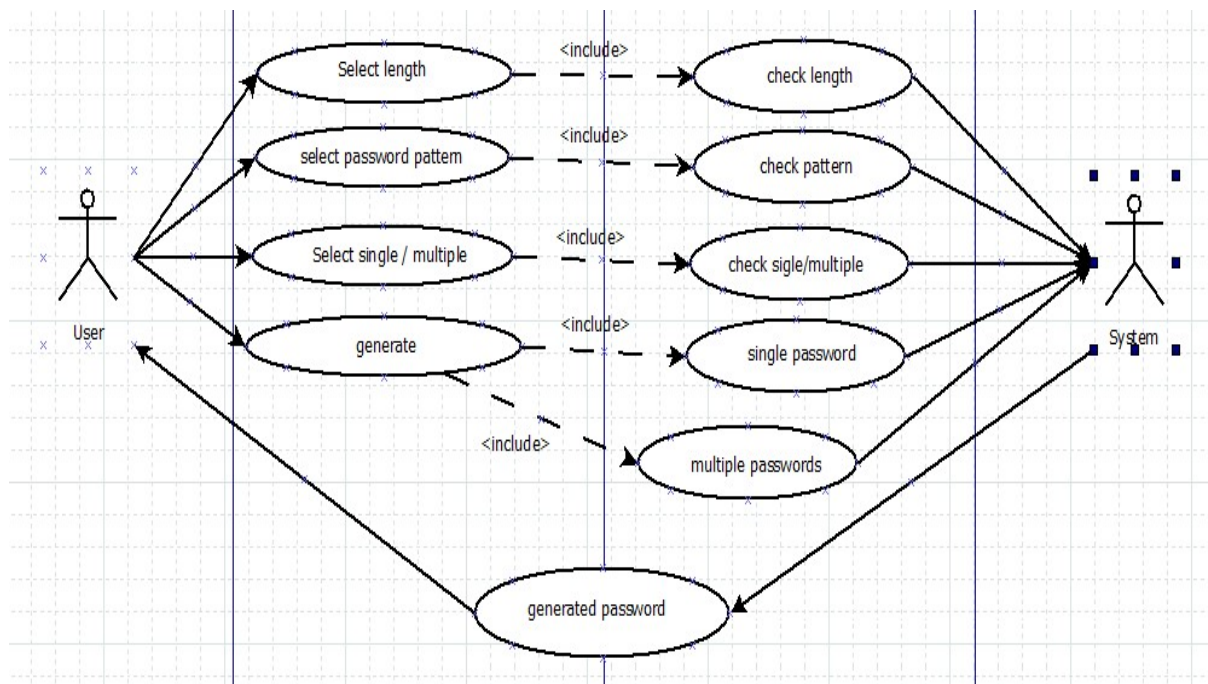


Fig 4.4.1 Use Case Diagram

4.4.2 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modelling 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

The class shape itself consists of a rectangle with three rows. The top row contains the name of the class, the middle row contains the attributes of the class, and the bottom section expresses the methods or operations that the class may use. Classes and subclasses are grouped together to show the static relationship between each object

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

The purpose of the class diagram can be summarized as

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

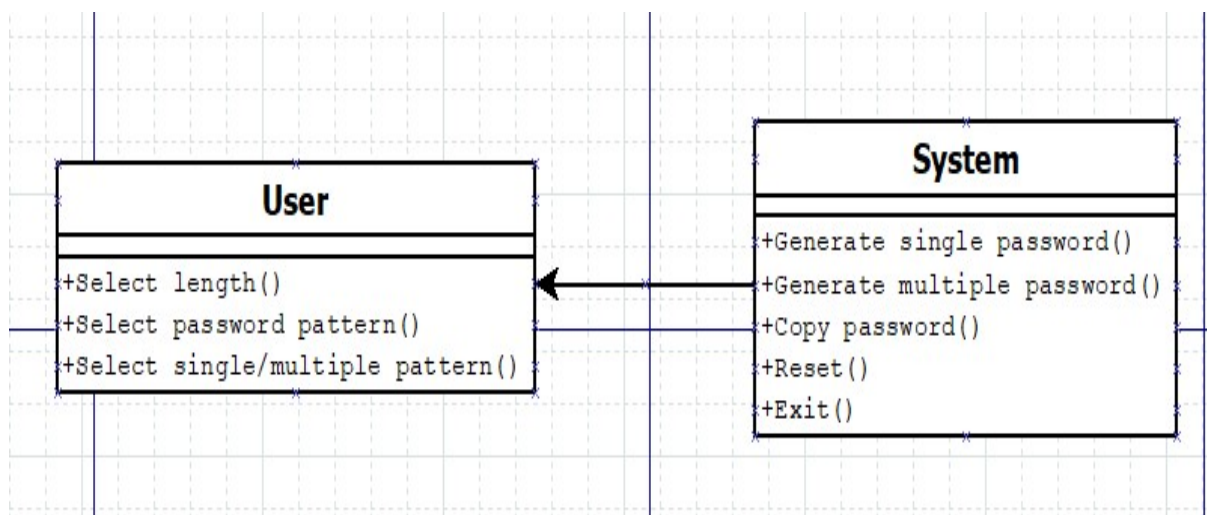


Fig 4.4.2 Class Diagram

4.4.3 ACTIVITY DIAGRAM

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

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques.

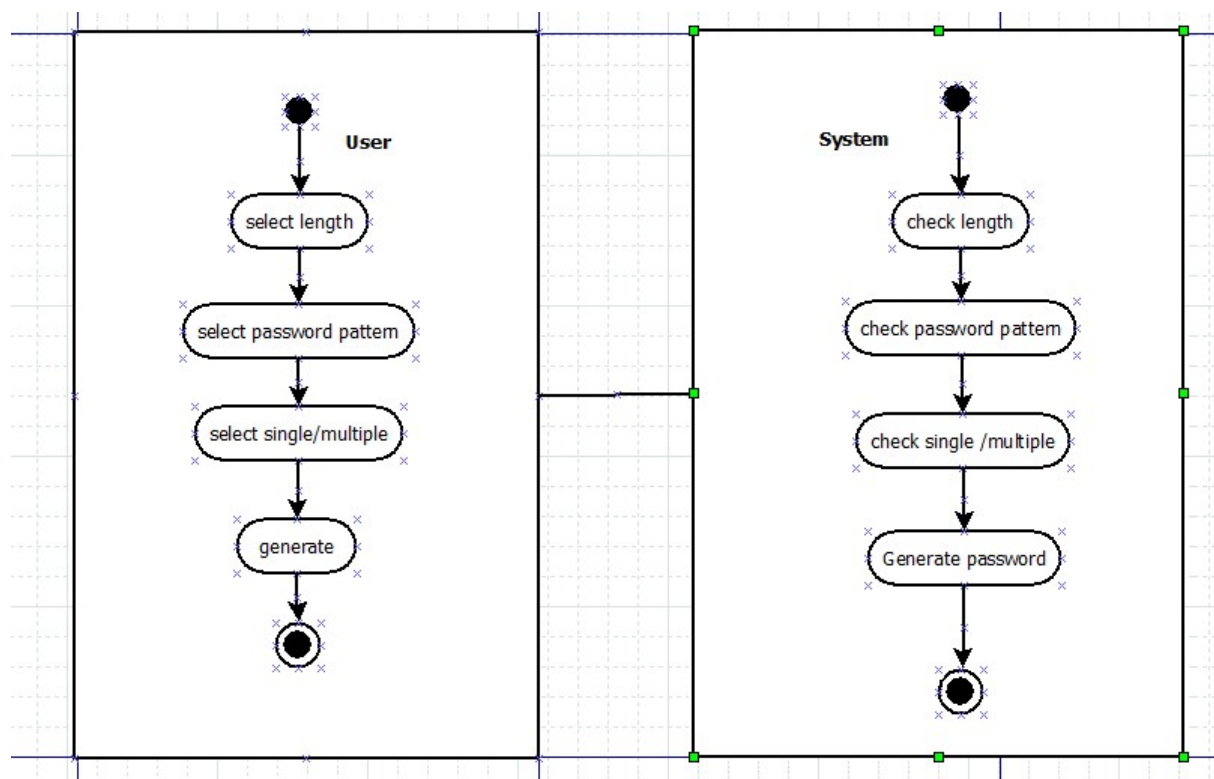


Fig 4.4.3 Activity Diagram

4.4.4 SEQUENCE DIAGRAM

A sequence diagram in Unified Modelling 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.

Sequence diagrams can be useful references for businesses and other organizations. Try drawing a sequence diagram to:

- Represent the details of a UML use case.
- Model the logic of a sophisticated procedure, function, or operation.
- See how objects and components interact with each other to complete a process.
- Plan and understand the detailed functionality of an existing or future scenario.

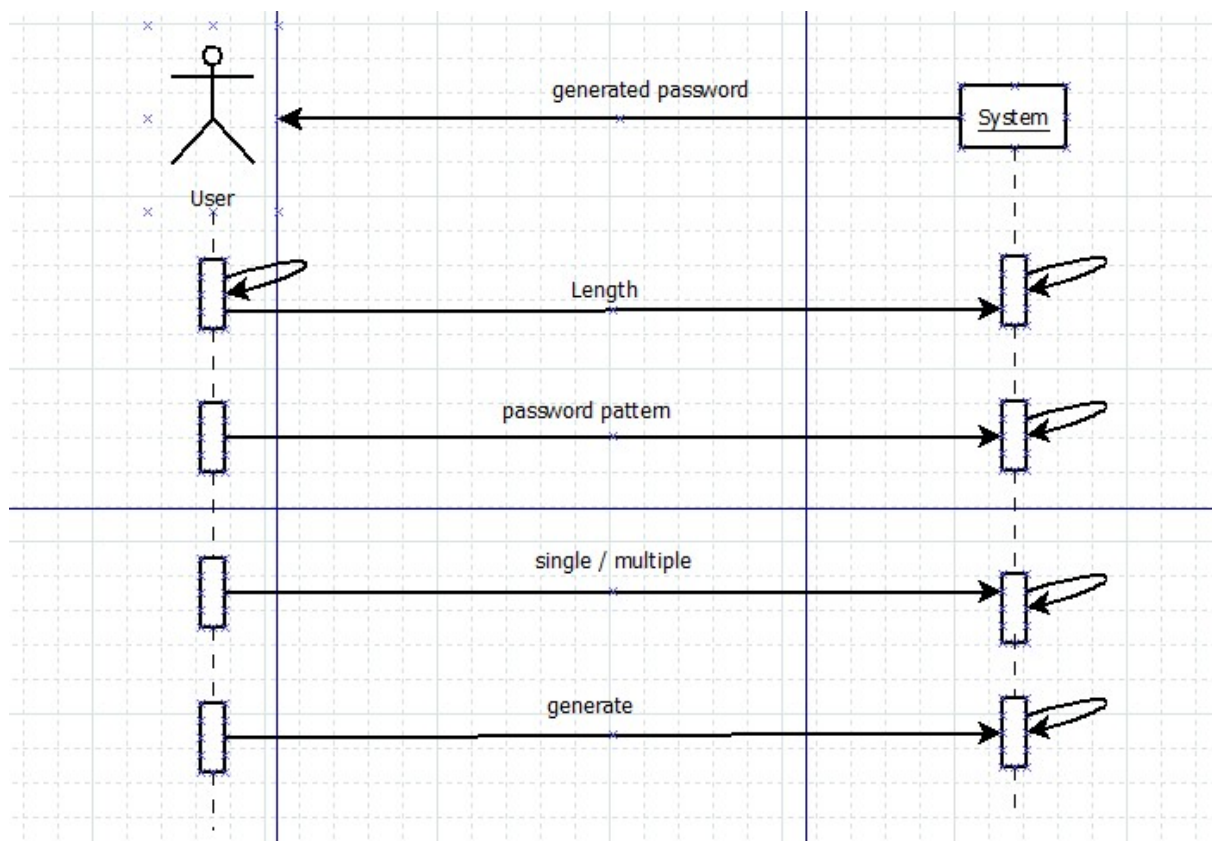


Fig 4.4.4 Sequence Diagram

4.4.5 COLLABORATION DIAGRAM

A collaboration diagram resembles a flow chart that portrays the roles, functionality and behaviour of individual objects as well as the overall operation of the system in real time. The relationship between the objects is shown as lines connecting the rectangles. The messages between objects are shown as arrows connecting the relevant rectangles along with labels that define the message sequencing.

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles.

Collaboration diagrams are best suited to the portrayal of simple interaction among relatively small numbers of objects. As the number of objects and messages grows, a collaboration diagram can become difficult to read. Several vendors offer software for creating and editing collaboration diagrams.

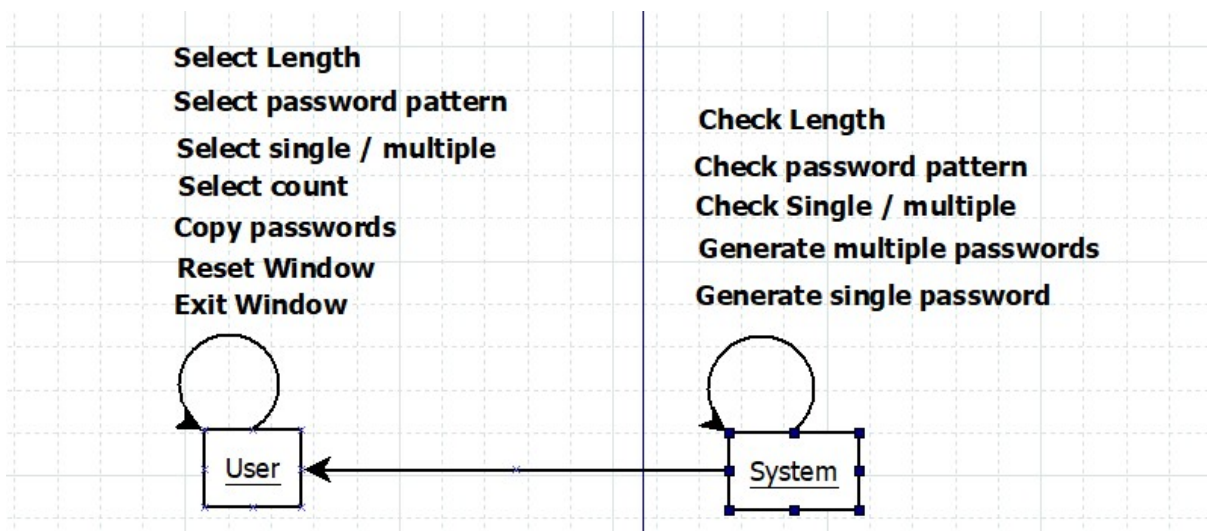


Fig 4.4.5 Collaboration Diagram

4.4.6 COMPONENT DIAGRAM

Component diagrams are different in terms of nature and behavior. Component diagrams are used to model the physical aspects of a system. Physical aspects are the elements such as executables, libraries, files, documents, etc. which reside in a node.

Component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

The purpose of the component diagram can be summarized as -

- Visualize the components of a system.
- Construct executables by using forward and reverse engineering.
- Describe the organization and relationships of the components.

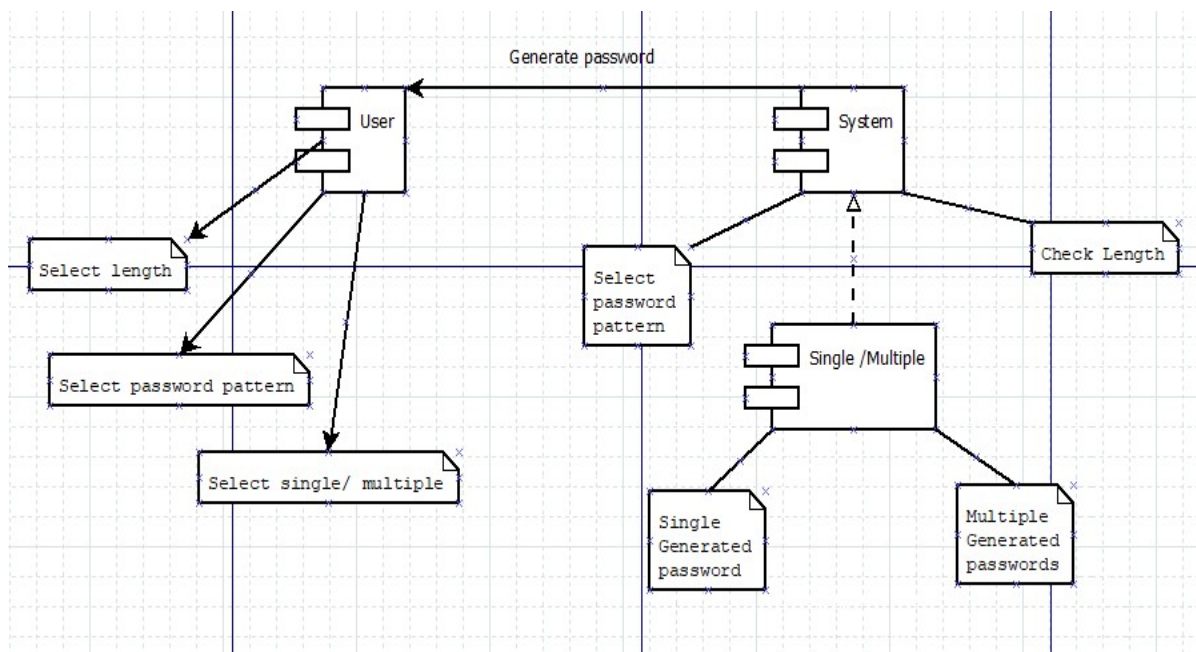


Fig 4.4.6 Component Diagram

4.4.7 DEPLOYMENT DIAGRAM

Deployment diagram is a structure diagram which shows architecture of the system as deployment(distribution) of software artifacts to deployment targets.

Artifacts represents concrete elements in the physical world that are the result of a development process. Example of artifacts are executable files, libraries, archives, database schemes, configuration files, etc.

Deployment target is usually represented by a node which is either hardware device or some software execution environment. Nodes could be connected through communication paths to create networked systems of arbitrary Complexity, Components are deployed to nodes indirectly through artifacts.

Deployment diagrams could describe architecture at specification level (also called type level) or at instance level (similar to class diagrams and object diagrams).

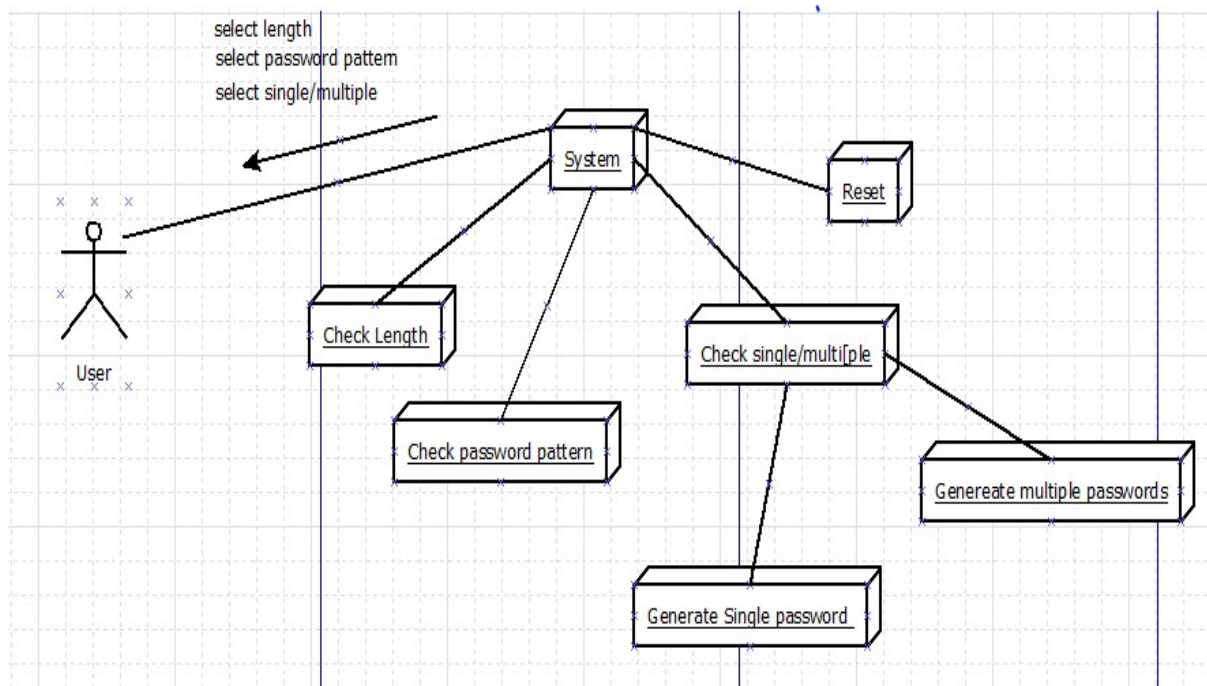


Fig 4.4.7 Deployment Diagram

CODING

CODING

'''PROJECT TITLE: PASSWORD GENERATOR

CODE LANGUAGE: PYTHON 3.7.0a3(32-bit)

FILE NAME: passwordGenerator.py'''

'''SOURCE CODE:'''

import random

from tkinter import *

from tkinter.ttk import *

import time

import datetime

def qwe(len,ch):

 def copy():

 import pyperclip

 c=""

 s=(lis.get(0,END))

 for i in s:

 c+=i+"\n"

 pyperclip.copy(c.strip())

 def Reset():

 lis.delete(0,END)

 E1.delete(0,END)

 entry5.delete(0,END)

 entry5.insert(0,"Enter the count")

 def msg(s):

 entry5.insert(END, s)

 def reqpw_except(cc):

 reqPWs=cc

 try:

 aa=int(reqPWs)

```

        if(aa==0):
            si="Enter count greater than 0"
            return msg(si)
        elif aa<0:
            si="Error:Negative value!"
            return msg(si)
        elif aa>100000:
            si="maximum 100000 only"
            return msg(si)
        else:return aa
    except ValueError:
        si="Error:Invalid Input count"
        return msg(si)
def qExit():
    root1.destroy()
def multiple(len,ch):
    entry5.delete(0,END)
    count=E1.get()
    count4=reqpw_except(count)
    password=""
    for i in range(0, count4):
        for j in range(len):
            password+=random.choice(ch)
        password+="\n"
    entry5.insert(0,"Copy your passwords")
    lis.delete(0,END)
    for index,i in enumerate(password.split("\n")):
        lis.insert(index,i)

# create GUI window
root1=Tk()

```



```

root1.title("Multiple Password generator")

fm1 = Frame(root1, width = 20, relief = SUNKEN)

fm1.pack(side = TOP,padx=10,pady=10)

fm = Frame(root1, width = 20, relief = SUNKEN)

fm.pack(side = TOP,padx=10,pady=10)

fm2 = Frame(root1, width = 20, relief = SUNKEN)

fm2.pack(side = TOP,padx=10,pady=10)

fm3 = Frame(root1, width = 20, relief = SUNKEN)

fm3.pack(side = TOP,padx=10,pady=10)

fm4 = Frame(root1, width = 20, relief = SUNKEN)

fm4.pack(side = TOP,padx=10,pady=10)

password_count = Label(fm1, text=" Password Count: ",font = ('timesnewroman', 12,'bold'))

password_count.pack(side=LEFT,padx=2,pady=2)

E1 = Entry(fm1,font = ('timesnewroman', 15))

E1.pack(side = LEFT)

lblInfo =Label(fm, font=('timesnewroman',12,'bold'),text = " Message: ")

lblInfo.pack(side=LEFT,padx=2,pady=2)

entry5 =Entry(fm,width=35,font = ('timesnewroman', 15, 'bold'))

entry5.pack(side=LEFT)

entry5.insert(0,"Enter password count")

scrollbar=Scrollbar(fm2)

scrollbar.pack(side=RIGHT, fill=Y)

lis=Listbox(fm2,width=50,height=10,font=('timesnewroman,14,bold'))

lis.pack(side=LEFT,padx=5,pady=5,expand=1)

lis.config(yscrollcommand=scrollbar.set)

scrollbar.config(command=lis.yview)

generate_button = Button(fm3,text="Generate",command=lambda:multiple(len,ch))

generate_button.pack(side=LEFT)

copy_button1 = Button(fm3, text="Copy",command=copy)

copy_button1.pack(side=LEFT)

```

```

reset1= Button(fm3, text="Reset",command=Reset)

reset1.pack(side=LEFT)

exit1= Button(fm3, text="Exit",command=qExit)

exit1.pack(side=LEFT)

localtime = time.asctime(time.localtime(time.time()))

lblInfo = Label(fm4, font=('timesnewroman',12,'bold'),text = localtime)

lblInfo.pack(side=BOTTOM)

root.mainloop()

#single password generator

def length_except(a):
    length=a
    try:
        bb=int(length)
        if(bb==0 or bb==1 or bb==2 or bb==3):
            si="Select length(4-32)"
            return msg(si)
        elif bb>32:
            si="Enter the Length less than 32"
            return msg(si)
        elif bb<0:
            si="Error:Negative Value Length"
            return msg(si)
        else:
            return bb
    except ValueError:
        si="Error:Invalid Input Length"
        return msg(si)

def msg(s):
    entry1.insert(END, s+"\n")

```

```
def single(length,ch):
    password=""
    for i in range(0, length):
        password +=random.choice(ch)
    return password
```

```
    if value==1:
        return single(length,ch)
    else:
        s="select single/multiple"
        return msg(s)
```

```
def check(val,ch,length):
    if val==1:
        return single(length,ch)
    elif val==2:
        if length>=4 and length<=32:
            qwe(length,ch)
    else:
        s="select single/multiple"
        return msg(s)
```

```
def calculation():
    entry.delete(0, END)
    entry1.delete(0,END)
    # Get the length of password
    length=var1.get()
    value=var.get()
    length=length_except(length)
    password=""
    num='0123456789'
    SLet='abcdefghijklmnopqrstuvwxyz'
```

```

CLet='ABCDEFGHIJKLMNOPQRSTUVWXYZ'
punc="!\"#$%&()*+,-./:<=>?@[\\]^_`{|}~"

if v1.get()== 1 and v2.get()== 0 and v3.get()== 0 and v4.get()== 0:
    ch=num
    return check(value,ch,length)

elif v1.get()== 0 and v2.get()== 0 and v3.get()== 0 and v4.get()== 4:
    ch=punc
    return check(value,ch,length)

elif v1.get()== 0 and v2.get()== 0 and v3.get()== 3 and v4.get()== 0:
    ch=SLet
    return check(value,ch,length)

elif v1.get()== 0 and v2.get()== 2 and v3.get()== 0 and v4.get()== 0:
    ch=CLet
    return check(value,ch,length)

elif v1.get()== 1 and v2.get()== 2 and v3.get()== 3 and v4.get()== 4:
    ch=num+punc+SLet+CLet
    return check(value,ch,length)

elif v1.get()== 0 and v2.get()== 2 and v3.get()== 3 and v4.get()== 0:
    ch=CLet+SLet
    return check(value,ch,length)

elif v1.get()== 1 and v2.get()== 0 and v3.get()== 3 and v4.get()== 0:
    ch=num+SLet
    return check(value,ch,length)

elif v1.get()== 1 and v2.get()== 2 and v3.get()== 0 and v4.get()== 0:
    ch=num+CLet
    return check(value,ch,length)

elif v1.get()== 0 and v2.get()== 0 and v3.get()== 3 and v4.get()== 4:
    ch=punc+SLet
    return check(value,ch,length)

elif v1.get()== 0 and v2.get()== 2 and v3.get()== 0 and v4.get()== 4:
    ch=punc+CLet

```

```

        return check(value,ch,length)
elif v1.get()== 1 and v2.get()== 0 and v3.get()== 0 and v4.get()== 4:
    ch=punc+num
    return check(value,ch,length)
elif v1.get()== 1 and v2.get()== 0 and v3.get()== 3 and v4.get()== 4:
    ch=punc+num+SLet
    return check(value,ch,length)
elif v1.get()== 1 and v2.get()== 2 and v3.get()== 0 and v4.get()== 4:
    ch=punc+CLet+num
    return check(value,ch,length)
elif v1.get()== 1 and v2.get()== 2 and v3.get()== 3 and v4.get()== 0:
    ch=num+CLet+SLet
    return check(value,ch,length)
elif v1.get()== 0 and v2.get()== 2 and v3.get()== 3 and v4.get()== 4:
    ch=punc+SLet+CLet
    return check(value,ch,length)
else:
    return msg("Tick options")

def generate():
    password = calculation()
    entry.insert(END, password)
    msg("copy your password")

def copy1():
    import pyperclip
    random_password = entry.get()
    pyperclip.copy(random_password)

def qExit():
    root.destroy()
    exit(0)

def Reset():
    v1.set(0)

```

```
v2.set(0)
v3.set(0)
v4.set(0)
var.set(0)
var1.set("4")
entry.delete(0,END)
entry1.delete(0,END)
```

```
# Main Function
```

```
# GUI window
```

```
root = Tk()
root.geometry("590x370")
root.title("Password Generator")
top = Frame(root, width = 20, relief = SUNKEN)
top.pack(side = TOP,padx=10,pady=10)
tops = Frame(root, width = 20, relief = SUNKEN)
tops.pack(side = TOP,padx=10,pady=10)
radio = Frame(root, width = 20, relief = SUNKEN)
radio.pack(side = TOP,padx=10,pady=10)
f2 = Frame(root, width = 20, height = 20,relief = SUNKEN)
f2.pack(side = TOP,padx=10,pady=10)
fw = Frame(root, width = 20, height = 20,relief = SUNKEN)
fw.pack(side = TOP,padx=10,pady=10)
last = Frame(root, width = 20, height = 20,relief = SUNKEN)
last.pack(side = TOP,padx=10,pady=10)
f5 = Frame(root, width = 20, height = 20,relief = SUNKEN)
f5.pack(side = TOP,padx=10,pady=10)
f4= Frame(root, width = 20, height = 20,relief = SUNKEN)
f4.pack(side = TOP,padx=10,pady=10)
```

```
var=IntVar()
v1=IntVar()
v2=IntVar()
v3=IntVar()
v4=IntVar()
var1=StringVar()
```

```
Random_password = Label(last, text="Generated Password: ",font = ('timesnewroman', 12,'bold'))
Random_password.pack(side=LEFT,padx=5)
entry =Entry(last,width=30,font = ('timesnewroman', 15, 'bold'))
entry.pack(side=LEFT,padx=2,fill=Y)
c_label = Label(top, text="Length: ",font = ('timesnewroman', 12, 'bold'))
c_label.pack(side=LEFT,padx=5)
combo = Combobox(top, textvariable=var1,font = ('timesnewroman', 15))
combo['values'] = (4,5,6,7,8, 9, 10, 11, 12, 13, 14, 15, 16,
                  17, 18, 19, 20, 21, 22, 23, 24, 25,
                  26, 27, 28, 29, 30, 31, 32)
combo.current(0)
combo.bind('<<ComboboxSelected>>')
combo.pack(side=LEFT,padx=2)
a=Checkbutton(tops, text="DIGITS",variable=v1,onvalue=1,offvalue=0)
a.pack(side=LEFT,fill=BOTH)
b=Checkbutton(tops, text="UPPER", variable=v2,onvalue=2,offvalue=0)
b.pack(side=LEFT,fill=BOTH)
c=Checkbutton(tops, text="LOWER", variable=v3,onvalue=3,offvalue=0)
c.pack(side=LEFT,fill=BOTH)
d=Checkbutton(tops, text="SYMBOLS", variable=v4,onvalue=4,offvalue=0)
d.pack(side=LEFT,fill=BOTH)
radio_low = Radiobutton(radio,text="SINGLE PASSWORD", variable=var, value=1)
radio_low.pack(side=LEFT,fill=BOTH)
radio_strong = Radiobutton(radio, text="MULTIPLE PASSWORDS", variable=var, value=2)
```

```
radio_strong.pack(side=LEFT,fill=BOTH)
generate_button = Button(f2,text="Generate",command=generate)
generate_button.pack(side=LEFT,fill=BOTH)
copy_button = Button(f5, text="Copy",command=copy1)
copy_button.pack(side=LEFT)
copy_button1 = Button(f5, text="Exit",command=qExit)
copy_button1.pack(side=RIGHT)
btnReset = Button(f5,text= "Reset",command = Reset).pack(side=RIGHT)
lblInfo =Label(fw,font=('timesnewroman',12,'bold'),text = " Message: ")

lblInfo.pack(side=LEFT,padx=2,pady=2)
entry1 =Entry(fw,width=36,font = ('timesnewroman', 15, 'bold'))
entry1.pack(side=LEFT)
localtime = time.asctime(time.localtime(time.time()))
lblInfo = Label(f4, font=('timesnewroman',12,'bold'),text = localtime)

lblInfo.pack(side=BOTTOM)
root.mainloop()

'''END CODE'''
```


TESTING

6. TESTING

6.1 TESTING METHODOLOGIES

Testing Methodologies are the methods or approaches to testing that includes from Unit testing through System Testing.

Software Testing Methodology is defined as strategies and testing types used to certify that the Application Under Test meets client expectations. Test Methodologies include functional and non-functional testing to validate the AUT.

The following are the testing methodologies:

- Unit Testing
- Integration Testing
- User Acceptance Testing
- Output Testing
- Validation Testing

6.1.1 UNIT TESTING

Unit testing is the first level of testing and is often performed by the developers themselves. It is the process of ensuring individual components of a piece of software at the code level are functional and work as they were designed to. Developers in a test-driven environment will typically write and run the tests prior to the software or feature being passed over to the test team. Unit testing can be conducted manually, but automating the process will speed up delivery cycles and expand test coverage. Unit testing will also make debugging easier because finding issues earlier means they take less time to fix than if they were discovered later in the testing process. Test Left is a tool that allows advanced testers and developers to shift left with the fastest test automation tool embedded in any IDE.

6.1.2 INTEGRATION TESTING

After each unit is thoroughly tested, it is integrated with other units to create modules or components that are designed to perform specific tasks or activities. These are then tested as group through integration testing to ensure whole segments of an application behave as expected (i.e. the interactions between units are seamless). These tests are often framed by user scenarios, such as logging into an application or opening files. Integrated tests can be

conducted by either developers or independent testers and are usually comprised of a combination of automated functional and manual tests.

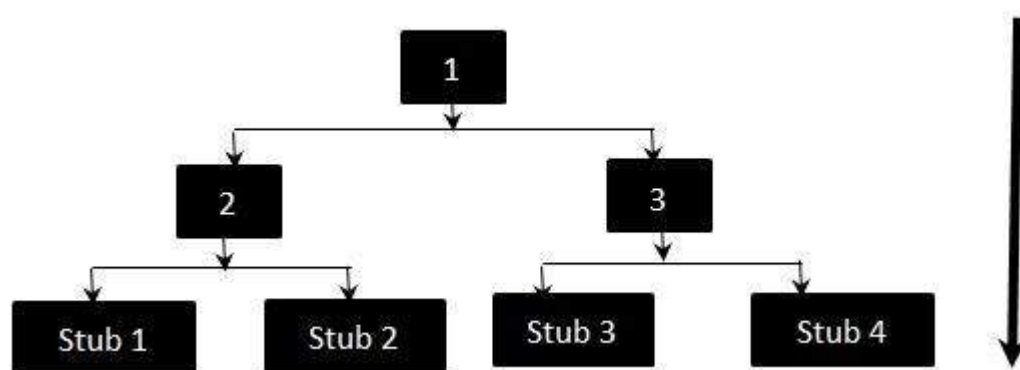
The following are the two types of Integration Testing:

1.Top Down Integration:

Top-down integration testing is an integration testing technique used in order to simulate the behaviour of the lower-level modules that are not yet integrated. Stubs are the modules that act as temporary replacement for a called module and give the same output as that of the actual product.

The replacement for the 'called' modules is known as 'Stubs' and is also used when the software needs to interact with an external system.

Stub - Flow Diagram:

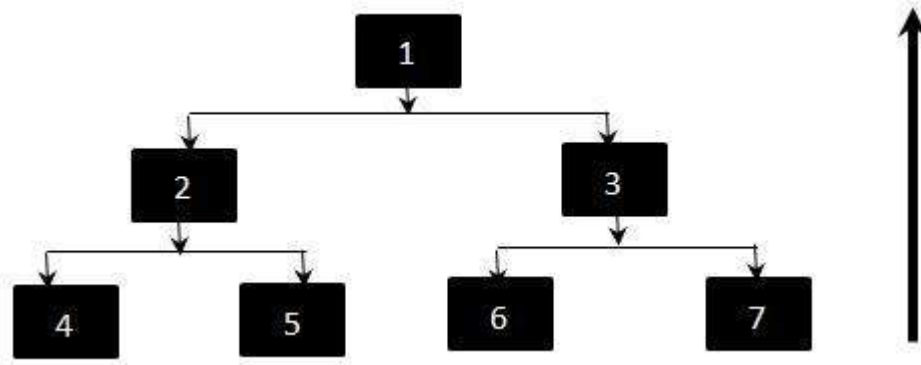


The above diagrams clearly states that Modules 1, 2 and 3 are available for integration, whereas, below modules are still under development that cannot be integrated at this point of time. Hence, Stubs are used to test the modules.

2.Bottom Up Integration

Each component at lower hierarchy is tested individually and then the components that rely upon these components are tested.

Bottom Up Integration - Flow Diagram



The order of Integration by Bottom-down approach will be:

Though Top level components are the most important, yet tested last using this strategy. In Bottom-up approach, the Components 2 and 3 are replaced by drivers while testing components 4,5,6,7. They are generally more complex than stubs.

6.1.3 ACCEPTANCE TESTING

Acceptance testing is the last phase of functional testing and is used to assess whether or not the final piece of software is ready for delivery. It involves ensuring that the product is in compliance with all of the original business criteria and that it meets the end user's needs. This requires the product be tested both internally and externally, meaning you'll need to get it into the hands of your end users for beta testing along with those of your QA team. Beta testing is key to getting real feedback from potential customers and can address any final usability concerns.

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes whether required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

User acceptance testing also important because failure to do it places an unnecessary burden on system developers who, while they be experts in development, are not familiar with the vagaries of running an organization or the day to day difficulties that the software they are developing must account for.

6.1.4 OUTPUT TESTING

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways: one is on screen and another in printed format.

6.1.6 VALIDATION TESTING

Validation checks are performed on the following fields.

Text Field:

The text field can contains only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

Numeric Field:

The numeric field acan contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with simple data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

Preparation of Test Data

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under the study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

6.1.7 TESTING STRATEGY

Testing Strategies is an overview of the key issues that occur in the testing process and is to be taken into consideration by the project manager, a team of developers and testers.

The above-discussed Software Testing Methods are used to implement n number of testing strategies.

SYSTEM TESTING:

System testing is a black box testing method used to evaluate the completed and integrated system, as a whole, to ensure it meets specified requirements. The functionality of the software is tested from end-to-end and is typically conducted by a separate testing team than the development team before the product is pushed into production.

Software once validated must be combined with other system elements (e.g. Hardware, people and database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

UNIT TESTING:

In unit testing different modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goal is to test the internal logic of the modules. Using the detailed design description as a guide, important control paths are tested to uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactory as regards to the expected output from the module.

In due course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this. The future holds a lot to offer to the development and refinement of this project.

6.2 TEST CASES FOR UNIT TESTING AND VALIDATION TESTING

S.No of the test case	R1
Name of the test case	Selecting Length
Sample Input	6ytr889
Expected Output	Error: Invalid Input Length
Actual Output	Same as expected
Remarks	This component displays error message if it is a alphanumeric value

S.No of the test case	R2
Name of the test case	Selecting Length
Sample Input	2
Expected Output	Select length (4-32)
Actual Output	Same as expected
Remarks	This component displays error message if the value is not between (4-32)

S.No of the test case	R3
Name of the test case	Selecting Length
Sample Input	-76
Expected Output	Error: Negative Value Length
Actual Output	Same as expected
Remarks	This component displays error message if the value is less than 0/Negative

S.No of the test case	R4
Name of the test case	Selecting Length
Sample Input	36
Expected Output	Enter the Length less than 32
Actual Output	Same as expected
Remarks	This component displays error message if the value is greater than 32

S.No of the test case	R5
Name of the test case	Selecting password pattern
Sample Input	If combo box are not selected
Expected Output	Tick Options
Actual Output	Same as expected
Remarks	This component displays error message if the digit/upper/lower/symbol options are not selected

S.No of the test case	R6
Name of the test case	Selecting radio button
Sample Input	If radio box is not selected
Expected Output	Select single/multiple
Actual Output	Same as expected
Remarks	This component displays error message if the single / multiple options are not selected

S.No of the test case	R7
Name of the test case	Choosing multiple passwords and selecting the count
Sample Input	Null
Expected Output	Enter password count
Actual Output	Same as expected
Remarks	This component displays error message if the count is not selected.

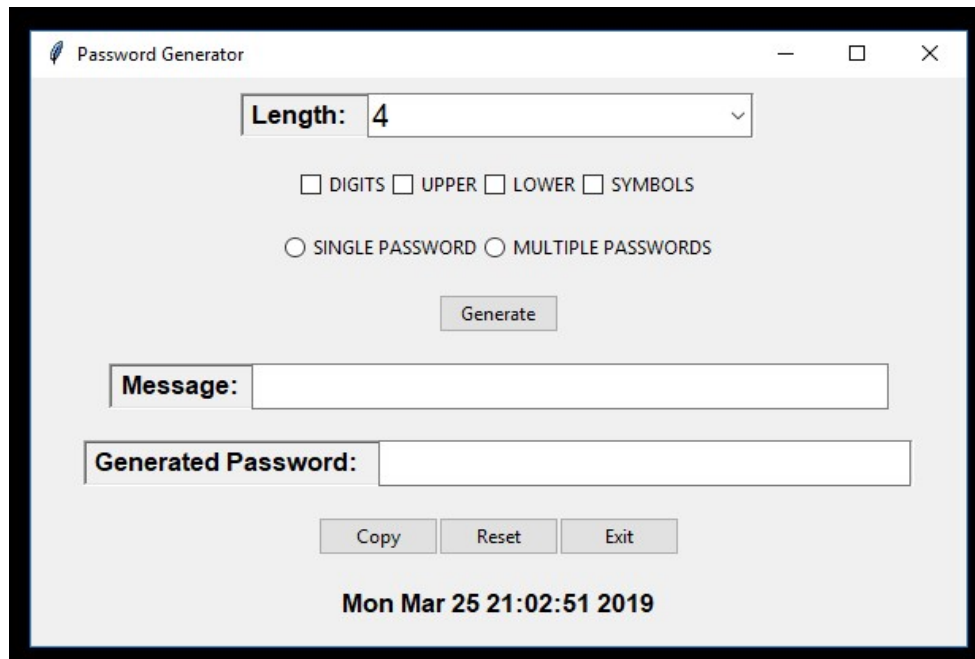
S.No of the test case	R8
Name of the test case	Choosing multiple passwords and selecting the count
Sample Input	-88
Expected Output	Error: Negative value!
Actual Output	Same as expected
Remarks	This component displays error message if the count value is negative

S.No of the test case	R9
Name of the test case	Choosing multiple passwords and selecting the count
Sample Input	0
Expected Output	Enter count greater than 0
Actual Output	Same as expected
Remarks	This component displays error message if the count value is 0

S.No of the test case	R10
Name of the test case	Choosing multiple passwords and selecting the count
Sample Input	U89
Expected Output	Error: Invalid Input count
Actual Output	Same as expected
Remarks	This component displays error message if the count value is alphanumeric

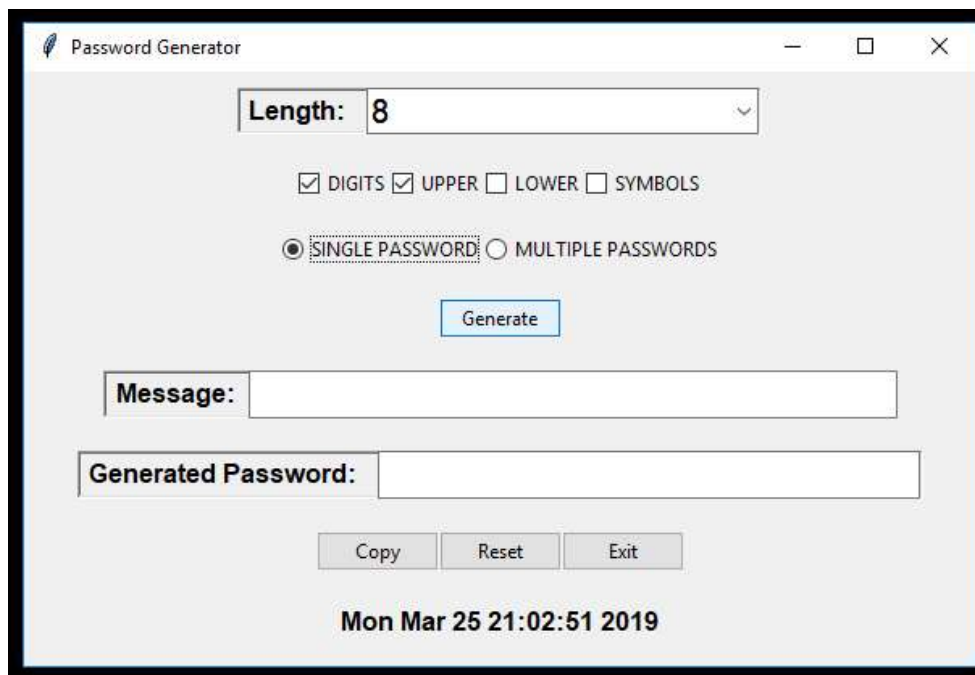
OUTPUT SCREENS

OUTPUT SCREENS



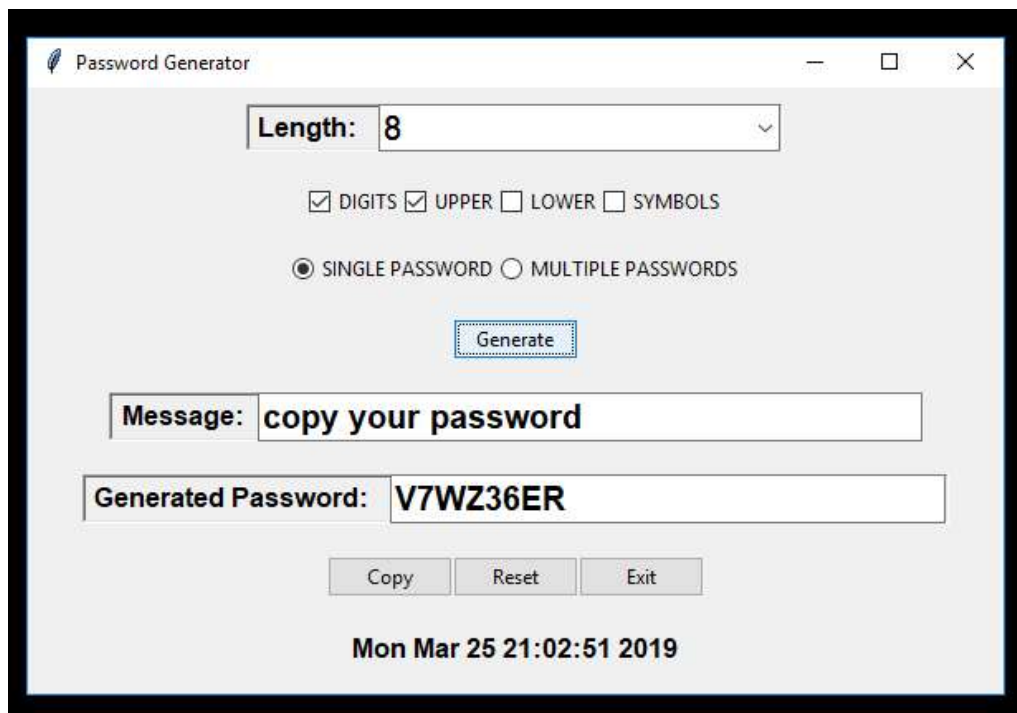
The screenshot shows the 'Password Generator' application window. The 'Length' dropdown is set to '4'. The checkboxes for 'DIGITS', 'UPPER', 'LOWER', and 'SYMBOLS' are all unchecked. The radio buttons for 'SINGLE PASSWORD' and 'MULTIPLE PASSWORDS' are both unselected. A 'Generate' button is visible. Below it, the 'Message:' and 'Generated Password:' text boxes are empty. At the bottom, there are 'Copy', 'Reset', and 'Exit' buttons, and a timestamp 'Mon Mar 25 21:02:51 2019'.

This is the default home page



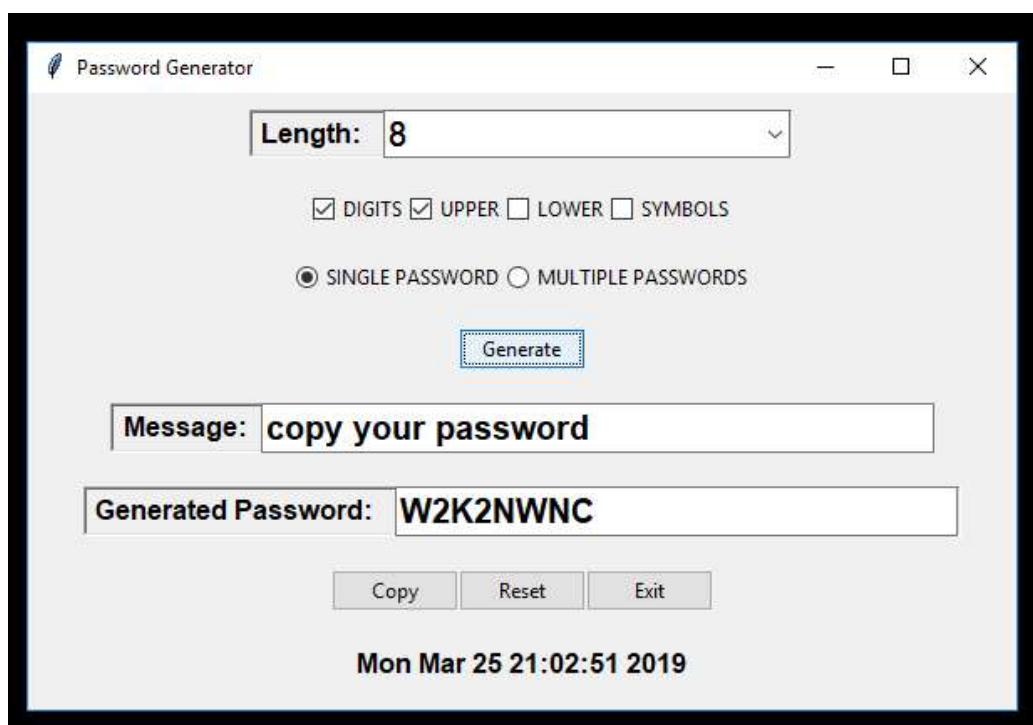
The screenshot shows the 'Password Generator' application window with updated settings. The 'Length' dropdown is now set to '8'. The checkboxes for 'DIGITS' and 'UPPER' are checked, while 'LOWER' and 'SYMBOLS' remain unchecked. The 'SINGLE PASSWORD' radio button is now selected. The 'Generate' button is highlighted with a blue border. The 'Message:' and 'Generated Password:' text boxes remain empty. The 'Copy', 'Reset', and 'Exit' buttons are still present at the bottom, along with the same timestamp 'Mon Mar 25 21:02:51 2019'.

After selecting the options click on generate



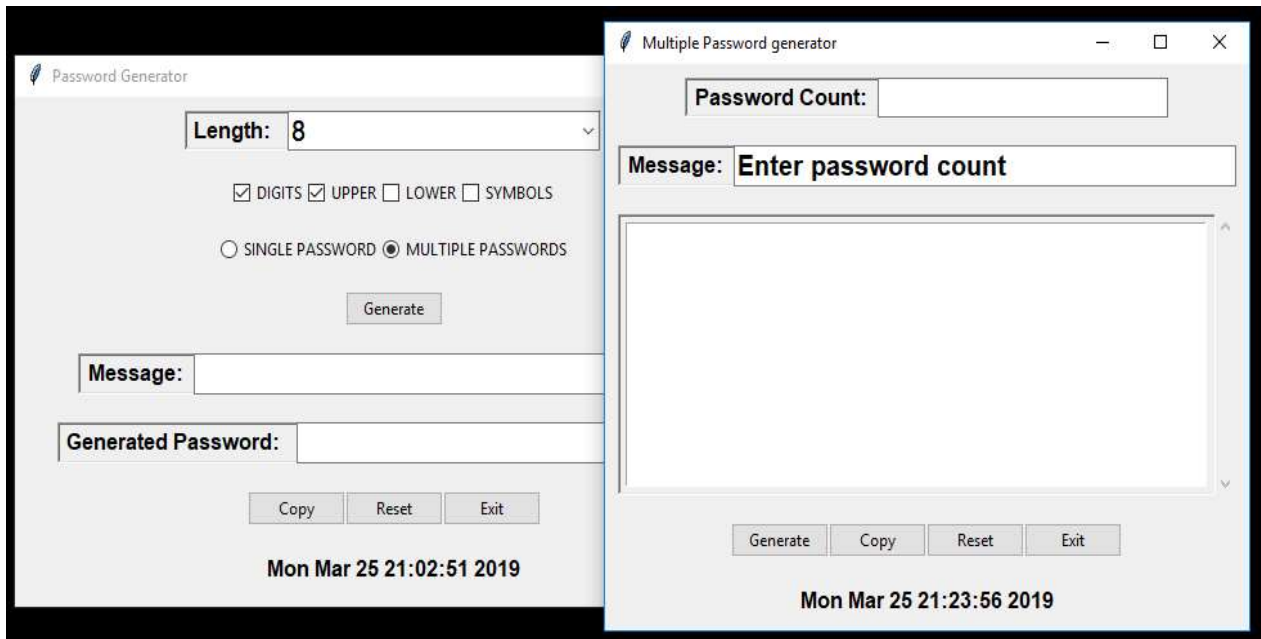
The screenshot shows a window titled "Password Generator" with a feather icon. It features a "Length:" dropdown menu set to "8". Below it are checkboxes for "DIGITS", "UPPER", "LOWER", and "SYMBOLS", all of which are checked. There are two radio buttons: "SINGLE PASSWORD" (selected) and "MULTIPLE PASSWORDS". A "Generate" button is highlighted with a blue dashed border. Below the button is a "Message:" label followed by a text box containing "copy your password". Underneath is a "Generated Password:" label followed by a text box containing "V7WZ36ER". At the bottom are three buttons: "Copy", "Reset", and "Exit". The timestamp "Mon Mar 25 21:02:51 2019" is displayed at the very bottom.

Password will Generated after clicking on Generate Button

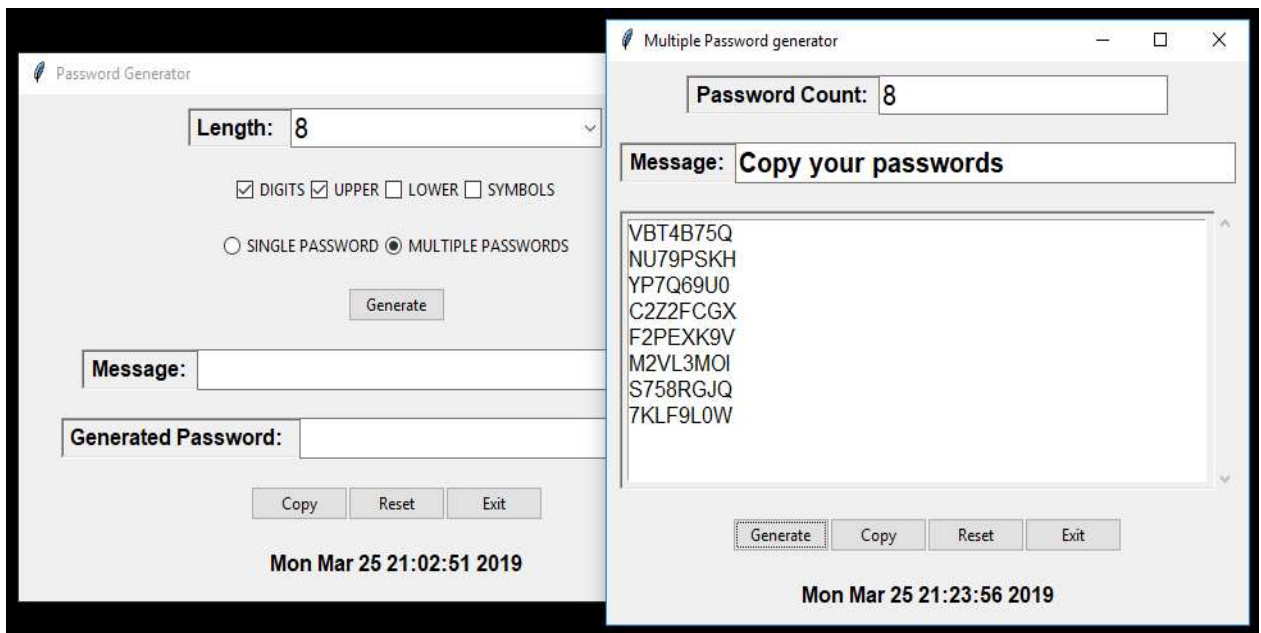


This screenshot shows the same "Password Generator" application window. The settings are identical to the first screenshot: "Length" is 8, "DIGITS", "UPPER", and "SYMBOLS" are checked, and "SINGLE PASSWORD" is selected. The "Generate" button is again highlighted. The "Message:" text box still says "copy your password". However, the "Generated Password:" text box now displays "W2K2NWNC", indicating a new password has been generated. The "Copy", "Reset", and "Exit" buttons remain at the bottom, along with the same timestamp "Mon Mar 25 21:02:51 2019".

Every time when we click on Generate Button a new password will be generated



This is for the Multiple Password Generator



After entering the count, click on generate Button in Multiple Password Generator to generate multiple passwords

CONCLUSION

CONCLUSION

Password Generator is a utility that allows you to generate a random password with a single click. This software program is useful when you often need to create passwords or passcodes.

You can think of a password yourself; however, these passwords may be vulnerable to security flaws. People often include their name or favorite number in the password, which makes the password insecure. The advantage of a password generator is that it generates random passwords, which are not based on your life experience. Even if the person who tries to break into your personal account knows a lot about you, he or she cannot guess the password generated with Password Generator.

You can include numbers and upper-case letters into your password to make it even stronger. Also, you can select the password length. Remember, the longer the password is the harder it is to break. However, it is not easy to remember a very long random password. So make sure you can remember your passwords. If in doubt, write the password down and store it in a safe place.

REFERENCES

REFERENCES

1. Password Authentication

https://en.wikipedia.org/wiki/Category:Password_authentication

2. Application of randomness

https://en.wikipedia.org/wiki/Category:Applications_of_randomness

3. RFC 4086 on Randomness Recommendations for Security (Replaces earlier RFC 1750.)

<https://www.ietf.org/rfc/rfc4086.txt>

4. "Linux / UNIX: Generating Random Password With mkpasswd / makepasswd / pwgen". www.cyberciti.biz. Retrieved 2016-03-25.

<http://www.cyberciti.biz/faq/generating-random-password/>

5. "StrongPasswords - Community Help Wiki". help.ubuntu.com. Retrieved 2016-03-25.

<https://help.ubuntu.com/community/StrongPasswords>

6. NIST. Automated Password Generator standard FIPS 181

<http://www.itl.nist.gov/fipspubs/fip181.htm>

7. Shay, Richard; Kelley, Patrick Gage; Komanduri, Saranga; Mazurek, Michelle L.; Ur, Blase; Vidas, Timothy; Bauer, Lujo; Christin, Nicolas; Cranor, Lorrie Faith (2012). Correct horse battery staple: Exploring the usability of system-assigned passphrases (PDF). SOUPS '12 Proceedings of the Eighth Symposium on Usable Privacy and Security. doi:10.1145/2335356.2335366.

<http://www.andrew.cmu.edu/user/nicolasc/publications/Shay-SOUPS12.pdf>

<https://doi.org/10.1145%2F2335356.2335366>

8. Ganesan, Ravi; Davies, Chris (1994). "A New Attack on Random Pronounceable Password Generators" (PDF). Proceedings of the 17th {NIST}-{NCSC} National Computer Security Conference. NIST: 184–197. Retrieved 2014-12-17.

<http://csrc.nist.gov/publications/history/nissc/1994-17th-NCSC-proceedings-vol-1.pdf>

9. "9.6. random — Generate pseudo-random numbers — Python 3.5.1 documentation". docs.python.org. Retrieved 2016-03-25.

<https://docs.python.org/py3k/library/random.html>

10. "16.1. os — Miscellaneous operating system interfaces — Python 3.5.1 documentation". docs.python.org. Retrieved 2016-03-25.

<https://docs.python.org/py3k/library/os.html#os.urandom>

11. Levine, John R., Ed.: Internet Secrets, Second edition, page 831 ff. John Wiley and Sons

12. Schneier, B: Applied Cryptography, Second edition, page 233 ff. John Wiley and Sons.

13. "Electronic Authentication Guideline" (PDF). NIST. Retrieved March 27, 2008.

http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf

14. Code club project

<https://codeclubprojects.org/en-GB/python/password-generator/>

15. Password Generator Solutions (Exercise 16)

<https://www.practicepython.org/solution/2014/06/06/16-password-generator-solutions.html>

16. GeeksForGeeks

<https://www.geeksforgeeks.org/python-random-password-generator-using-tkinter/>

17. Trinket (Put Interactive Python Anywhere on the Web)

<https://trinket.io/python/08c0ad3359>

18. Password generator using Python · GitHub

<https://gist.github.com/rahulrajaram/1479f4f017e2ca752f1f45cdb878b3d2>

19. Python software Foundation(pip install random-password-generator)

<https://pypi.org/project/random-password-generator/>

20. Python GUI - tkinter - GeeksforGeeks

<https://www.geeksforgeeks.org/python-gui-tkinter/>

21. Graphical User Interfaces with Tk — Python 3.7.3rc1 documentation

<https://docs.python.org/3/library/tk.html>

22. Introduction To GUI With Tkinter In Python (article) - DataCamp

<https://www.datacamp.com/community/tutorials/gui-tkinter-python>

23. Python GUI Programming (Tkinter) - Tutorialspoint

https://www.tutorialspoint.com/python/python_gui_programming.htm

24. Secure Password Generator

<https://passwordsgenerator.net/>

25. RANDOM.ORG (Randomness and Integrity services Ltd)

<https://www.random.org/passwords/>

26. C. Herley and P. C. van Oorschot, “A research agenda acknowledging the persistence of passwords,” IEEE Security & Privacy, vol. 10, no. 1, pp. 28–36, 2012.

27. D. Florêncio, C. Herley, and P. C. van Oorschot, “Password portfolios and the finite-effort user: Sustainably managing large numbers of accounts,” in Proc. 23rd USENIX Security Symposium. USENIX Association, 2014, pp. 575–590.

28. J. A. Halderman, B. Waters, and E. W. Felten, “A convenient method for securely managing passwords,” in Proc. WWW 2005, A. Ellis and T. Hagino, Eds. ACM, 2005, pp. 471–479.

29. A. H. Karp, “Site-specific passwords,” HP Laboratories, Palo Alto, Tech. Rep. HPL-2002-39 (R.1), May 2003.

30. M. Mannan and P. C. van Oorschot, “Passwords for both mobile and desktop computers: ObPwd for Firefox and Android,” USENIX ;login, vol. 37, no. 4, pp. 28–37, August 2012.

31. B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell, “Stronger password authentication using browser extensions,” in Proc. 14th USENIX Security Symposium, P. McDaniel, Ed. USENIX Association, 2005, pp. 17–32.

32. R. Wolf and M. Schneider, “The passwordsitter,” Fraunhofer Institute for Secure Information Technology (SIT), Tech. Rep., May 2006.
33. K.-P. Yee and K. Sitaker, “Passpet: Convenient password management and phishing protection,” in Proc. SOUPS 2006, L. F. Cranor, Ed. ACM, 2006, pp. 32–43.
34. F. A. Maqbali and C. J. Mitchell, “Password generators: Old ideas and new,” in Proc. WISTP 2016, ser. LNCS, S. Foresti and J. Lopez, Eds., vol. 9895. Springer, 2016, pp. 245–253.
35. D. McCarney, “Password managers: Comparative evaluation, design, implementation and empirical analysis,” Master’s thesis, Carleton University, August 2013, available at <https://danielmccarney.ca/assets/pubs/McCarney.MCS.Archive.pdf>.
36. R. Biddle, M. Mannan, P. C. van Oorschot, and T. Whalen, “User study, analysis, and usable security of passwords based on digital objects,” IEEE Trans. Inf. Forensics & Security, vol. 6, no. 3, pp. 970–979, 2011.

