**PASSWORD MANAGER**

**PROJECT REPORT**

**Submitted By**

KARUNAGARAN V 21500147

KARTHIK S21500145

RAJAGOPAL S 21500160

ARAVINDAN D 21500131

ARJUNAN D 21500132

BHUVANESH K 21500134

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**PROJECT REPORT 2022-2023**



Under the Guidance of

**MRS.R.BAKIALAKSHMY M.Tech.,**

Lecturer,

Department of Information Technology

In Partial Fulfilment for Requirement of Award of

**DIPLOMA IN INFORMATION TECHNOLOGY**

**MOTILAL NEHRU GOVT. POLYTECHNIC COLLEGE LAWSPET**

**PUDUCHERRY-605008**

**MOTILAL NEHRU GOVT. POLYTECHNIC COLLEGE LAWSPET,**

**PUDUCHERRY-605008**

**DEPARTMENT OF INFORMATION TECHNOLOGY**



**BONAFIDE CERTIFICATE**

Certified that the project work titled PASSWORD MANAGER is the Bonafide Record of work done by KARUNAGARAN V [21500147], KARTHIK S [21500145], RAJAGOPAL S [21500160], ARAVINDAN D [21500131], ARJUNAN D [21500132], BHUVANESH K [21500134] in the partial fulfilment of the requirement for the award of Diploma in Information Technology during the year 2022-2023.

HEAD OF THE DEPARTMENT PROJECT GUIDE

Mrs.R.BAKIALAKSHMY, M.Tech,

LECTURER DEPT.OF IT.

Submitted for the Practical Exam held on ……………………………..

INTERNAL EXAMINER EXTERNAL EXAMINER

**ACKNOWLEGEMENT**

We would like to express our sincere thanks to our beloved Principal

**Mr. V. PASSOUPATHI, M.Tech., FF(AUS).,** for having provided necessary facilities we needed to complete this project.

We wish to extend our sincere gratitude to our Head of Department

**Dr. S. JANARTHANAN, M.Tech., Ph.D.,** and the Academic In

Charge **Mr. S. MAHENDRAN, M.Tech.,** for providing me the guidelines to complete the project.

We are internally indebted to our Project guide **Mrs. R. BAKIALAKSHMY M.Tech.,** for her extensive patience and excellent guidance throughout this project work. We also extend our heart-felt gratitude for her constant encouragement, inspiration, co-operation and valuable directions that helped us in successful completion of this project.

We would like to extend our sincere thanks to the following faculty members of Department of Information Technology.

**Mr.S.DATCHINAMOORTHY M.Tech.,**

**Mr.K. JANAKIRAMAN B.Tech.,**

**Dr.P.BANUMATHY PhD.,**

We thank all our family members and friends, who have been with us in all our trials and tribulations, without whose co-operations and encouragement, this project would not have come to lime light.

Above all we are grateful to thank “THE ALMIGHTY” for giving us courage and strength to complete this project successfully.

**ABSTRACT**

The purpose of this project is to provide a secure password and manage all the accounts. The given password will be stored in the database using an encryption algorithm. The user will store their online account passwords for things like Gmail accounts, Office accounts, all web application accounts, and much more. A QR code generator will also be provided in this system.

**TABLE OF CONTENTS**

1. INTRODUCTION………………………………………………………….1
   1. OVERVIEW OF THE PROJECT
   2. PROBLEM DESCRIPTION
2. SYSTEM SPECIFICATION……………………………………………….2

2.1 HARDWARE REQUIREMENTS

2.2 SOFTWARE REQUIREMENTS

1. FEASIBILITY STUDIES………………………………………………….3

3.1 EXISTING SYSTEM

3.2 PROPOSED SYSTEM

1. SOFTWARE DESCRIPTION……………………………………………..4

4.1 FRONT END

4.2 BACK END

1. PROJECT DESCRITION…………………………………………………10

5.1 NAME OF THE MODULES

5.2 MODULE DESCRIPTION

1. DATA DICTIONARY……………………………………………………12

6.1 DATAFLOW DIAGRAM

6.2 DATABASE DESIGN

1. SYSTEM TESTING………………………………………………………15

7.1 TYPES OF TESTING

1. CONCLUSION AND FUTURE ENHANCEMENT……………………..16
2. BIBLIOGRAPHY

BOOK

REFERENCE

CODING

SCREENSHOTS…………………………………………………………..25

1. **INTRODUCTION**
   1. **OVERVIEW OF THE PROJECT**

The main objective of developing this project is to provide a Secure Password Manager. This application will manage your digital accounts. It will create a secure password and store it in database. The given password will be stored in the database using an encryption algorithm. The user will store their online account ID, PASSWORDS, MAIL for things like Gmail accounts, Office accounts, all web application accounts, and much more. This makes you to easily store user accounts. We provide two features in our project. The features are **QR code generator and password generator**. A QR code generator using url will create a qr code for given link to access the web application anywhere. A password generator will also be provided in this system. In this feature user can make it easy to access any website. User can generate QR code and scan it in any mobile. The whole credentials will encrypted and decrypted when we want.

**1.2 PROBLEM DESCRIPTION**

Passwords are stolen all the time. Sites and services are at constant risk of breaches as much as you are to phishing attacks that try to trick you into handing over your password. Although companies are meant to scramble your password when they store it a process known as hashing not all use strong or modern algorithms, making it easy for hackers to reverse that hashing and read your password in plain text. Some companies don’t bother to hash at all! That puts your accounts at the risk of fraud or your data at risk of being used against you for identity thefts. But the longer and more complex your passwords are, the longer it takes for hackers to unscramble them. That can be a password that’s a mix of uppercase and lowercase characters, numbers, symbols and punctuation — or, as many are moving towards, the use of deliberately lengthy pass-phrases, which make up several unique words that can be easily remembered but can be far stronger than shorter passwords.

**2.SYSTEM SPECIFICATION**

**2.1 HARDWARE REQUIREMENTS**

* Processor : Dual core processor 2.6.0 GHZ
* RAM : 4GB
* Hard Disk : 160 GB
* Compact Disk : 650 MB
* Keyboard : Standard Keyboard
* Monitor : 15 inch color monitor.

**2.2 SOFTWARE REQUIREMENTS**

* Operating System : Windows 10
* Front End : HTML, CSS, JAVASCRIPT
* Back End : Python with Django
* Tool : Visual Studio Code, Django server

1. **FEASIBILITY STUDIES**

**3.1 EXISTING SYSTEM**

The existing password management system only store their passwords. The users will create their password strongly that will complicate the user to create so they create a simple password like karan, karan.2005. In our password manager we generate a secure password to the user and using CRUD method. In existing system they charge to use their product.

Limitations in Existing System:

* They provide few access for free users then we need to pay for all features.
* The current system only provide to store passwords
* In current system is manually need to create passwords

DISADVANTAGES OF EXISTING SYSTEM

* User manually need to create password
* They charge amount for subscriptions

**3.2 PROPOSED SYSTEM**

In the proposed system we implement the following features: User will not always create their passwords we provide them to generate strong passwords. We provide QR Code generating system by giving web application url this will help when we want to change the application in any device we use this feature to change easily. We provide all the features for free.

ADVANTANGES OF PROPOSED SYSTEM

* The proposed system is completely free to use
* The user generate and store their passwords easily
* This system will provide QR code generator for access website in any device

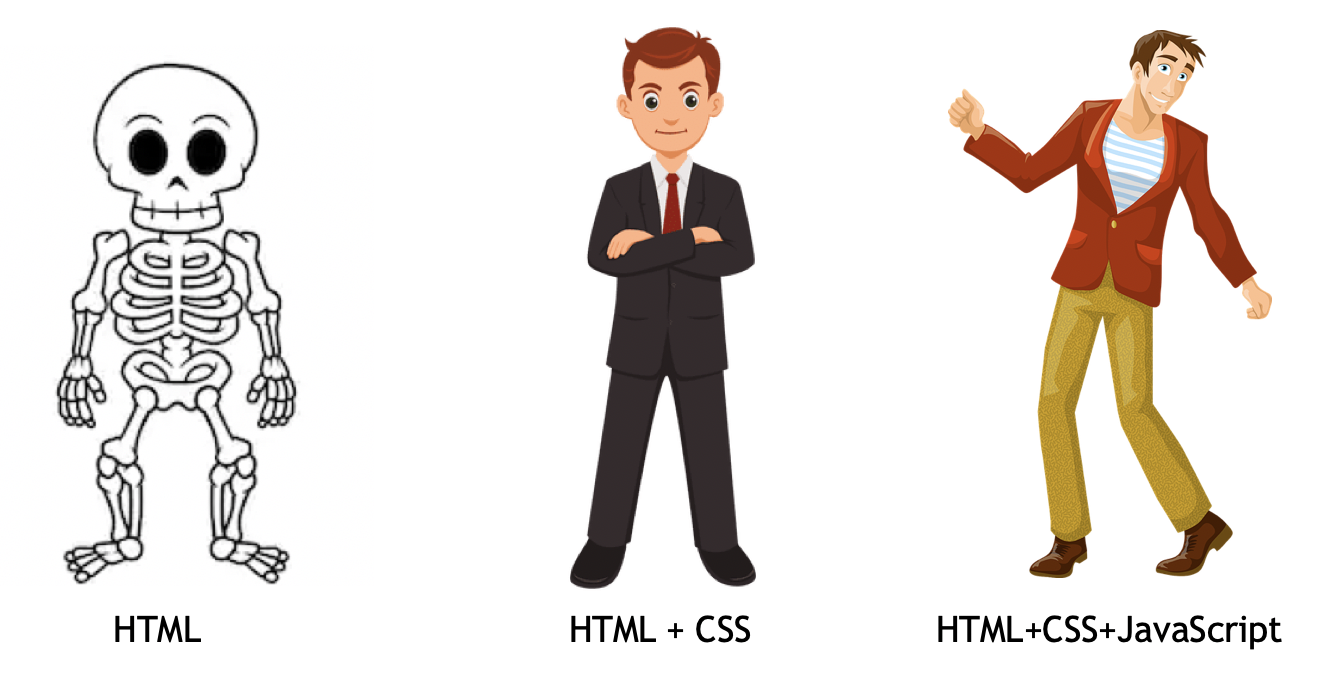
**4. SOFTWARE DESCRIPTION**

**4.1 FRONT END**

HTML: **HTML** stands for **HyperText Markup Language**. It is used to design web pages using the **markup language**. HTML is the combination of **Hypertext** and **Markup language**. Hypertext defines the link between the web pages and markup language defines the text document within the tag that define the structure of web pages.

CSS : **CSS (Cascading Style Sheets)**is used to apply styles to web pages. Cascading Style Sheets are fondly referred to as CSS. It is used to make web pages presentable. The reason for using this is to simplify the process of making web pages presentable. It allows you to apply styles on web pages. More importantly, it enables you to do this independently of the HTML that makes up each web page.

JAVASCRIPT : **JavaScript (JS)** is the most popular lightweight, interpreted compiled programming language. It can be used for both [**Client-side**](https://www.geeksforgeeks.org/server-side-client-side-programming/) as well as [**Server-side**](https://www.geeksforgeeks.org/server-side-client-side-programming/) developments. JavaScript also known as a scripting language for web pages.



Accessing an HTML Page

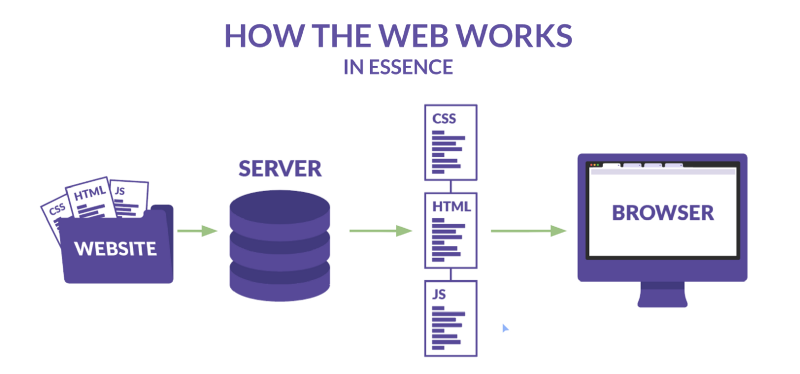
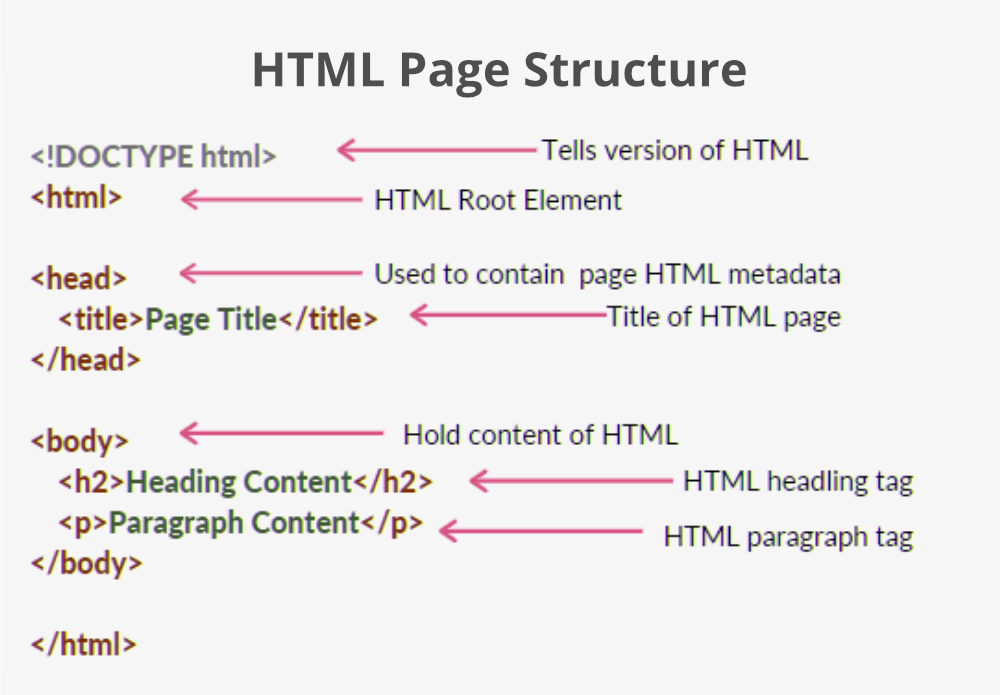


Figure shows: Accessing an HTML, CSS, JAVASCRIPT Page

* Your browser sends a request to that web page’s server (computer) for the file you wish to view.
* The web server sends the file requested back to your computer.
* Your browser displays the file appropriately.

HTML Syntax:

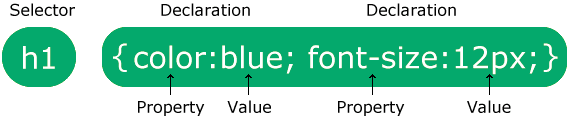
The basic structure of an HTML page is laid out below. It contains the essential building-block elements (i.e. doctype declaration, HTML, head, title, and body elements) upon which all web pages are created.



* [**<DOCTYPE! html>**](https://www.geeksforgeeks.org/html-doctypes/) **–** A doctype or document type declaration is an instruction that tells the web browser about the markup language in which the current page is written. It is not an element or tag. The doctype declaration is not case-sensitive.
* [**<html>**](https://www.geeksforgeeks.org/html-html-tag/) **–** This tag is used to define the root element of HTML document. This tag tells the browser that it is an HTML document. It is the second outer container element that contains all other elements within it.
* [**<head>**](https://www.geeksforgeeks.org/html-head-tag/) **–** This tag is used to define the head portion of the HTML document that contains information related to the document. Elements within the head tag are not visible on the front-end of a webpage.
* [**<body>**](https://www.geeksforgeeks.org/html-body-tag/) **–** The body tag is used to enclose all the visible content of a webpage. In other words, the body content is what the browser will show on the front end.

CSS Syntax:

A CSS rule consists of a selector and a declaration block.



The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

Example:

p {  
color : red;  
text-align : center;  
}

Example explained:

* p is a selector in CSS (it points to the HTML element you want to style: <p>).
* color is a property, and red is the property value
* text-align is a property, and center is the property value

Javascript Syntax:

JavaScript syntax is the set of rules, how JavaScript programs are constructed:

// How to create variables:  
var x;  
let y;  
  
// How to use variables:  
x = 5;  
y = 6;  
let z = x + y;

# JavaScript Data Types

1. String  
2. Number  
3. Bigint  
4. Boolean  
5. Undefined  
6. Null  
7. Symbol  
8. Object

### The Object Datatype

The object data type can contain:

1. An object  
2. An array  
3. A date

**The Concept of Data Types**

In programming, data types is an important concept. To be able to operate on variables, it is important to know something about the type. Without data types, a computer cannot safely solve this:

let x = 16 + "Volvo";

Does it make any sense to add "Volvo" to sixteen? Will it produce an error or will it produce a result?

JavaScript will treat the example above as:

let x = "16" + "Volvo";

JAVASCRIPT FUNCTION :

A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function is executed when "something" invokes it (calls it).

**JavaScript Function Syntax**

A JavaScript function is defined with the function keyword, followed by a **name**, followed by parentheses **()**.

Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

The parentheses may include parameter names separated by commas:  
**(*parameter1, parameter2, ...*)**

The code to be executed, by the function, is placed inside curly brackets: **{}**

CODE :

function *name*(*parameter1,parameter2,parameter3*) {  
  // *code to be executed*  
}

Function **parameters** are listed inside the parentheses () in the function definition.

Function **arguments** are the **values** received by the function when it is invoked.

Inside the function, the arguments (the parameters) behave as local variables.

**4.2 BACK END**

**PYTHON**

Python is a high-level, general-purpose, and very popular programming language. Python programming language (latest Python 3) is being used in web development, Machine Learning applications, along with all cutting-edge technology in Software Industry.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following:

* [Machine Learning](https://www.geeksforgeeks.org/machine-learning/)
* GUI Applications (like [Kivy](https://www.geeksforgeeks.org/kivy-tutorial/), Tkinter, PyQt etc. )
* Web frameworks like [Django](https://www.geeksforgeeks.org/django-tutorial/) (used by YouTube, Instagram, Dropbox)
* Image processing (like [OpenCV](https://www.geeksforgeeks.org/opencv-python-tutorial/), Pillow)
* Web scraping (like Scrapy, BeautifulSoup, Selenium)
* Test frameworks
* Multimedia
* Scientific computing
* Text processing and many more..

## Features in Python

There are many features in Python, some of which are discussed below as follows:

### 1. Free and Open Source

[Python](https://www.geeksforgeeks.org/python-programming-language/) language is freely available at the official website and you can download it from the given download link below click on the **Download Python** keyword. [Download Python](https://www.python.org/downloads/) Since it is open-source, this means that source code is also available to the public. So you can download it, use it as well as share it.

### 2. Easy to code

Python is a [high-level programming language](https://www.geeksforgeeks.org/difference-between-high-level-and-low-level-languages/). Python is very easy to learn the language as compared to other languages like C, C#, Javascript, Java, etc. It is very easy to code in the Python language and anybody can learn Python basics in a few hours or days. It is also a developer-friendly language.

### 3. Easy to Read

As you will see, learning Python is quite simple. As was already established, Python’s syntax is really straightforward. The code block is defined by the indentations rather than by semicolons or brackets.

### 4. Object-Oriented Language

One of the key features of [Python is Object-Oriented programming](https://www.geeksforgeeks.org/python-oops-concepts/). Python supports object-oriented language and concepts of classes, object encapsulation, etc.

### 5. GUI Programming Support

Graphical User interfaces can be made using a module such as [PyQt5](https://www.geeksforgeeks.org/pyqt5-qaction/), PyQt4, wxPython, or [Tk in python](https://www.geeksforgeeks.org/python-gui-tkinter/). PyQt5 is the most popular option for creating graphical apps with Python.

### 6. High-Level Language

Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture, nor do we need to manage the memory.

### 7. Extensible feature

Python is an **Extensible** language. We can write some Python code into C or C++ language and also we can compile that code in C/C++ language.

### 8. Easy to Debug

Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program’s issues once you understand how to [interpret](https://www.geeksforgeeks.org/difference-between-compiled-and-interpreted-language/) Python’s error traces. Simply by glancing at the code, you can determine what it is designed to perform.

### 9. Python is a Portable language

Python language is also a portable language. For example, if we have Python code for windows and if we want to run this code on other platforms such as [Linux](https://www.geeksforgeeks.org/introduction-to-linux-operating-system/), Unix, and Mac then we do not need to change it, we can run this code on any platform.

**5. PROJECT DESCRIPTION**

**5.1 NAME OF THE MODULES**

* Sign up
* Sign in
* Add Password
* Copy
* Delete Password
* QR Code Generator
* Logout

**5.2 MODULE DESCRIPTION**

* Sign Up

In this module, the user can sign up in the system using his/her username, email, password and confirm password.

* Sign In

In this module, the user can login in the system using his/her username and password.

* Add Password

In this module, the user can add the account information like social media account name, mail id, account password.

* Delete Password

In this module, the user can delete his/her password card.

* Copy Password

In this module, the user can copy the credentials.

* Generate Password

In this module, the user can generate a strong password with multiple options.

* QR Code Password

In this module, the user can generate a QR code using UR.

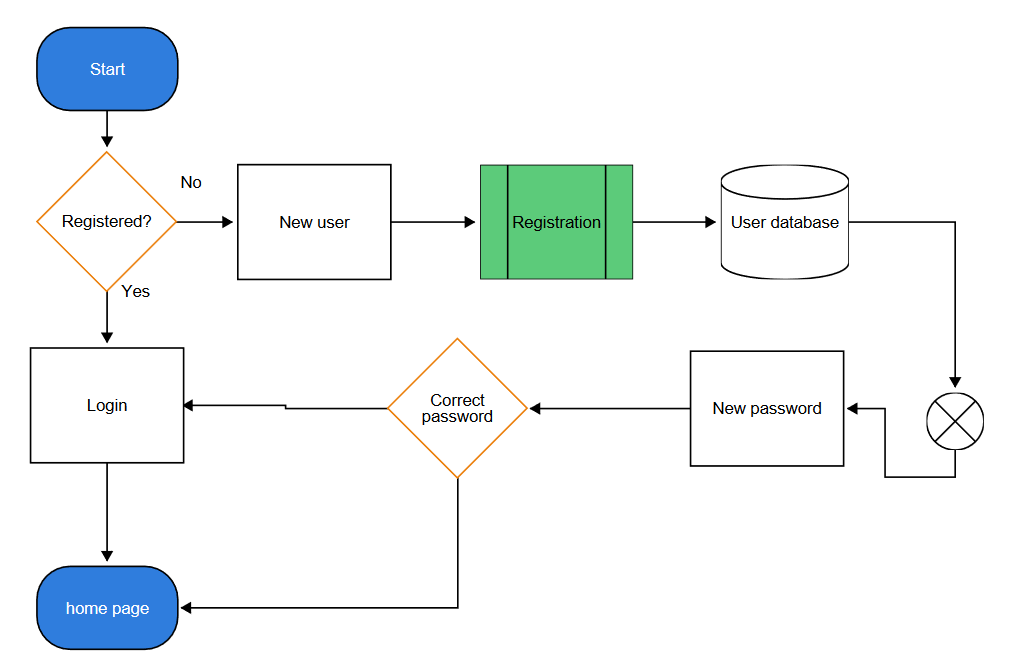
* Logout

In this module, the user can logout.

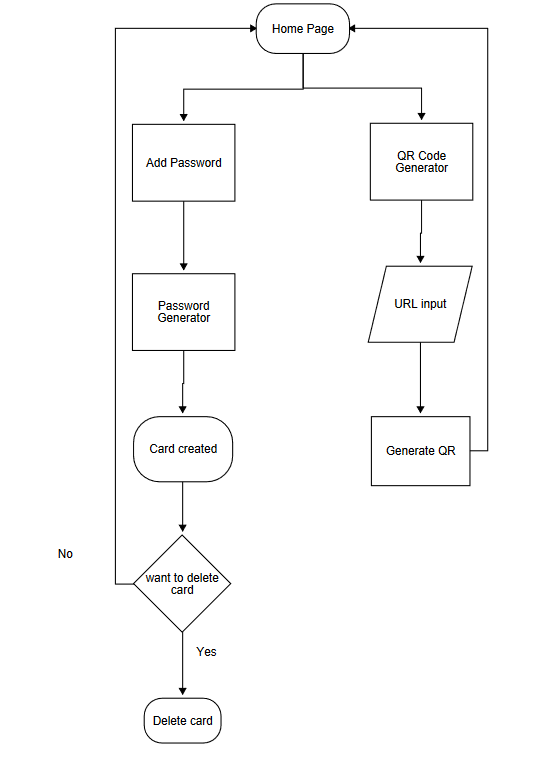
**6. DATA DICTIONARY**

**6.1 DATAFLOW DIAGRAM**

* Registration Page



* Home Page



**6.2 DATABASE DESIGN**

A table is a data structure that organizes information into rows and columns. It can be used to both store and display data in a structured format. For example, databases store data in tables so that information can be quickly accessed from specific rows. Websites often use tables to display multiple rows of data on page. Spreadsheets combine both purposes of a table by storing and displaying data in a structured format.

Databases often contain multiple tables, with each one designed for a specific purpose. For example, a company database may contain separate tables for employees, clients, and suppliers. Each table may include its own set of fields, based on what data the table needs to store. In database tables, each field is considered a column, while each entry, is considered a row. A specific value can be accessed from the table by requesting data from an individual column and row.

**Table name: User account**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Default** |
| User Name | int(20) | No | None |
| Email | varchar(20) | Yes | NULL |
| Password | varchar(30) | Yes | NULL |

**Table Name: Add account password**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Default** |
| Title | varchar(30) | No | None |
| url | varchar(40) | No | None |
| Email | varchar(30) | No | None |
| Password | varchar(50) | No | None |

**7. SYSTEM TESTING**

**7.1 TYPES OF TESTING**

* SYSTEM TESTING

System testing is a type of software testing that evaluates the overall functionality and performance of a complete and fully integrated software solution. It tests if the system meets the specified requirements and if it is suitable for delivery to the end-users. This type of testing is performed after the integration testing and before the acceptance testing.

**System Testing** is a type of [software testing](https://www.geeksforgeeks.org/software-testing-basics/) that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behavior of a component or a system when it is tested. **System Testing** is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. System testing tests the design and behavior of the system and also the expectations of the customer. It is performed to test the system beyond the bounds mentioned in the [software requirements specification (SRS)](https://www.geeksforgeeks.org/software-engineering-quality-characteristics-of-a-good-srs/). System Testing is basically performed by a testing team that is independent of the development team that helps to test the quality of the system impartial. It has both functional and non-functional testing. **System Testing is a black-box testing**. System Testing is performed after the integration testing and before the acceptance testing.

* UNIT TESTING

**Unit Testing** is a software testing technique by means of which individual units of software i.e. group of computer program modules, usage procedures, and operating procedures are tested to determine whether they are suitable for use or not. It is a testing method using which every independent module is tested to determine if there is an issue by the developer himself. It is correlated with the functional correctness of the independent modules. Unit Testing is defined as a type of software testing where individual components of a software are tested. Unit Testing of the software product is carried out during the development of an application. An individual component may be either an individual function or a procedure. Unit Testing is typically performed by the developer. In SDLC or V Model, Unit testing is the first level of testing done before integration testing. Unit testing is such a type of testing technique that is usually performed by developers. Although due to the reluctance of developers to test, quality assurance engineers also do unit testing.

* INTEGRATION TESTING

Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.

* ACCEPTANCE TESTING

Acceptance testing is aquality assurance (QA) process that determines to what degree an application meets end users' approval. Depending on the organization, acceptance testing might take the form of beta testing, application testing, field testing or end-user testing.

1. **CONCLUSION**

This project entitled as “Password Manager” has been developed to satisfy all the proposed requirements. The process of storing accounts credentials is more simple and easy. The system reduces the possibility of errors to a great extent and maintains the data is an efficient manner. User friendliness is the unique feature of this system. The system has a feature of secure password, QR code when required. The system is highly interactive and flexible for further enhancement. The coding is done in a simplified and easy to understandable manner so that other team trying to enhance the project can do so without facing much difficulty. The documentation will also assist in the process as it has also been carried out in a simplified and concise way.

**FUTURE ENHANCEMENT**

In future we can develop this project in extension type and bot application with extra feature using secure algorithm.

1. **BIBLIOGRAPHY**

**REFERENCES**

**BOOK REFERENCE**

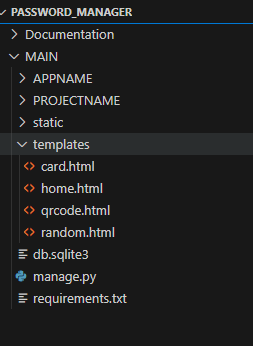
* + The author William S. Web Development with Django.
  + Andrew Hoffman, 2020. Web Development.

**WEBSITE REFERENCE**

* + <https://www.w3schools.com/>
  + <https://www.javatpoint.com/>

**CODING**

**FOLDER ARCHITECTURE**



Fig(1)

home.html

{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Password Manager</title>

<!--LINKING FILE 🡪

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css">

<link rel="stylesheet" href="{% static '../static/bootstrap/css/bootstrap.min.css ' %}">

<link rel="stylesheet" href="{% static '../static/styles/style.css' %}">

</head>

<!— CONTENT OF THE BODY -->

<body>

<!—HEADER SECTION -->

<header>

<nav class="navbar navbar-expand-lg navbar-light bg-dark" style="height:90px">

<div class="container-fluid">

<a href="/" class="navbar-brand" style="font-size:30px;margin-left: 10%;color: white;">Password Manager</a>

<div class=" collapse navbar-collapse" id="mynavbar">

<ul class="navbar-nav" style="margin-left: auto;">

{% if request.user.is\_anonymous %}

<li class="nav-item">

<a href="" class="nav-link navopt" style="color: white;">Home</a>

</li>

<li class="nav-item">

<a href="javascript:;" class="nav-link navopt" style="color: white;" onclick="displayModal('login-modal');">Login</a>

</li>

{% else %}

<li class="nav-item">

<a href="javascript:;" class="nav-link navopt" style="color: white;" onclick="displayModal('add-password-modal'); mymodel();">Add</a>

</li>

<li class="nav-item">

<a href="random.html" target="\_top" class="nav-link navopt" style="color:white;" >Gen Pass</a>

</li>

<li class="nav-item">

<a href="qrcode.html" target="\_top" class="nav-link navopt" style="color:white;" >QRCode</a>

</li>

<li class="nav-item">

<a href="javascript:;" class="nav-link navopt" style="color: white;" onclick="document.getElementById('logout-btn').click();">Logout</a>

</li>

<form hidden="true" action="." method="POST">

{% csrf\_token %}

<input type="submit" id="logout-btn" name="logout">

</form>

{% endif %}

</ul>

</div>

</div>

</nav>

</header>

<!--Display messages-->

{% if messages %}

<div class="messages" style="text-align: center;">

{% for message in messages %}

<p>

{% if message.tags == "error" %}

<i class="fa fa-exclamation" aria-hidden="true"></i>

{% else %}

<i class="fa fa-check" aria-hidden="true"></i>

{% endif %}

&nbsp;{{message}}

</p>

{% endfor %}

</div>

{% endif %}

**<!--Modals-->**

<div class="modals-wrapper" style="display:none">

<span id="close-modal" title="close"><i class="fa fa-times" aria-hidden="true" style="font-size:35px"></i></span>

**<!--login modal-->**

<div class="container container1" style="margin-top:90px" >

<div class="form-container sign-in-container" id="login-modal">

<form action="." role="form" method="POST" autocomplete="off">

{% csrf\_token %}

<h1>Login</h1>

<div class="infield">

<input type="text" placeholder="User Name" name="username"/>

</div>

<div class="infield">

<input type="password" placeholder="Password" name="password" id="pswd" class="hello"/>

</div>

<button type="submit" name="login-form">SIGN IN</button>

</form>

</div>

<!-- sign up -->

<div class="form-container sign-up-container">

<form action="." role="form" method="POST" autocomplete="off">

{% csrf\_token %}

<h1>Create Account</h1>

<div class="infield">

<input type="text" placeholder="Name" name="username">

</div>

<div class="infield">

<input type="text" placeholder="Email address" name="email">

</div>

<div class="infield">

<input type="password" placeholder="Password" name="password">

</div>

<div class="infield">

<input type="password" placeholder="Confirm Password" name="password2">

</div>

<button type="submit" name="signup-form">Sign up</button>

</form>

</div>

<div class="overlay-container" id="overlayCon">

<div class="overlay">

<div class="overlay-panel overlay-left">

<h1 style="margin-top:-15%">Welcome Back!</h1>

<button style="margin-top: 33%;">Sign In</button>

</div>

<div class="overlay-panel overlay-right">

<h1 style="margin-top: -30%;">Password Manager</h1>

<p>Enter your personal details and start journey with us</p>

<button style="margin-top:7%">Sign Up</button>

</div>

</div>

<button id="overlayBtn"></button>

</div>

</div>

**<!--add password modal-->**

<div id="add-password-modal" >

<section style="margin-left:5%;">

<div class="mask d-flex align-items-center h-100 gradient-custom-3" >

<div class="container h-100">

<div class="row d-flex justify-content-center align-items-center h-100">

<div class="col-12 col-md-9 col-lg-7 col-xl-6" style="width:90%">

<div class="card" style="border-radius: 15px;width:fit-content">

<div class="card-body p-5" style="background-color:#212529;border-radius:8px">

<h2 class="text-uppercase text-center mb-4" style="color:white">Add Password</h2>

<form action="." role="form" method="POST">

{% csrf\_token %}

<div class="form-outline mb-3">

<input type="text" id="form3Example1cg" class="form-control form-control-lg" placeholder="Website Name" name="web\_name" required>

</div>

<div class="form-outline mb-3">

<input type="url" id="form3Example2cg" class="form-control form-control-lg" placeholder="Url of website" name="url" required>

</div>

<div class="form-outline mb-3">

<input type="email" id="form3Example3cg" class="form-control form-control-lg" placeholder="Email" name="email" required>

</div>

<div class="form-outline mb-3">

<input type="password" id="form3Example4cg" class="form-control form-control-lg" placeholder="Password" name="password" required>

</div>

<button type="submit" value="Save" class="btn btn-block btn-lg gradient-custom-4 text-body d-flex justify-content-center"

style="color:white;background-color: rgb(146, 141, 141);" name="add-password" onclick="viewmodel()">ADD</button>

</form>

</div>

</div>

</div>

</div>

</div>

</div>

</section>

</div>

</div>

<div class="container-fluid row" id="card-modal">

<!--One card-->

{% for password in passwords %}

<div class="container-fluid col-3" id="card-model" style="margin-top:30px;">

<div class="card bg-dark card-design"

style="color:white;width:fit-content;

margin-left:3%;

border-radius:1rem;

display:flex;

">

<div class="card-body">

<img width="60" height="60" src="{{password.logo}}" style="margin-left:37%">

<br />

<h5 class="card-title" style="margin-left:34%;margin-top:20px;">{{password.name}}</h5>

<form hidden="true" action="." method="POST">

{% csrf\_token %}

<input type="text" hidden="true" name="password-id" value="{{password.id}}" >

<input type="submit" id="delete-btn{{password.id}}" name="delete" style="width:30px">

</form>

<div style="display:flex">

<input type="email" name="email" value="{{password.email}}">

<i class="fas fa-copy copy" title="copy to clipboard"></i>

</div>

<div style="display:flex;margin-top:5px">

<input type="text" name="username" value="{{password.password}}">

<i class="fas fa-copy copy" title="copy to clipboard"></i>

</div>

<div style="display:flex;justify-content:center;width:fit-content;margin-left:35%">

<a href="javascript:;"

onclick="document.getElementById('delete-btn{{password.id}}').click();"

class="btn btn-primary" style="">Delete</a>

</div>

</div>

</div>

</div>

{% endfor %}

</div></body></html>

**<!—Python file 🡪**

**views.py**

from django.contrib.auth.models import User

from django.shortcuts import render

from django.conf import settings

from django.contrib import messages

from django.http import HttpResponseRedirect

from django.contrib.auth import authenticate, login, logout

import random

from django.core.mail import send\_mail

from cryptography.fernet import Fernet

from mechanize import Browser

import favicon

from .models import Password

br = Browser()

br.set\_handle\_robots(False)

fernet = Fernet(settings.KEY)

def home(request):

if request.method == "POST":

if "signup-form" in request.POST:

username = request.POST.get("username")

email = request.POST.get("email")

password = request.POST.get("password")

password2 = request.POST.get("password2")

#if password are not identical

if password != password2:

msg = "Please make sure you're using the same password!"

messages.error(request, msg)

return HttpResponseRedirect(request.path)

#if username exists

elif User.objects.filter(username=username).exists():

msg = f"{username} already exists!"

messages.error(request, msg)

return HttpResponseRedirect(request.path)

#if email exists

elif User.objects.filter(email=email).exists():

msg = f"{email} already exists!"

messages.error(request, msg)

return HttpResponseRedirect(request.path)

else:

User.objects.create\_user(username, email, password)

new\_user = authenticate(request, username=username, password=password2)

if new\_user is not None:

login(request, new\_user)

msg = f"{username}. Thanks for subscribing."

messages.success(request, msg)

return HttpResponseRedirect(request.path)

elif "logout" in request.POST:

msg = f"{request.user}. You logged out."

logout(request)

messages.success(request, msg)

return HttpResponseRedirect(request.path)

elif 'login-form' in request.POST:

username = request.POST.get("username")

password = request.POST.get("password")

new\_login = authenticate(request, username=username, password=password)

if new\_login is None:

msg = f"Login failed! Make sure you're using the right account."

messages.error(request, msg)

return HttpResponseRedirect(request.path)

else:

login(request, User.objects.get(username=new\_login))

msg = f"{request.user} welcome again."

messages.success(request, msg)

return HttpResponseRedirect(request.path)

elif "add-password" in request.POST:

website\_name = request.POST.get("web\_name")

url = request.POST.get("url")

email = request.POST.get("email")

password = request.POST.get("password")

#ecrypt data

encrypted\_email = fernet.encrypt(email.encode())

encrypted\_password = fernet.encrypt(password.encode())

#get title of the website

try:

br.open(url)

title = br.title()

except:

title = url

#get the logo's URL

try:

icon = favicon.get(url)[0].url

except:

icon = "https://cdn-icons-png.flaticon.com/128/1006/1006771.png"

#Save data in database

new\_password = Password.objects.create(

user=request.user,

name=website\_name,

logo=icon,

email=encrypted\_email.decode(),

password=encrypted\_password.decode(),

)

msg = f"{website\_name} added successfully."

messages.success(request, msg)

return HttpResponseRedirect(request.path)

elif "delete" in request.POST:

to\_delete = request.POST.get("password-id")

msg = f"{Password.objects.get(id=to\_delete).name} deleted."

Password.objects.get(id=to\_delete).delete()

messages.success(request, msg)

return HttpResponseRedirect(request.path)

context = {}

if request.user.is\_authenticated:

passwords = Password.objects.all().filter(user=request.user)

for password in passwords:

password.email = fernet.decrypt(password.email.encode()).decode()

password.password = fernet.decrypt(password.password.encode()).decode()

context = {

"passwords":passwords,

}

return render(request, "home.html", context)

def qrcode(request):

return render(request,"qrcode.html")

def passgen(request):

return render(request,"random.html")

**SCREENSHOTS**

