International Journal of Advances in Engineering and Management (IJAEM)
Volume 4, Issue 6 June 2022, pp: 59-64 www.ijaem.net ISSN: 2395-5252

Password Manager

Nipun Jadhav[1], Pratik Chopde[2], Suman Puri[3]
Department Of Computer Engineering, JSPM's Rajarshi Shahu College of Engineering Polytechnic, Tathawade, Pune, India.

Submitted: 25-05-2022 Revised: 01-06-2022 Accepted: 05-06-2022

ABSTRACT: This research paper is about to create a password manager web app that will store, retrieve and generate passwords for users. This web app is protected by Google's OAuth 2.0 which supports user to sign-in using there own Google account. In a world where every service is on the internet, there is a need for unique and complex password for each and every service that we use.

KEYWORDS: Django, Encryption, Google OAuth 2.0™, SQLite, Python, Security, Passwords

I. INTRODUCTION

In this world with increasing risk of being hacked and what we really need is a strong password, but remembering this passwords is a hassle. We use Google Pay, Amazon, Flipkart, Facebook, Instagram etc. that requires one to have different passwors for each site and to memorize them is a task impossible for a average human being. To overcome these problems, we proposed a web app that will handle both tasks, generating a complex password and saving it. A complex password is a password that was randomly generated and consist of atleast a uppercase letter, lowercase letter, digits and symbols. In order to use this app, the user will have to sign-up giving his personal details such as name and email-id and set a password in order to access the app. In addition to this user can also sign-up using his/her own Google account and get started. After signing up, user can immediatly start generating and saving his passwords. One user can have infinite number of passwords saved and search them whenever he needs them. User can generate new passwords and update existing ones any time he wants. The following major features were used in this web app:

[1].Generating password:

Now a days lots of websites have password policy that forces users to have a very complex, secure passwords. Our webapp allows user to create a password which consist of atleast one upper case letter, one lower case letter, a digit and a symbol. These fout symbols are chosen at random and the remaining required digits are also chosen at random. The obtained sequence is then shuffled for increased randomness. The code was written on the client side using Javascript and a generate button was given to the user to invoke the generation method.

[2].Database Operations:

The Django Framework was utilized by us in order to handle serverside coding and database interactions. The Django ORM (Object and Relational Mapping) is a powerful tool which enables us to create, store and retrive database objects. Django also comes with a built in template engine which allows us to inject server side objects into the front end. In order to insert data into database, one only needs to create the required object and call its save method.

[3].Database Schema:

The database had two tables, A user table for authentication and security of the web app, other table passwords, which had the details of the saved passwords. Entity 'user' had different attributes which stored personal information of the user. The password entity had attributes such as username, password, and website for which the password was to be saved. The entity also had a attribute 'usr' which was a foreign key pointing to the primary key of the user table. Thus there was a one to many relationship between a user and his/her passwords.(One user could have multiple passwords)

Objectives

The main objective of the system is to provide a means to save, generate, save and update passwords by using just a web app. It also provides secure way to save passwords which is protected by Google's OAuth 2.0.

Software Requirements

1. Operating System: Windows 10/11 or Linux
2. User Interface: HTML, CSS, Bootstrap
3. Scripting: JavaScript Progrraming Language: Python
4. Web Application: Django

International Journal of Advances in Engineering and Management (IJAEM)
Volume 4, Issue 6 June 2022, pp: 59-64 www.ijaem.net ISSN: 2395-5252

5. Database: SQLite

Proposed Methodology:

In this digital world with increasing threat of being hacked digitally is so high that there is a need for complex password. If one is using a simple password, he might as well be living in a house made of glass.

To avoid this we are proposing a web app that will gnerate complex password and store them in database securely. In order to use this app, the user must sign-up giving his personal details such as name and email-id and set a password in order to access the app. In addition to this user can also sign-up using his/her own Google account and get started. After signing up, user can immediatly start generating and saving his passwords. One user can have infinite number of passwords saved and search them whenever he needs them. User can generate new passwords and update existing ones any time he wants.

Now a days lots of websites have password policy that forces users to have a very complex, secure passwords. Our webapp allows user to create a password which consist of atleast one upper case letter, one lower case letter, a digit and a symbol. These fout symbols are chosen at random and the remaining required digits are also chosen at random. The obtained sequence is then shuffled for increased randomness. The code was written on the client side using Javascript and a generate button was given to the user to invoke the generation method.

```
class Usr(models.Model):
    f_name = models.CharField(max_length=50)
    l_name = models.CharField(max_length=50)
    usr_name = models.CharField(max_length=50,unique=True)
    is_google_user = models.BooleanField()
    email_id = models.EmailField(max_length=254,unique=True)
    email_verified = models.BooleanField()
    password = models.CharField(max_length=15)
    profile_pic = models.URLField()
```

fig.2.1

The Django Framework was utilized by us in order to handle serverside coding and database interactions. The Django ORM (Object and Relational Mapping) is a powerful tool which enables us to create, store and retrive database objects. Django also comes with a built in template engine which allows us to inject server side objects into the front end. In order to insert data into database, one only needs to create the required object and call its save method.

```
class Passwords(models.Model):
    website = models.CharField(max_length=50)
    password = models.CharField(max_length=50)
    usr = models.ForeignKey('Usr',on_delete=models.CASCADE)
```

fig.2.2

The database had two tables, A user table for authentication and security of the web app, other table passwords, which had the details of the saved passwords. Entity 'user' had different attributes which stored personal information of the user. The password entity had attributes such as username, password, and website for which the password was to be saved. The entity also had a attribute 'usr' which was a foreign key pointing to the primary key of the user table. Thus there was a one to many relationship between a user and his/her passwords.(One user could have multiple passwords)

International Journal of Advances in Engineering and Management (IJAEM)
Volume 4, Issue 6 June 2022, pp: 59-64 www.ijaem.net ISSN: 2395-5252

```
function password_generate(){
    const n = parseInt(document.getElementById('n').value) || 8
    const password_input = document.getElementById('password')
    if(n < 8 || n > 12){
        alert("password must be 8 to 12 characters long")
        return
    }
    let allowed_symbols = ['@', '#', '$', '%', '=', ':', '?', '.', '/', '|', '~', '>',
    '*', '(', ')', '<']
    let digits = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
    let upper_case_letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H',
                              'I', 'J', 'K', 'M', 'N', 'O', 'P', 'Q',
                              'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y',
                              'Z']
    let lower_case_letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h',
                              'i', 'j', 'k', 'm', 'n', 'o', 'p', 'q',
                              'r', 's', 't', 'u', 'v', 'w', 'x', 'y',
                              'z']

    let characters = [...allowed_symbols,...digits,...upper_case_letters,...lower_case_letters]
    let password = Array(n)
    password[0] = allowed_symbols[Math.floor(Math.random() * allowed_symbols.length)]
    password[1] = digits[Math.floor(Math.random() * digits.length)]
    password[2] = upper_case_letters[Math.floor(Math.random() * upper_case_letters.length)]
    password[3] = lower_case_letters[Math.floor(Math.random() * lower_case_letters.length)]
    for(let i=4;i<n;i++){
        password[i] = characters[Math.floor(Math.random() * characters.length)]
    }
    result = ""
    password = shuffel_password(password)
    for(let i = 0; i < password.length; i++){
        result = result+password[i]
    }
    password_input.value = result
}
```

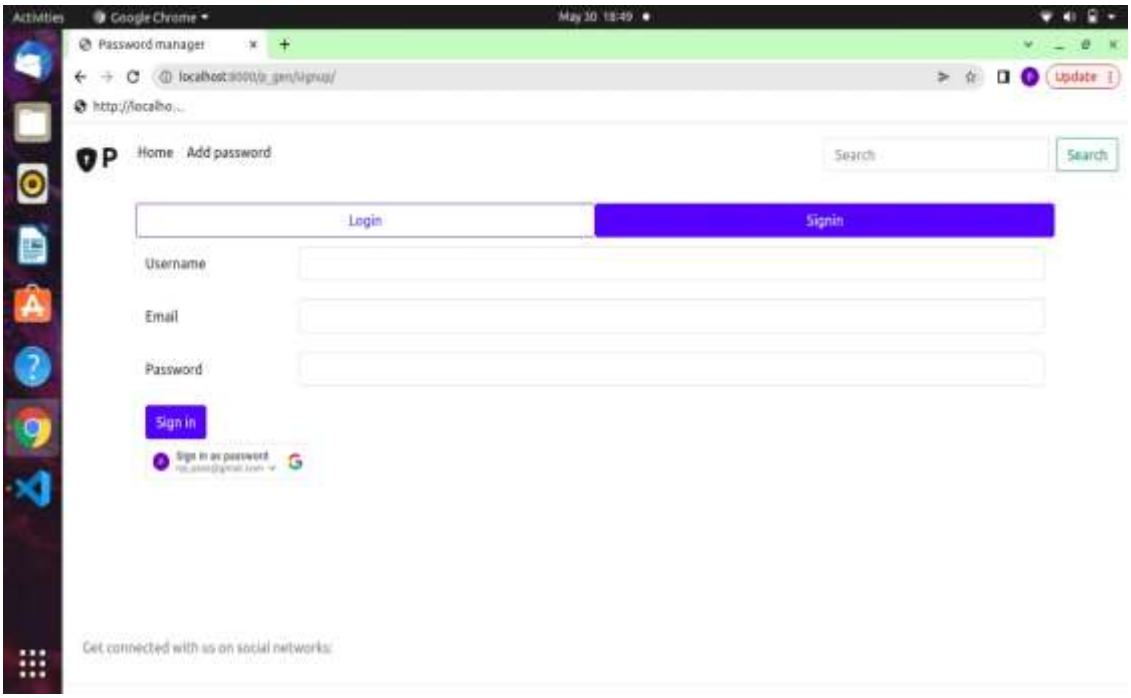
fig.2.3

System Architecture

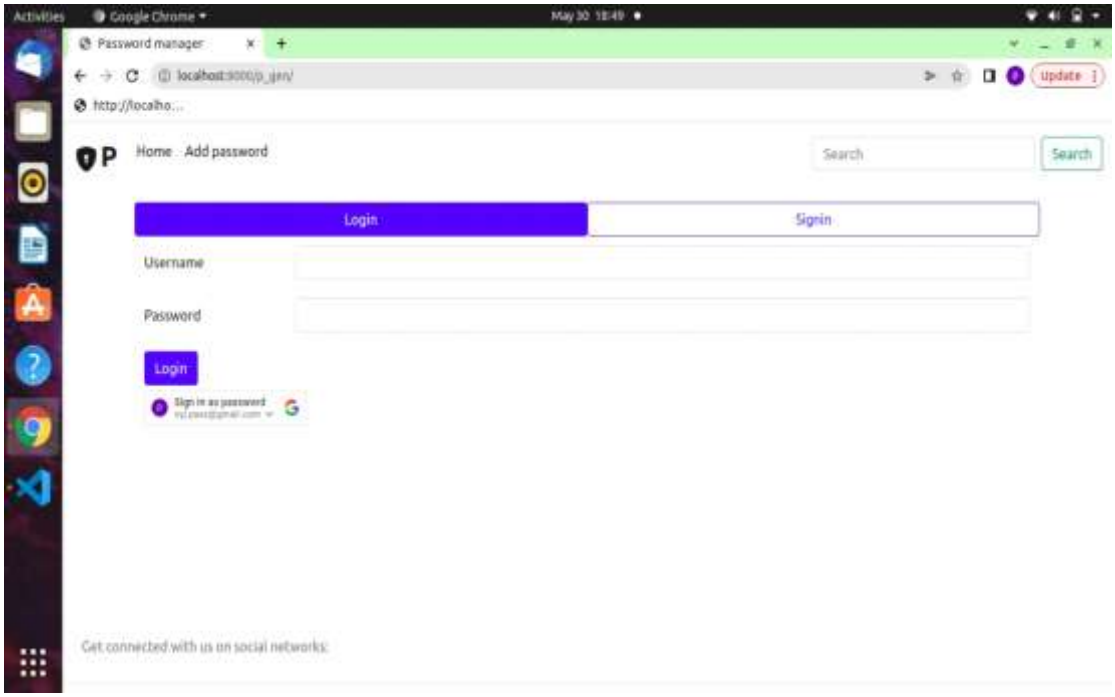
fig. Flow Diagram

International Journal of Advances in Engineering and Management (IJAEM)
Volume 4, Issue 6 June 2022, pp: 59-64 www.ijaem.net ISSN: 2395-5252

Step 1: Registration



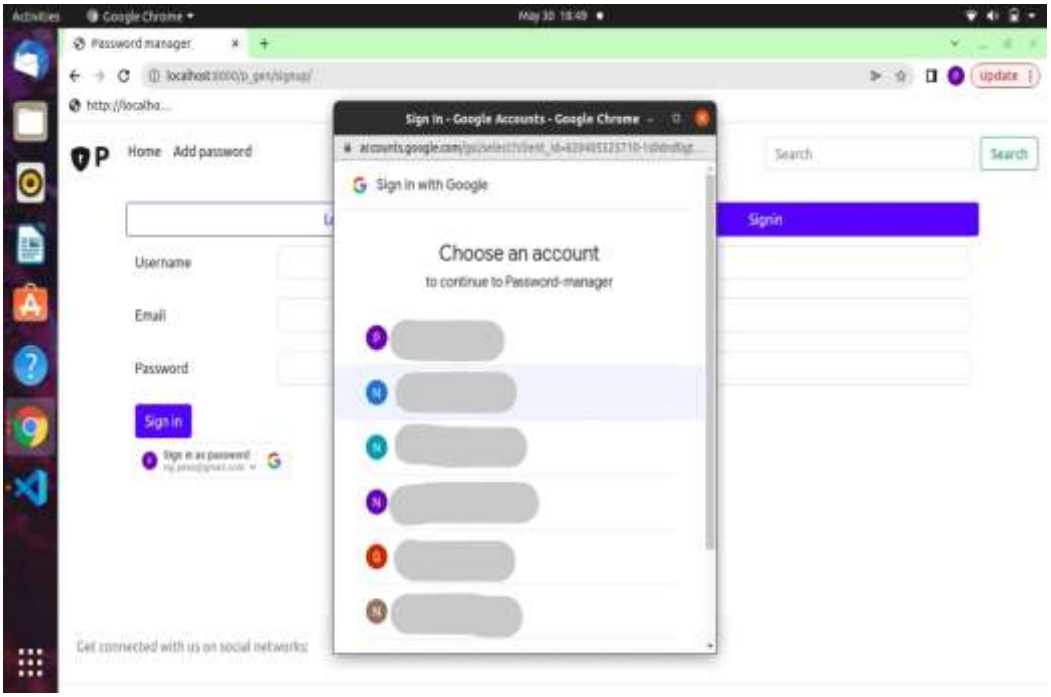
Step 2: Login



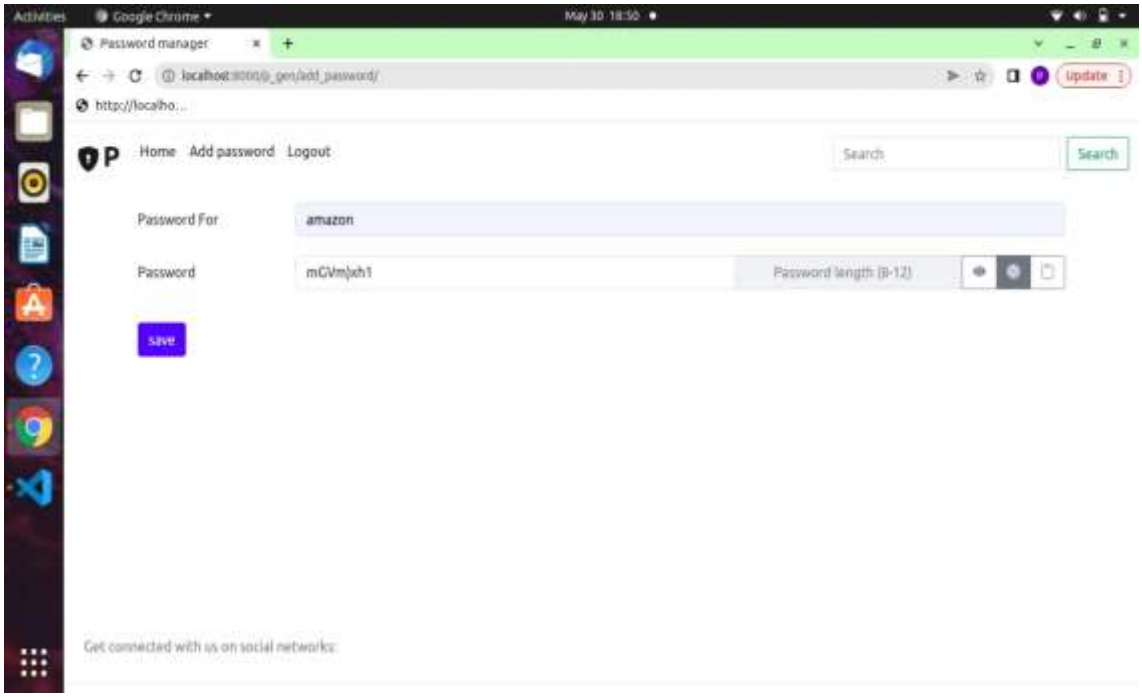
OR

International Journal of Advances in Engineering and Management (IJAEM)
Volume 4, Issue 6 June 2022, pp: 59-64 www.ijaem.net ISSN: 2395-5252

Sign-up with Google

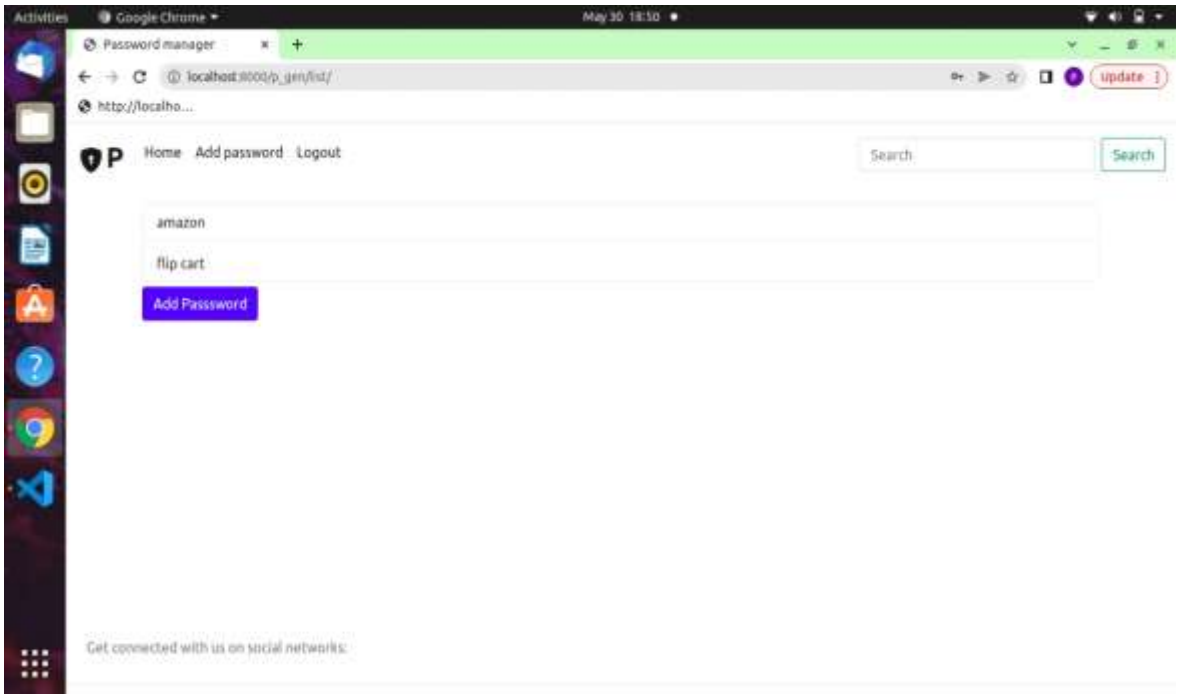


Step 3: Generate Password



International Journal of Advances in Engineering and Management (IJAEM)
Volume 4, Issue 6 June 2022, pp: 59-64 www.ijaem.net ISSN: 2395-5252

Step 5: List of Passwords



II. CONCLUSION

We all know that we can't have 100% secure environment, but we can introduce and establish a sequence of protection mechanisms which will certainly decrease the possible attack vectors in our system. A password manager can be a great tool to keep our user credentials safe, but let's not forget that even a 4096 bit key can be cracked under special circumstances. To push our solution to industrial standards we can pair it with a authenticar app which will create temporary one time passwords which will further incrrease our security. .Keeping our passwords secure and most important complex enough will give us peace of mind and knowing the possible attack vectors furthermore reduces the possibility of successful malicious activity.

REFERENCES

https://www.geeksforgeeks.org/generating-strong-password-using-python/
http://www.math.bas.bg/infres/cib80/book/cib80-p14.pdfGould, Password Manager
with 3-Step Authentication System by Zhelyazko Petrov, Razvan Ragazan Paper.
https://docs.djangoproject.com/en/4.0