

1. INTRODUCTION

1.1. PURPOSE

The purpose of the project is to provide a platform where the internet users can create strong passwords for their different online accounts and it suggests, manages and stores the passwords for the user which they can use in their online accounts.

It assists in generating and retrieving complex passwords, storing such passwords in an encrypted database, or calculating them on demand.

It is basically a localhost web application. It has an indicator which indicates the strength of the password. As we store any password then a button is created on the page by the name of the website through which we can see the password by clicking on it. We can store different passwords for different websites and applications.

The main advantage of this project is we need not to remember passwords for different websites as it stores them into a database for us.

1.2. PROJECT SCOPE

The password diary is a computer program that allows users to store, suggest passwords, and manage their passwords for local applications and online services.

A password manager is an advanced tool that helps individuals and businesses securely store and manage all of their login credentials. This tool is commonly used to prefer strong, unique passwords for web applications. Once it is done then, they are put in a centralized vault.

In this digital era, everyone has accounts on different websites and applications. Remembering all the passwords of different websites is a difficult task. So this password diary helps in storing all the passwords of different websites so that we do not need to remember them. Besides this, it also helps in selecting strong passwords in order to ensure security.

1.3. PRODUCT FEATURES

1.3.1. Multi-Platform Support

Support for various platforms is crucial to ensure that no matter which OS you're using, you can still access your password vault. At the minimum, a password manager should support the four major platforms: Android, iOS, Windows, and macOS.

1.3.2. Password Strength Tester

This project has a password strength estimator. Password strength estimator is essential and plays a critical role in helping users create strong, unique passwords. This removes so much overhead, and users don't have to remember all the tips for creating secure and strong passwords. It offers customization options, letting users adjust the length of a generated password and even choose whether it should have special characters, numbers, lowercase and uppercase letters. Main aim of this feature is to ensure security because easy passwords can easily be guessed. Hence we need strong passwords to protect our accounts.

1.3.3. Vault Storage Location

The passwords which are being encrypted are stored in an online database through which the user can access the password from any online device.

Even if due to any security fault the database gets hacked then also the actual passwords will not be shown rather it will be shown in its encrypted form.

1.3.4. Security

Given that you'll be storing important passwords for all your accounts, including important platforms like banks, security is

of utmost importance. Our project password diary leverages AES encryption protocols to ensure your data is safe and secure from the bad guys. AES 256-bit encryption, the current industry standard, is used in the encryption process.

2. SYSTEM ANALYSIS

This section provides system requirements for installing and running Password Manager and its components.

2.1. HARDWARE REQUIREMENTS

Here are the hardware requirements for smooth operation of the project:-

Processor - Minimum:-800MHz or Higher Intel Pentium-Compatible CPU

Recommended:-1.9 gigahertz(GHz)

x86-or x64bit dual core processor

Memory(RAM) - Minimum-512 MB RAM

Recommended-1GB RAM or More

Hard Disk Space - Minimum- 100 MB

Recommended-500MB or More

2.2. SOFTWARE REQUIREMENTS

Here are the software requirements for smooth operation of the project:-

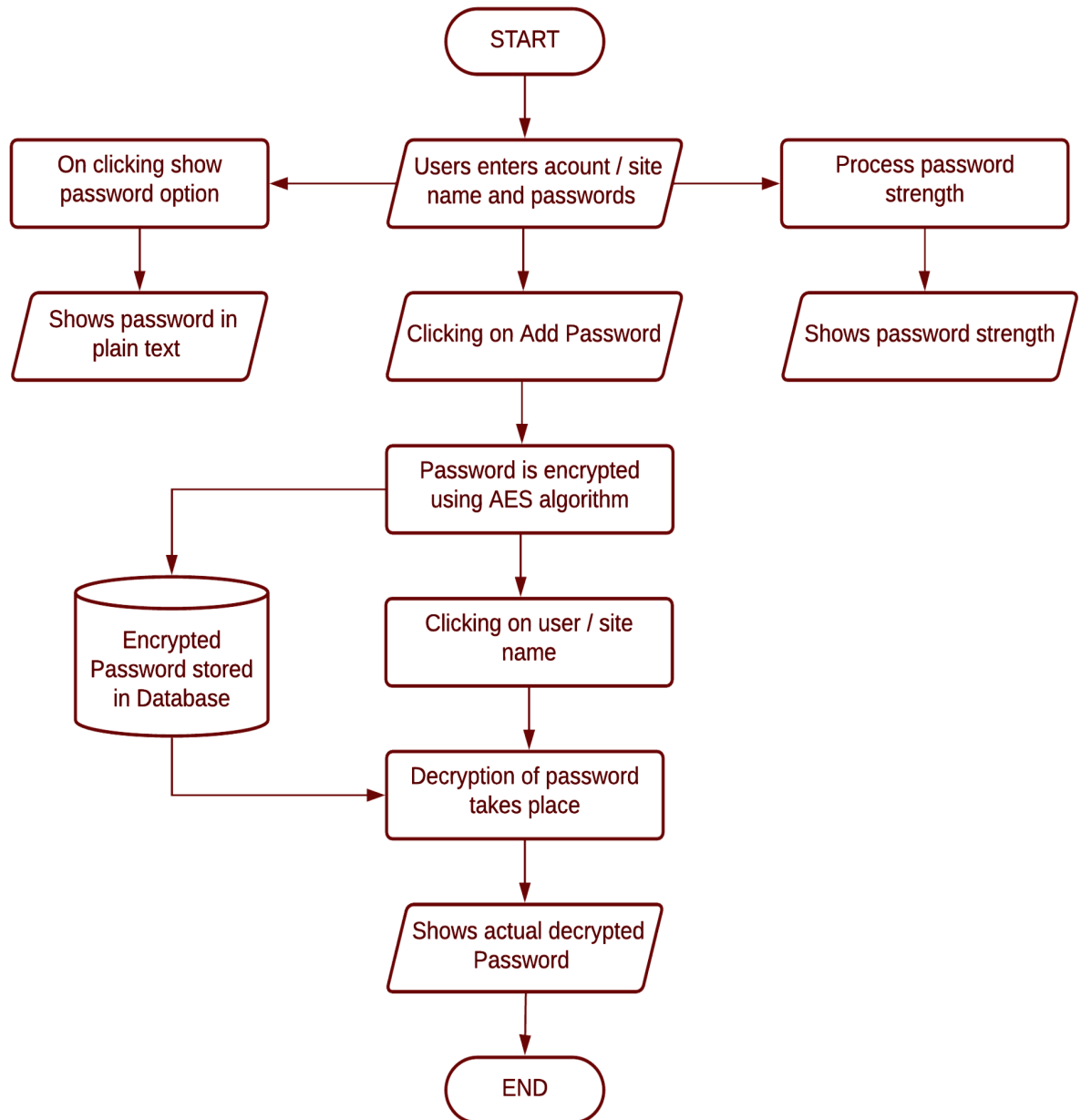
Operating Systems - Windows, Linux, MacOS

Web Browser - Google Chrome, Microsoft Edge, Firefox, Yahoo,etc

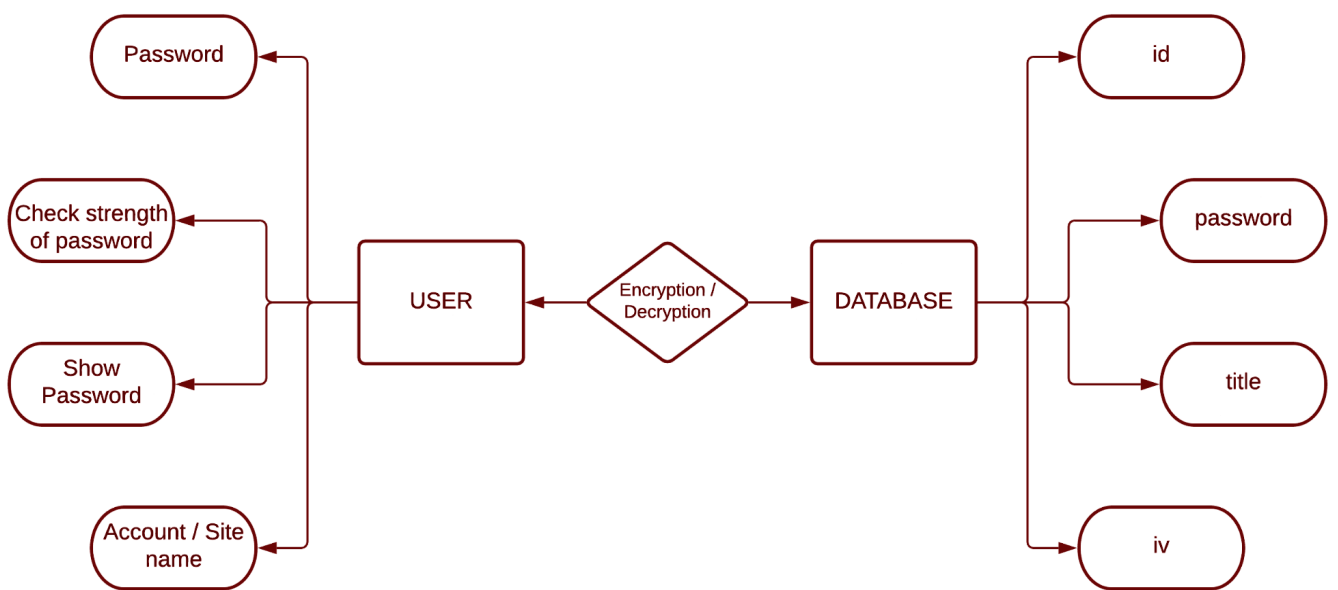
3. SYSTEM DESIGN AND SPECIFICATIONS

3.1. HIGH LEVEL DESIGN (HLD)

3.1.1. FLOW CHART



3.1.2. E-R DIAGRAM



3.2. LOW LEVEL DESIGN (LLD)

3.2.1. PROCESS SPECIFICATIONS (PSEUDO CODE / ALGORITHM)

Advanced Encryption Standard Algorithm

Each round consists of four byte-oriented cryptographic transformations:

1. Byte Substitution
2. Shifting rows of the State Array
3. Mixing data within a column of the State Array
4. Round Key addition to the State Array

Cipher(byte in[4 * Nb],

byte out[4 * Nb], word w[Nb * (Nr + 1)])

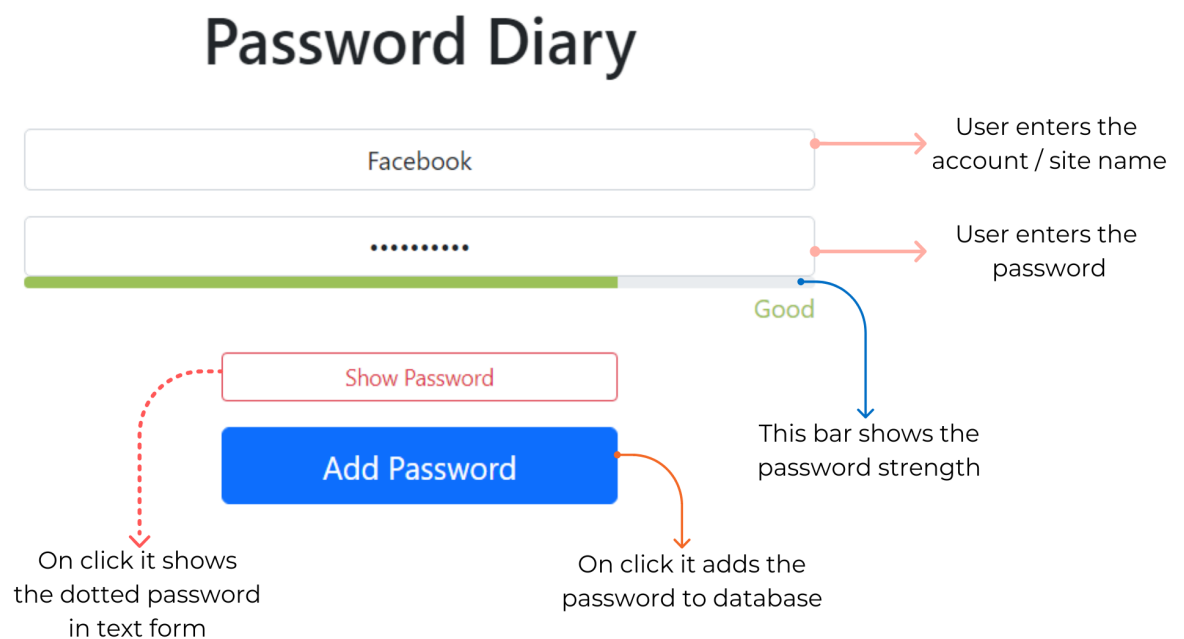
begin byte state[4,Nb] state = in AddRoundKey(state, w)

for round = 1 step 1 to Nr - 1 SubBytes(state) ShiftRows(state)
MixColumns(state) AddRoundKey(state, w + round * Nb) end
for

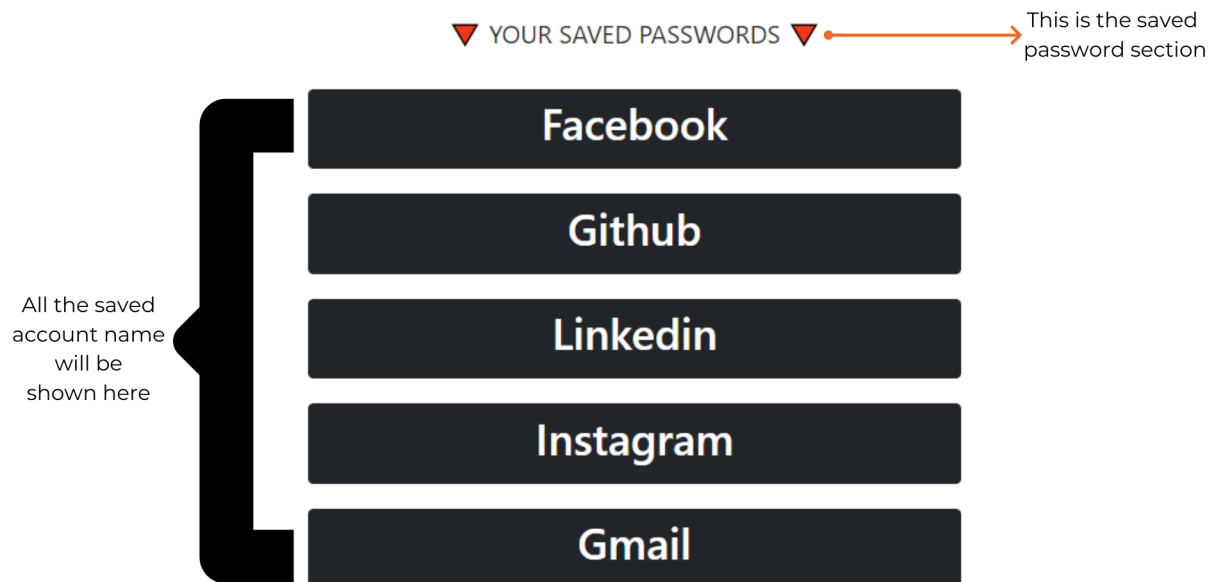
SubBytes(state) ShiftRows(state) AddRoundKey(state, w + Nr
* Nb) out = state end

3.2.2. SCREEN-SHOT DIAGRAM

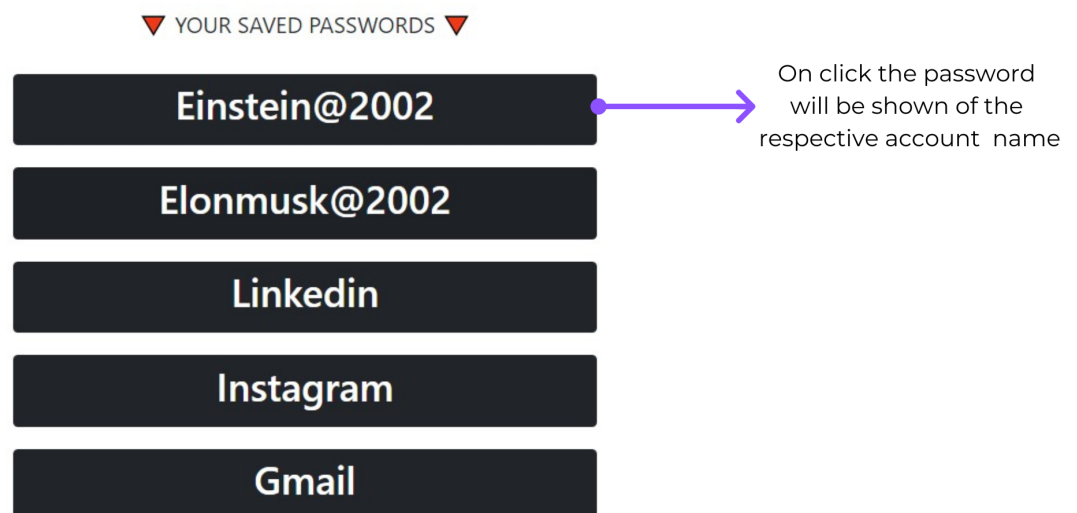
This below screen-shot shows the basic front-end part that are like where the users need to enter the account or site name, password. Below the password section there is a password strength estimator which shows the strength of the password provided by the user. After that there is a show password button which shows the dotted form password into text form password which adds a bit of privacy if someone is near you. Then finally there is an add password button which on click forwards the password to the backend for encryption and storage purpose.



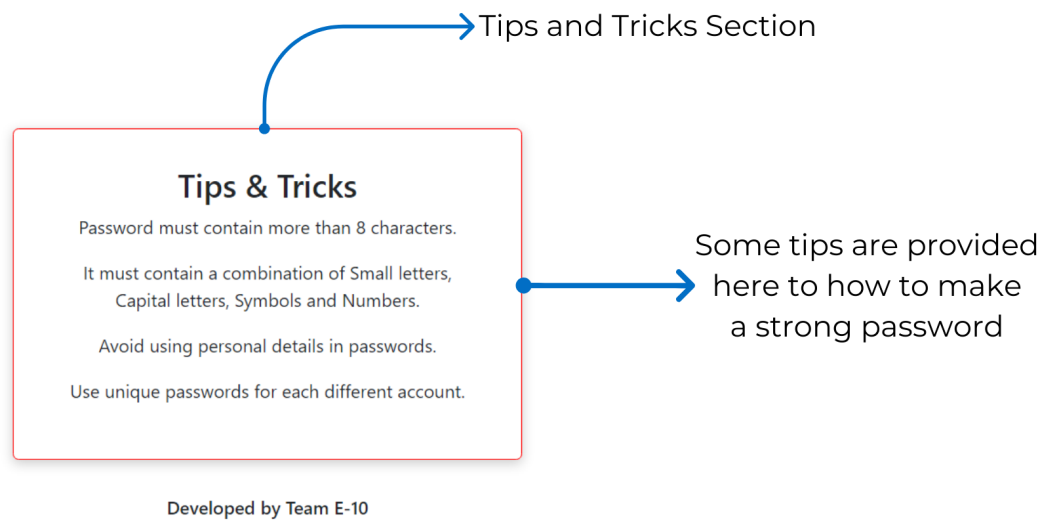
After adding the password with a particular account name, the account name will be shown in the password section as shown below in the below screenshot.



Now, clicking onto the account name the decryption process will take place and the actual original password will be shown to the user in plain text form.



At last there is a tips and tricks section which provides some useful tips to the user to make a strong password .



4. CODING

This section contains the front-end codes and back-end codes used in the development of the project :

4.1. Front-End Coding

\password-diary\client\src\App.js

Imported modules for making the frontend part :

```
import './App.css';  
  
import { useState, useEffect } from 'react';  
  
import Axios from 'axios';  
  
import PasswordStrengthMeter from './components/PasswordStrengthMeter';
```

Connection code to backend :

```
function App() {  
  
  const [password, setPassword] = useState('')  
  
  const [title, setTitle] = useState('')  
  
  const [passwordList, setPasswordList] = useState([])  
  
  const [passwordShown, setPasswordShown] = useState(false);  
  
  
  const togglePassword = () => {  
  
    // When the handler is invoked  
  
    // change inverse the boolean state passwordShown  
  
    setPasswordShown(!passwordShown);  
  
  };  
  
  
  useEffect(() => {  
  
    Axios.get("http://localhost:3001/showpassword").then((response) =>  
    {
```

```

setPasswordList(response.data);

    });

    }, []);

    const addPassword = () => {

        Axios.post("http://localhost:3001/addpassword", {
password: password,
title: title,

        });

    };

    const decryptPassword = (encryption) => {

        Axios.post("http://localhost:3001/decryptpassword", {
password: encryption.password,
iv: encryption.iv,

        }).then((response) => {
setPasswordList(passwordList.map((val) => {

            return val.id === encryption.id ? {id: val.id, password: val.password, title:
            response.data, iv: val.iv} : val;

        }));

    });

    };

    };

```

Structure of the page :

```

return (

    <div className="containers">

    <h1 className="text-center my-1">Password Diary</h1>

    <div className="col-md-4 mx-auto text-center password-diary">

```

```

<div className="form-group">
  <input
    type="text"
    className="form-control shadow-none mb-3"
    placeholder="Enter user/site Name"
    onChange={(event) => {
      setTitle(event.target.value);
    }}
  />

```

```

  <input
    type={passwordShown ? "text" : "password"}
    className="form-control shadow-none"
    placeholder="Enter Password"
    onChange={(event) => {
      setPassword(event.target.value);
    }}
  />
</div>

```

```

<span className="text-end">
  <PasswordStrengthMeter password={ password } />
</span>

```

```

<div class="d-grid col-6 mx-auto">
  <button
    type="button" class="btn btn-outline-danger shadow-none btn-sm mb-3"
    onClick={togglePassword}>Show Password

```

```

</button>

<button
    type="button" class="btn btn-primary shadow-none btn-lg"
onClick={addPassword}>Add Password
</button>

</div>

<div className="m-4">

<p> ▼ YOUR SAVED PASSWORDS ▼ </p>

</div>

<div className = "Password">

    {passwordList.map((val, key) => {
return (
    <div class="d-grid col-10 mx-auto m-3">

        <button type="button" class="btn btn-dark btn-sm shadow-none">

            <div className = "password" onClick = {} => {decryptPassword({password:
            val.password, iv: val.iv, id: val.id})}} key={key}>

            <h3> {val.title} </h3>

        </div>

    </button>

</div>

);

    })}

</div>

<div className="extras ">

<h3>Tips & Tricks</h3>

<p>Password must contain more than 8 characters.</p>

    <p>It must contain a combination of Small letters, <br/> Capital letters, Symbols
    and Numbers.</p>

```

```

<p>Avoid using personal details in passwords.</p>

    <p>Use unique passwords for each different account.</p>

</div>

<div className="extra">

<h6>Developed by Team E-10</h6>

</div>

</div>

</div>

);
}

```

\password-diary\client\src\App.css

Designing code part :

```

.password-diary{

margin-top: 2rem;

}

.extras{

border: 1px solid red;

border-radius: 5px;

padding: 2rem;

margin: 2rem;

box-shadow: 0 3px 10px rgb(0 0 0 / 0.2);

}

```



```
input{  
  text-align: center;  
}
```

\password-diary\client\src\components\passwordStrengthEstimator.js

zxcvbn password strength estimator code :

```
import React from 'react';  
import zxcvbn from 'zxcvbn';  
  
const PasswordStrengthMeter = ({ password }) => {  
  const testResult = zxcvbn(password);  
  const num = testResult.score * 100/4;  
  
  const createPassLabel = () => {  
    switch(testResult.score) {  
      case 0:  
        return 'Very weak';  
      case 1:  
        return 'Weak';  
      case 2:  
        return 'Fair';  
      case 3:  
        return 'Good';  
      case 4:  
        return 'Strong';  
      default:  
        return '';  
    }  
  }  
}
```

```

const funcProgressColor = () => {
  switch(testResult.score) {
    case 0:
      return '#828282';
    case 1:
      return '#EA1111';
    case 2:
      return '#FFAD00';
    case 3:
      return '#9bc158';
    case 4:
      return '#00b500';
    default:
      return 'none';
  }
}

```

```

const changePasswordColor = () => ({
  width: `${num}%`,
  background: funcProgressColor(),
  height: '7px'
})

```

```

return (
  <div className="progress" style={{ height: '7px' }}>
    <div className="progress-bar" style={changePasswordColor()}></div>
  </div>
)

```

```

        <p style={{ color: funcProgressColor() }}>{createPassLabel()}</p>

    </>

    )

}

export default PasswordStrengthMeter

```

4.2. Back-End Coding

\password-diary\server\index.js

Code for connection to the database :

```

const express = require("express");

const app = express();

const mysql = require("mysql");

const cors = require("cors");

const PORT = 3001;

const { encrypt, decrypt } = require("./EncryptionHandler") //importing the
encryption and decryption process

app.use(cors());

app.use(express.json())

const db = mysql.createConnection({ //created a connection with the mySql
database

    user: "root",

    host: "localhost",

    password: "2362@Mysql",

    database: "password-tester"

});

```

```
app.post("/addpassword", (req, res) => { // To get the data from the front-End and
pass it to the database
```

```
    const {password, title} = req.body;
```

```
    const hashedPassword = encrypt(password);
```

```
    db.query("INSERT INTO password (password, title, iv) VALUES (?, ?, ?)",
[hashedPassword.password, title, hashedPassword.iv],
```

```
    (err, result) => {
```

```
        if(err){
```

```
            console.log(err);
```

```
        } else {
```

```
            res.send("Success");
```

```
        }
```

```
    });
```

```
});
```

```
app.get("/showpassword", (req, res) => {
```

```
    db.query("SELECT * FROM password;", (err, result) => {
```

```
        if(err){
```

```
            console.log(err);
```

```
        } else {
```

```
            res.send(result);
```

```
        }
```

```
    });
```

```
});
```

```
app.post('/decryptpassword', (req, res) => {
```

```
    res.send(decrypt(req.body));
```

```
});
```

```
app.listen(PORT, () => {
  console.log("Server is running.")
});
```

\password-diary\server\EncryptionHandler.js

Code for encryption process of the password :

```
const crypto = require("crypto"); // imported crpto module used to get the AES
algorithm
```

```
const secret = "pppppppppppppppppppppppppppppppppppppp";
```

```
const encrypt = (password) => { //encription function to encrypt the passed value
i.e, password
```

```
const iv = Buffer.from(crypto.randomBytes(16));
```

```
const cipher = crypto.createCipheriv("aes-256-ctr", Buffer.from(secret), iv);
```

```
const encryptedPassword = Buffer.concat([
    cipher.update(password),
    cipher.final(),
]);
```

```
return {  
    iv: iv.toString("hex"),  
    password: encryptedPassword.toString("hex"),  
};  
;
```

```
const decrypt = (encryption) => {  
  const decipher = crypto.createDecipheriv(  
    algorithm,
```

```
    "aes-256-ctr",  
    Buffer.from(secret),  
    Buffer.from(encryption.iv, "hex")  
  );  
  
  const decryptedPassword = Buffer.concat([  
    decipher.update(Buffer.from(encryption.password, "hex")),  
    decipher.final(),  
  ]);  
  
  return decryptedPassword.toString();  
};  
  
module.exports = {encrypt, decrypt};
```

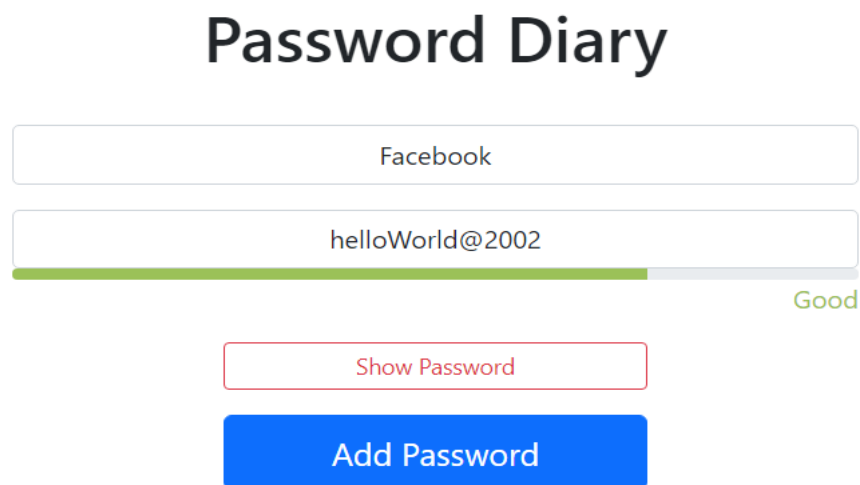
5. TESTING

5.1. UNIT TESTING

INPUT (I / P) : The input includes account or site name and a password in the respective input fields.

OUTPUT (O / P) : The output includes whether the encryption and decryption process is taking place or not.

Test 1 (I / P) : We have entered 'Facebook' as the site name and password as 'helloWorld@2002'.



Facebook

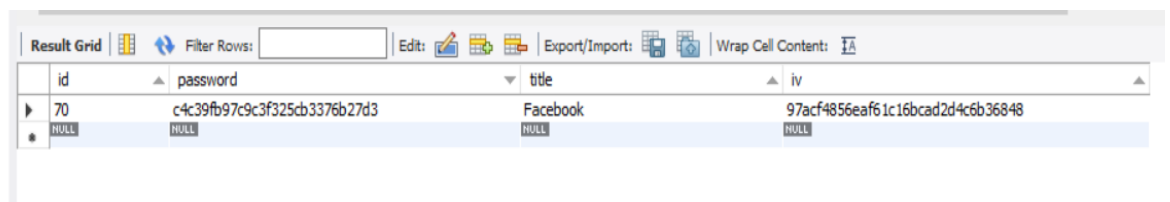
helloWorld@2002

Good

Show Password

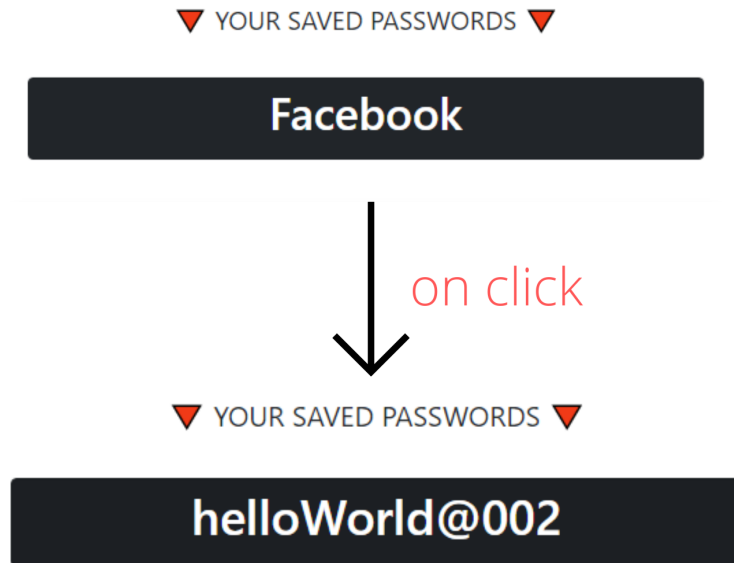
Add Password

Test 1 (O / P) : Now the added password is stored in the database in encrypted form.



	id	password	title	iv
▶	70	c4c39fb97c9c3f325cb3376b27d3	Facebook	97acf4856eaf61c16bcad2d4c6b36848
•	NULL	NULL	NULL	NULL

And the user can get back its actual plain text password from the ‘YOUR SAVED PASSWORD’ section.



Test 2 (I / P) : We have entered ‘Linkedin’ as the site name and password as ‘Iamstanding@12oclock’.

Password Diary

Strong

Test 2 (O / P) : Now the added password is stored in the database in encrypted form.

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	id	password	title	iv
▶	70	c4c39fb97c9c3f325cb3376b27d3	Facebook	97acf4856eaf61c16bcad2d4c6b36848
	71	ad003120a6bc49b36d09f9f9a98c59fd51725d	Github	153742f70ad00fc2dc3c094ad864df86
•	NULL	NULL	NULL	NULL

And the user can get back its actual plain text password from the ‘YOUR SAVED PASSWORD’ section.



6. CONCLUSION AND LIMITATIONS

6.1. LIMITATIONS

6.1.1. NO UPDATE AND DELETE FACILITY

Currently there is no update and delete option for the user.

6.1.2. NOT HOSTED ONLINE

It is a locally hosted Web application ,there is no online connection of the web application.

6.1.3. NO OFFLINE STORAGE

There is no offline storage system for the passwords. In case if the DB gets hacked or erased it will be difficult to retrieve back the passwords.

6.1.4. NO DETECTION FOR REPEATED PASSWORD

There is no detection for repetition of passwords so even if the user uses the same password for the new accounts it will not alert the user, which makes it a bit vulnerable.

7. REFERENCE AND BIBLIOGRAPHY

7.1. WEBSITES

7.1.1. Password encryption using mysql database in node js

LINK :

<https://medium.com/@technologies4.trending/password-encryption-using-mysql-database-in-node-js-30f736ea1de8>

7.1.2. Advance encryption Standard (AES) Related

LINKS :

https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

<https://www.thesslstore.com/blog/advanced-encryption-standard-aes-what-it-is-and-how-it-works/>

7.1.3. Front-End related

LINKS :

<https://reactjs.org/docs/getting-started.html>

<https://getbootstrap.com/docs/5.0/getting-started/introduction/>

<https://www.npmjs.com/package/axios/>

7.1.4. Password Strength Estimator

LINK :

<https://github.com/dropbox/zxcvbn>

7.1.5. Back-End related

LINKS :

<https://nodejs.org/en/docs/>

<https://expressjs.com/>

<https://www.npmjs.com/package/cors>

<https://www.npmjs.com/package/nodemon>