I

**PASSWORD MANAGER**

**PROJECT WORK I**

**Submitted by**

**PRANAVDEEP A M**

**22CSR148**

**RAHUL SIDDHARTH D H**

**22CSR178**

**SARAN AVINASH B**

**22CSR179**

***in partial fulfilment of the requirements***

***for the award of the degree of***

**BACHELOR OF ENGINEERING IN COMPUTER SCIENCE**

**AND ENGINEERING**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

****

**KONGU ENGINEERING COLLEGE**

**(Autonomous)**

**PERUNDURAI ERODE – 638 060**

**MAY 2024**

II

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING KONGU ENGINEERING COLLEGE**

**(Autonomous)**

**PERUNDURAI ERODE – 638 060**

**MAY 2024**

**BONAFIDE CERTIFICATE**

This is to certify that the Project report entitled **PASSWORD MANAGER** is the bonafide record of the project work done by **PRANAVDEEP A M (Register No: 22CSR148), RAHUL SIDDHARTH D H (Register No: 22CSR158), SARAN AVINASH B (Register No:22CSR179),** in the partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in **Computer Science and Engineering** of Anna University Chennai during the year 2023-2024.

**SUPERVISOR HEAD OF THE DEPARTMENT**

**Date:** (**Signature with seal**)

III

**ABSTRACT**

A password manager is a software tool designed to store and manage users' passwords and other sensitive information securely. With the increasing number of online accounts and complex password requirements, users often face difficulties in remembering and safely storing their credentials. A password manager addresses this challenge by allowing users to save their passwords in an encrypted database, accessible with a single master password. The project aims to enhance user security by implementing encryption algorithms, automatic password generation, and easy retrieval features to prevent unauthorized access and reduce the risk of password reuse across different platforms.

This project will focus on developing a user-friendly password manager with an intuitive interface for both novice and advanced users. Key functionalities will include secure password storage, cross-device synchronization, and multi-factor authentication to protect user data. The manager will also offer automated password generation, ensuring strong and unique passwords for each account. By prioritizing security and usability, this project will empower users to safeguard their online presence effectively, enhancing overall digital security and minimizing vulnerabilities due to poor password practices.

.

IV

**KONGU ENGINEERING COLLEGE**

(**Autonomous**)

**PERUNDURAI ERODE – 638 060**

**MAY 2024**

**DECLARATION**

We affirm that the Project Report titled **PASSWORD MANAGER** being submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering is the original work carried out by us. It has not formed the part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion this or any other candidate.

**Date: PRANAVDEEP A M**

**(22CSR148)**

**RAHUL SIDDHARTH D H**

(**22CSR158)**

**SARAN AVINASH B**

**(22CSR179)**

I certify that the declaration made by the above candidate is true to the best of my knowledge.

Date: Name and Signature of the Supervisor

V

**ACKNOWLEDGEMENT**

We express our sincere thanks and gratitude to **Thiru. A. K. ILANGO B.Com., M.B.A., LLB.,** our beloved Correspondent, and all other philanthropic trust members of Kongu Vellalar Institute of Technology Trust who have always encouraged us in academic and co-curricular activities.

We are extremely thankful with no words of a formal nature to the dynamic Principal **Dr. V. BALUSAMY, M.Tech., Ph.D**., for providing the necessary facilities to complete our work.

We would like to express our sincere gratitude to our respected Head of the Department **Dr. S. MALLIGA, M.E., Ph.D.**, for providing the necessary facilities.

We extend our thanks to **Mr. M. MUTHURAJA, M.E.,** Assistant Professor, Computer Science Engineering, Project Coordinator for his encouragement and valuable advice that made us carry out the project work successfully.

We extend our gratitude to our Supervisor **Mr. M. MUTHURAJA, M.E.,** Assistant Professor of Computer Science Engineering, for his valuable ideas and suggestions, which have been very helpful in the project. We are grateful to all the faculty members of the Computer Science and Engineering Department, for their support.

# VI

# TABLE OF CONTENTS

**CHAPTER No. TITLE PAGE No.**

**ABSTRACT iii**

**LIST OF FIGURES viii**

## 1. INTRODUCTION 1

1.1 EXISTING SYSTEM 1

1.2 SYSTEM STUDY 2

1.3 OBJECTIVE 2

1.4 SCOPE 3

## 2. GENERAL DESCRIPTION 4

2.1 PROJECT PERSPECTIVE 4

2.2 USER CHARACTERISTICS 4

2.3 DESIGN AND IMPLEMENTATION 4

CONSTRAINTS

## 3. REQUIREMENTS 6

3.1 FUNCTIONAL REQUIREMENTS 6

3.2 NON-FUNCTIONAL REQIREMENTS 7

3.3 USER INTERFACES 8

## 4. DETAILED DESIGN 13

4.1 ARCHITECTURAL DESIGN 13

4.1.1 MODULE CHARACTERISTICS 1

4.1.1.1 USER MODULE 14

4.1.1.2 ADMIN MODULE 14

VII

4.1.2 USE CASE DIAGRAM 15

4.1.3 SEQUENCE DIAGRAM 16

4.2 INTERFACE DESIGN 17

4.3 DATABSE DESIGN 18

4.4 OUTPUT DESIGN 18

**5. TESTING 19**

* 1. UNIT TESTING 19

5.2 REGRESSION TESTING 19

5.3 VALIDATION TESTING 19

* 1. VERIFICATION TESTING 20

5.5 INTEGRATION TESTING 20

## 

## 6. CONCLUSION AND FUTURE SCOPE 21

APPENDIX 1 22

APPENDIX 2 50

REFERENCES 57

## 

## VIII

## LIST OF FIGURES

**FIGURE NO. TITLE PAGE NO.**

4.1 FLOW DIAGRAM 15

4.2 USE CASE DIAGRAM 16

4.3 SEQUENCE DIAGRAM 17

4.4 DATABASE DESIGN DIAGRAM 18

1

**CHAPTER 1**

**INTRODUCTION**

Welcome to “LockSafe,” your ultimate companion for secure and effortless password management. LockSafe blends cutting-edge encryption with intuitive design, offering a seamless experience that keeps your digital life protected. With a single master password, unlock a vault that securely stores all your credentials, from social media logins to financial accounts, ensuring you’re safeguarded across every platform. Featuring automatic password generation, cross-device synchronization, and easy access controls, LockSafe empowers you to navigate the digital world with confidence and peace of mind. Join a community that prioritizes digital security—experience the simplicity and strength of LockSafe.

* 1. **EXISTING SYSTEM**

From the requirement analysis it is clear that there is no existing system currently in place for Dream Look to manage the items. The company is currently relying on a manual book to keep track of their items, which can be inefficient and prone to errors. The new database management website will provide a digital platform to streamline the product management process, improve efficiency, and reduce the likelihood of errors. By implementing this new system, Dream Look can better manage their products and improve the overall effectiveness of their business operations.

2

**1.2 SYSTEM STUDY**

**1.2.1 Understanding the Business Requirements**

The first step in the system study is to understand the business requirements of the project. This involves identifying the business objectives, target audience, and the features and functionalities required in the website.

**1.2.2Analyzing the Existing Systems and Processes**

The next step is to analyze the existing systems and processes in place, including the current website and any related software systems. This will help identify any limitations or areas for improvement in the new website.

**1.2.3 Identifying User Needs**

The system study should also involve identifying the needs of the users who will be accessing the website. This can be done through surveys, focus groups, or other user research methods.

**1.2.4 Developing Use Cases**

Use cases can be developed to illustrate how the website will function and how users will interact with it. This can help identify any potential issues or areas for improvement in the design and functionality.

**1.3 OBJECTIVE**

The objective of the “LockSafe” Password Manager project is to create a secure, efficient, and user-friendly tool that enables users to store, generate, and manage passwords. By implementing encryption protocols, seamless cross-device synchronization, and intuitive access controls, the project aims to empower users to protect their digital assets, reduce password fatigue, ultimately enhancing their overall online security.

3

**1.4 SCOPE**

The “LockSafe” Password Manager project is designed to deliver a secure, efficient, and user-friendly solution for managing passwords and sensitive information in the modern digital landscape. LockSafe will securely store a wide range of credentials, including social media, email, and banking logins, along with sensitive data like payment information and secure notes, all within a single encrypted vault accessible across multiple devices. To enhance security, it includes a built-in password generator that creates strong, unique passwords, reducing the risks associated with weak or reused credentials. LockSafe will also incorporate advanced encryption protocols and multi-factor authentication to protect against unauthorized access, adding crucial layers of security. With a simple, intuitive interface, LockSafe is designed to be accessible to users of all technical levels, making password management straightforward. Features like automated backup and recovery will ensure that users retain access to their vault if a device is lost, providing peace of mind. Regular security updates and user notifications will keep users informed and secure as threats evolve. By combining accessibility, convenience, and robust protection, LockSafe aims to empower users to take control of their digital security, making the digital world easier and safer to navigate confidently.

4

**CHAPTER 2**

**GENERAL DESCRIPTION**

**2.1 PROJECT PERSPECTIVE**

With the rising need for digital security, “LockSafe” provides a comprehensive solution that enables users to securely manage and store passwords in a single, encrypted vault. It encourages strong, unique passwords, reducing vulnerabilities across accounts, and offers cross-device accessibility for convenient, secure access anywhere. By integrating advanced encryption, multi-factor authentication, and regular security updates, LockSafe adapts to evolving threats. Ultimately, LockSafe empowers users to enhance their digital security with ease and confidence.

**2.2 USER CHARACTERISTICS**

For a password manager, understanding the needs of users and administrators is crucial for developing a platform that is both secure and intuitive. Users, who come with varying levels of technical expertise, prioritize simplicity and strong security features. They seek reliable encryption and multi-factor authentication, which build trust in the platform. A user-friendly interface enables easy saving, organizing, and retrieval of passwords, while features like one-click autofill and secure password generation streamline the login process. Multi-device synchronization is also important for users who need access across different platforms. On the other hand, administrators need advanced security capabilities to monitor user activity and enforce security policies effectively. Role-based access control allows them to safeguard sensitive data by managing permissions. Admins also rely on comprehensive dashboards and analytics tools to track usage, monitor potential threats, and ensure compliance with security standards.

5

**2.3 DESIGN AND IMPLEMENTATION**

**CONSTRAINTS**

**2.3.1 Time**

For Dream Look, Meeting each milestone prevents delays that entry and competitiveness. Efficient collaboration in design, structured development, and thorough testing will ensure a smooth deployment. This approach safeguards Dream Look’s market impact and customer satisfaction.

**2.3.2 Budget Limitations**

There may be budget constraints that limit the resources available for development, procurement of technology, and marketing initiatives. This could affect the selection of features, technologies, and third-party services integrated into the platform.

**2.3.3 Security**

Strong security measures, including data encryption, secure authentication, and access control, to protect user data and ensure privacy. Additional requirements, such as TLS encryption and multi-factor authentication, may impact development but are crucial for regulatory adherence and user trust.

**2.3.4 Scalability**

The architecture and design of the online store must be scalable to accommodate future growth in user traffic, product catalog size, and concurrent transactions. Scalability constraints could arise if the system architecture is not designed to handle increased load.

6

**CHAPTER 3**

**REQUIREMENTS**

**3.1 FUNCTIONAL REQUIREMENTS**

**3.1.1 User Authentication**

Users should be able to register, log in, and log out securely to access their accounts securely.

**3.1.2 Password generator**

The website should generate the password as per the user need and the user can add the length of the password ,upper case ,lowercase letters ,etc…,for generating secure password.

**3.1.2.1 Generate password**

Users can generate the strong password, he can choose the length of the password and the user can choose the combination of upper case, lower case, symbols as per the user requirments .

**3.1.2.2 Update password**

The User can easily update his password by using this application safely and securely.

**3.1.3 Dashboard**

Users should be able to filter and sort data by various categories, such as date range, product type, and region, for customized insights.

7

**3.2 NON-FUNCTIONAL REQUIREMENTS**

**3.2.1 Performance**

The website should load quickly and respond promptly to user interactions, even during periods of high traffic. Page load times should be optimized to enhance the user experience and minimize bounce rates.

**3.2.2 Usability**

Usability ensures the system is intuitive, efficient, and user-friendly, enabling users to accomplish tasks easily and without frustration.

**3.2.3 Reliability**

The website should be reliable and stable, with robust error handling and recovery mechanisms in place to prevent data loss or corruption in the event of system failures or unexpected errors.

**3.2.4 Availability**

The website should have high availability, with minimal downtime for maintenance. Redundant systems, failover mechanisms, and automated backups for uninterrupted service.

8

**3.3 USER INTERFACES**

**SIGN UP PAGE:**

This allows new users to create an account enabling secure retrieval of password.

A screenshot of a login screen

Description automatically generated

FIGURE 3.3.0

9

**LOGIN PAGE:**

This allows users to access their accounts using unique credentials for data protection .

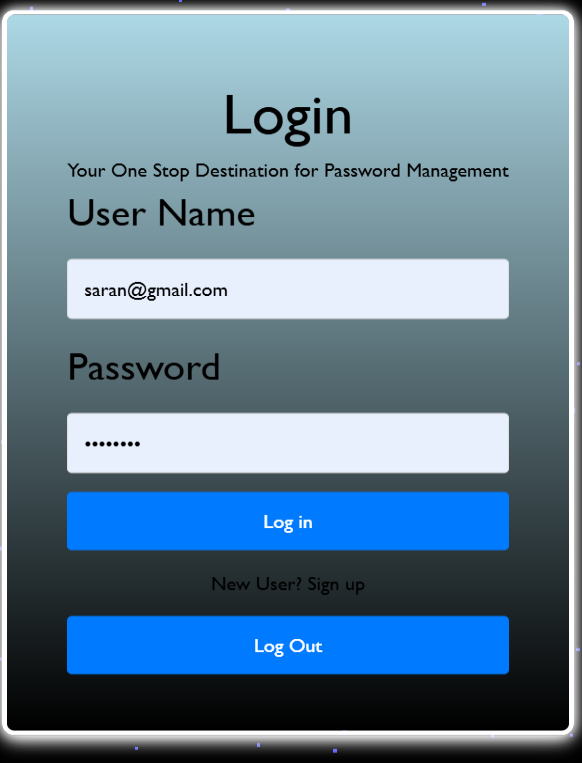


FIGURE 3.3.1

10

**PASSWORD GENERATOR:**

It creates secure, unique, customizable, and complex passwords to protect user data.

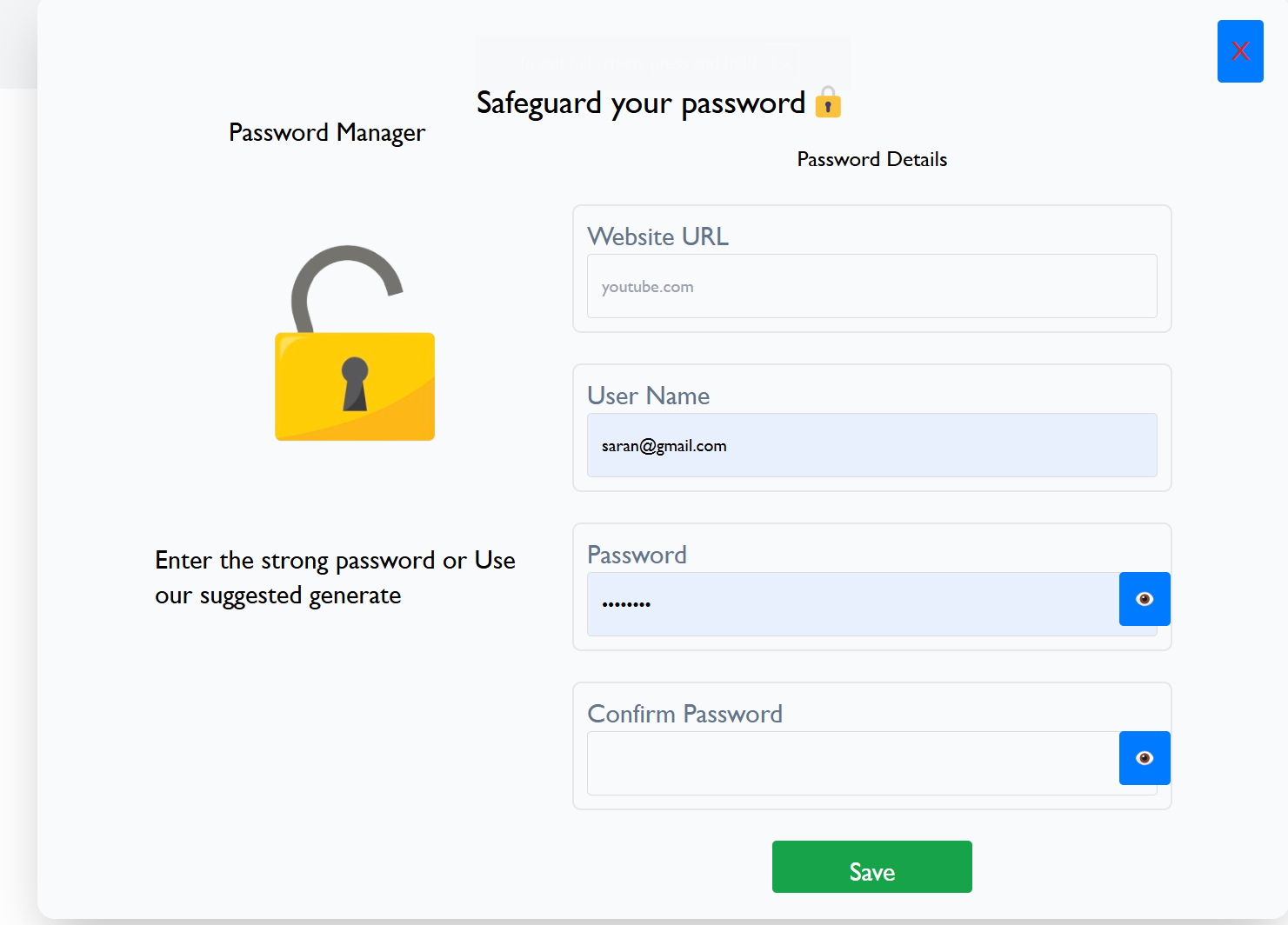


Figure 3.3.2

11

**PASSWORD MANAGER**:

In this page the user can update ,view and delete his password.

A white rectangular object with a black border

Description automatically generated

FIGURE 3.3.3

**UPDATE PASSWORD**:

It securely organizes users passwords, ensuring easy access and enhanced data protection.

A screenshot of a computer screen

Description automatically generated FIGURE 3.3.4

12

**HELP PAGE**:

The help page provides users with clear instructions, FAQs, and tips to navigate the system effectively.

A screenshot of a computer

Description automatically generated

FIGURE 3.3.5

13

**DASHBOARD**

Securely stores, organizes, and auto-fills passwords, help users protect their credentials .

A screenshot of a computer

Description automatically generated

FIGURE 3.3.6

14

**CHAPTER 4**

**DETAILED DESIGN**

**4.1 ARCHITECTURAL DESIGN**

The architectural design includes various diagrams such as use case diagram, sequence diagram and activity diagram for both user and admin modules.

**4.1.1 MODULE CHARACTERISTICS**

The project contains two modules as follows and their description is given below.

User module

i) Sign up

ii) Login

iii) Password generator

iv) Add password

v) Help page

**4.1.1 User Module**

The user module contains a Sign up Page login page, Password generator page, add password page, view password page and help page. The user can view all the pages as mentioned in the user module flow FIGURE 4.1

A diagram of a computer program

Description automatically generated

FIGURE 4.1 FLOWCHART

16

**4.1.1.2 USE CASE DIAGRAM**

A use case diagram is a dynamic or behaviour diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are set of actions, services, and functions that the system needs to perform. A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. The use cases are represented by either circles or ellipses. Due to their simplistic nature, use case diagrams can be a good communication tool for stakeholders

A diagram of a login and password manager system

Description automatically generated

FIGURE 4.2 USE CASE DIAGRAM

17

**4.1.1.3 SEQUENCE DIAGRAM**

Sequence chart diagram is one of the five UML diagrams used to model the dynamic nature of a system. The admin defines the sequence of an object during its lifetime. FIGURE 4.3

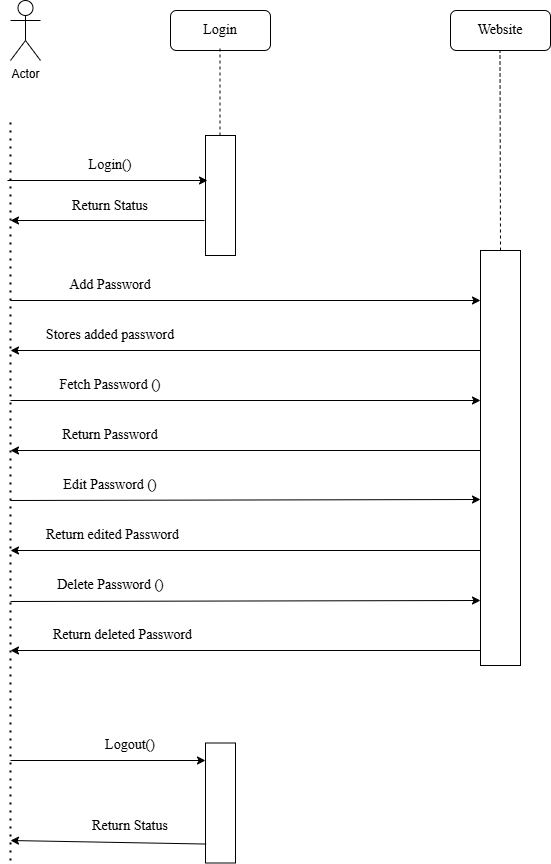


FIGURE 4.3 SEQUENCE DIAGRAM

**4.2 INTERFACE DESIGN**

The application provides various interfaces to the users that make them more convenient with the application. The user interface contains Home, About, Projects done by the company. Further the user can contact the company by giving their details, and approach the company. This is simple to use and clear to understand. The user can further provide feedback regarding the company. The user interface is more attractive and consistent on all interface screens. The interface is completely responsive.

18

**4.3 DATABASE DESIGN**

In this password manager system, Firebase serves as the primary database, organizing data into collections and documents. Key collections include Services, which manages password storage and retrieval functions; Admin Management, where administrators can be added or removed; User Feedback, which stores customer feedback; and User Details, which captures contact information such as names, emails, and phone numbers for users who interact with the system. Firebase’s real-time data syncing ensures that users always access the most up-to-date information across devices..

**4.4 OUTPUT DESIGN**

For the password manager system, output design is crucial in presenting secure, clear, and user-friendly information to end-users managing their passwords and sensitive data. The output should be organized in a format that is easy to understand, allowing users to view, update, and manage their stored passwords, security settings, and account details with confidence. Previewing the output, such as password strength indicators, recent activity logs, and encryption status, is essential for users, as they need to quickly verify the accuracy and security of their information. The design is therefore optimized to be both informative and visually intuitive, ensuring users have a seamless experience and peace of mind in managing their credentials.

19

**CHAPTER 5**

**TESTING**

**5.1 UNIT TESTING**

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unit testing can be done manually, but is often automated. In this process, each module is considered as individual units and are tested for proper operation. If each module meets up with the user’s requirement, then it is subjected to integration testing where more than one module is integrated and tested.

**5.2 REGRESSION TESTING**

Regression testing helps in identifying any unintended side effects or issues that may arise due to code changes, updates to dependencies, or modifications in the underlying infrastructure. By rerunning test cases, the development team can ensure that previously implemented features and functionality have not been negatively impacted by recent updates.The primary objective of regression testing is to ensure that the website remains fully functional, reliable, and user-friendly, even after modifications or enhancements have been made.Overall, regression testing serves as an essential quality assurance measure to maintain the integrity and performance of the website throughout its lifecycle. It helps in minimizing the risk of functional regressions, ensuring that the website continues to meet the expectations and requirements of the end users and stakeholders.

**5.3 VALIDATION TESTING**

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements. Validation Testing ensures that the service actually meets the client's needs. It can also be defined as to demonstrate that the service fulfils its intended use when deployed on appropriate environment.

20

**5.4 VERIFICATION TESTING**

Verification is the process of evaluating work-service of a development phase to determine whether they meet the specified requirements. When developing each module, the individuality of the service is checked at its development stage. Thus ,the modules must be verified at the development stage.

**5.5 INTEGRATION TESTING**

Integration tests are designed to test the integrated software components to determine if they actually run as a program. It is specifically aimed at exposing problems that arise from the combination of components. Integration testing is done after integration of the model with the core service. The integration testing can be done for our project by integrating the user module.

21

**CHAPTER 6**

**CONCLUSION AND FUTURE WORK**

The development of the password manager application represents a pivotal step toward meeting the growing need for secure digital management solutions. By utilizing advanced technologies like ReactJS, Firebase, Node.js, and AES encryption, the project aims to improve security, enhance accessibility, and establish a trusted online presence. The application offers a seamless user experience, robust encryption, and intuitive credential management, providing users with convenience and peace of mind. Moving forward, ongoing refinement and adaptation will be essential to remain competitive in the cybersecurity market. Future plans include enhancing personalization through advanced algorithms, integrating biometric authentication, promoting data privacy initiatives, and expanding cross-platform support. Additionally, efforts toward international expansion, voice-activated access, advanced analytics, community engagement, and premium membership programs will help maintain a competitive advantage and drive sustained growth in the evolving digital landscape.

22

**APPENDIX 1**

**CODING:**

**App.jsx:**

import { useState } from 'react'

import Login from './Login'

import { BrowserRouter, Route, Routes } from 'react-router-dom'

import Signup from './SignUp'

import Dashboard from './Dashboard'

import { AuthProvider } from '../context/AuthContext'

import PasswordGenerator from './PasswordGenerator'

function App() {

return (

<BrowserRouter>

<AuthProvider>

<Routes>

<Route path="/" Component={Login}/>

<Route path='/login' Component={Login}/>

<Route path='/signup' Component={Signup}/>

<Route path='/dashboard/user/:uid' Component={Dashboard}/>

<Route path='/dashboard/user/:uid/password\_generator' Component={PasswordGenerator}/>

</Routes>

</AuthProvider>

</BrowserRouter>

23

**)**

**}**

**Background.js:**

// Background.js

import { useEffect, useRef } from 'react';

import \* as THREE from 'three';

import styles from '../styles/login.module.css'; // Assuming CSS Modules are used

export default function Background() {

const mountRef = useRef(null); // Reference to attach the Three.js renderer to a div

useEffect(() => {

// Scene setup

const scene = new THREE.Scene();

// Camera setup

const camera = new THREE.PerspectiveCamera(

75,

window.innerWidth / window.innerHeight,

0.1,

1000

);

camera.position.z = 5;

const renderer = new THREE.WebGLRenderer();

renderer.setSize(window.innerWidth, window.innerHeight);

renderer.setPixelRatio(window.devicePixelRatio);

24

mountRef.current.appendChild(renderer.domElement); // Attach renderer to the div

const particleCount = 1000;

const positions = new Float32Array(particleCount \* 3);

const velocities = new Float32Array(particleCount);

for (let i = 0; i < particleCount; i++) {

positions[i \* 3] = (Math.random() - 0.5) \* 20;

positions[i \* 3 + 1] = Math.random() \* 20;

positions[i \* 3 + 2] = (Math.random() - 0.5) \* 20;

velocities[i] = 0.006 + Math.random() \* 0.05;

}

const particles = new THREE.BufferGeometry();

particles.setAttribute('position', new THREE.BufferAttribute(positions, 3));

const particleMaterial = new THREE.PointsMaterial({ color: 0x5555ff, size: 0.1 });

const particleSystem = new THREE.Points(particles, particleMaterial);

scene.add(particleSystem);

const animate = () => {

requestAnimationFrame(animate);

// Update particle positions to create raindrop effect

const positionAttribute = particles.getAttribute('position');

for (let i = 0; i < particleCount; i++) {

25

positionAttribute.array[i \* 3 + 1] -= velocities[i];

if (positionAttribute.array[i \* 3 + 1] < -10) {

positionAttribute.array[i \* 3 + 1] = 10;

}

}

positionAttribute.needsUpdate = true;

renderer.render(scene, camera);

};

animate();

**Dashboard.jsx**

import React, { useState } from 'react'

import InsertData from './InsertData'

export default function Hero() {

const [isAddPassword, setIsAddPassword] = useState(false)

function handleAddPassword(){

setIsAddPassword(!isAddPassword)

}

return (

<>

{isAddPassword && <InsertData isAddPassword={isAddPassword} setIsAddPassword={setIsAddPassword}/>}

<div className='flex flex-col max-w-custom mx-auto items-start h-1/3 bg-white py-6 gap-5 '>

26

<h1 className='text-2xl left-100 box-border z-50 bg'>Password Manager</h1>

<div>

<p className='text-zinc-400 mt-11'>See, save and edit your passwords</p>

</div>

<div className='flex justify-evenly rounded-2xl items-center w-full h-48 border-2 content-center'>

<div className='flex flex-col w-full h-full justify-evenly basis-1/2'>

<h1 className='text-2xl'>Add Password</h1>

<p>Click here to Save Passwords</p>

<button onClick={handleAddPassword} className="w-56 relative h-auto p-3 rounded-md border-2 after:content-['+'] after:absolute after:right-3 text-xl hover:shadow-lg text-white bg-green-500 hover:bg-green-400">Add New Password</button>

</div>

<div>

<img src="/icons8-show-password-96.png" alt="" />

</div>

</div>

</div>

</>

)

}

i import React, { useEffect, useState } from 'react';

import Navbar from './Navbar';

import { useNavigate } from 'react-router-dom';

import { useAuth } from '../context/AuthContext';

const PasswordGenerator = ({}) => {

const navigate = useNavigate()

const { currentUser,

setCurrentUser,

signUp,

logOut, 27

logIn,

url,

setUrl,

setWebsites,

websites,

fetchData} = useAuth()

const [password, setPassword] = useState('');

const [purpose, setPurpose] = useState('');

const [length, setLength] = useState(10);

const [includeLowercase, setIncludeLowercase] = useState(true);

const [includeUppercase, setIncludeUppercase] = useState(true);

const [includeNumbers, setIncludeNumbers] = useState(true);

const [includeSymbols, setIncludeSymbols] = useState(true);

const [passwordStrength, setPasswordStrength] = useState('');

const lowercaseLetters = 'abcdefghijklmnopqrstuvwxyz';

const uppercaseLetters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';

const numbers = '0123456789';

const symbols = '!@#$%^&\*()\_+{}:"<>?[];,./';

useEffect(()=>{

if(currentUser == null){

navigate('/login')

}

}, [currentUser])

const generatePassword = () => {

let charPool = '';

if (includeLowercase) charPool += lowercaseLetters;

if (includeUppercase) charPool += uppercaseLetters;

if (includeNumbers) charPool += numbers;

if (includeSymbols) charPool += symbols;

if (charPool === '') {

alert("Please select at least one character type!");

return;

}

let newPassword = '';

for (let i = 0; i < length; i++) {

const randomIndex = Math.floor(Math.random() \* charPool.length);

newPassword += charPool[randomIndex];

}

setPassword(newPassword);

if (includeLowercase && includeUppercase && includeNumbers && includeSymbols) {

setPasswordStrength('strong');

} else {

setPasswordStrength('weak');

}

};

28

Const passwordBorderClass = passwordStrength === 'strong' ? 'border-green-500' : 'border-red-

1500';

return (

<>

<Navbar/>

<div className="flex flex-col items-center justify-center min-h-screen bg-gray-100">

<div className="bg-white p-6 rounded-lg shadow-md w-80">

<h2 className="text-2xl font-semibold mb-4 text-center text-gray-800">

Password Generator

</h2>

<label className="block mb-4">

<span className="text-gray-700 font-medium">Purpose of the Password</span>

<input

type="text"

value={purpose}

onChange={(e) => setPurpose(e.target.value)}

placeholder="e.g., email, bank account"

className="mt-1 block w-full px-3 py-2 bg-white border border-gray-300 rounded-md shadow-sm placeholder-gray-400 focus:outline-none focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm"

/>

</label>

<label className="block mb-4">

<span className="text-gray-700 font-medium">Password Length</span>

<input

type="number"

value={length}

onChange={(e) => setLength(Number(e.target.value))}

min="4"

max="20"

className="mt-1 block w-full px-3 py-2 bg-white border border-gray-300 rounded-md shadow-sm placeholder-gray-400 focus:outline-none focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm"

/>

</label>

<div className="mb-4">

<label className="inline-flex items-center">

<input

type="checkbox"

checked={includeLowercase}

onChange={(e) => setIncludeLowercase(e.target.checked)}

className="form-checkbox text-indigo-600"

/>

<span className="ml-2 text-gray-700">Include Lowercase Letters</span>

</label>

</div>

<div className="mb-4"> 29 <label className="inline-flex items-center">

<input

type="checkbox"

checked={includeUppercase}

onChange={(e) => setIncludeUppercase(e.target.checked)}

className="form-checkbox text-indigo-600"

/>

<span className="ml-2 text-gray-700"> Include Uppercase Letters</span>

</label>

</div>

<div className="mb-4">

<label className="inline-flex items-center">

<input

type="checkbox"

checked={includeNumbers}

onChange={(e) => setIncludeNumbers(e.target.checked)}

className="form-checkbox text-indigo-600"

/>

<span className="ml-2 text-gray-700">Add Numbers</span>

</label>

</div>

<div className="mb-4">

<label className="inline-flex items-center">

<input

type="checkbox"

checked={includeSymbols}

onChange={(e) => setIncludeSymbols(e.target.checked)}

className="form-checkbox text-indigo-600"

/>

<span className="ml-2 text-gray-700">Need Symbols</span>r

</label>

</div>

<button

onClick={generatePassword}

className="w-full bg-indigo-600 text-white py-2 rounded-md text-lg font-medium hover:bg-indigo-500 focus:outline-none focus:ring-2 focus:ring-indigo-500 focus:ring-opacity-50"

>

Generate Password

</button>

{password && (

<div className={`mt-6 text-center border-2 p-3 rounded-md ${passwordBorderClass}`}>

<h3 className="text-lg font-semibold text-gray-700">Generated Password for "{purpose}":</h3>

<p className="mt-2 text-lg font-bold text-indigo-600 break-words">{password}</p>

<p className="mt-2 text-md font-medium text-gray-600">

This password is {passwordStrength}.

</p>

</div>

)}

</div> 30

</div>

</>

);

};

export default PasswordGenerator;mport React, { useEffect, useState } from 'react';

import Navbar from './Navbar';

import { useNavigate } from 'react-router-dom';

import { useAuth } from '../context/AuthContext';

const PasswordGenerator = ({}) => {

  const navigate = useNavigate()

  const {  currentUser,

    setCurrentUser,

    signUp,

    logOut,

    logIn,

    url,

    setUrl,

    setWebsites,

    websites,

    fetchData} = useAuth()

  const [password, setPassword] = useState('');

  const [purpose, setPurpose] = useState('');

  const [length, setLength] = useState(10);

  const [includeLowercase, setIncludeLowercase] = useState(true);

  const [includeUppercase, setIncludeUppercase] = useState(true);

  const [includeNumbers, setIncludeNumbers] = useState(true);

  const [includeSymbols, setIncludeSymbols] = useState(true);

  const [passwordStrength, setPasswordStrength] = useState('');

  const lowercaseLetters = 'abcdefghijklmnopqrstuvwxyz';

  const uppercaseLetters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';

  const numbers = '0123456789';

  const symbols = '!@#$%^&\*()\_+{}:"<>?[];,./';

  useEffect(()=>{

    if(currentUser == null){

      navigate('/login')

    }

  }, [currentUser])

  const generatePassword = () => {

    let charPool = '';

    if (includeLowercase) charPool += lowercaseLetters;

    if (includeUppercase) charPool += uppercaseLetters;

    if (includeNumbers) charPool += numbers;

    if (includeSymbols) charPool += symbols;

    if (charPool === '') {

      alert("Please select at least one character type!");

      return;

    }

31

    let newPassword = '';

    for (let i = 0; i < length; i++) {

      const randomIndex = Math.floor(Math.random() \* charPool.length);

      newPassword += charPool[randomIndex];

    }

    setPassword(newPassword);

    if (includeLowercase && includeUppercase && includeNumbers && includeSymbols) {

      setPasswordStrength('strong');

    } else {

      setPasswordStrength('weak');

    }

  };

  const passwordBorderClass = passwordStrength === 'strong' ? 'border-green-500' : 'border-red-500';

  return (

    <>

      <Navbar/>

      <div className="flex flex-col items-center justify-center min-h-screen bg-gray-100">

        <div className="bg-white p-6 rounded-lg shadow-md w-80">

          <h2 className="text-2xl font-semibold mb-4 text-center text-gray-800">

            Password Generator

          </h2>

          <label className="block mb-4">

            <span className="text-gray-700 font-medium">Purpose of the Password</span>

            <input

              type="text"

              value={purpose}

              onChange={(e) => setPurpose(e.target.value)}

              placeholder="e.g., email, bank account"

              className="mt-1 block w-full px-3 py-2 bg-white border border-gray-300 rounded-md shadow-sm placeholder-gray-400 focus:outline-none focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm"

            />

          </label>

          <label className="block mb-4">

            <span className="text-gray-700 font-medium">Password Length</span>

            <input

              type="number"

              value={length}

              onChange={(e) => setLength(Number(e.target.value))}

              min="4"

              max="20"

              className="mt-1 block w-full px-3 py-2 bg-white border border-gray-300 rounded-md shadow-sm placeholder-gray-400 focus:outline-none focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm"

            32

          </label>

          <div className="mb-4">

            <label className="inline-flex items-center">

              <input

                type="checkbox"

                checked={includeLowercase}

                onChange={(e) => setIncludeLowercase(e.target.checked)}

                className="form-checkbox text-indigo-600"

              />

              <span className="ml-2 text-gray-700">Include Lowercase Letters</span>

            </label>

          </div>

          <div className="mb-4">

            <label className="inline-flex items-center">

              <input

                type="checkbox"

                checked={includeUppercase}

                onChange={(e) => setIncludeUppercase(e.target.checked)}

                className="form-checkbox text-indigo-600"

              />

              <span className="ml-2 text-gray-700"> Include Uppercase Letters</span>

            </label>

          </div>

          <div className="mb-4">

            <label className="inline-flex items-center">

              <input

                type="checkbox"

                checked={includeNumbers}

                onChange={(e) => setIncludeNumbers(e.target.checked)}

                className="form-checkbox text-indigo-600"

              />

              <span className="ml-2 text-gray-700">Add Numbers</span>

            </label>

          </div>

          <div className="mb-4">

            <label className="inline-flex items-center">

              <input

                type="checkbox"

                checked={includeSymbols}

                onChange={(e) => setIncludeSymbols(e.target.checked)}

                className="form-checkbox text-indigo-600"

              />

              <span className="ml-2 text-gray-700">Need Symbols</span>r

            </label>

          </div>

          <button

            onClick={generatePassword}

33

className="w-full bg-indigo-600 text-white py-2 rounded-md text-lg font-medium hover:bg-indigo-500 focus:outline-none focus:ring-2 focus:ring-indigo-500 focus:ring-opacity-50"

          >

            Generate Password

          </button>

          {password && (

            <div className={`mt-6 text-center border-2 p-3 rounded-md ${passwordBorderClass}`}>

              <h3 className="text-lg font-semibold text-gray-700">Generated Password for "{purpose}":</h3>

              <p className="mt-2 text-lg font-bold text-indigo-600 break-words">{password}</p>

              <p className="mt-2 text-md font-medium text-gray-600">

                This password is {passwordStrength}.

              </p>

            </div>

          )}

        </div>

      </div>

    </>

  );

};

export default PasswordGenerator;

**Database:**

const mongoose = require("mongoose");

const connectDatabase = () => {

mongoose

.connect(process.env.DB\_URL, {

useNewUrlParser: true,

useUnifiedTopology: true,

})

.then((data) => {

console.log(`mongod connected with server: ${data.connection.host}`); });

};

module.exports = connectDatabase;

**34**

**Login:**

import React, { useEffect, useState } from "react";

import styles from "../../styles/styles";

const AdminDashboardMain = () => {

const dispatch = useDispatch();

const { adminOrders,adminOrderLoading } = useSelector((state) => state.order); const { sellers } = useSelector((state) => state.seller);

useEffect(() => {

dispatch(getAllOrdersOfAdmin());

dispatch(getAllSellers());

}, []);

31

const adminEarning = adminOrders && adminOrders.reduce((acc,item) => acc + item.totalPrice \* .10, 0);

const adminBalance = adminEarning?.toFixed(2);

const row = [];

return (

<>

{

adminOrderLoading ? (

<Loader />

) :

<h5 className="pt-2 pl-[36px] text-[22px] font-[500]">$ {adminBalance}</h5> </div>

35

<div className="w-full mb-4 800px:w-[30%] min-h-[20vh] bg-white shadow rounded px-2 py-5">

<div className="flex items-center">

<MdBorderClear size={30} className="mr-2" fill="#00000085" /> <h3

className={`${styles.productTitle} !text-[18px] leading-5 !font-[400] text- [#00000085]`}

>

All Sellers

</h3>

</div>

</Link>

</div>

<div className={`mt-6 text-center border-2 p-3 rounded-md ${passwordBorderClass}`}>

<h3 className="text-lg font-semibold text-gray-700">Generated Password for "{purpose}":</h3>

<p className="mt-2 text-lg font-bold text-indigo-600 break-words">{password}</p>

<p className="mt-2 text-md font-medium text-gray-600">

This password is {passwordStrength}.

</p>

</div>

)}

36

export default PasswordGenerator;

**Checkout:**

import React, { useState } from "react";

import styles from "../../styles/styles";

import { Country, State } from "country-state-city";

import { useNavigate } from "react-router-dom";

import { useSelector } from "react-redux";

import { useEffect } from "react";

import axios from "axios";

import { server } from "../../server";

import { toast } from "react-toastify";

const paymentSubmit = () => {

if (

address1 === "" ||

address2 === "" ||

zipCode === null ||

country === "" ||

city === ""

) {

toast.error("Please choose your delivery address!");

} else {

const shippingAddress = {

address1,

address2,

zipCode,

37

city,

};

const orderData = {

cart,

totalPrice,

subTotalPrice,

shipping,

discountPrice,

shippingAddress,

user,

};

// update local storage with the updated orders array

localStorage.setItem("latestOrder", JSON.stringify(orderData));

navigate("/payment");

}

};

const subTotalPrice = cart.reduce(

(acc, item) => acc + item.qty \* item.discountPrice,

0

);

// this is shipping cost variable

const shipping = subTotalPrice \* 0.1;

const handleSubmit = async (e) => {

e.preventDefault();

const name = couponCode;

await axios.get(`${server}/coupon/get-coupon-value/${name}`).then((res) => { const shopId = res.data.couponCode?.shopId;

const couponCodeValue = res.data.couponCode?.value;

38

if (res.data.couponCode !== null) {

const isCouponValid =

cart && cart.filter((item) => item.shopId === shopId);

return (

<div className="w-full flex flex-col items-center py-8">

<div className="w-[90%] 1000px:w-[70%] block 800px:flex"> <div className="w-full 800px:w-[65%]">

<ShippingInfo

user={user}

country={country}

setCountry={setCountry}

city={city}

/>

</div>

</div>

<div

className={`${styles.button} w-[150px] 800px:w-[280px] mt-10`} onClick={paymentSubmit}

>

<h5 className="text-white">Go to Payment</h5>

</div>

</div>

);

};

return (

<div className="w-full 800px:w-[95%] bg-white rounded-md p-5 pb-8"> <h5 className="text-[18px] font-[500]">Shipping Address</h5> <br />

<div className="w-full flex pb-3">

39

<div className="w-[50%]">

<label className="block pb-2">Country</label>

<select

className="w-[95%] border h-[40px] rounded-[5px]"

value={country}

onChange={(e) => setCountry(e.target.value)}

>

))}

</select>

</div>

<div className="w-[50%]">

<label className="block pb-2">City</label>

<select

className="w-[95%] border h-[40px] rounded-[5px]"

value={city}

onChange={(e) => setCity(e.target.value)}

>

<option className="block pb-2" value="">

Choose your City

</option>

{State &&

State.getStatesOfCountry(country).map((item) => (

<option key={item.isoCode} value={item.isoCode}>

{item.name}

</option>

))}

</select>

40

</div>

<div className="w-full flex pb-3">

<div className="w-[50%]">

<label className="block pb-2">Address1</label>

<input

type="address"

required

value={address1}

onChange={(e) => setAddress1(e.target.value)}

className={`${styles.input} !w-[95%]`}

/>

</div>

</div>

<div></div>

</form>

<h5

className="text-[18px] cursor-pointer inline-block"

onClick={() => setUserInfo(!userInfo)}

>

Choose From saved address

</h5>

</div>

);

};

return (

<div className="w-full bg-[#fff] rounded-md p-5 pb-8">

41

<div className="flex justify-between">

<h3 className="text-[16px] font-[400] text-[#000000a4]">subtotal:</h3> <h5 className="text-[18px] font-[600]">Rs.{subTotalPrice}</h5> </div>

<br />

<div className="flex justify-between">

<h3 className="text-[16px] font-[400] text-[#000000a4]">shipping:</h3> <h5 className="text-[18px] font-[600]">Rs.{shipping.toFixed(2)}</h5> </div>

</div>

);

};

export default Checkout;

**Firebase.jsxx:**

// Import the functions you need from the SDKs you need

import { initializeApp } from "firebase/app";

import { getAnalytics } from "firebase/analytics";

import { getAuth } from "firebase/auth";

import { getFirestore } from "firebase/firestore";

// TODO: Add SDKs for Firebase products that you want to use

42

pp's Firebase configuration

// For Firebase JS SDK v7.20.0 and later, measurementId is optional

const firebaseConfig = {

apiKey: import.meta.env.VITE\_API\_KEY,

authDomain: import.meta.env.VITE\_AUTH\_DOMAIN,

projectId: import.meta.env.VITE\_PROJECT\_ID,

storageBucket: import.meta.env.VITE\_STORAGE\_BUCKET,

messagingSenderId: import.meta.env.VITE\_MESSAGE\_SENDER\_ID,

appId: import.meta.env.VITE\_APP\_ID,

measurementId: import.meta.env.VITE\_MEASUREMENT\_ID

};

// Initialize Firebase

const app = initializeApp(firebaseConfig);

export const auth = getAuth(app)

43

port const db = getFirestore(app)

(app);**Route.js:**

import LoginPage from "../pages/Login";

import SignupPage from "../pages/SignupPage";

import ActivationPage from "../pages/ActivationPage";

import HomePage from "../pages/HomePage";

import ProductsPage from "../pages/ProductsPage";

import BestSellingPage from "../pages/BestSellingPage";

import ShopCreatePage from "../pages/ShopCreate";

import SellerActivationPage from "../pages/SellerActivationPage";

import ShopLoginPage from "../pages/ShopLoginPage";

import OrderDetailsPage from "../pages/OrderDetailsPage";

import TrackOrderPage from "../pages/TrackOrderPage";

import Feedback from "../pages/Feedback.jsx";

export {

LoginPage,

SignupPage,

ActivationPage,

HomePage,

ProductsPage,

BestSellingPage,

EventsPage,

ContactPage,

FAQPage,

44

AboutPage,

};

**Data.js:**

// navigation Data

export const navItems = [

{

title: "Home",

url: "/",

},

{

title: "About Us",

url: "/about",

},

{

title: "Products",

url: "/products",

},

{

title: "FAQ",

url: "/faq",

},

{

title: "Contact Us",

url: "/contact",

},

];

45

{

id: 8,

title: "Jeans",

subTitle: "",

image\_Url:

"https://images.lee.com/is/image/Lee/112343021-HERO?$KDP-LARGE2$", },

];

export const footerSupportLinks = [

{

name: "FAQ",

link: "/faq",

},

{

name: "Contact Us",

link: "/contact",

},

];

**Server.js:**

import React from "react";

import ReactDOM from "react-dom";

import App from "./App";

import reportWebVitals from "./reportWebVitals";

import { Provider } from "react-redux";

import Store from "./redux/store";

ReactDOM.render(

<Provider store={Store}>

**46**

**APPENDIX 2**

**SNAPSHOTS**

A screenshot of a login screen

Description automatically generated

FIGURE A2.0 SIGNUP PAGE

FIGURE A2.1 LOGIN PAGE

47

A screenshot of a computer screen

Description automatically generated

FIGURE A2.2 PASSWORD MANAGER

A white rectangular object with a black border

Description automatically generated

FIGURE 4.2.5 PASSWORD VIEWER

48

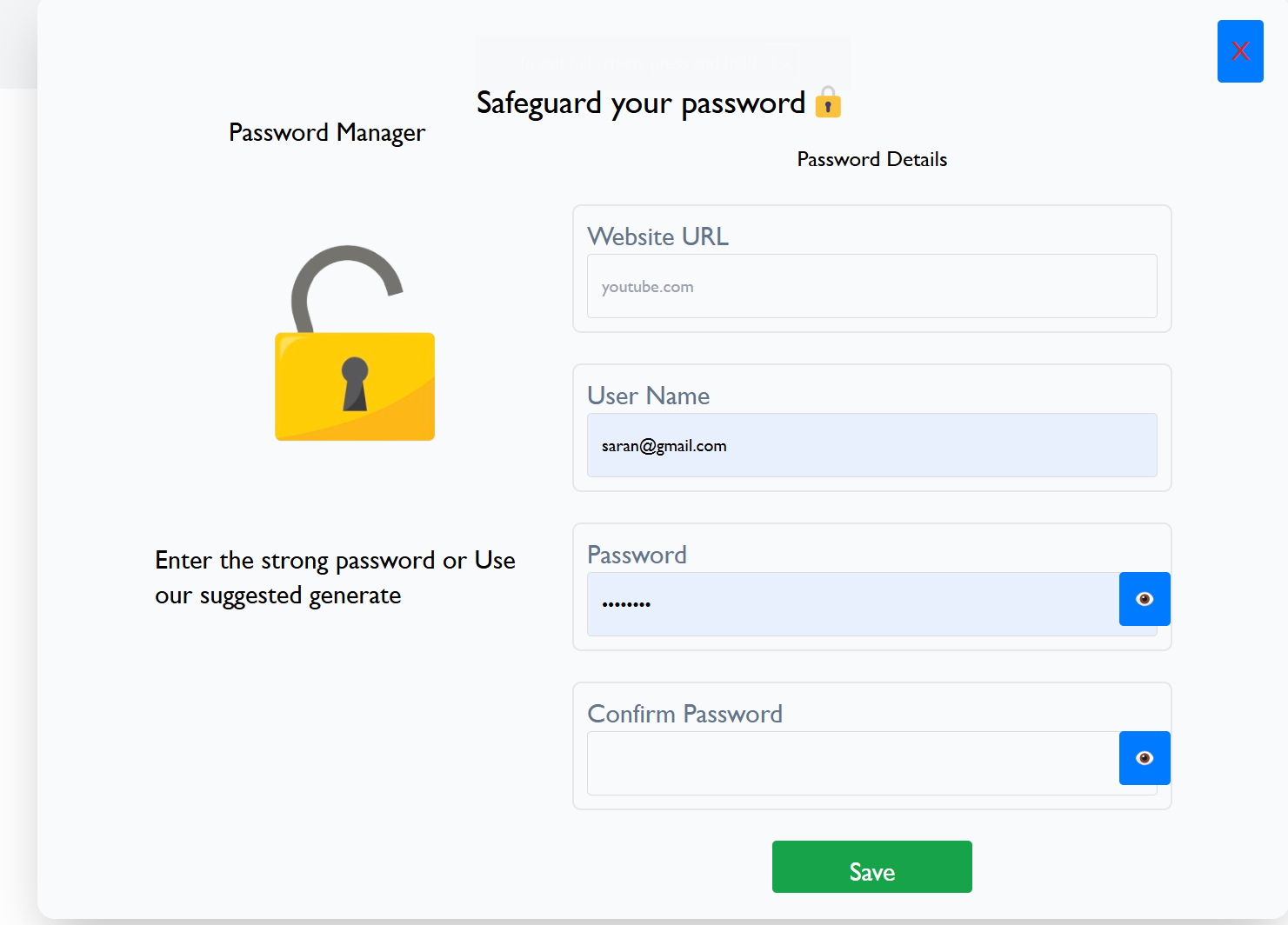


FIGURE A2.4 ADD PASSWORD

49

A screenshot of a computer

Description automatically generated

FIGURE A2.5 USER DASHBOARD

54

50

**REFERENCES**

|  |  |
| --- | --- |
| [1] | Firebase.google.com official documentation of firebase and all its tools and  Services as Baas  https://firebase.google.com/docs |