Submitted by:

Sibtul Hussain
01-135211-076
Aizaz Ahmed
01-135211-009

SAFEPASS-Secure Authentication Framework and Password Evaluation System

Bachelor of Science in Computer Science

Supervisor: Sir Syed Ur Rehman

Department of Computer Science
Bahria University, Islamabad

Certificate

We accept the work contained in the report titled **SAFEPASS-Secure Authentication Framework and Password Evaluation System**, written by Aizaz Ahmed **AND** Sibtul Hussein as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Computer Science.

Approved by:

Supervisor: Sir Syed Ur Rehman

Internal Examiner: Name of the Internal Examiner

External Examiner: Name of the External Examiner

Project Coordinator: Name of the Project Coordinator

Head of the Department: Dr Muhammad Khurram Ehsan

---

th, 2024

Dedication

I dedicate this work to my parents, family and teachers who have always been my greatest supporters, motivators, and inspirations. They are the most important people in my life. My success has been fueled by their support and love, and I am forever indebted to them.

Abstract

The SAFEPASS Password Manager is an advanced, web-based application designed to provide comprehensive password management solutions for both individuals and businesses. It offers a range of features that make it easier for users to manage their passwords securely, with various scanning and monitoring tools that help maintain strong security practices.

With SAFEPASS, users can safely store, share, and manage their passwords—an essential need in today's world where cyber-attacks are increasingly common. The application allows users to create strong, unique passwords for every account and store them securely using encryption. By doing so, users don't have to remember multiple passwords, as they can access everything with just one click.

Additionally, SAFEPASS includes tools for real-time monitoring of network activities, which help detect potential security risks and send alerts if anything suspicious is found. This enables users to respond to security threats quickly, minimizing the chances of damage.

The application also streamlines password management by allowing users to generate secure passwords, update them regularly, and share them safely with others when necessary. This eliminates the need to share passwords through less secure methods or keep track of numerous credentials.

SAFEPASS ultimately improves users' daily workflows by enhancing security, privacy, and productivity. By automating best practices for password management, users can focus on other tasks, confident that their passwords are well-protected.

For anyone serious about security, SAFEPASS Password Manager is a must-have tool. It offers a suite of features that help both individuals and businesses strengthen their security measures, reduce the risk of cyber-attacks, and boost efficiency. Whether you're managing passwords for personal use or for a larger organization, SAFEPASS provides a comprehensive and secure solution.

Acknowledgments

I would like to express my heartfelt gratitude to Allah for His blessings and guidance throughout thisproject. I could not have achieved this feat without his support. I am deeply grateful to my parents,family, and teachers, whose encouragement, motivation, and unwavering support played an important role in my academic and personal growth. My sincere thanks to my team members and supervisors. Dedication and hard work are essential in bringing SAFEPASS Password Manager to life. Finally, I would like to thank everyone who has directly or indirectly supported this project. Your valuable insights and feedback greatly improve the quality and effectiveness of our work.

Sibtul Hussein
Islamabad, Pakistan

Aizaz Ahmed
Islamabad, Pakistan

2024

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Project Background

SAFEPASS Password Manager is a powerful web-based application designed to help users manage their passwords across multiple accounts. It offers a complete solution for safely storing and managing passwords in a convenient location. One of its standout features is the ability to detect data breaches and notify users if any of their data has been compromised. It provides an additional layer of protection from online threats. The software allows users to securely create, share, and manage passwords. At the same time, it automatically performs recommended password management. In doing so, it helps users improve their security, privacy, and overall performance in their daily activities. SAFEPASS is more than just a simple password manager. It includes additional resources such as two-factor authentication, data breach investigation, and the ability to configure devices more reliably. All of which helps improve network security. In summary, SAFEPASS Password Manager is a trusted tool for those looking to improve their cybersecurity practices and threat management.

1.2 Motivations and Challenges

A key component of protecting user accounts from fraudsters is the usage of passwords. According to recent data breaches [**?**], weak passwords are a major contributor to data breaches. Unfortunately, many people use passwords that are simple to guess, including popular dictionary phrases, pet names, or months of their birth. This makes it easier for hackers to access user accounts. Furthermore, a common error is the use of the same secure password on numerous websites. Although this makes it easier to remember passwords, it also makes all accounts that use that password susceptible if the password is compromised [**?**].

Password managers were created to solve this issue. These automatic password managers are secure and help users create strong passwords that they can save in a single master password hash. However, password managers themselves have had several data breaches [**?**] and security flaws. As a result, people are less likely to trust freely available technologies and are in need of a more secure solution.

Developing a safe password manager was difficult for several reasons. First, it was challenging to acquire a high level of unpredictability while creating cryptographic passwords without sacrificing usability. Additionally, a strong security architecture was needed to

guarantee that user passwords are transferred and stored safely. The password manager must also be easy to use in order to promote adoption and continued use. These issues must be resolved to develop a user-friendly and secure password manager.

1.3 Project Plan

The project schedule was made up of two phases: FYP I and FYP II. In each of these phases, several tasks were completed. The project plan was created with the intention of being finished within a year while accomplishing all goals. Our team successfully completed it within this time in several phases.

1.4 Problem Statement

Many password managers, including well-known ones like Passbolt, do a good job of offering solid security features. However, they often miss out on important functionalities that people need to fully protect their sensitive information. For example, not all password managers offer a "security kit" or recovery option for users who forget their passwords, which can leave them locked out and frustrated with no easy way to regain access.

Another common issue is the lack of two-factor authentication (2FA), which is an extra layer of security that's essential for keeping accounts safe from hackers. Surprisingly, many password managers don't include this feature. Additionally, secure password sharing is often missing, meaning users have to rely on less secure methods like email or messaging apps to share their credentials, which can expose them to risks.

There's also the problem of sign-in activity monitoring. Without this feature, users can't easily track if someone is trying to access their account without permission. Some password managers also don't allow users to set trusted devices, which is another layer of protection by ensuring only approved devices can access accounts. Finally, many tools don't offer alerts or monitoring for password breaches, so users might not even know if their credentials have been exposed in a data breach.

Overall, while password managers are useful, these gaps in features show that they often fall short in providing the complete security package that users need for true peace of mind.

1.5 Project Scope

SafePass is a comprehensive web-based application specifically designed to empower users in managing their passwords securely while ensuring the adoption of robust security measures. This tool offers an array of advanced scanning and monitoring functionalities tailored to address the needs of individuals who require a reliable solution for storing, sharing, and managing their passwords effectively. SafePass aims to automate and streamline best practices in password management, enabling users to achieve optimal security and ease of use.

Beyond its core functionality, SafePass is designed to enhance the overall security posture, safeguard user privacy, and boost productivity in daily operations. By combining convenience with cutting-edge security features, SafePass provides users with a holistic

approach to digital safety, ensuring their sensitive information remains protected while simplifying their day-to-day interactions with passwords and accounts.

1.6 Goals and Objectives

The goal of the proposed solution was to develop a web-based password manager tool that can ensure the user's privacy in terms of password security through data encryption and provide password security analysis through different scanning modules searching over the cloud and alerting users if any password is reported on the cloud.

The main objectives of the project were:

1. Study the working and functionality of existing available Password Manager Tools.
2. Develop a GUI-based password manager web-application.
3. Test the functionality of the developed tool with real crime scene scenarios.
4. Compare the performance and functionality of the developed tool with state-of-the-art available tools.

1.7 Work Breakdown Structure

The SAFEPASS development project plan covers the tasks and timetable. The following phases are part of the plan:

- **Gathering of Requirements**: In this initial stage, we collected information from various stakeholders, including end users and cybersecurity professionals. We also conducted market research to identify pain points lacking in existing password managers.
- **Design and Development**: According to our needs, we proceeded with the design and development of the SAFEPASS password manager. This phase included the creation of necessary resources such as secure storage, management, and safety analysis modules.
- **Testing**: During this phase, we conducted a series of tests to ensure that SAFEPASS met performance, safety, and functionality expectations. We followed the OWASP 10 Principles to ensure product safety and reliability.
- **Improvements**: After the initial development, we focused on product improvement, accelerating safety, adding new features, and resolving identified issues. We also solicited feedback from users and incorporated their suggestions into future SAFEPASS updates.
- **Documentation**: During this stage, we provided technical, user, and installation manuals for the SAFEPASS password manager.

1.8 Report Outline

This report is divided into several chapters, each offering a thorough analysis of the relevant project components. Each chapter addresses the material intended for it and reflects the goals and results. The report will cover the introduction, software requirements, use cases, representations, testing, results, etc. Each chapter offers a thorough analysis and documentation that will help the reader understand every single element of the project.

Chapter 2

Literature Review

In a nutshell, there are many password managers, and among them, Passbolt password manager has good security features compared to other open-source password managers. However, there are some features that are missing.

The following are the shortcomings found in the existing password managers. Although it's not that all password managers lack the following features, they do provide some of them, but there's always something missing:

I. A security kit may not be available with all password managers in the event that a user forgets their password.

II. Two-factor authentication is not available in all password managers.

III. Secure password sharing is not available in all password managers.

IV. Sign-in activity monitoring is not available with all password managers.

V. Certain password managers don't let you specify trusted devices.

VI. Some password managers don't offer monitoring for password breaches.

2.1 Gap Analysis

Based on a study of the features offered for different password managers, it appears that some of them are lacking, and it is always suggested that for creation of a brand-new tool it should offer extra capabilities to boost security. The study of features offered by various password managers reveals that while most provide core functionalities like secure storage, encryption, and synchronization, significant gaps remain in advanced security measures, user customization, and enterprise-level support. Inconsistent implementation of features like dark web monitoring, biometric authentication, and robust two-factor authentication leaves room for improvement. Furthermore, innovative functionalities, such as deception technology to detect cyber threats or dynamic password rotation for enhanced security, are largely absent. A new password manager should address these shortcomings by integrating real-time breach alerts, customizable password policies, advanced threat detection, and enterprise-grade collaboration tools, ensuring a comprehensive, scalable, and future-ready solution. Here's a comparison table of features among the studied password managers:

#	Features	Chrome	1Password	Last Pass	Nord Pass	RoboForm	Pass Bolt	KeePass	Bit Warden	SP
1	Generate Strong Passwords	✓	✓	✓	✓	✓	✓	✓	✓	✓
2	Security Kit (in case if customer forget password)		✓				✓			✓
3	2-Factor Authentication		✓	✓	✓		✓	✓	✓	✓
4	Secure Password Sharing		✓	✓	✓	✓	✓		✓	✓
5	Sign in Activity Monitoring		✓							✓
6	Set Trusted Devices			✓						✓
7	Password Breach Monitoring		✓	✓	✓	✓	✓		✓	✓
8	Encryption		✓	✓	✓	✓	✓	✓	✓	✓

Password Managers Comparison

1Password

1Password is a well-established password manager that excels at generating strong passwords, supporting 2-factor authentication, secure password sharing, and encryption. However, it falls short when it comes to offering sign-in activity monitoring and the ability to set trusted devices. These missing features can leave users with less control over the security of their accounts.

LastPass

Known for its user-friendly interface and comprehensive feature set, LastPass offers password generation, secure sharing, breach monitoring, and encryption. However, it lacks a security kit for users who forget their master password and does not include functionality to monitor sign-in activity or configure trusted devices.

NordPass

NordPass places significant emphasis on encryption and overall security. While it delivers on the basics of password management, it offers limited features for monitoring sign-in activity and setting trusted devices, which are crucial for modern security practices.

RoboForm

RoboForm is known for its ease of use, especially with features like form-filling, which saves time for users. However, it lags in offering advanced security tools such as password breach monitoring or detailed activity tracking. It is best suited for users prioritizing convenience over security.

PassBolt

PassBolt is an open-source password manager designed primarily for teams and collaboration. While it performs well in areas like secure password sharing and encryption, it lacks key features for individual users, such as trusted device configuration and sign-in monitoring.

KeePass

KeePass is a free, open-source password manager with a focus on customizability and offline usage. While it is highly flexible, it misses essential modern features like 2-factor authentication, breach monitoring, and user-friendly interfaces, making it less accessible

to non-technical users.

Bitwarden
Bitwarden offers a balance of strong encryption, transparency as an open-source tool, and feature richness, including password breach monitoring and 2-factor authentication. However, its interface can feel unintuitive for some users, and it lacks a security kit to help users recover their accounts if they forget their master password.

2.2 How SafePass Stands Out

SafePass builds on the strengths of these tools while addressing their weaknesses, providing a more holistic solution for password management. It is designed to meet modern security needs without compromising usability. Here's how SafePass improves upon existing options:

1. **Security Kit for Password Recovery**: Unlike most competitors, SafePass includes a comprehensive security kit to help users recover their accounts if they forget their master password. This ensures users can regain access securely without the stress of losing their data, a feature that many leading password managers overlook.

2. **Sign-in Activity Monitoring**: SafePass introduces detailed sign-in activity monitoring, allowing users to track when and where their accounts are accessed. This feature adds an extra layer of transparency and security by helping users detect suspicious login attempts early.

3. **Set Trusted Devices**: To further secure user accounts, SafePass allows users to configure trusted devices. Once a device is marked as trusted, login attempts from unfamiliar devices can be flagged or blocked, minimizing the risk of unauthorized access.

4. **Password Breach Monitoring**: SafePass incorporates real-time password breach detection powered by AI-driven algorithms. This feature alerts users immediately if their credentials appear in a data breach, empowering them to take swift action to secure their accounts.

5. **User-Friendly Interface with Advanced Encryption**: SafePass combines advanced encryption methods with an intuitive interface, ensuring strong security without overwhelming users. Its design makes it accessible to everyone, from beginners to tech-savvy individuals, bridging the gap that many other tools fail to address.

Chapter 3

Software Requirement Specification

3.1 Introduction

This Chapter focuses on the software requirement specification for the proposed product SAFEPASS.

3.1.1 Purpose

The purpose of this document is to develop a SAFEPASS Password Manager, a web-based tool that will assist users in managing and carrying out adequate security measures through various scanning and monitoring functions.

3.1.2 Document Conventions

The documentation convention followed in this report is as follows:

- Font is Times New Roman.
- Text size is followed according to the standard at different areas in the document.
- Alignment is set to Justify.
- Document is divided into different chapters and sections.
- The line spacing is set to 1.15.
- All requirement statements have their own priority assigned based on their level of importance.

3.1.3 Intended Audience and Reading Suggestions

SafePass is a password manager web application that helps users create strong, secure passwords and store them safely using encryption and multi-factor authentication. It also includes advanced password monitoring features to keep your accounts protected. SafePass is designed for anyone who wants to secure their passwords, whether it's individuals managing their personal accounts or IT professionals handling organizational password security. It's built to be a reliable tool for keeping sensitive information safe and easy to manage.

3.2 Overall Description

This section has all the description required related to the project.

3.2.1 Product Perspective

SAFEPASS is a web application password manager that provides secure storage and management of the user's login credentials. SAFEPASS will interact with the following systems:

- Web browsers from where users will access SAFEPASS.
- Multi-factor authentication system for user login and signup.
- Data Breach Scanner API: SAFEPASS using the "Have I Been Pwned" data breach dataset for scanning breached passwords.
- Users' data will be stored encrypted by strong encryption methods in the SAFEPASS database.

3.2.2 User Classes and Characteristics

Users of the SAFEPASS will be able to:

- Generate strong cryptographic passwords.
- Store login credentials in an encrypted form in the database.
- Check their password health and perform data breach scanning for breached passwords, if any.
- Set user trusted devices to their account.
- Share passwords securely with other SAFEPASS users.
- Monitor login activity for their account.

3.2.3 Operating Environment

The recommended operating environment for SAFEPASS is as follows:

- Windows 10 or higher OS.
- Client (any web browser).
- SQLite3 Database.
- Python3.
- Visual Studio Code.
- Django Framework.

3.2.4 Design and Implementation Constraints

SAFEPASS can currently only be accessed via localhost. It must be hosted on a cloud service to be made publicly accessible. Additionally, SAFEPASS currently uses the free versions of the APIs, which have restrictions like a maximum number of requests per minute. It is necessary to buy subscriptions for the APIs to improve scanning capabilities.

3.2.5 Assumptions and Dependencies

Assume our product has been hosted using a cloud service. A user will be able to access the SAFEPASS web application from their browser and perform the following actions:

- Signup to the system.
- Login to the system.
- Generate cryptographic passwords.
- Save passwords in the database.
- Check password health.
- Check for passwords in recent data breaches.
- Share passwords securely with other users.
- Set trusted devices.
- Check login activity.
- Logout.

3.3 System Features

The following are the features implemented in the project.

3.3.1 Storing Passwords in Database Encrypted

Description and Priority

Storing user's passwords in encrypted form using strong encryption methods is the basics of a password manager. It is a high-priority feature of the system.

Stimulus/Response Sequences

- User must be logged in.
- User clicks on the add password button.
- User fills in the add password form.
- All the data is checked for validation.
- Master password is used to generate an encryption key at runtime.

- The generated encryption key is used to encrypt the password.
- The data is then stored in the database.
- The data is displayed on the dashboard.

Functional Requirements

- REQ-SF1-1: Allow logged-in user to add password.
- REQ-SF1-2: Validate all form data.
- REQ-SF1-3: Generate encryption key.
- REQ-SF1-4: Generate hash value of password.
- REQ-SF1-5: Check password health and assign a score to the password.
- REQ-SF1-6: Encrypt password before saving it to the database.
- REQ-SF1-7: Save password to the database.

3.3.2 Password Generator

Description and Priority

Generating strong cryptographic passwords for users is one of the basics of a password manager. It is a medium-priority feature of the system.

Stimulus/Response Sequences

- User must be logged in.
- User clicks on the password generator tab.
- User selects the size and checkboxes on the password generator GUI.
- User clicks on the generate button.
- System will generate a random string of the size user requested according to NIST password standards.
- Display the generated password to the user.

Functional Requirements

- REQ-SF2-1: Allow logged-in user to generate a password.

3.3.3 Password Health Check

Description and Priority

Check the user's saved passwords in the database according to the NIST password standards to see if the password is strong or not. This is a high-priority feature of the system.

Stimulus/Response Sequences

- User must be logged in.
- User clicks on the password health tab.
- User clicks on the check passwords button.
- System will check the scores of each password that was generated during the user's password storage.
- System will also compare the hashes of stored passwords to check for any repetitions.
- Display the list of password results along with a detailed recommendation for weak passwords.

Functional Requirements

- REQ-SF3-1: Allow logged-in user to check password health.
- REQ-SF3-2: System should get data from the database to the server.
- REQ-SF3-3: System should check the score and show messages accordingly.
- REQ-SF3-4: System should compare hashes and display repeated passwords.

3.3.4 Data Breach Scanner

Description and Priority

Check if the user's saved passwords are reported in a data breach and notify the user accordingly. This is a high-priority feature of the system.

Stimulus/Response Sequences

- User must be logged in.
- User clicks on the data breach scanner tab.
- User clicks on the scan passwords button.
- System will get the hashes of the saved passwords from the database.
- System will send the first 5 prefixes of the hashes to the Have I Been Pwned (HIBP) API.
- The API will return data related to those 5 prefixes.
- System will compare the hashes of stored passwords and the data received through the API.
- Display the list of password results along with a detailed explanation of breached passwords.

Functional Requirements

- REQ-SF4-1: Allow logged-in user to scan for data breaches.
- REQ-SF4-2: System should get data from the database to the server.
- REQ-SF4-3: System should request data from the HIBP API and display it.
- REQ-SF4-4: System should compare the hashes and display the emails and hostname of breached passwords.

3.3.5 Secure Password Sharing

Description and Priority

Users will securely share their passwords with other users in encrypted form. This is a high-priority feature of the system.

Stimulus/Response Sequences

- User must be logged in.
- User clicks on the actions on the displayed password and selects "Share Password".
- User will be redirected to the share password form.
- User enters their master password and the receiver's email.
- System checks the validations.
- System checks if the receiver is registered on SafePass.
- System generates the encryption key from the user's master password and decrypts the password.
- System fetches the receiver's public key from the database and encrypts the password.
- System saves the encrypted password and related data in the receiver's table.
- System sends an email saying that the user has shared a password with the receiver.

Functional Requirements

- REQ-SF5-1: Allow logged-in user to add and share a password.
- REQ-SF5-2: Validate all form data.
- REQ-SF5-3: Generate encryption key.
- REQ-SF5-4: Generate hash value of the password.
- REQ-SF5-5: Check password health and assign a score.
- REQ-SF5-6: Encrypt password before saving it to the database.
- REQ-SF5-7: Save password to the database.

3.4 Nonfunctional Requirements

3.4.1 Performance Requirements

Implementation of SAFEPASS involves the following steps:

- Install all the required libraries.
- Run the server using Python3.
- Signup to SAFEPASS using the web GUI.
- Login to the user dashboard using the web GUI.

3.4.2 Security Requirements

Following security measures have been added:

- Admin role.
- 2-Factor authentication system.
- Email validation check.
- Strong password encryption using AES 256 GCM.
- Adding trusted devices.
- Sign-in activity monitoring.

3.4.3 Usability Requirements

SAFEPASS is Python3-based and uses the following libraries (the cryptographic libraries used are FIPS 140-2 certified):

- Django
- render, redirect
- contrib.auth
- messages
- core.email
- cryptography
- hashlib
- ensure_csrf_cookie
- pyotp
- cache
- Pyperclip
- SQLite3
- datetime

3.4.4 Efficiency Requirements

During the development we used efficient methods and libraries for maximum efficiency of the system. We tried over best to optimize the code and make it more secure through adding different validation checks.

3.5 Use Case Model:

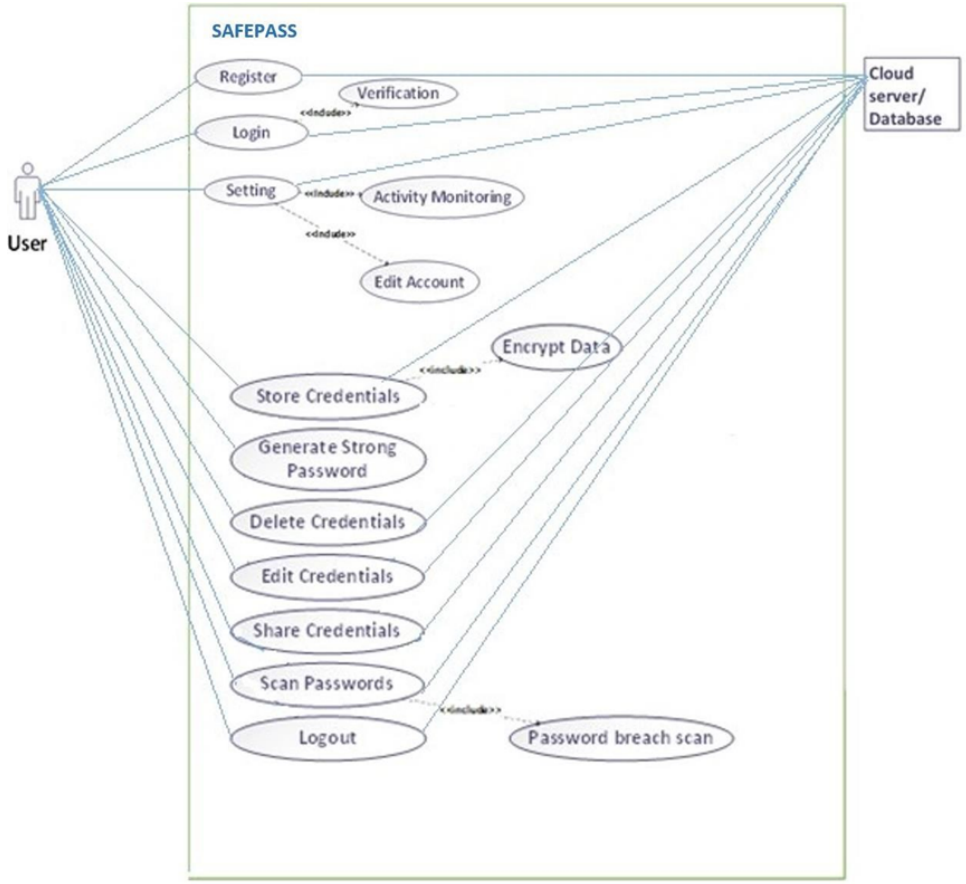

Figure 3.1: Use Case Model

Use Case Name	Login
Primary Actor	User
Goal in Context	Take user credentials to login the user to the web application.

Preconditions	• The system must be connected to the internet and server. • The system must be fully configured; appropriate user ID and passwords must be obtained. • User already registered.
Postconditions	• After successful login, a notification will pop up on the user's trusted device. • User must allow login on the device. • A confirmation mail will be sent to the user's email account, and the user must confirm it.
Priority	High, it should be implemented at the start.
Frequency of Use	Mandatory for new users.
Normal Course of Events	The user will register themselves before using other functionalities.
Alternative Courses	None
Scenario	1. The user will open the web application and click on the register option. 2. The application will display the registration form. 3. The application will allow the user to enter data in the form. 4. The user will enter the data and press "Register Me/Submit Form". 5. The user's password will be hashed by the application before being transferred to the cloud database.
Exceptions	• Form is not submitted due to an internet connectivity issue. • Wrong credentials. • Wrong six-digit pin.

Assumption	Mandatory for new users to register before using other functionalities of the application.

Setting

Use Case Name	Setting
Primary Actor	User
Goal in Context	Edit profile Information
Preconditions	• User must be logged in.
Postconditions	• After editing profile, data should be saved, and a notification will pop up with the message "profile updated".
Priority	Medium, it should be implemented after login start.
Frequency of Use	Mandatory
Normal Course of Events	• Update the user information on user requirement. It includes use cases 03, 04, 05, 06, 07, 08.
Alternative Courses	None
Scenario	1. After Login, if user wants to change his account information. 2. He will click on Edit and then select the required use case/option and change the information and then save the information.
Exceptions	• Form is not submitted due to internet connectivity issue. • Select other use case (04, 05, 06, 07, 08).
Assumption	Not Mandatory

Activty Monitoring

Use Case Name	Activity Monitoring
Primary Actor	User
Goal in Context	Monitoring account.
Preconditions	• User must be logged in.
Postconditions	• User should get monitoring information about his account.
Priority	Medium, it should be implemented after login start.
Frequency of Use	Mandatory
Normal Course of Events	• User will get information like last login date time and location.
Alternative Courses	None
Scenario	1. After Login, if user wants to check his account activity. 2. He will click on settings and from there to check account activity.
Exceptions	• Request failed due to internet connectivity issue.
Assumption	Not Mandatory

Edit Account

Use Case Name	Edit Account
Primary Actor	User
Goal in Context	User can edit his accounts details and can set trusted devices and can delete account.
Preconditions	• User must be logged in.
Postconditions	• User should be able to edit his accounts details and can set trusted devices and can delete account.
Priority	Medium, it should be implemented after login start.
Frequency of Use	Mandatory
Normal Course of Events	• User will be able to edit his account.
Alternative Courses	None
Scenario	1. If user wants to edit his account. 2. He will click on settings and from there to edit account.
Exceptions	• Request failed due to internet connectivity issue.
Assumption	Not Mandatory

Store Credentials

Use Case Name	Store Credentials
Primary Actor	User
Goal in Context	Add information
Preconditions	• User must be logged in.
Postconditions	• After adding information, data will be encrypted and saved, and a notification will pop up with the message "Saved".
Priority	High, It should be implemented after login.
Frequency of Use	Mandatory
Normal Course of Events	• Add user information on user requirement.
Alternative Courses	None
Scenario	1. After Login, if user wants to add credentials. 2. He will click on add and then fill the form and save the information.
Exceptions	• Form is not submitted due to internet connectivity issue. • Warning on weak password. • Warning on any empty field of form. • Information is not in valid form. • Email isn't contain "@". • Warning if words limitation exceed in entries.
Assumption	N/A

Encrypt Data

Use Case Name	Encrypt Data
Primary Actor	User
Goal in Context	Encrypt data before storing it in database.
Preconditions	• Take store/edit form information as input.
Postconditions	• Apply algorithm and encrypt the data and generate the output for storing it into database.
Priority	High, It should be implemented after Store Credential use case.
Frequency of Use	Mandatory
Normal Course of Events	Encrypt user data.
Alternative Courses	None
Scenario	1. After Login, whenever user adds or edits credentials, its data will be encrypted with different encryption algorithms before storing it into the database.
Exceptions	N/A
Assumption	Not Mandatory

Generate Strong Password

Use Case Name	Generate Strong Password
Primary Actor	User
Goal in Context	Generate strong password for user
Preconditions	• User should be logged in.
Postconditions	• A strong password will be generated for the user.
Priority	High
Frequency of Use	Medium
Normal Course of Events	• Generate password on click, following rules.
Alternative Courses	None
Scenario	1. User wants to generate a strong password. 2. User clicks on "Generate Password" button. 3. User gets a strong password suggestion.
Exceptions	• Issue generating the password due to internet connectivity issue.
Assumption	Not Mandatory

Use Case: Delete Credentials

Primary Actor	User
Goal in Context	Delete the user's stored credentials
Preconditions	• User must be logged in. • User must have stored credentials. • A confirmation popup should be generated before deletion.

Postconditions	• After deletion, the database is updated. • A popup is generated with the message "Successfully deleted".
Priority	High
Frequency of Use	Mandatory
Normal Course of Events	Delete user credentials.
Alternative Courses	None
Scenario	• The user selects credentials to delete, clicks delete, and confirms via the popup. • The credentials are deleted, and the database is updated.
Exceptions	• Credentials are not deleted due to an internet connectivity issue.
Assumption	Not Mandatory

Use Case: Edit Credentials

Primary Actor	User
Goal in Context	Edit user's stored credentials.
Preconditions	• User must be logged in. • Credentials must already be stored.
Postconditions	• Edited credentials are encrypted and saved. • A notification pops up with the message "Information Updated".
Priority	High
Frequency of Use	Up to user requirement
Normal Course of Events	Update the user's stored credentials.

Alternative Courses	Delete Credentials
Scenario	• After logging in, the user selects stored credentials and clicks edit. • A form opens for making changes. • After saving, a popup confirms "Information is updated successfully".
Exceptions	• Form not submitted due to internet connectivity issue. • Warning for weak password or empty fields. • Invalid information format. • Email does not contain "@". • Word limit exceeded in entries.
Assumption	Not Mandatory

Use Case: Share Credentials

Primary Actor	User
Goal in Context	Share credentials with another user securely.
Preconditions	• A link to the shared folder must be shared. • The second user must click the link and create an account.
Postconditions	The second user gains access to the shared credentials.
Priority	Medium
Frequency of Use	Mediocre
Normal Course of Events	Share credentials with another user.
Alternative Courses	Available but not too secure.

Scenario	• After logging in, the user adds credentials to the shared folder. • The user creates and sends a link to the second user. • The second user clicks the link, creates an account, and accesses the credentials.
Exceptions	• Link expired. • Internet connectivity issue. • Unauthorized party tries to access the shared folder, resulting in blocking and session timeout.
Assumption	Not Mandatory

Use Case: Scan Passwords

Primary Actor	User
Goal in Context	User saved passwords should be checked for a data breach.
Preconditions	• User must be logged in.
Postconditions	• After scanning passwords, a notification will pop up with a customized message.
Priority	Medium, it should be implemented after login start.
Frequency of Use	Mandatory
Normal Course of Events	• Check user passwords. • Includes use cases 03, 04, 05, 06, 07, and 08.
Alternative Courses	None
Scenario	• After login, the user clicks on any of the sub-use cases to scan passwords.

Exceptions	• Scanning failed due to internet connectivity issues.
Assumption	Not Mandatory

Use Case: Data Breach Scanning

Primary Actor	User
Goal in Context	Scan user credentials on websites to check if they are reported in a data breach.
Preconditions	• User must be logged in.
Postconditions	• After scanning, a notification will pop up with a customized message.
Priority	Medium, it should be implemented after login start.
Frequency of Use	Mandatory
Normal Course of Events	Check credentials if reported in a data breach.
Alternative Courses	None
Scenario	• After login, the user clicks the data breach scanning button to perform the scan.
Exceptions	• Scanning failed due to internet connectivity issues.
Assumption	Not Mandatory

Use Case: Logout

Primary Actor	User
Goal in Context	User logs out from their account.

Preconditions	• The system must be connected to the internet and server. • The system must be fully configured with appropriate user ID and passwords. • User must be logged in.
Postconditions	• User logs out from their account.
Priority	Medium
Frequency of Use	Depends on user
Normal Course of Events	• User logs out from their account by clicking the logout button.
Alternative Courses	None
Scenario	• User clicks the logout button and is logged out of their account.
Exceptions	• Internet connectivity issues.
Assumption	User must be logged in.

3.5.1 Admin Side

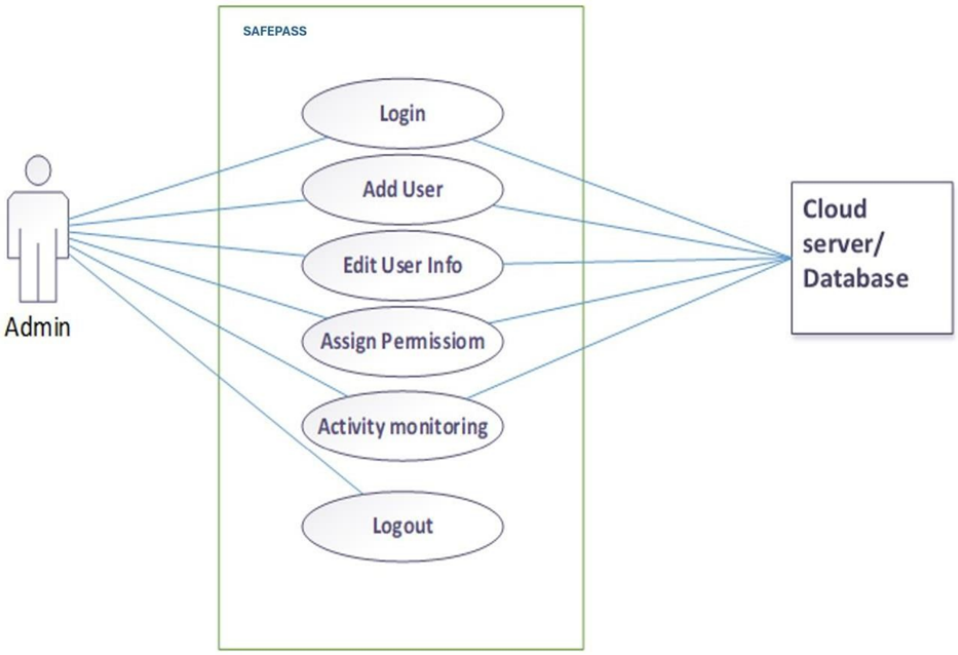

Figure 3.2: Admin Side UseCase

Add User (Login)

Use Case Name	Add User (Login)
Primary Actor	ADMIN
Goal in Context	Take user credentials to log in to the admin panel.
Preconditions	• The system must be connected to the internet and server. • Must be logged in to the admin panel.
Postconditions	• After successful login, a notification will pop up on the user's trusted device. • Users must allow login on the device.
Priority	High, it should be implemented at the start.
Frequency of Use	Mandatory for new user.
Normal Course of Events	Have the permissions to act as admin.
Alternative Courses	None
Scenario	1. The user will open the web admin login panel. 2. The application will display the form. 3. The application will allow the admin to enter data in the form. 4. The user will enter the data and press "Login."
Exceptions	• Form is not submitted due to internet connectivity issues. • Login credentials are incorrect.
Assumption	Have permissions to access the admin panel.

Add User (Creation)

Use Case Name	Add User (Creation)
Primary Actor	ADMIN
Goal in Context	Create a new user.
Preconditions	• The system must be connected to the internet and server. • The system must be fully configured; appropriate user ID and passwords must be obtained. • Logged in as admin.
Postconditions	Admin creates a new user from the admin panel.
Priority	Low
Frequency of Use	Depends on admin.
Normal Course of Events	Have the permissions to act as admin.
Alternative Courses	None
Scenario	1. Admin will click the "Add New User" button. 2. The application will display the form. 3. The application will allow the admin to enter data in the form. 4. The admin will enter the data and press "Create."
Exceptions	• Form is not submitted due to internet connectivity issues. • Entered information is not valid.
Assumption	Logged in as admin.

Edit User Info

Use Case Name	Edit User Info
Primary Actor	Admin
Goal in Context	Create a new user.
Preconditions	• The system must be connected to the internet and server. • The system must be fully configured; appropriate user ID and passwords must be obtained. • Logged in as admin.
Postconditions	Admin edits a user's information from the admin panel.
Priority	Medium
Frequency of Use	Depends on admin.
Normal Course of Events	Have the permissions to act as admin.
Alternative Courses	None
Scenario	1. Admin will select user and click the edit user button. 2. The application will display the user information form. 3. The application will allow the admin to enter data in the form. 4. The admin will enter the data and press "update".
Exceptions	• Form is not submitted due to internet connectivity issue. • Entered information is not valid.
Assumption	Logged in as admin panel.

Assign Permissions

Use Case Name	Assign Permissions
Primary Actor	Admin
Goal in Context	Manage user permissions.
Preconditions	• The system must be connected to the internet and server. • The system must be fully configured; appropriate user ID and passwords must be obtained. • Logged in as admin.
Postconditions	Admin assigns permissions to the user through the admin panel.
Priority	Medium
Frequency of Use	Depends on admin.
Normal Course of Events	Have the permissions to act as admin.
Alternative Courses	None
Scenario	1. Admin will select user and click the manage permissions button. 2. Admin can add or remove permissions to the selected user. 3. The admin will enter the data and press “submit”.
Exceptions	• Form is not submitted due to internet connectivity issue. • Entered information is not valid.
Assumption	Logged in as admin panel.

Activity Monitoring

Use Case Name	Activity Monitoring
Primary Actor	Admin
Goal in Context	Monitoring user activity.
Preconditions	• The system must be connected to the internet and server. • The system must be fully configured; appropriate user ID and passwords must be obtained. • Logged in as admin.
Postconditions	Admin monitors user through the admin panel.
Priority	Medium
Frequency of Use	Depends on admin.
Normal Course of Events	Have the permissions to act as admin.
Alternative Courses	None
Scenario	1. Admin will select user and click the manage permissions button. 2. Admin can monitor the last logged-in time of the selected user.
Exceptions	• Internet connectivity issue.
Assumption	Logged in as admin panel.

Logout

Use Case Name	Logout
Primary Actor	Admin
Goal in Context	Admin logs out.
Preconditions	• The system must be connected to the internet and server. • The system must be fully configured; appropriate user ID and passwords must be obtained. • Logged in as admin.
Postconditions	Admin logs out from the admin panel.
Priority	Medium
Frequency of Use	Depends on admin.
Normal Course of Events	Logged in as admin.
Alternative Courses	None
Scenario	1. Admin will click on the logout button. 2. Admin will be logged out.
Exceptions	• Internet connectivity issue.
Assumption	Logged in as admin panel.

Chapter 4

System Design

4.1 High-Level Design

Design of the proposed framework is shown. Users will interact with the SAFEPASS server through SAFEPASSweb application after two factor authentication. The server will contain different modules likepassword generation, storing passwords and other different security modules and will respond according to the user's request. In the monitoring module the server will scan through dark web and webto see if the user data is available there and will generate an alert if any user related data is found.

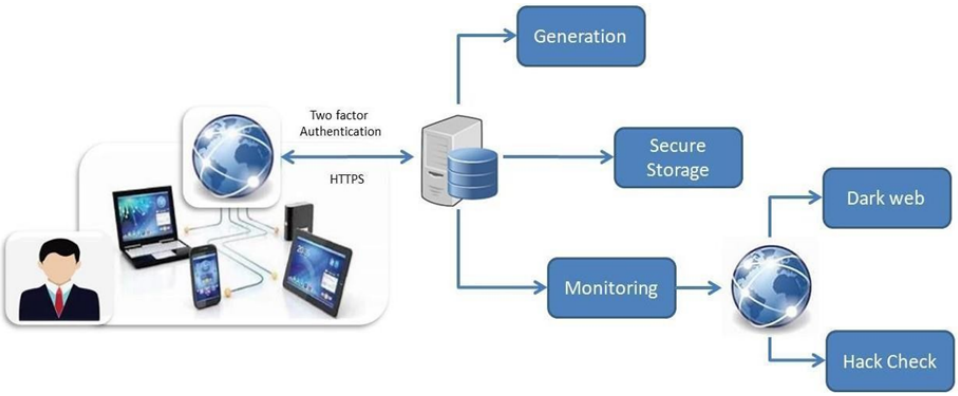

Figure 4.1: High Level Architecture

4.2 System Architecture Diagram

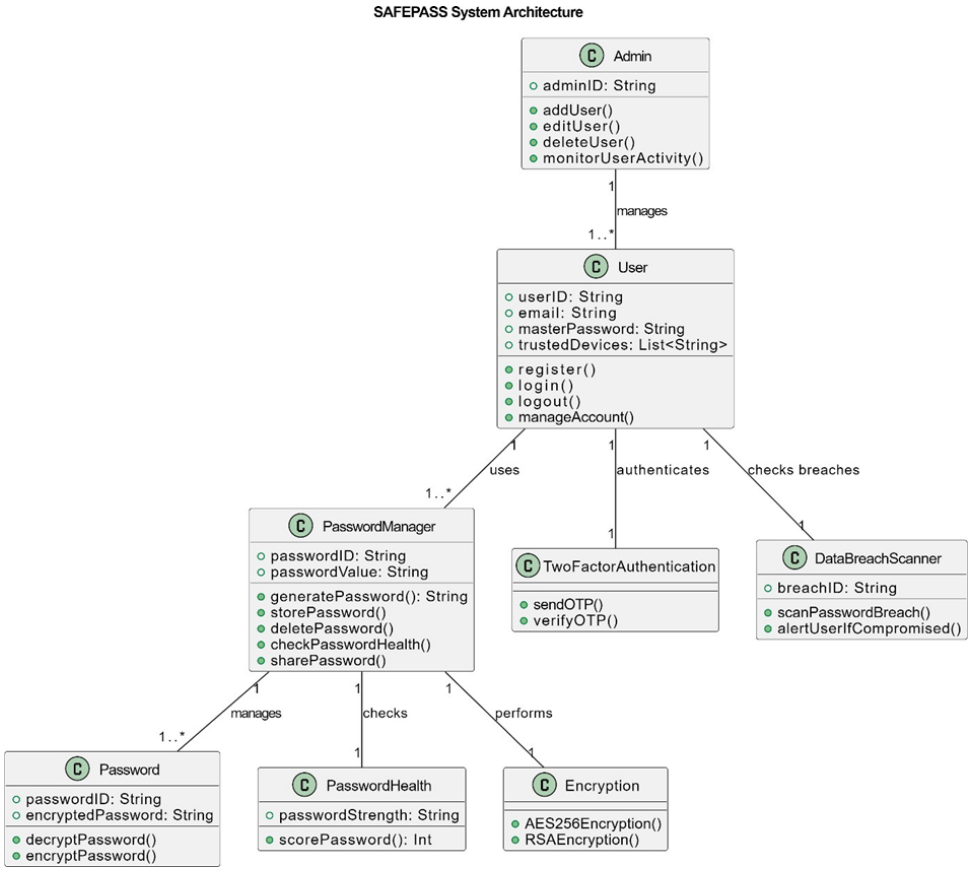

Figure 4.2: System Architecture

The SafePass system architecture diagram outlines the structure and functionality of a secure password management application. At the core is the User component, which allows individuals to register, log in, and manage their accounts. Users interact with the PasswordManager to securely store, generate, delete, and share passwords, while the PasswordHealth component evaluates the strength of stored passwords. For added security, the system employs TwoFactorAuthentication, which sends and verifies one-time passwords (OTPs) during the login process. To safeguard user credentials, Encryption mechanisms like AES256 and RSA are used for secure password storage. Additionally, the DataBreachScanner monitors for potential breaches, alerts users if their data is compromised, and ensures proactive security. The Admin oversees the system by managing users and monitoring activity, ensuring the application remains efficient and secure. This architecture emphasizes robust security and user convenience.

4.3 Layer Architecture Diagram

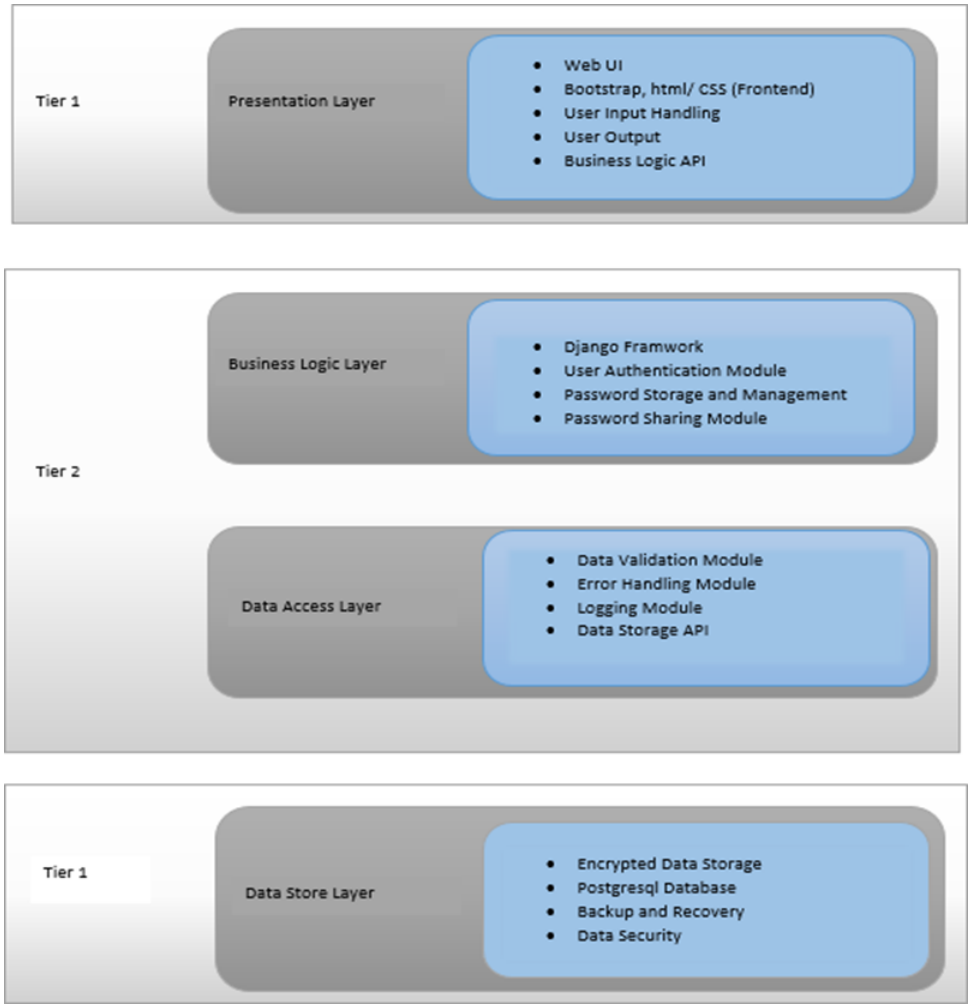

Figure 4.3: Layer Architecture

This diagram illustrates the four-tier architecture of the SafePass application. The Presentation Layer (Tier 1) focuses on user interaction through a web-based UI using technologies like HTML, CSS, and Bootstrap for input handling and output display. The Business Logic Layer (Tier 2) is built on the Django framework, managing user authentication, password storage, and sharing functionalities. The Data Access Layer handles data validation, error logging, and secure communication with the database through APIs. Finally, the Data Store Layer ensures encrypted storage of passwords, leverages a PostgreSQL database, and includes backup and recovery mechanisms to maintain data integrity and security.

4.4 Component Diagram

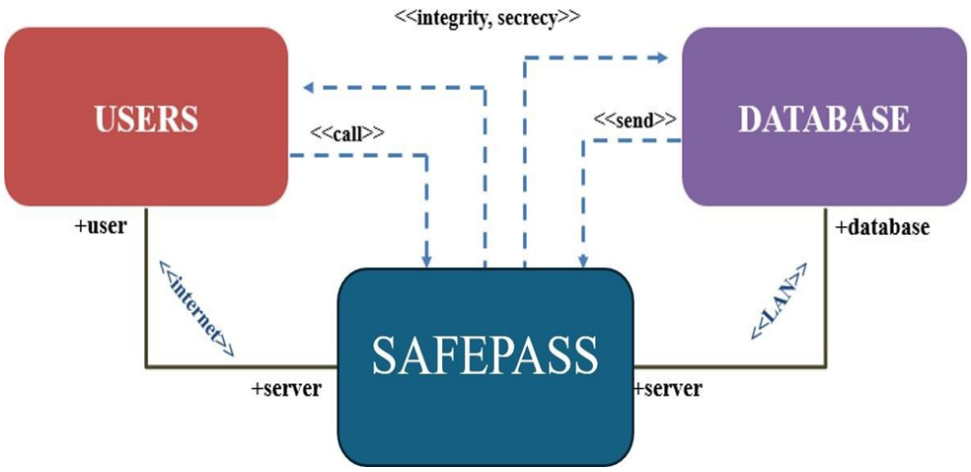

Figure 4.4: Component Diagram

4.5 Sequence/Collaboration Diagram

This SAFEPASS product Sequence diagram shows the actions a user can perform while using the SAFEPASS app. The customer opens the app and send signup request at the beginning of the diagram, Then he receive confirmation email, After email confirmation he will redirect to login page and give credentials, and he have to be verified by 2f authentication, then he will redirected to dashboard. Similarly, by following different steps mention in diagram user can use our product functionality

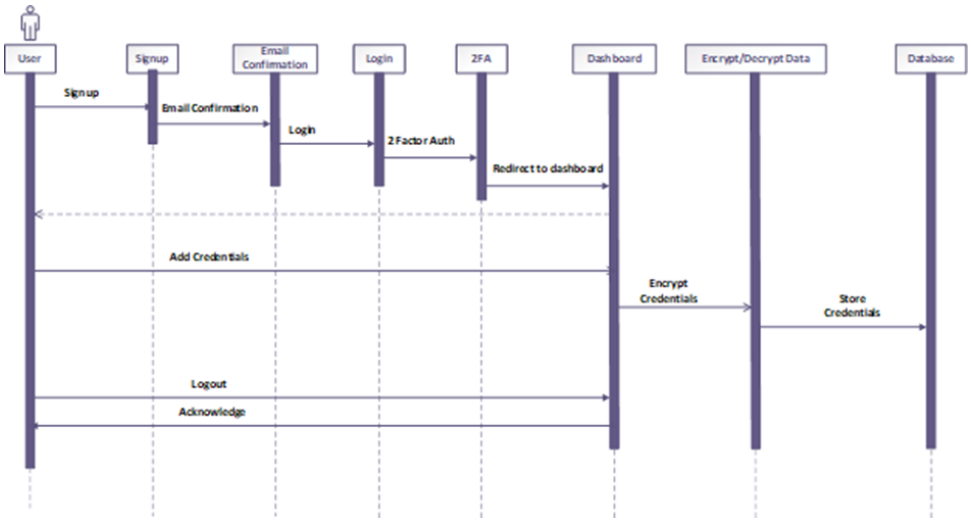

Figure 4.5: Sequence Diagram

4.6 Activity Diagram

This SAFEPASS product activity diagram lists the numerous actions a user can perform while using the SAFEPASS app. The customer opens the app at the beginning of the

diagram, which then shows the main menu with options for password creation, credential storage, password monitoring, data breach scanning, secure vault sharing, two-factor authentication, and password health checks. Each step in the diagram is shown as an action that either requests input from the user or does an automated task within the app. The password health check option, which gives the customer a report on the strength and complexity of their saved passwords, is shown at the conclusion of the diagram. In general, the diagram offers a distinct visual depiction of the various traits and skills of the of SAFEPASS app.

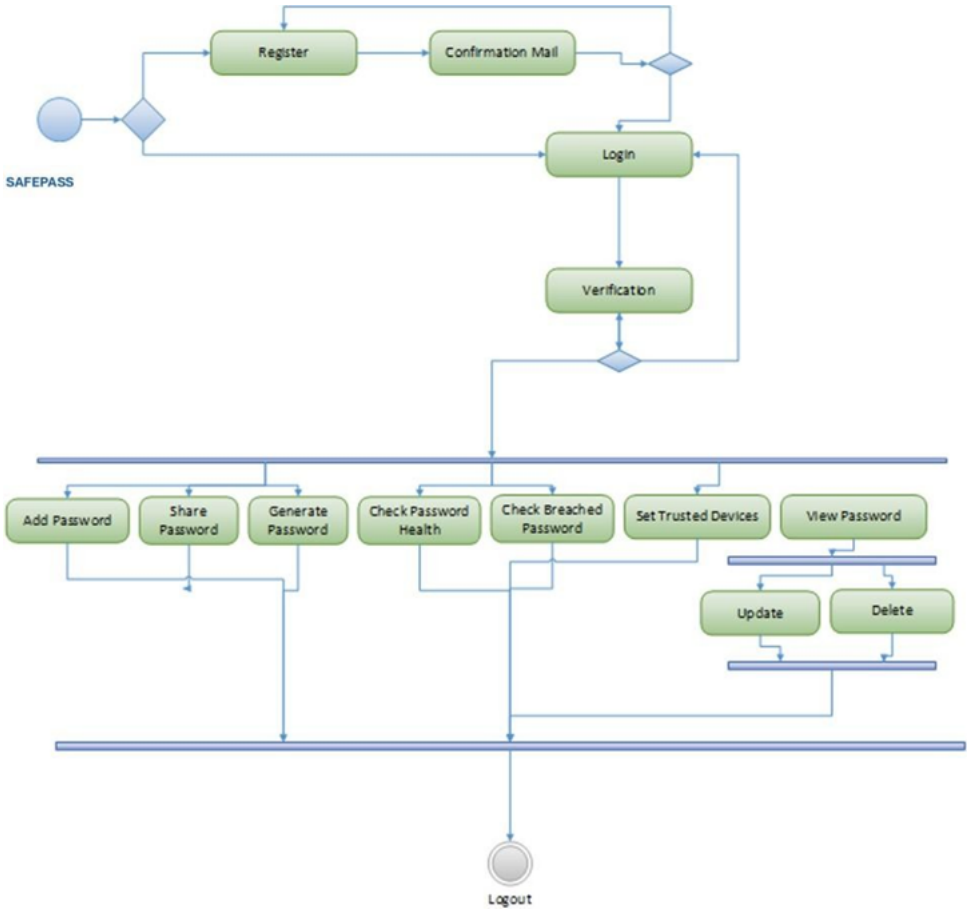

Figure 4.6: Activity Diagram

4.7 Database Design

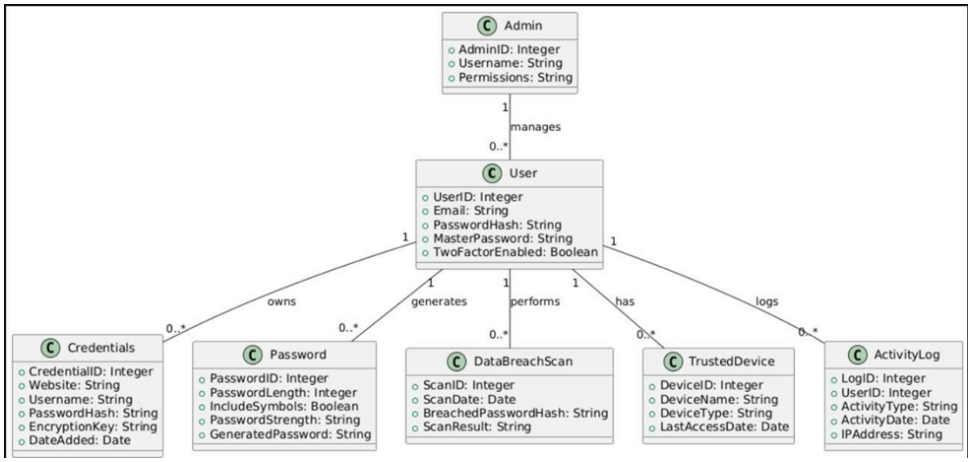

Figure 4.7: Database Design

4.8 External Interface

This section defines the interface requirements of the software.

4.8.1 User Interfaces

We have implemented the following user interfaces in our project:

- Frontend: HTML, CSS, JavaScript.
- Backend: Django Framework, SQLite3.

This is the GUI of our project:

- Our Home page has basic information with buttons that will redirect the user to the different sections.

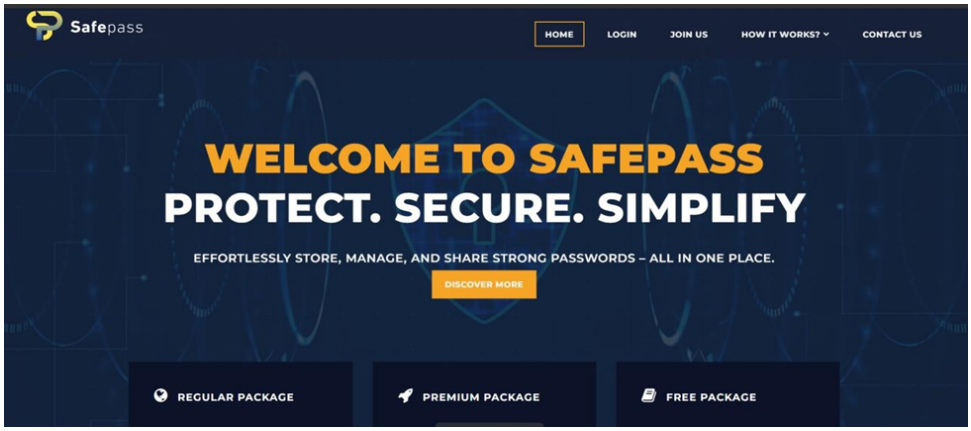

Figure 4.8: User Interface 1

Figure 4.9: User Interface 2

Now very first user will click join us he will be redirected to Signup Page.

Here In the signup page will check username first if it is already taken or not and then check if the email is valid and then Check password according to NIST standards and it be stored in hash in database. After everything is clear server will send a confirmation link to check if the email user entered is valid.

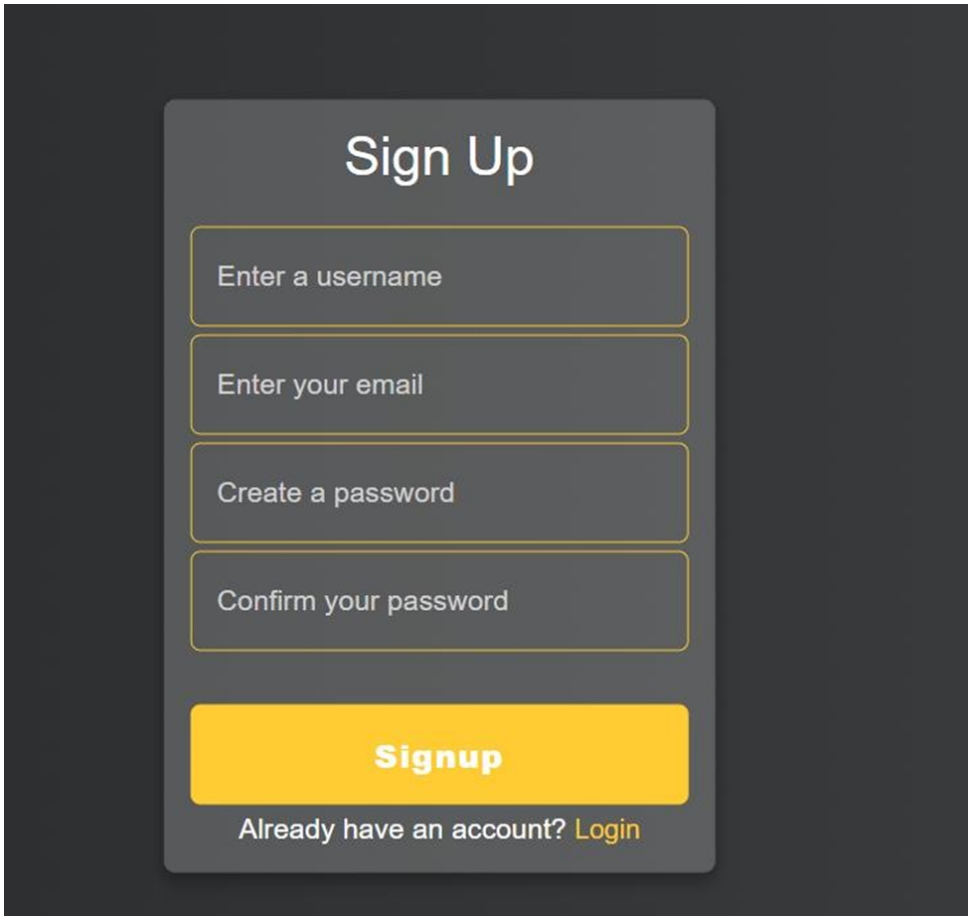

Figure 4.10: User Interface 3

If user clicks the link his account will be activated, and he will be redirecting to sign in page here user will enter his login credentials as server will authenticate user and if

his credentials matched, server will send 6 digit opt and user will only have one try if he enters the wrong code he have to try sign in again.

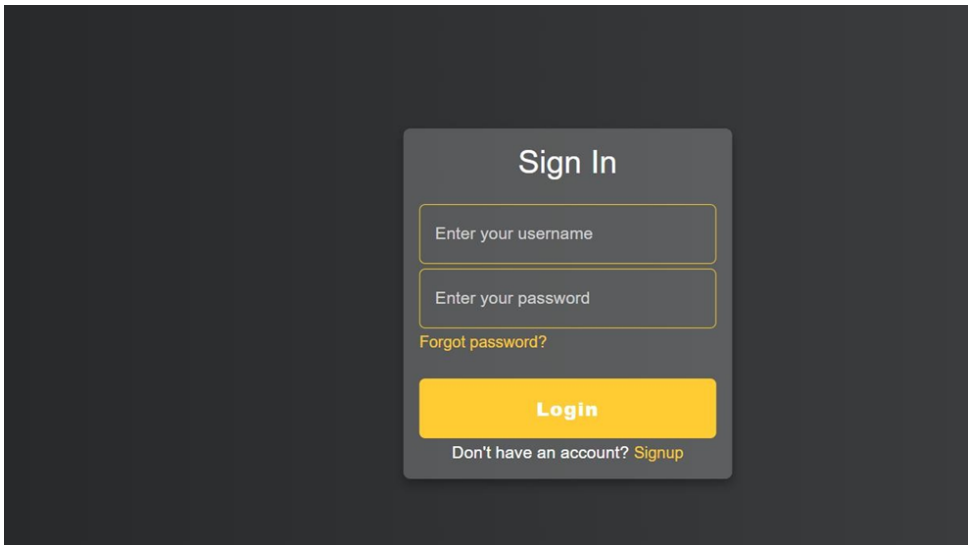

Figure 4.11: User Interface 4

After login is successful, user will be redirected to his dashboard.

This is our dashboard here user saved passwords will be saved and for till now he can add or delete passwords but with time we will add more features. If a user wants to add a new password.

User will click on the add password button, and he will get this:

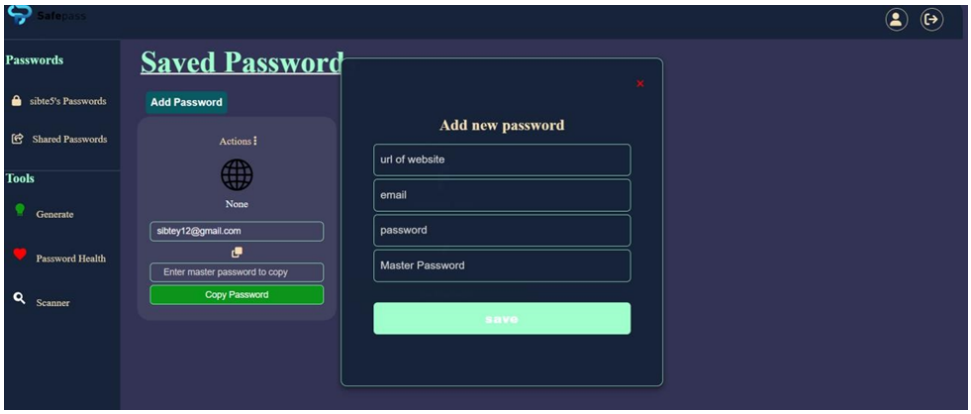

Figure 4.12: User Interface 5

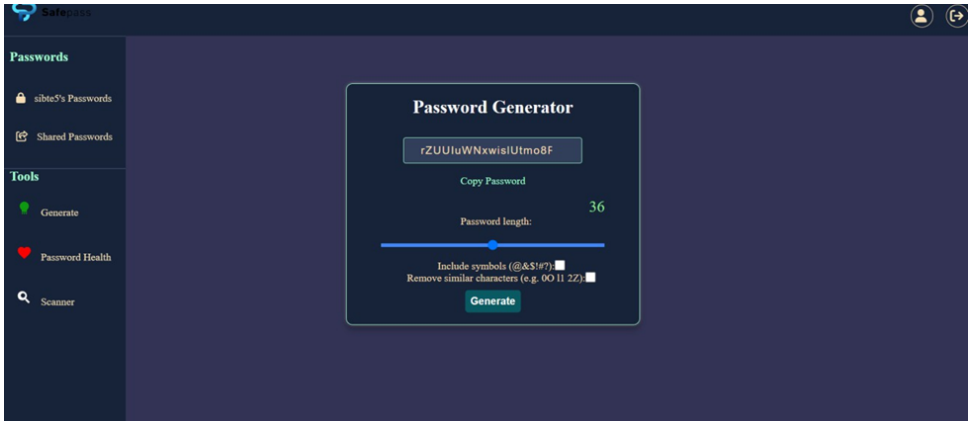

Figure 4.13: User Interface 6

Here server will generate random passwords for the user. It has followed the standards like added numbers, upper case and lowercase letters and user can add special characters and can exclude similar characters from password. If user clicks on password health check:

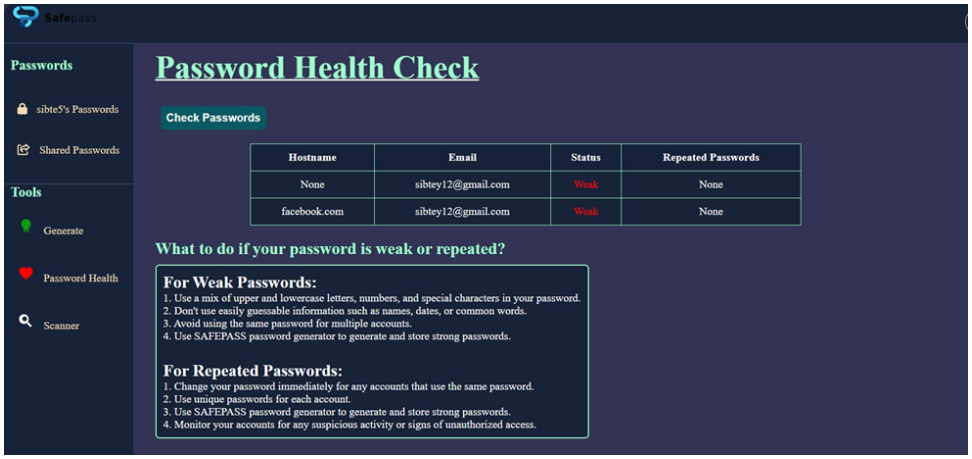

Figure 4.14: User Interface 7

Here user will click on the check password button, and we will check if any of his password is weak or being reused like if it matches to users other passwords. If user clicks on Data breach scanner:

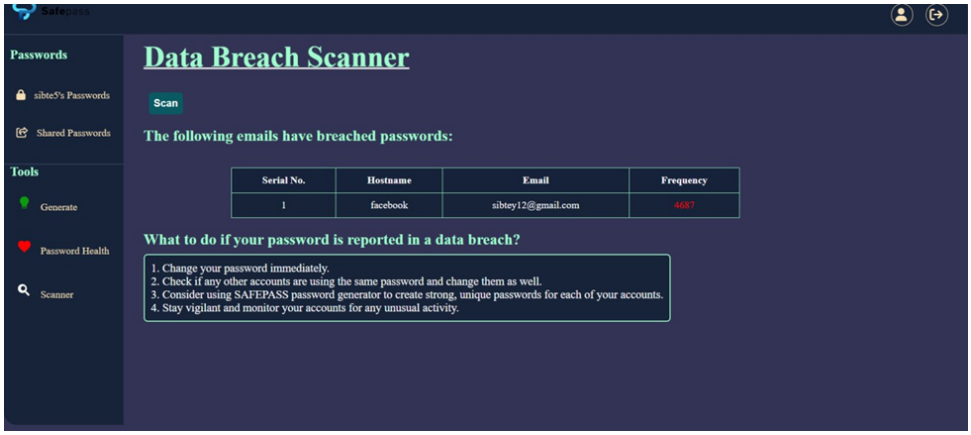

Figure 4.15: User Interface 8

Here user will click on the scan button, and we will check if any of his password is being reported in a data breach. Now if user clicks on the user icon on the top he will go to user's profile:

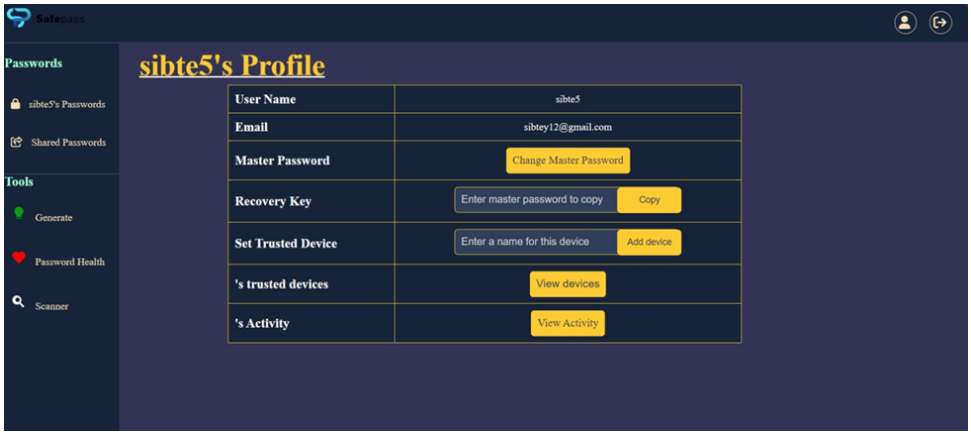

Figure 4.16: User Interface 9

Here he can see his personal information and can change his master password, or can copy the recovery key which he can use if he forgets the password later. And also the user set trusted devices, like if user wants to set the currently login device as trusted device he will give the device name and click add device like this:

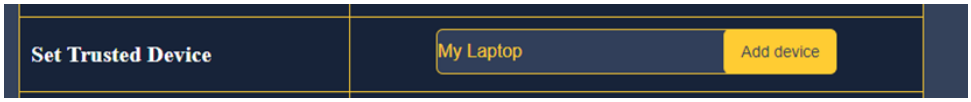

Figure 4.17: User Interface 10

The device will be added to his trusted devices with a unique id. Now if he wants to check the trusted devices he will click the view devices and he will get: Now if the user goes back to the user' password and click on share password he will get:

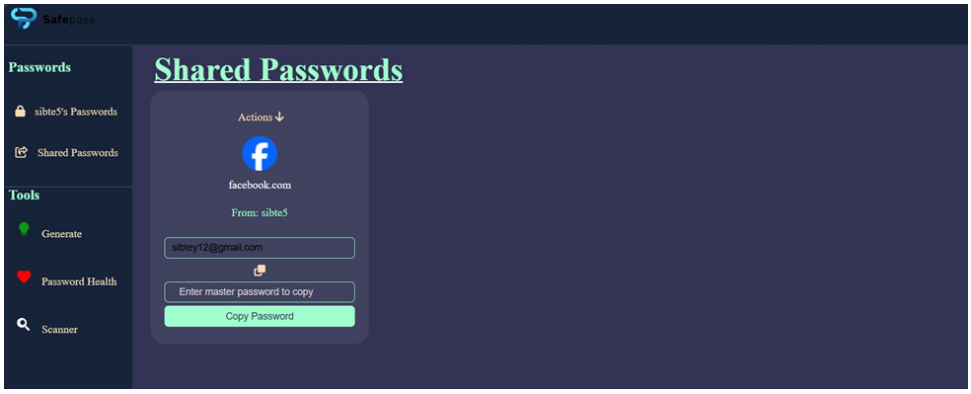

Figure 4.18: User Interface 11

Here the user will enter email of the user and his master password and if the email is registered to SAFEPASS, he will get a mail saying he got a shared password from user. And after the other user logs into his account and click of shared passwords he will get: Here he will see all the passwords being shared with him and from there he can copy and use that password by giving his master password. Note that all this data will be encrypted in database and no one will see the passwords except the user.

4.8.2 Admin Interfaces

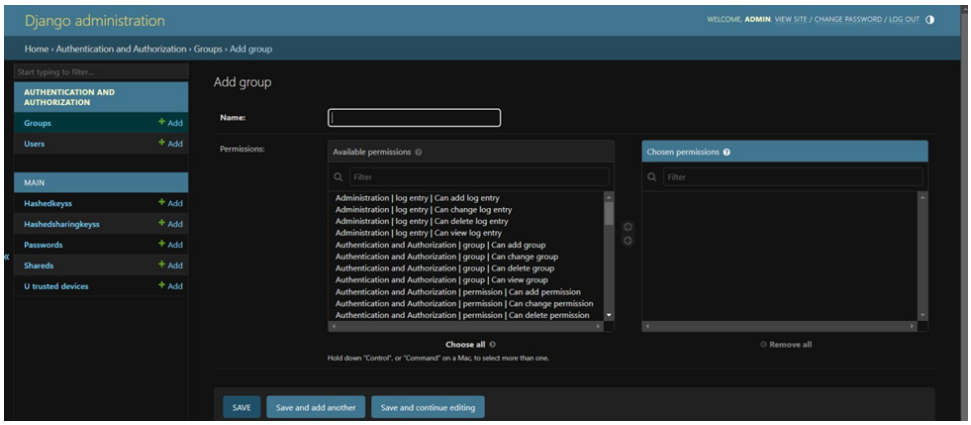

Figure 4.19: admin interface

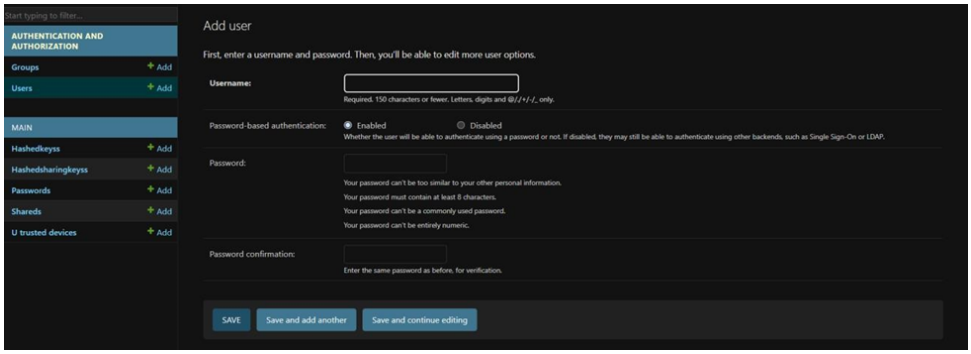

Figure 4.20: admin interface

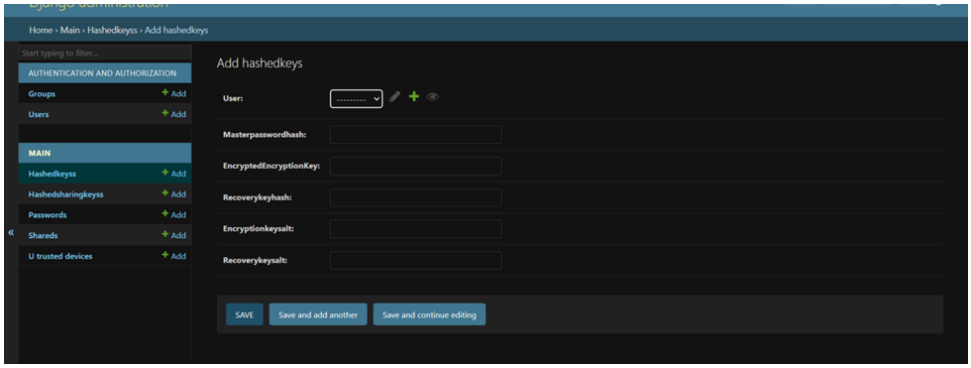

Figure 4.21: admin interface

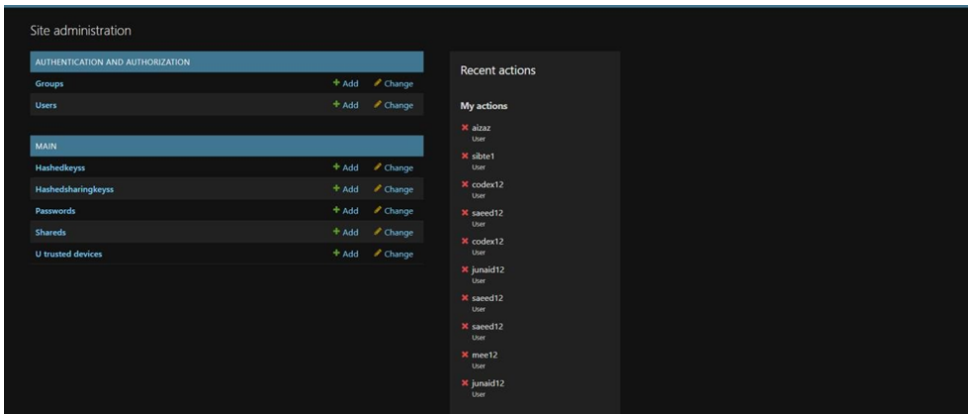

Figure 4.22: admin interface

Chapter 5

System Implementation

We considered upcoming innovations and current market trends in the context of password managers when we implemented this project. In order to make module integration in further advancements simple, a modular structure was chosen.

5.1 Components, Libraries, Web Services, and Stubs

Following are the components of this application:

- 2-Factor Authentication by OTP sent to user's email.
- Password Generator.
- Password Sharing.
- Strong Encryption.
- Recovery kit.
- Password breach scanning.
- Encrypted user data.

Following libraries are used:

- Django.
- pycryptodome.
- hashlib.
- pyotp.
- ensure_csrf_cookie.
- csrf_exempt.
- contrib.auth.
- SQLite3.
- Core.email.
- Utils.encoding.

5.2 Deployment Environment

We now use localhost to launch our web application password manager on our own Windows computer. However, in order to make it available via the internet and to benefit from its scalability and security characteristics, we want to host it on the cloud computing platform Amazon Web Services (AWS) in the near future.

5.3 Tools and Techniques

Following tools and techniques are used in the application:

- Python3.
- Django Framework.
- HTML, CSS, JavaScript.
- AES-256 GCM Encryption.
- RSA Encryption.

5.3.1 Softwares

We used the following software and technologies for this project:

OS:

Windows 11.

Platform:

Visual Studio Code.

Database:

SQLite3.

Language:

Python3, HTML, CSS, JavaScript.

5.3.2 Communications Interfaces

SAFEPASS can be run on any web browser, however, recommended and tested browsers are Chrome and Firefox. Also, for mailing users, we are using an SMTP server at the backend with TLS enabled.

Chapter 6

Testing and Evaluation

In this chapter we are going to run some tests on our product and analyze its behavior. All the test cases were successful at the time of testing.

Test Case - 1

Test Case ID	**SAFEPASS_001**
Test Case Description	Test the 2-factor authentication functionality
Created By	Sibtul
Reviewed By	Aizaz
Version	1.0

QA Tester's Log	tested

Tester's Name	Sibtul
Date Tested	6-Sep-2024
Test Case (Pass/Fail/Not Executed)	Pass

S #	**Prerequisites**
1	Access to Chrome Browser

S #	**Test Data**
1	User name = sib12
2	Pass = sdawS@.123.
3	Generated OTP: 803451

Test Scenario	Validating valid user name and password, the customer will get otp and verify the otp then user can login.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to `http://127.0.0.1:8000/signin`	Site should open	As Expected	Pass
2	Enter User name & Password	Credential can be entered	As Expected	Pass
3	Click Submit	OTP is generated and sent to user.	As Expected	Pass
4	Enter OTP	OTP can be entered	As Expected	Pass
5	Click Submit	Customer is logged in	As Expected	Pass

Table 6.1: Test Case 1

Test Case - 2

Test Case ID	SAFEPASS_002
Test Case Description	Add a new password
Created By	Aizaz
Reviewed By	Sibtul
Version	1.0

QA Tester's Log	Tested

Tester's Name	Bahria
Date Tested	1-Oct-2024
Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites
1	User logged in

S #	Test Data
1	url = https://www.facebook.com
2	Email = Sibtul@gmail.com
3	Password = .MyPassword@123.

Test Scenario	Verify on entering valid data and master password, the password encrypted with user encryption key and saved to database.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to `http://127.0.0.1:8000/dashboard`	Site should open	As Expected	Pass
2	Click on add password	Add password form displayed.	As Expected	Pass
3	Enter master password and data	Data can be entered.	As Expected	Pass
4	Click save.	Data validated and encrypted and password is hashed.	As Expected	Pass
5	Display message	Save data to database.	As Expected	Pass

Table 6.2: Test Case 2

Test Case – 3

Test Case ID	SAFEPASS_003
Test Case Description	Data Breach Scanning
Created By	Sibtul Hussain
Reviewed By	Aizaz Ahmad
Version	1.0

QA Tester's Log	Tested

Tester's Name	Aizaz
Date Tested	01-Oct-2024
Test Case (Pass/Fail/Not Executed)	Pass

Test Scenario	Send passwords hashed first five prefix values to "have I been pawned" site to check in their data breaches database and send all values related to those prefixes and alert user

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to `http://127.0.0.1:8000/scanner`	Site should open	As Expected	Pass
2	Click on Scan	Hashes will be compared one by one.	As Expected	Pass
3	Display Breached Passwords if any	Show the table of breached passwords	As Expected	Pass

Table 6.3: Test Case 3

Chapter 7

Conclusion

7.1 Achievements and Improvements

We successfully developed our secure password management web application, SAFEPASS, which provides users with enterprise-level features. It generates strong passwords according to NIST standards and uses encryption to securely store user credentials. SAFEPASS also includes features like data breach scanning, secure vault sharing, password monitoring, and health checks. Additionally, we've implemented two-factor authentication and a password health check tool. We take pride in SAFEPASS because it offers superior security and functionality compared to other password managers.

7.2 Critical Review

To ensure SAFEPASS meets high standards for security and functionality, we conducted extensive testing and analysis. We are confident that SAFEPASS provides users with a high level of security and usability. However, we recognize that there is always room for improvement, so we will continue to evaluate and enhance SAFEPASS over time.

7.3 Lessons Learned

Throughout this project, we learned the importance of thorough research, testing, and gathering user feedback when developing a web application. Additionally, we realized how critical it is to stay up to date with the latest cybersecurity trends and standards to maintain the relevance and effectiveness of our software.

7.4 Future Enhancements/Recommendations

SAFEPASS has already provided users with a safe and secure online password management tool. In the future, we plan to further enhance the product by introducing additional features.

One of the upcoming improvements we aim to implement is password-less authentication. This feature will allow users to access their accounts without needing to type a password. Instead, they will receive a one-time password via email, SMS, or an authenticator app, which is more secure than traditional passwords.

Another feature we plan to introduce is dark monitoring, a security tool that detects suspicious activity and immediately alerts users. This will help prevent cyber-attacks and protect users' data from unauthorized access.

In conclusion, we are committed to providing our users with the best security measures available, while continuously improving our product. Our ultimate goal is to make SAFEPASS the most feature-rich password management tool on the market.

Bibliography

[1] Breaches Were Related to Password Issues. (2021). *Cloud Nine*. Retrieved from `https://cloudnine.com/ediscoverydaily/electronic-discovery/80-percent-hacking-related-breachesrelated-password-issues-cybersecurity-trends/`.

[2] Wang, C., T. S. (n.d.). Empirical Analysis of User Passwords across Online Services. *Department of Computer Science, Virginia Tech*. Retrieved from `https://people.cs.vt.edu/gangwang/pass`.

[3] Silver, D., S. (n.d.). Password Managers: Attacks and Defenses, pp. 5-6.

[4] Josh. (2024). Are Password Managers Safe in 2024. *All Things Secured*. Retrieved from `https://www.allthingssecured.com/identity-protection/are-password-managers-safe/`.

[5] PassBolt For Teams. (n.d.). *PassBolt*. Retrieved from `https://github.com/passbolt`.