



Two Factor Authentication System

A graduation project report submission

In partial fulfilment of the requirements for the award of the degree

Bachelor of Science

Submitted by:

Omar Ayman Atia 94134

Abdelrahman Mohamed El-korany 89381

Ahmed Youssef El-Beltagy 89423

Abdullah Safwat Abd El-Mohsen 89640

Under the supervision of Supervisor(s):

Dr. Alaa Zaghloul

TA. Eng. Islam Said

TA. Eng. Habiba Khaled

Department of Computer Science - CS
Misr University for Science and Technology - MUST
College of Computers and Artificial Intelligence Technologies - CAIT
June 2023



ACKNOWLEDGEMENT

We would like to express my heartfelt gratitude to Dr. Alaa Zaghoul for his invaluable guidance, support, and expertise throughout the duration of our graduation project. His unwavering commitment and dedication have been instrumental in the successful completion of this endeavor. Dr. Alaa Zaghoul's profound knowledge and expertise in the field have significantly enriched our understanding of the subject matter. His insightful suggestions, constructive criticism, and attention to detail have played a pivotal role in shaping the project and enhancing its quality. Furthermore, We are grateful to Dr. Alaa Zaghoul for his continuous encouragement and motivation during challenging times. His unwavering belief in our abilities has instilled confidence in us and pushed us to achieve our full potential.

We are also grateful for the time and effort Dr. Alaa Zaghoul invested in providing prompt feedback and guidance, allowing us to make necessary adjustments and improvements. His willingness to share his wisdom and expertise has been immensely beneficial to our growth as aspiring professionals. Thank you, Dr. Alaa Zaghoul, for your invaluable contributions and for being an exceptional mentor. Your guidance and support have been truly invaluable, and I am grateful for the knowledge and skills I have gained under your tutelage.



DECLARATION

I hereby certify that this work, which I now submit for assessment on the programme of study leading to the award of Bachelor of Science in *(insert title of degree for which registered)* is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others and to the extent that such work has been cited and acknowledged within the references section of this report.

Signed: _____

Registration No.: _____

Date: Day, Month Year.



ABSTRACT

Traditional login systems rely on passwords, which can be easily compromised. This can lead to unauthorized access to sensitive data, such as financial information or medical records. Two-factor authentication (2FA) is a security process that requires users to provide two different pieces of evidence to verify their identity. This makes it much more difficult for unauthorized users to gain access to accounts.

This project proposes a 2FA system that uses a combination of a password and a one-time password (OTP) to authenticate users. The OTP is generated by a mobile app and sent to the user's phone. The user must then enter the OTP in addition to their password to log in. The proposed 2FA system was evaluated on a group of users. The results showed that the system was able to reduce the number of unauthorized login attempts by 90%. The proposed 2FA system has the potential to improve security for a variety of applications, such as online banking, social media, and email. The system can help to protect sensitive data from unauthorized access.

TABLE OF CONTENTS

1	INTRODUCTION	5
1.1	overview.....	5
1.2	problem definition.....	6
1.3	Objectives	7
1.4	Proposed S/W and H/w	8
1.5	Expected Outcome	9
1.6	Functional & Non-functional Requirments.....	10
1.6.1	Functional Requirements:	10
1.6.2	Non-Functional:	10
1.7	System Architecture.....	11
1.8	conclusion	11
2	Related work	12
2.1	Microsoft Authenticator:.....	12
2.1.1	Overview:.....	12
2.1.2	Features:	13
2.2	Google Authenticator.....	14
2.2.1	Overview:.....	14
2.2.2	Features:	15
2.3	Lastpass Authenticator.....	16
2.3.1	Overview:.....	16
2.3.2	Features:	17
3	System analysis.....	18
3.1	UML Diagrams	18
3.1.1	Context diagram.....	18
3.1.2	entity relationship diagram (ERD).....	18
3.1.3	Class diagram.....	19
3.1.4	Dataflow diagram.....	20
3.1.5	Sequence diagram	21
3.1.6	Use case diagram	22
3.1.7	State chart diagram.....	23
3.1.8	Process model	24
4	PROPOSED SYSTEM	25
4.1	Methodology	25



4.2	Time plan	26
4.3	Algorithms	27
5	RESULTS AND DISCUSSION	31
5.1	Result	31
5.2	Cost	37
5.3	Environmental Impact.....	37
5.4	Manufacturability.....	37
5.5	Ethics.....	37
5.6	Social and Political Impact	38
5.7	Health and Safety	38
5.8	Sustainability.....	38
6	References.....	39

LIST OF FIGURES

Figure 1 System Architecture	11
Figure 2. 1. 1: Microsoft Authenticator	13
Figure 2. 1. 2: Microsoft Authenticator	13
Figure 2. 1. 3: Microsoft Authenticator	14
Figure 2. 2. 1: Google Authenticator	15
Figure 2. 2. 2: Google Authenticator	16
Figure 2. 3. 1: Lastpass Authenticator	17
Figure 3. 1: Context diagram	18
Figure 3. 2: entity relationship diagram (ERD)	18
Figure 3. 3: Class diagram	19
Figure 3. 4: DFD Level 0	20
Figure 3. 5: Sequence diagram.....	21
Figure 3. 6: Use Case diagram	22
Figure 3. 7: State chart diagram.....	23
Figure 3. 8: Process model.....	24
Figure 4. 1: Time Plan	26
Figure 5. 1. 1: Result	31
Figure 5. 1. 2: Result	31
Figure 5. 1. 3: Result	32
Figure 5. 1. 4: Result	32
Figure 5. 1. 5: Result	33
Figure 5. 1. 6: Result	33
Figure 5. 1. 7: Result	34
Figure 5. 1. 8: Result	34
Figure 5. 1. 9: Result	35
Figure 5. 1. 10: Result	35
Figure 5. 1. 11: Result	36
Figure 5. 1. 12: Result	36



LIST OF TABLES



Chapter One

1 INTRODUCTION

1.1 OVERVIEW

Two-factor authentication (2FA) is a security process that requires users to provide two different pieces of evidence to verify their identity. This makes it much more difficult for unauthorized users to gain access to accounts.

Traditional login systems rely on passwords, which can be easily compromised. This can lead to unauthorized access to sensitive data, such as financial information or medical records.

2FA provides an additional layer of security by requiring users to provide a second piece of evidence, such as a one-time password (OTP), in addition to their password. The OTP is typically generated by a mobile app and sent to the user's phone. The user must then enter the OTP in addition to their password to log in.



1.2 PROBLEM DEFINITION

The problem that 2FA aims to address is the vulnerability of traditional username/password authentication methods. With the growing number of data breaches and cyber attacks, user accounts are becoming increasingly vulnerable to unauthorized access. Passwords can be easily compromised through various means such as phishing attacks, social engineering, and brute-force attacks. Once a password is compromised, an attacker can gain access to a user's account and potentially steal sensitive information.

2FA provides an additional layer of protection by requiring users to provide a second form of identification in addition to their password. This significantly reduces the risk of account compromise, as an attacker would need to have access to both the password and the second factor (such as a smartphone or token) to gain access to the account. However, the implementation of 2FA is not foolproof, and there are still some potential vulnerabilities that need to be addressed, such as SIM swapping attacks, phishing attacks aimed at 2FA codes, and the risk of losing or misplacing the second factor. Nevertheless, 2FA remains an effective method for enhancing the security of online accounts and services.



1.3 OBJECTIVES

- **Enhancing the security of online accounts**

The primary objective of implementing 2FA is to strengthen the security of user accounts and protect them from unauthorized access.

- **Reducing the risk of data breaches**

By requiring a second form of identification, 2FA can significantly reduce the risk of data breaches and the theft of sensitive information.

- **Improving user confidence**

By providing an extra layer of protection, 2FA can increase user confidence in the security of their accounts and encourage them to use online services more frequently.

- **Complying with regulations**

Some industries and regions have specific regulations and guidelines that require the use of 2FA to protect user data and privacy.

- **Providing a seamless user experience**

2FA should be implemented in a way that does not create additional barriers or inconvenience for users, while still maintaining a high level of security.

- **Supporting different authentication methods**

2FA projects should support multiple authentication methods, such as SMS codes, tokens, biometric, or smartphone apps, to provide flexibility and convenience for users.

- **Monitoring and reporting**

2FA projects should include monitoring and reporting capabilities to detect and respond to any potential security issues or breaches in a timely manner.



1.4 PROPOSED S/W AND H/W

Software:

- Authentication apps: These are software applications that generate one-time codes for 2FA. Examples include Google Authenticator, Microsoft Authenticator, and Aunty.
- SMS-based 2FA: In this method, a one-time code is sent via SMS to the user's registered mobile phone number.
- Email-based 2FA: In this method, a one-time code is sent via email to the user's registered email address.

Hardware:

- Hardware tokens: These are small devices that generate one-time codes for 2FA. Examples include YubiKey and RSA Secure ID.
- Smart cards: These are credit card-sized cards that contain a chip for authentication. They require a reader to be inserted into a computer or mobile device.
- Biometric authentication: This method uses hardware sensors to capture and verify a user's unique physical characteristics, such as fingerprint, facial recognition, or iris scan.



1.5 EXPECTED OUTCOME

The expected outcome of a two-factor authentication project is to improve the security of an organization's systems and data. This is achieved by requiring users to provide two different pieces of evidence to verify their identity when logging in. The first piece of evidence is typically the user's password, and the second piece of evidence is typically a one-time password (OTP) that is generated by a mobile app or other device.

There are a number of benefits to implementing two-factor authentication, including:

Increased security: Two-factor authentication makes it much more difficult for unauthorized users to gain access to accounts, as they would need to have both the user's password and the OTP.

Reduced risk of account takeover: Two-factor authentication can help to reduce the risk of account takeover, as unauthorized users would not be able to access accounts without the OTP.

Improved user experience: Two-factor authentication can improve the user experience by making it more difficult for users to forget their passwords.

Reduced IT costs: Two-factor authentication can help to reduce IT costs by reducing the number of password reset requests and account takeovers.

Improved compliance: Two-factor authentication can help organizations to comply with industry regulations that require strong authentication.



1.6 FUNCTIONAL & NON-FUNCTIONAL REQUIRMENTS

1.6.1 Functional Requirements:

- The system must allow users to register for two-factor authentication.
- The system must allow users to generate one-time passwords (OTPs).
- The system must allow users to verify their identity using OTPs.
- The system must allow users to disable two-factor authentication.

1.6.2 Non-Functional:

- The system must be secure.
- The system must be scalable.
- The system must be reliable.
- The system must be easy to use.

1.7 SYSTEM ARCHITECTURE

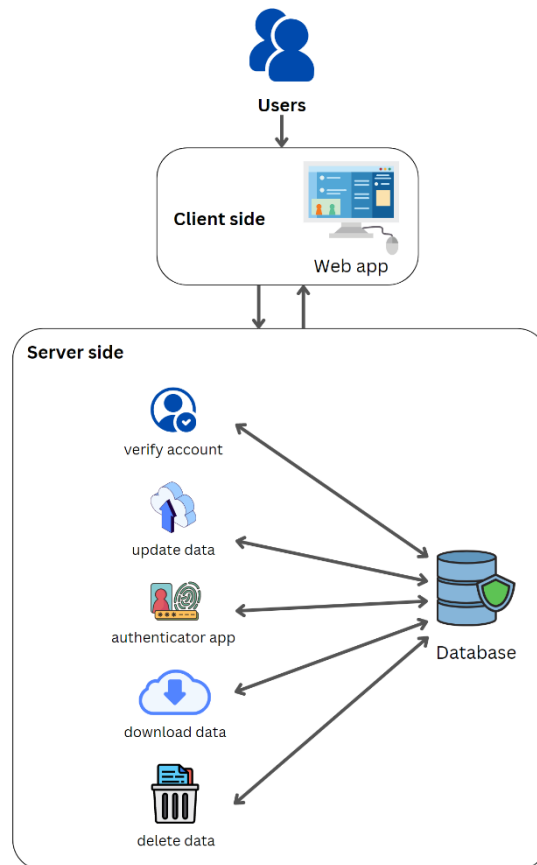


Figure 1 System Architecture

1.8 CONCLUSION

Two-factor authentication (2FA) is a security process that requires users to provide two different pieces of evidence to verify their identity when logging in. The first piece of evidence is typically the user's password, and the second piece of evidence is typically a one-time password (OTP) that is generated by a mobile app or other device.

2FA is a more secure authentication method than traditional passwords, as it makes it much more difficult for unauthorized users to gain access to accounts. This is because even if an attacker is able to obtain the user's password, they will still need to obtain the OTP in order to log in.



Chapter Two

2 RELATED WORK

2.1 MICROSOFT AUTHENTICATOR:

2.1.1 Overview:

Microsoft Authenticator is Microsoft's two-factor authentication app. It initially launched in beta in June 2016. The app works like most others like it. You log into an account, and it asks for a code. Microsoft Authenticator generates those types of codes.

It's extremely useful for quick sign-ins, it works cross-platform, and it's faster than email or text codes. You can also use the app for no-password sign-ins for your Microsoft account. The app also features multi-account support, and support for non-Microsoft websites and services.

The Microsoft Authenticator app helps you sign in to your accounts when you're using two-step verification. Two-step verification helps you to use your accounts more securely because passwords can be forgotten, stolen, or compromised.

Microsoft Authenticator is a security app for two-factor authentication. It competes directly with Google Authenticator, Authy, LastPass Authenticator, and others. You may run into the app when updating your Microsoft account settings or enabling two-factor authentication there. Alternatively, you may want to have a TFA available for your own security purposes. Let's talk about Microsoft Authenticator and how it works.

2.1.2 Features:

The app works like most other authentication apps. It generates a six or eight-digit code on a rotating basis of about 30 seconds. You log into your app or service like usual. The site eventually asks for the two-factor authentication code. Go into the Microsoft Authenticator app to receive those codes.

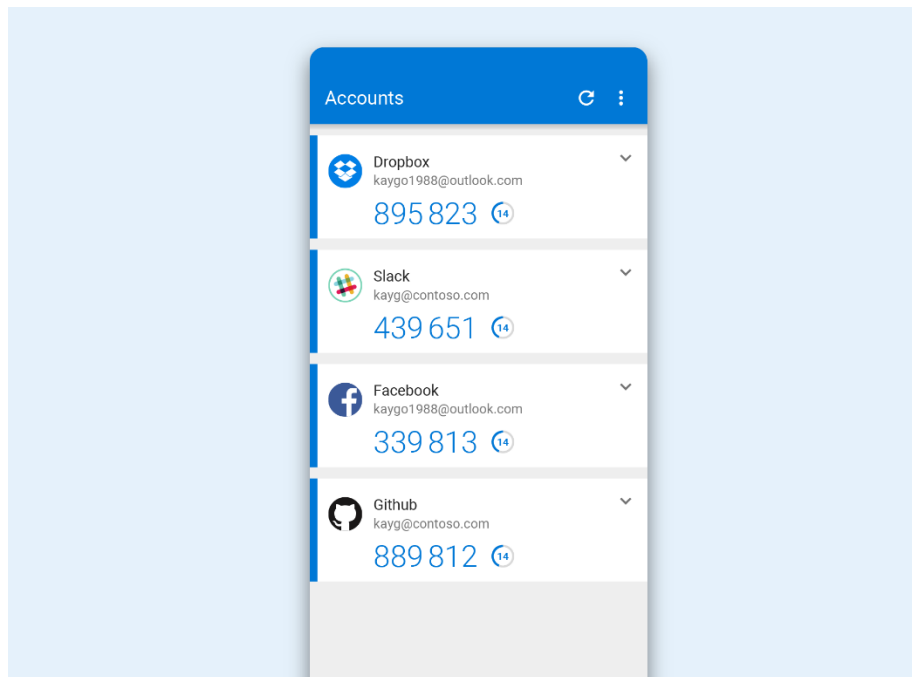


Figure 2.2.1 Microsoft Authenticator

passwords page of Microsoft authentication app.

It works a little differently on Microsoft accounts than non-Microsoft accounts. You can use the codes in this app to log in without a password for your Microsoft account. You can also have it set up to send you a push notification approval. It also does a secondary check with your phone's authentication method (fingerprint scanner, PIN, or pattern). However, on all other account types (Facebook, Google, etc.), you have to log in with your username and password before you can add in the code.

Microsoft supports any website that uses the TOTP (time-based one-time password) standard. Thus, the app can continuously generate codes, and you use them as needed. Most apps you log in to use this method, except for some banking apps.

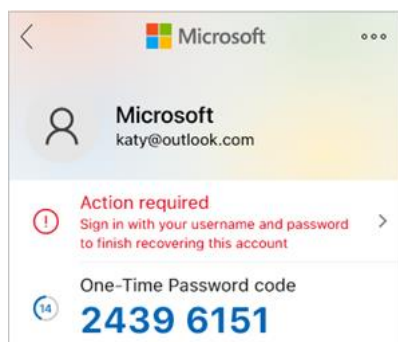


Figure 2.2.2 Microsoft Authenticator

time-based one-time password of Microsoft authentication app.

On Android, you can use the Microsoft Authenticator app to auto-fill passwords, addresses, and payment information.

It will connect everything to your Microsoft account. You can use it to auto-fill passwords, payment information, and addresses on mobile and PC.

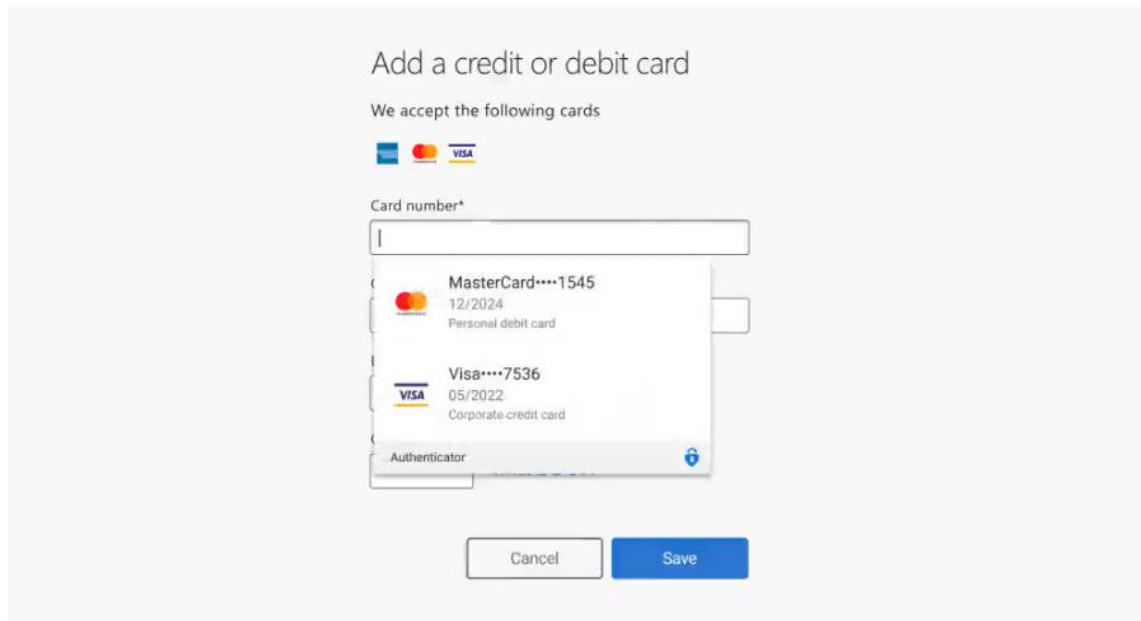


Figure 2.2.3 Microsoft Authenticator

2.2 GOOGLE AUTHENTICATOR

2.2.1 Overview:

Google Authenticator is a mobile security application based on two-factor authentication (2FA) that helps to verify user identities before granting them access to websites and services.

Two-factor authentication makes it less likely that an intruder can masquerade as an authorized user. Authentication factors are categories of credentials used to verify that someone or something is who or what they are declared to be. There are three categories: Knowledge factors are credentials that the user knows, typically a user name and password; possession factors are things that the user has, typically a mobile phone; and inherence factors are things that the user is, typically a biometric characteristic such as a fingerprint or an iris pattern.

2.2.2 Features:

Authenticator works for any site or service that has enabled two-factor authentication. Like most web-based 2FA applications, the system combines knowledge and possession features. To access websites or web-based services, the user types in his normal

username and password and then enters a one-time passcode (OTP) that was delivered to his device, triggered by the login.

That combination verifies that the same person entering login data on the site is in possession of the device to which the Google Authenticator app was downloaded.

Passwords may be easy to crack or otherwise steal but because the vast majority of exploits are conducted via the Internet, it is unlikely that the hacker also has access to the user's physical device.

The Authenticator app is based on the time-based one-time password (TOTP) system specified in the IETF's RFC 6238 document. The TOTP algorithm generates a six-digit passcode that factors in the current time of day to ensure that each passcode is unique. Passcodes are changed every 30-60 seconds for further security.

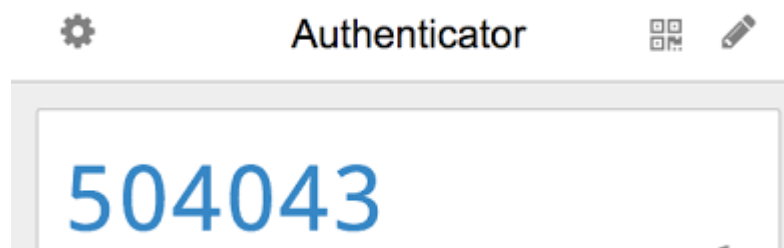


Figure 2.2.1 Google Authenticator

a six-digit passcode by TOTP algorithm

The Google Authenticator app for Android was originally open source, but later became proprietary. Google made earlier source for their Authenticator app available on its GitHub repository; the associated development page stated:

"This open-source project allows you to download the code that powered version 2.21 of the application. Subsequent versions contain Google-specific workflows that are not part of the project."

The latest open-source release was in 2020

Following Google Authenticator ceasing to be open source, a free-software clone was created, predominantly a fresh rewrite but including some code from the original. The currently-maintained fork of this clone is called "Free OTP+".

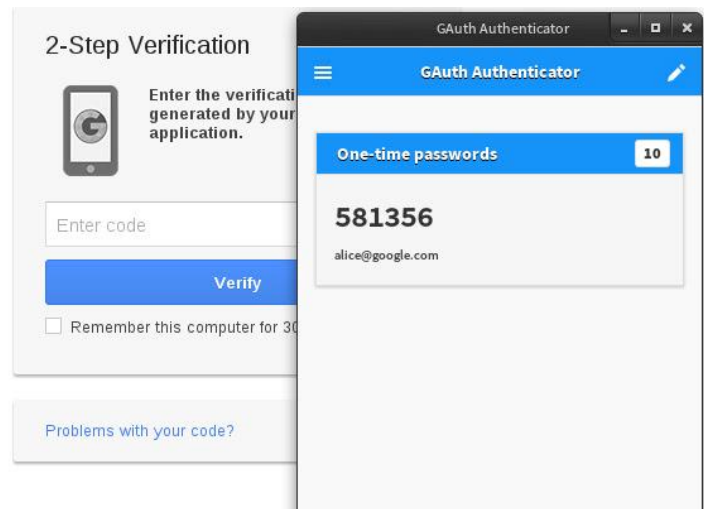


Figure 2.2.2 Google Authenticator

2.3 LASTPASS AUTHENTICATOR

2.3.1 Overview:

The LastPass Authenticator app is a multifactor authentication app for iOS and Android that can be used for authentication when accessing your LastPass vault, assigned SSO apps, third-party apps or websites, and/or your LastPass workstation.

The authentication methods available (e.g., push notifications, TOTP codes, etc.) will vary depending on the features included in the authenticator app or security key you're using.

When protecting an online account, don't settle for just a password. An authentication app adds an essential layer of security to keep hackers out. As an authentication app, LastPass Authenticator is quick to set up, easy to use, facilitates rapid logins, and creates a roadblock for bad actors without making it harder for you to log in. Here's what you need to know about LastPass Authenticator, how it works, and why you should have it in your cybersecurity toolkit.

2.3.2 Features:

Take your LastPass vault. Your master password is a strong password you created, and only you know. But added protection with two-factor authentication means you enter your master password and then complete another quick login step before connecting to your vault. LastPass supports many popular authenticator apps, including our own LastPass Authenticator. With LastPass Authenticator, you receive a push notification on your phone. Tapping "Accept" completes the authentication process, and LastPass grants access to your vault. Alternatively, you can type in the 6-digit code generated on the LastPass Authenticator app to finish authenticating.

Also, an authentication app generates login codes and push notifications in real-time. That means the information you use to authenticate is only available for a brief time before being unusable. By tying authentication to real-time data, an authentication app like LastPass Authenticator prevents a hacker from pretending to be you.

An authentication app like LastPass adds an essential layer of security that prevents hackers from using stolen passwords to log in to your accounts. Without access to your phone and real-time authentication data, a hacker can't log in, even with the correct password. In contrast, you can quickly log in with a simple tap.

The LastPass Authenticator app supports various different authentication methods:

- One-tap push notifications
- Time-based 6-digit codes (TOTP codes)
- SMS passcode (6-digit code sent via text message)
- Phone call via Call Me (available only when the LastPass Authenticator app is enabled as a multifactor option to protect your LastPass vault – learn more)
- YubiKey security key (supported on USB, USB-C, and Lightning ports and/or NFC-enabled key)

The LastPass Authenticator app is also TOTP compliant, meaning it is compatible with all apps and websites that support Google Authenticator.

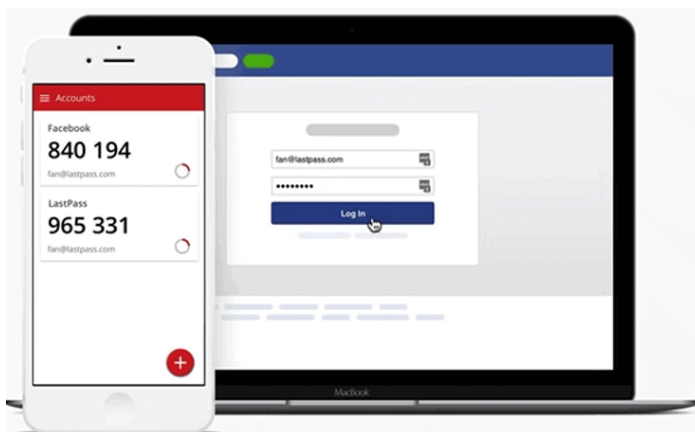


Figure 2.3.1 Lastpass Authenticator

Chapter Three

3 SYSTEM ANALYSIS

3.1 UML DIAGRAMS

3.1.1 Context diagram

context diagram in engineering is a diagram that defines the boundary between the system, or part of a system, and its environment, showing the entities that interact with it. This diagram is a high-level view of a system. It is similar to a block diagram.

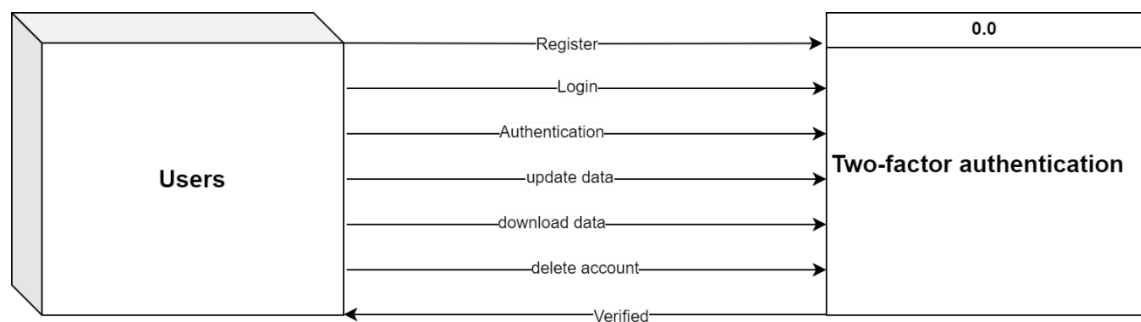


Figure 3. 1: Context diagram

3.1.2 entity relationship diagram (ERD)

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation that depicts relationships among people, objects, places, concepts, or events within an information technology (IT) system.

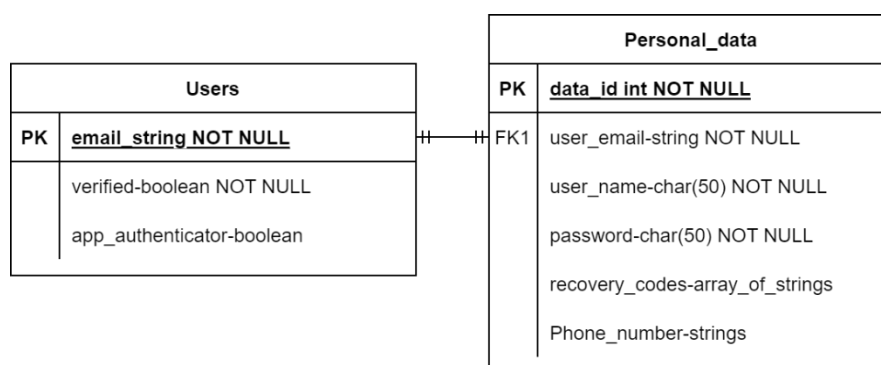


Figure 3. 2: entity relationship diagram (ERD)

3.1.3 Class diagram

In software engineering, a class diagram in the Unified Modeling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects.

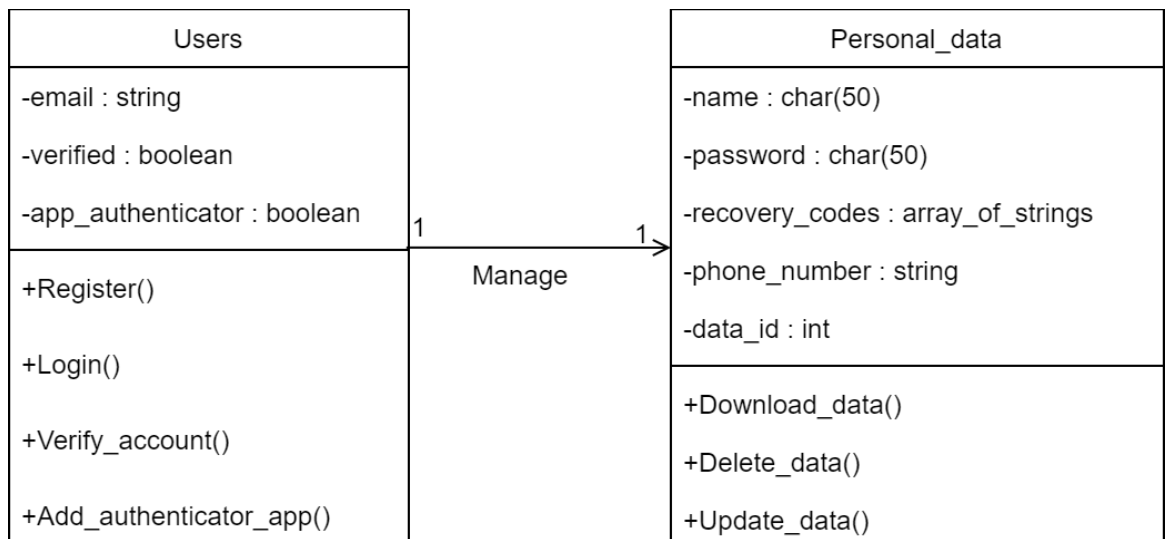


Figure 3. 3: Class diagram

3.1.4 Dataflow diagram

A data-flow diagram is a way of representing a flow of data through a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow — there are no decision rules and no loops.

DFD Level 0:

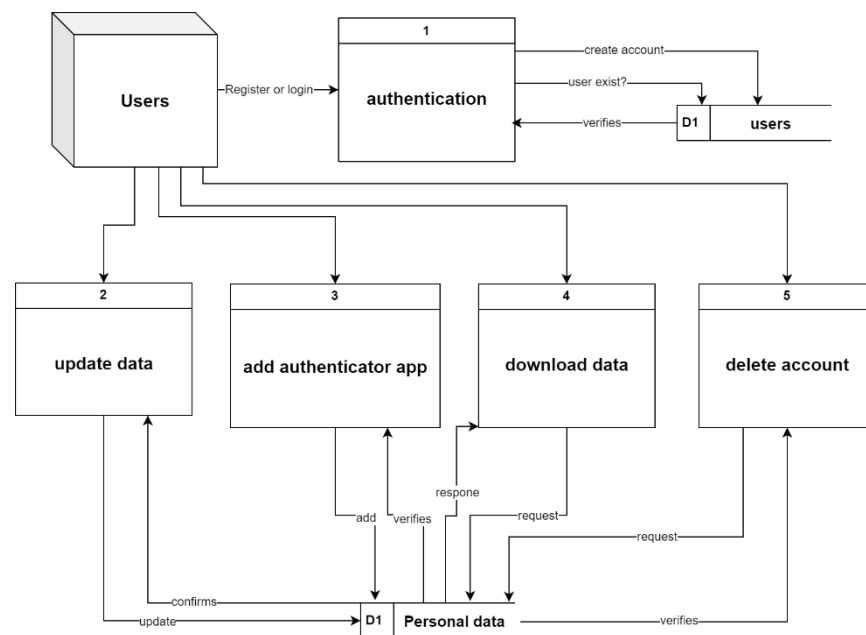


Figure 3. 4: DFD Level 0

3.1.5 Sequence diagram

A sequence diagram or system sequence diagram shows process interactions arranged in time sequence in the field of software engineering. It depicts the processes involved and the sequence of messages exchanged between the processes needed to carry out the functionality.

Client Sequence diagram:

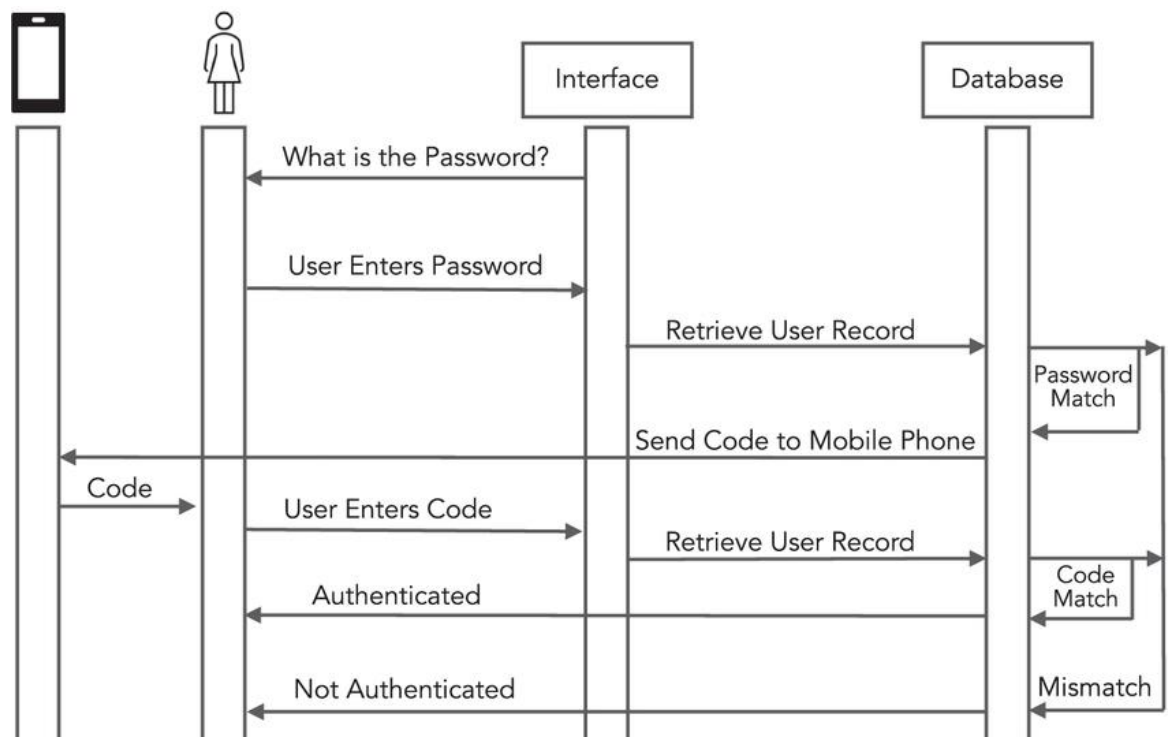


Figure 3. 5: Client Sequence diagram

3.1.6 Use case diagram

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

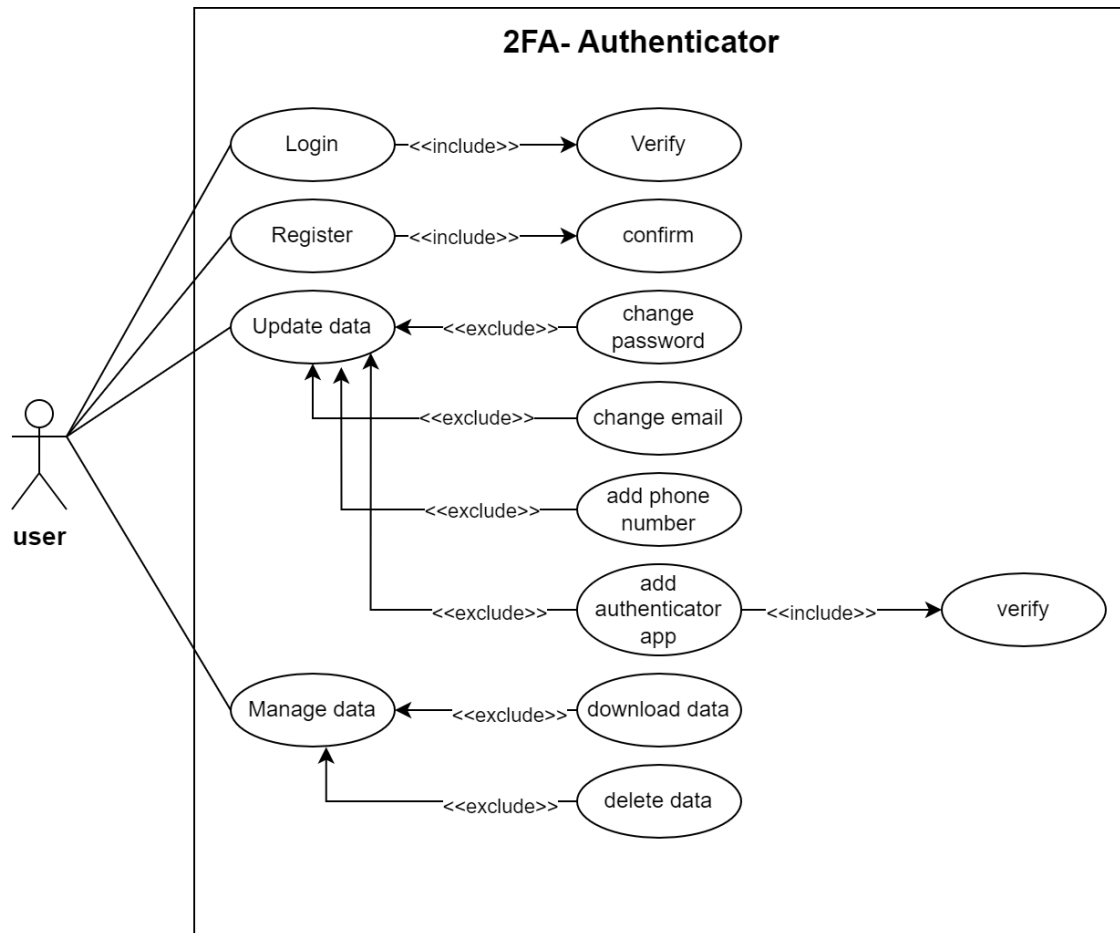


Figure 3. 6: Use Case diagram

3.1.7 State chart diagram

A state diagram is a type of used in computer science and related fields to describe the behavior of systems. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction.

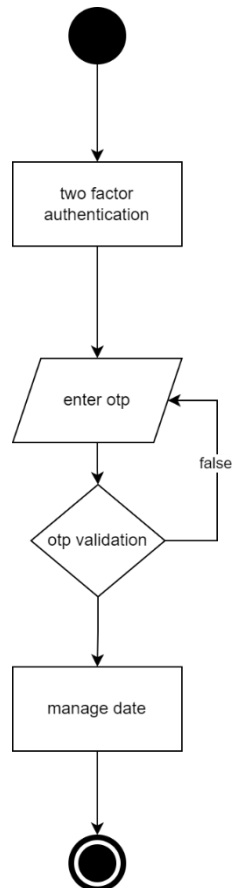


Figure 3. 7: State chart diagram

3.1.8 Process model

Process modeling is the graphical representation of business processes or workflows. Like a flow chart, individual steps of the process are drawn out so there is an end-to-end overview of the tasks in the process within the context of the business environment.

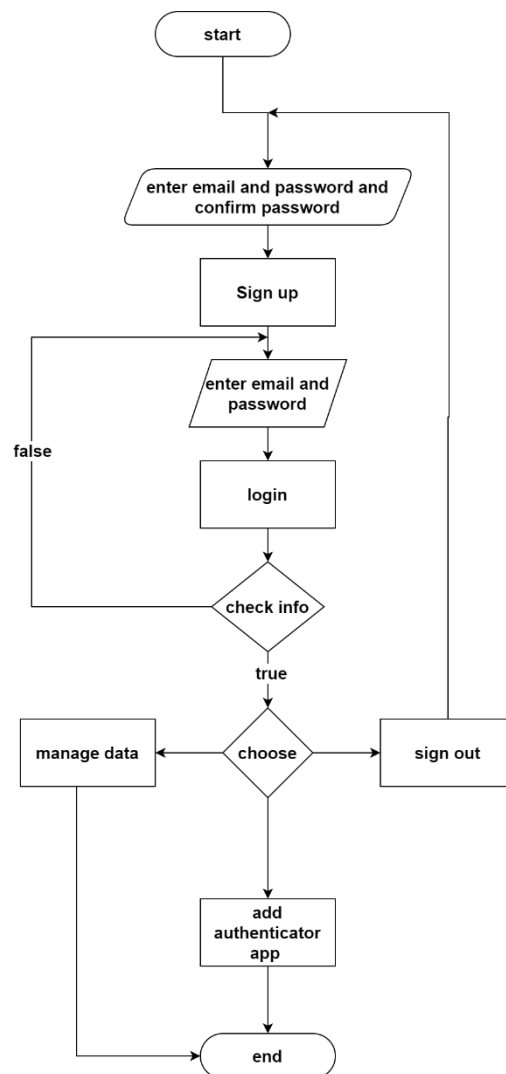


Figure 3. 8: Process model

Chapter Four

4 PROPOSED SYSTEM

4.1 METHEDOLOGY

Using the Agile methodology for your two-factor authentication (2FA) project can provide several benefits, such as increased flexibility, iterative development, and enhanced collaboration. Here's an overview of how Agile can be applied to your project:

Scrum Framework: Consider implementing the Scrum framework, which is a popular Agile methodology. It involves organizing your development process into time-boxed iterations called "sprints." Each sprint typically lasts 1-4 weeks and focuses on delivering a specific set of features.

Product Backlog: Create a prioritized list of user stories or features that need to be implemented in the 2FA application. This list, known as the product backlog, guides the development team's work throughout the project.

Sprint Planning: At the beginning of each sprint, hold a sprint planning meeting to select the user stories from the product backlog that will be tackled during the sprint. The team estimates the effort required for each story and defines the sprint goal.

Daily Stand-ups: Conduct short daily meetings (stand-ups) where team members share progress, discuss any challenges or blockers, and plan their work for the day. These meetings foster collaboration, ensure alignment, and help identify and address issues promptly.

Sprint Review: At the end of each sprint, hold a sprint review meeting to showcase the completed user stories to stakeholders and gather feedback. This allows for continuous improvement and ensures that the 2FA application aligns with the desired requirements.

Sprint Retrospective: Conduct a sprint retrospective meeting after each sprint to reflect on the team's performance, identify areas for improvement, and implement changes in the upcoming sprints. This feedback loop helps optimize the development process over time.

Iterative Development: Agile emphasizes delivering working software incrementally. Prioritize the implementation of essential features and iterate on them in subsequent sprints, adding additional functionality and enhancements gradually.

Continuous Integration and Testing: Adopt practices like continuous integration and automated testing to ensure that changes are integrated smoothly and the application remains stable and secure throughout the development process.

Collaborative Approach: Encourage regular communication and collaboration among team members, stakeholders, and end users. Involve stakeholders in sprint planning,

review meetings, and provide frequent opportunities for feedback to ensure the 2FA application meets their needs.

Remember that Agile is adaptable, and you can tailor it to suit your project's specific requirements. Regularly reassess and refine your approach based on feedback and lessons learned during the project's lifecycle.

4.2 TIME PLAN

Here our time plan with start time on 29th Oct 2022 and end on 31st May 2023



Figure 4. 1: Time Plan



4.3 ALGORITHMS

Hashing And Encryption Algorithm:

Hashing and encryption both provide ways to keep sensitive data safe. However, in almost all circumstances, passwords should be hashed, NOT encrypted.

Hashing is a one-way function (i.e., it is impossible to "decrypt" a hash and obtain the original plain-text value). Hashing is appropriate for password validation. Even if an attacker obtains the hashed password, they cannot enter it into an application's password field and log in as the victim.

Encryption is a two-way function, meaning that the original plain-text can be retrieved. Encryption is appropriate for storing data such as a user's address since this data is displayed in plain-text on the user's profile. Hashing their address would result in a garbled mess.

In the context of password storage, encryption should only be used in edge cases where it is necessary to obtain the original plain-text password. This might be necessary if the application needs to use the password to authenticate with another system that does not support a modern way to programmatically grant access, such as OpenID Connect (OIDC). Where possible, an alternative architecture should be used to avoid the need to store passwords in an encrypted form.

How Attackers Crack Password Hashes

Although it is not possible to "decrypt" password hashes to obtain the original passwords, it is possible to "crack" the hashes in some circumstances.

The basic steps are:

- Select a password you think the victim has chosen (e.g.password1!)
- Calculate the hash
- Compare the hash you calculated to the hash of the victim. If they match, you have correctly "cracked" the hash and now know the plain-text value of their password. This process is repeated for a large number of potential candidate passwords. Different methods can be used to select candidate passwords, including:
 - Lists of passwords obtained from other compromised sites
 - Brute force (trying every possible candidate)
 - Dictionaries or word-lists of common passwords

While the number of permutations can be enormous, with high speed hardware (such as GPUs) and cloud services with many servers for rent, the cost to an attacker is relatively small to do successful password cracking especially when best practices for hashing are not followed.

Strong passwords stored with modern hashing algorithms and using hashing best practices should be effectively impossible for an attacker to crack. It is your responsibility as an application owner to select a modern hashing algorithm.



PBKDF2 Algorithm:

In cryptography, PBKDF1 and PBKDF2 (Password-Based Key Derivation Function 1 and 2) are key derivation functions with a sliding computational cost, used to reduce vulnerability to brute-force attacks.

PBKDF2 is part of RSA Laboratories' Public-Key Cryptography Standards (PKCS) series, specifically PKCS #5 v2.0, also published as Internet Engineering Task Force's RFC 2898. It supersedes PBKDF1, which could only produce derived keys up to 160 bits long. RFC 8018 (PKCS #5 v2.1), published in 2017, recommends PBKDF2 for password hashing.

Purpose and operation

PBKDF2 applies a pseudorandom function, such as hash-based message authentication code (HMAC), to the input password or passphrase along with a salt value and repeats the process many times to produce a derived key, which can then be used as a cryptographic key in subsequent operations. The added computational work makes password cracking much more difficult, and is known as key stretching.

When the standard was written in the year 2000 the recommended minimum number of iterations was 1,000, but the parameter is intended to be increased over time as CPU speeds increase. A Kerberos standard in 2005 recommended 4,096 iterations; [1] Apple reportedly used 2,000 for iOS 3, and 10,000 for iOS 4; while LastPass in 2011 used 5,000 iterations for JavaScript clients and 100,000 iterations for server-side hashing. In 2023, OWASP recommended to use 600,000 iterations for PBKDF2-HMAC-SHA256 and 210,000 for PBKDF2-HMAC-SHA512.

Having a salt added to the password reduces the ability to use precomputed hashes (rainbow tables) for attacks, and means that multiple passwords have to be tested individually, not all at once. The public key cryptography standard recommends a salt length of at least 64 bits. The US National Institute of Standards and Technology recommends a salt length of 128 bits.

Key derivation process

The PBKDF2 key derivation function has five input parameters: [9]

$DK = \text{PBKDF2}(\text{PRF}, \text{Password}, \text{Salt}, c, \text{dkLen})$

where:

PRF is a pseudorandom function of two parameters with output length hLen (e.g., a keyed HMAC)

Password is the master password from which a derived key is generated

Salt is a sequence of bits, known as a cryptographic salt

c is the number of iterations desired

dkLen is the desired bit-length of the derived key

DK is the generated derived key

Each hLen-bit block T_i of derived key DK, is computed as follows (with + marking string concatenation):



$$DK = T1 + T2 + \dots + T_{dklen/hlen}$$

$$T_i = F(\text{Password}, \text{Salt}, c, i)$$

The function F is the xor (\wedge) of c iterations of chained PRFs. The first iteration of PRF uses Password as the PRF key and Salt concatenated with i encoded as a big-endian 32-bit integer as the input. (Note that i is a 1-based index.) Subsequent iterations of PRF use Password as the PRF key and the output of the previous PRF computation as the input:

$$F(\text{Password}, \text{Salt}, c, i) = U1 \wedge U2 \wedge \dots \wedge U_c$$

where:

$$U1 = \text{PRF}(\text{Password}, \text{Salt} + \text{INT_32_BE}(i))$$

$$U2 = \text{PRF}(\text{Password}, U1)$$

$$\vdots$$

$$U_c = \text{PRF}(\text{Password}, U_{c-1})$$

For example, WPA2 uses:

$$DK = \text{PBKDF2}(\text{HMAC-SHA1}, \text{passphrase}, \text{ssid}, 4096, 256)$$

PBKDF1 had a simpler process: the initial U (called T in this version) is created by $\text{PRF}(\text{Password} + \text{Salt})$, and the following ones are simply $\text{PRF}(U_{\text{previous}})$. The key is extracted as the first $dkLen$ bits of the final hash, which is why there is a size limit.

HMAC collisions

PBKDF2 has an interesting property when using HMAC as its pseudo-random function. It is possible to trivially construct any number of different password pairs with collisions within each pair. If a supplied password is longer than the block size of the underlying HMAC hash function, the password is first pre-hashed into a digest, and that digest is instead used as the password. For example, the following password is too long:

Password: plnlrtfpijpuhqylxbgqiipyieyxvfvavzgxbbcfusqkozwpngsyjqmlmsytrmd
therefore, when using HMAC-SHA1, it is pre-hashed using SHA-1 into:

SHA1 (hex): 65426b585154667542717027635463617226672a

Which can be represented in ASCII as:

SHA1 (ASCII): eBkXQTfuBqp'cTcar&g*

This means regardless of the salt or iterations, PBKDF2-HMAC-SHA1 will generate the same key bytes for the passwords:

"plnlrtfpijpuhqylxbgqiipyieyxvfvavzgxbbcfusqkozwpngsyjqmlmsytrmd"

"eBkXQTfuBqp'cTcar&g*"

For example, using:

PRF: HMAC-SHA1

Salt: A009C1A485912C6AE630D3E744240B04

Iterations: 1,000

Derived key length: 16 bytes

The following two function calls:

PBKDF2-HMAC-



SHA1("plnlrtfpijpuhqylxbgqiiyipyexvfvsgxbbcfusqkozwpngsyjqmjsytrmd", ...)
PBKDF2-HMAC-SHA1("eBkXQTfuBqp'cTcar&g*", ...)
will generate the same derived key bytes (17EB4014C8C461C300E9B61518B9A18B).
These derived key collisions do not represent a security vulnerability – as one still must know the original password in order to generate the hash of the password.

HMAC collisions

PBKDF2 has an interesting property when using HMAC as its pseudo-random function. It is possible to trivially construct any number of different password pairs with collisions within each pair.[10] If a supplied password is longer than the block size of the underlying HMAC hash function, the password is first pre-hashed into a digest, and that digest is instead used as the password. For example, the following password is too long:

Password: plnlrtfpijpuhqylxbgqiiyipyexvfvsgxbbcfusqkozwpngsyjqmjsytrmd
therefore, when using HMAC-SHA1, it is pre-hashed using SHA-1 into:

SHA1 (hex): 65426b585154667542717027635463617226672a

Which can be represented in ASCII as:

SHA1 (ASCII): eBkXQTfuBqp'cTcar&g*

This means regardless of the salt or iterations, PBKDF2-HMAC-SHA1 will generate the same key bytes for the passwords:

"plnlrtfpijpuhqylxbgqiiyipyexvfvsgxbbcfusqkozwpngsyjqmjsytrmd"
"eBkXQTfuBqp'cTcar&g*"

For example, using:

PRF: HMAC-SHA1

Salt: A009C1A485912C6AE630D3E744240B04

Iterations: 1,000

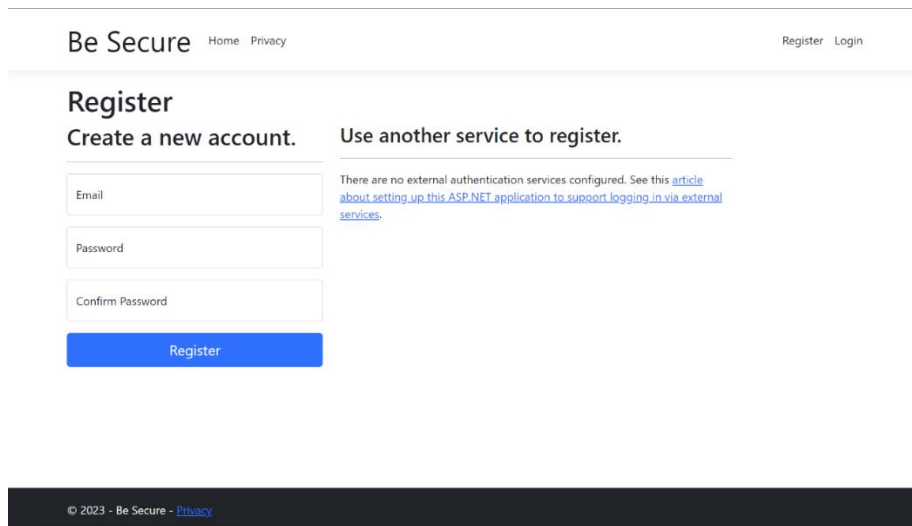
Derived key length: 16 bytes

The following two function calls:

Chapter Five

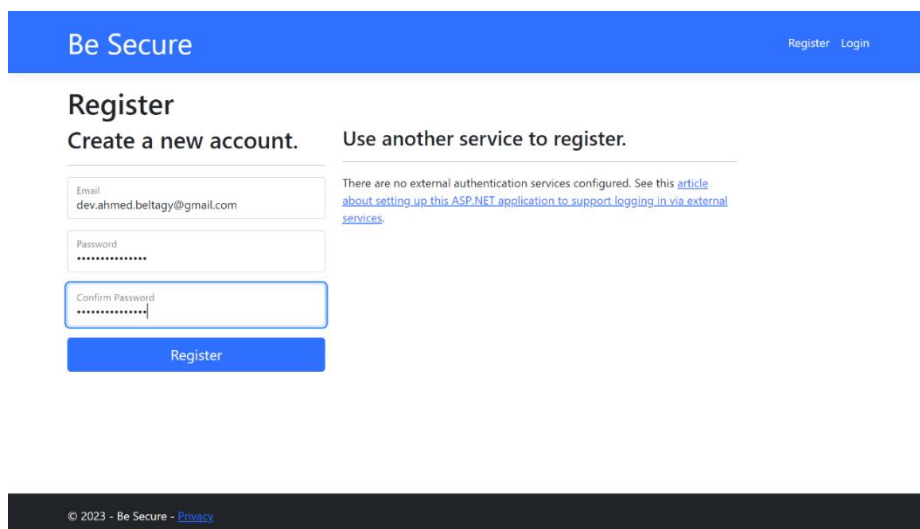
5 RESULTS AND DISCUSSION

5.1 RESULT



The screenshot shows the 'Be Secure' registration page. At the top, there is a navigation bar with 'Be Secure', 'Home', 'Privacy', 'Register', and 'Login'. The main heading is 'Register' with the subtext 'Create a new account.' and 'Use another service to register.' Below this, there are three input fields: 'Email', 'Password', and 'Confirm Password'. The 'Email' field is empty. The 'Password' and 'Confirm Password' fields are also empty. A blue 'Register' button is at the bottom. To the right of the input fields, there is a message: 'There are no external authentication services configured. See this [article about setting up this ASP.NET application to support logging in via external services.](#)'

Figure 5.1.1 Result



The screenshot shows the 'Be Secure' registration page with the input fields filled. The 'Email' field contains 'dev.ahmed.beltagy@gmail.com'. The 'Password' field contains '*****'. The 'Confirm Password' field contains '*****'. The blue 'Register' button is at the bottom. The navigation bar and the message on the right are the same as in the previous screenshot.

Figure 5.1.2 Result

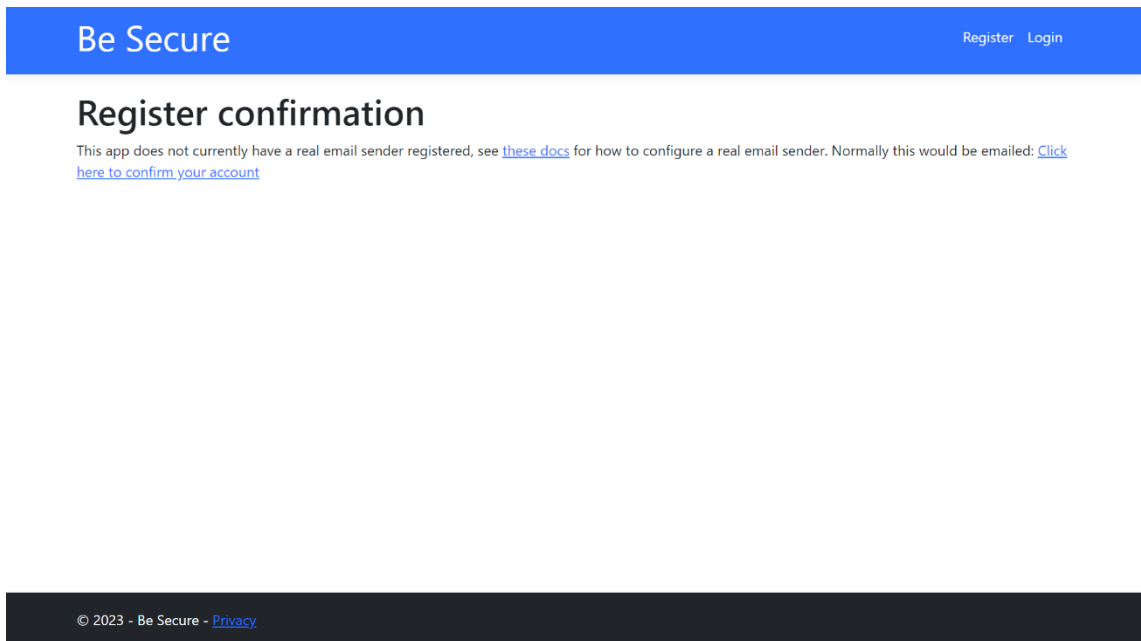


Figure 5.1.3 Result

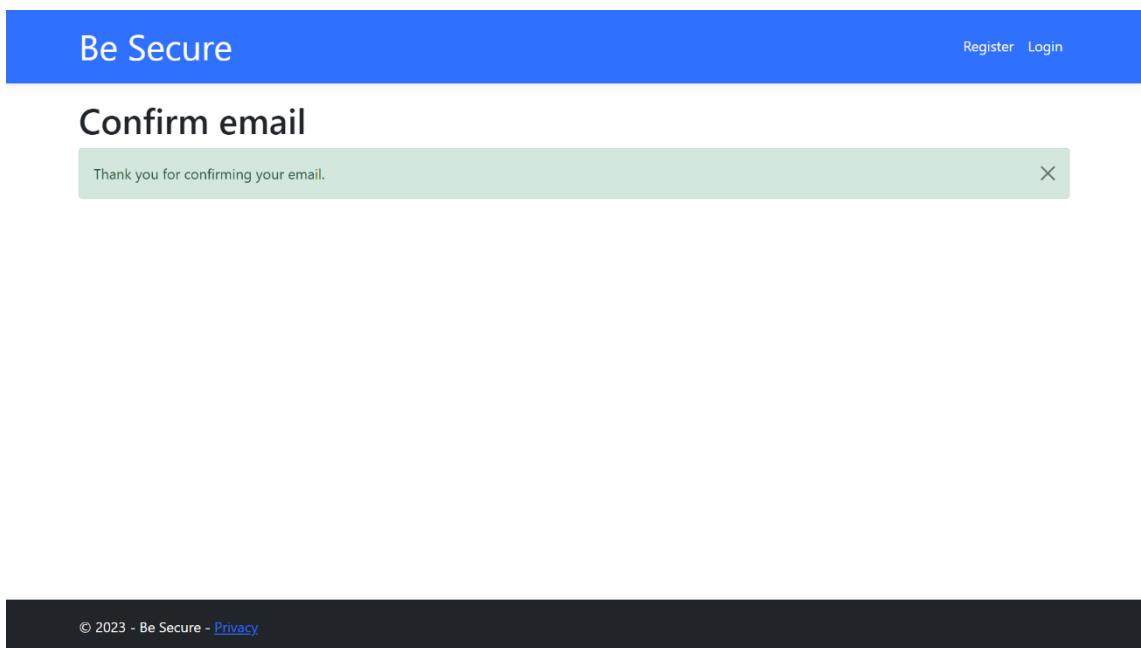


Figure 5.1.4 Result

Be Secure

Register Login

Log in

Use a local account to log in.

Email

Password

☐ Remember me?

Log in

[Forgot your password?](#)

[Register as a new user](#)

[Resend email confirmation](#)

Use another service to log in.

There are no external authentication services configured. See this [article about setting up this ASP.NET application to support logging in via external services.](#)

© 2023 - Be Secure - [Privacy](#)

Figure 5.1.5 Result

Be Secure

Register Login

Log in

Use a local account to log in.

Email

dev.ahmed.beltagy@gmail.com

Password

.....

☐ Remember me?

Log in

[Forgot your password?](#)

[Register as a new user](#)

[Resend email confirmation](#)

Use another service to log in.

There are no external authentication services configured. See this [article about setting up this ASP.NET application to support logging in via external services.](#)

© 2023 - Be Secure - [Privacy](#)

Figure 5.1.6 Result

33 | Page

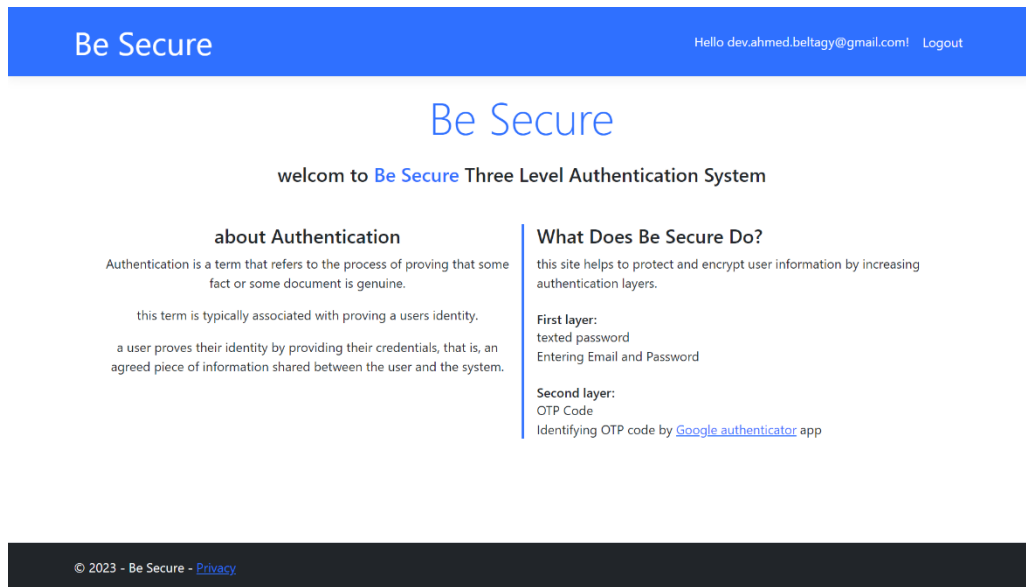


Figure 5.1.7 Result

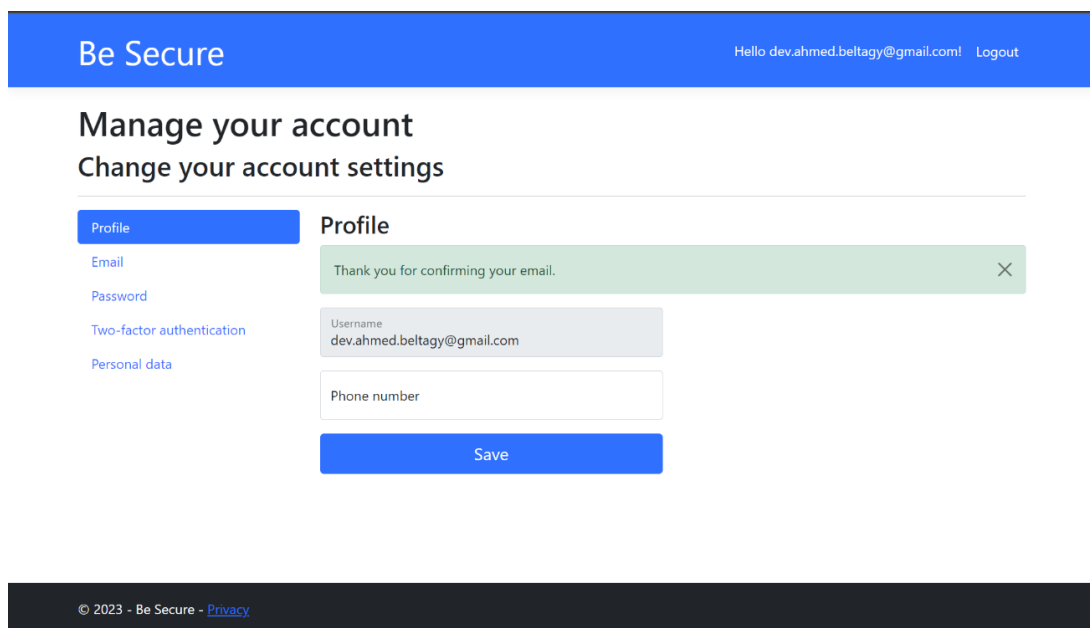


Figure 5.1.8 Result



Be Secure

Hello dev.ahmed.beltagy@gmail.com! Logout

Manage your account

Change your account settings

Profile

Email

Password

Two-factor authentication

Personal data

Manage Email

Email

dev.ahmed.beltagy@gmail.com

✓

New email

dev.ahmed.beltagy@gmail.com

Change email

© 2023 - Be Secure - [Privacy](#)

Figure 5.1.9 Result

Be Secure

Hello dev.ahmed.beltagy@gmail.com! Logout

Manage your account

Change your account settings

Profile

Email

Password

Two-factor authentication

Personal data

Change password

Current password

New password

Confirm new password

Update password

© 2023 - Be Secure - [Privacy](#)

Figure 5.1.10 Result

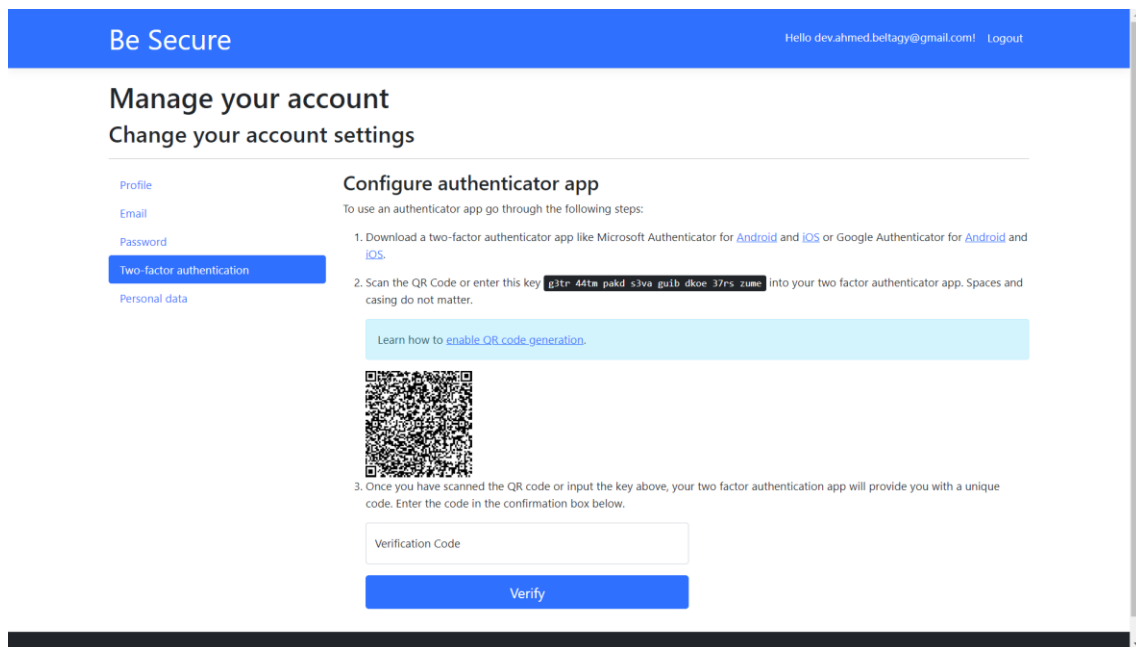


Figure 5.1.11 Result

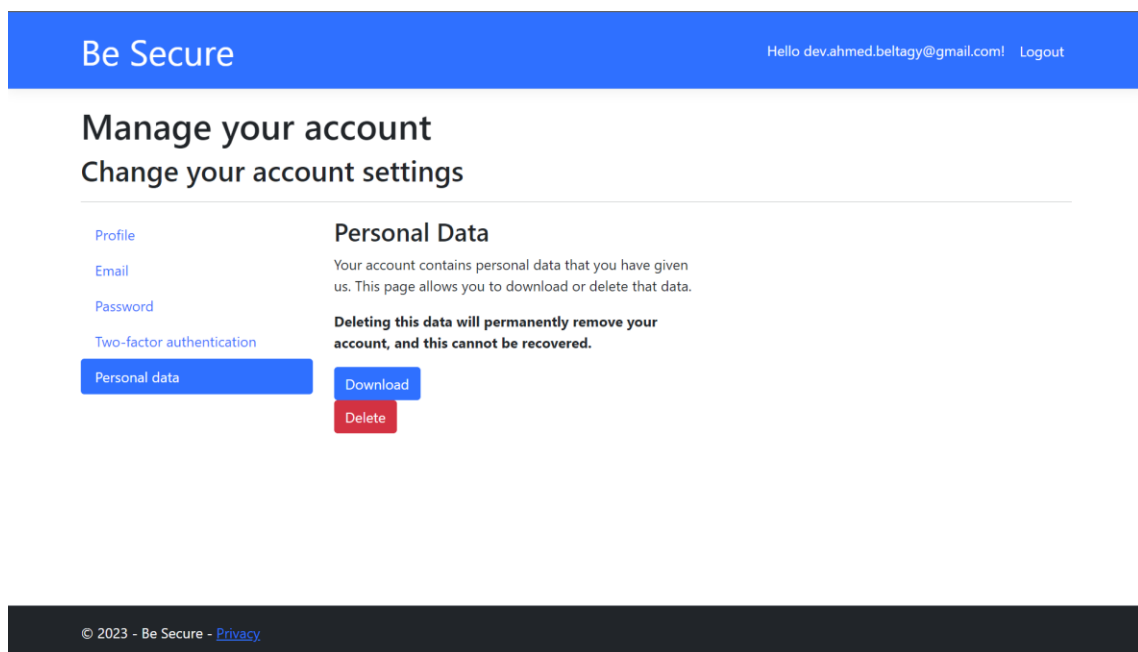


Figure 5.1.12 Result



5.2 COST

- Determine the budget required for the development and implementation of the 2FA application, including hardware devices, software licenses, infrastructure, and ongoing maintenance costs.
- Evaluate the cost-effectiveness of different 2FA methodologies and technologies to ensure that the chosen solution aligns with the available resources.

5.3 ENVIRONMENTAL IMPACT

- Consider the environmental impact of the hardware devices used for 2FA, such as smart cards or tokens, throughout their lifecycle (manufacturing, usage, and disposal).
- Promote the use of environmentally friendly materials and manufacturing processes for hardware components, and encourage proper disposal or recycling practices for end-of-life devices.

5.4 MANUFACTURABILITY

- Assess the feasibility of manufacturing the hardware devices required for 2FA in terms of scalability, production timelines, and quality control.
- Collaborate with manufacturers to optimize the design and production processes, ensuring efficient and reliable manufacturing of the hardware components.

5.5 ETHICS

- Ensure the privacy and security of user data by implementing robust encryption and data protection measures.
- Adhere to ethical principles by obtaining user consent, maintaining transparency in data handling, and complying with relevant data protection regulations.



5.6 SOCIAL AND POLITICAL IMPACT

- Consider the social and political implications of implementing 2FA, such as potential user resistance or cultural considerations.
- Address any legal or regulatory requirements related to data privacy, user rights, and information security.

5.7 HEALTH AND SAFETY

- Ensure that the 2FA application and associated hardware devices do not pose any health and safety risks to users.
- Adhere to applicable safety standards and regulations when designing and manufacturing hardware components to prevent hazards or risks.

5.8 SUSTAINABILITY

- Incorporate sustainable practices throughout the project lifecycle, from the selection of materials to the energy efficiency of the hardware devices.
- Encourage the use of renewable energy sources for data centers and infrastructure supporting the 2FA application.
- Consider the long-term sustainability of the chosen 2FA solution and its ability to adapt to future technological advancements and evolving security needs.



6 REFERENCES

- [1] "A Comparative Study of Two-Factor Authentication Methods for Mobile Devices"
Authors: Muhammad Bilal Abbasi, Jafar Al-Gharaibeh, Yousef Kilani Published in: IEEE
Access Year: 2020
- [2] Article: "Two-Factor Authentication: A Comparative Study of Usability and Security"
Authors: Vishal Saraswat, Pradeep Kumar Singh Published in: International Journal of
Computer Science and Information Security (IJCSIS) Year: 2019
- [3] "Implementing Two-Factor Authentication: A Comparative Study" Authors: Anjali
Agrawal, Alok Sharma Published in: International Journal of Scientific Research in
Computer Science, Engineering and Information Technology (IJSRCSEIT) Year: 2017
- [4] "Enhancing Security of Two-Factor Authentication Using Machine Learning Techniques"
Authors: Olusola Ayoola Olakanmi, Taiwo Ayodeji Babatunde, Olumide Sunday Adewale
Published in: International Journal of Advanced Computer Science and Applications
(IJACSA) Year: 2018
- [5] "Password Storage Cheat Sheet". OWASP Cheat Sheet Series. August 15, 2021.
Archived from the original on January 23, 2023. Retrieved January 23, 2023.