



University of Reading  
Department of Computer Science

# FortiVault: Architecting the Future of Digital Security through Password Management

Akatomisan Michaelangelo Ajuyah

*Supervisor:* Dr Martin Lester

A report submitted in partial fulfilment of the requirements of the University  
of Reading for the degree of  
Bachelor of Science in *Computer Science*

April 16, 2024

## Declaration

I, Akatomisan Ajuyah, of the Department of Computer Science, University of Reading, confirm that this is my own work and figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted, and referenced. I understand that if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

I give consent to a copy of my report being shared with future students as an exemplar.

I give consent for my work to be made available more widely to members of UoR and public with interest in teaching, learning and research.

Akatomisan Ajuyah  
April 16, 2024

## Abstract

In an era where data breaches are commonplace, the security of personal credentials is more critical than ever. FortiVault emerges as an innovative password manager designed to facilitate the secure storage and management of sensitive user passwords. Crafted using a harmony of contemporary technologies including Next.js 14, Prisma, Supabase, and OAuth 2.0 with

Google, FortiVault stands as a testament to advanced web development and cybersecurity principles. Through the employment of NextAuth for seamless authentication and PostgreSQL for resilient data persistence, the application guarantees both the integrity and accessibility of user data. This dissertation delineates the developmental journey of FortiVault, discussing its secure architecture, user experience, design strategies, and robust authentication mechanisms that collectively create a trustworthy digital safe for password storage. Additionally, it explores the challenges overcome in implementing a user-friendly interface without compromising on security, the integration of external services through Supabase, and the compliance with security standards to safeguard against cyber threats. FortiVault not only proposes a practical solution to individual password security but also contributes to the broader discussion surrounding personal data protection in the digital landscape.

Keywords: AES-256, Password, Authorization, Cybersecurity, User-friendly

Report's total word count: 10,951 excluding references and Appendix

## Acknowledgements

I would like to take the opportunity to express my sincere appreciation to the people who engaged in my journey through this project and my time at the University of Reading. First, I would like to extend my thanks towards my project supervisor, Dr Martin Lester for his help and guidance. I would also like to express my gratitude towards Dr Atta Badii, who continuously supported me throughout my time at the University, as both an Academic Mentor and a member of staff who always strived to make sure the students were challenged and engaged.

I would like to express a deep thanks to my colleagues, friends and family who have constantly supported me and pushed me to be the best that I could be, both academically and generally. Their continuous motivation and perseverance, alongside the examples that they have set throughout has been paramount to my achievements at the University. Special mentions to Max Bourne, Muath Khalifa, Rowan Avis, Joshua Roberts, and Alex Whitehead, who throughout our time at the University have spent countless hours and late nights beside me working on each coursework to ensure the best outcomes.

# Contents

Akatomisan Michaelangelo Ajuyah .....	1
Declaration .....	i
Abstract .....	i
Acknowledgements .....	ii
Chapter 1 .....	1
Introduction.....	1
1.1 Background .....	1
1.2 Problem statement .....	1
1.3 Questions and objectives .....	2
1.4 Justification for the Study.....	2
1.5 Scope of the Study .....	3
1.6 Organization of the report .....	3
Chapter 2 .....	4
Literature Review .....	4
2.1 Cybersecurity Landscape and Threats .....	4
2.2 Password Management Systems .....	4
2.2.1 What are Password Management Systems .....	4
2.2.2 Challenges in Password Management.....	5
2.2.3 Current Techniques in Password Management Systems.....	6
2.2.4 Current Technologies Utilized in Password Management Systems .....	7
2.2.5 Encryption Standards .....	7
2.2.6 Current Leading Password Management Systems in Cybersecurity Industry .....	8
2.3 The Rise of JavaScript Frameworks: Next.js and Modern Web Application Development .	9
2.4 Advanced Authentication with NextAuth: Secure and Efficient User Verification .....	9
2.5 Understanding Supabase in Real-Time Database Management .....	10
2.6 User Experience and Usability in Security Software .....	11
2.7 FortiVault: A Password Manager Leveraging AES-256 Encryption and Modern Web Technologies to Solve Current Issues in Digital Security .....	12
Methodology.....	13
3.1 AUTHENTICATION .....	13
3.1.1 Prisma Adapter Connection .....	13

3.1.2	Session Management .....	14
3.1.3	Credential-Based Authentication .....	14
3.1.4	OAuth 2.0 with GoogleProvider .....	15
3.2	PRISMA .....	15
3.3	CRUD .....	16
3.4	Password Generation.....	18
3.5	ENCRYPTION .....	19
3.6	EMAIL VERIFICATION(SMTP Server) .....	20
3.7	DEPLOYMENT .....	22
3.8	Summary.....	23
Results and Discussions.....		25
4.1	Application Features .....	25
4.1.1	HOMEPAGE.....	25
4.1.2	Signup Page .....	26
4.1.3	Sign-in Page .....	27
4.1.4	Forgot Password Page .....	28
4.1.5	GMAIL Signup/Sign-in .....	28
4.1.6	Password Page.....	30
4.2	Observed Results and User Feedback.....	31
4.3	Summary.....	32
Conclusions and Future Work .....		33
5.1	Conclusions .....	33
5.2	Future work .....	33
Reflection .....		35
References.....		37
Appendix 1- Link to Project Code .....		41
Appendix 2- Link to Web Application .....		41
Appendix 3 – User Feedback Form Results .....		42

# List of Figures

Figure 1: FortiVault Flowchart .....24

Figure 2: Homepage. ....25

Figure 3: Sign up page. ....26

Figure 4: Sign-in Page. ....27

Figure 5: Forgot Password. ....28

Figure 6: Gmail Signup/Sign-in. ....28

Figure 7: Email Verification.....29

Figure 8: Verification Email.....29

Figure 9: Confirmation Message. ....29

Figure 10: Password Page.....30

Figure 11: Password Details.....30

Figure 12: Password Table .....31

Figure 13: Decrypted Password Table .....31

Figure 14: Data from Database.....31

# List of Abbreviations

CRUD: Create, Read, Update, Delete

PMS: Password Management Systems

ORM: Object Relational Mapper

JSON: JavaScript Object Notation

JWT: JSON Web Token

AES: Advanced Encryption Standard

FIPS: Federal Information Processing Standards

NITS: National Institute of Standards and Technology

SMTP: Simple Mail Transfer Protocol



# Chapter 1

## Introduction

### 1.1 Background

In the digital era, cybersecurity is paramount, with the management and protection of passwords being a particularly vulnerable and critical aspect of online security. As cyber threats evolve, and data breaches become more sophisticated, traditional password management techniques no longer suffice. FortiVault emerges in response to this challenge, offering an advanced password management solution constructed with innovative web technologies, including Next.js 14, Prisma, PostgreSQL, Supabase, OAuth 2.0 with Google, and the implementation of NextAuth for secure authentication. Recognizing the inherent risks in password storage and retrieval, this dissertation seeks to illuminate the development and efficacy of FortiVault as a vault for personal credential security. With an increase in digital services requiring robust authentication methods, the implementation of secure and user-friendly password management systems has become crucial. FortiVault's innovative approach designs not only to secure credentials with stringent security measures but also to enhance user experience through seamless integration and intuitive operations. This research endeavours to contribute to the body of knowledge in cybersecurity by meticulously outlining the design, development, and deployment processes of FortiVault, demonstrating its potential to redefine the standards of password security in the modern web landscape.

### 1.2 Problem statement

Despite advancements in cybersecurity, password management remains a linchpin of vulnerability, with common storage solutions failing to meet the rising tide of sophisticated cyber threats. As incidents of data breaches and unauthorized access escalate, the inadequacy of conventional password managers in safeguarding sensitive information has become glaringly apparent. The problem that FortiVault seeks to address is the critical need for a more secure, dependable, and user-friendly password management system that can withstand the complexities of modern cyber threats while maintaining ease of user adoption and operation. Many current systems compromise either security for usability or vice versa, leaving users at a crossroads. This research identifies the security limitations of existing password management solutions, such as inadequate encryption standards, single points of failure, and suboptimal authentication methods, and posits the necessity of a novel approach embodied by the FortiVault project. The core problem is thus to conceive, develop, and validate a password management system that delivers on the vital twin objectives of fortified security and enhanced user experience in a digital landscape marked by escalating cyber risk scenarios.

## 1.3 Questions and objectives

### Research Questions

1. What are the primary vulnerabilities in current password management systems that contribute to user data breaches?
2. How can a password management system like FortiVault enhance security measures to mitigate these vulnerabilities?
3. In what ways can user experience be improved without compromising the security integrity of a password management solution?
4. What role does the integration of modern web technologies play in advancing the security and usability of password managers?

### Research Objectives

1. To identify and critically analyse the weaknesses and points of failure in conventional password management approaches.
2. To design and develop the FortiVault password management system utilizing a combination of Next.js 14, Prisma, PostgreSQL, Supabase, OAuth 2.0, and NextAuth to establish a new benchmark in security and usability.
3. To evaluate the effectiveness of FortiVault's security features against common cyberattack vectors and validate its robustness through rigorous testing protocols.
4. To assess user experience and adoption challenges of the FortiVault system by conducting usability testing and gather empirical user feedback.
5. To propose a framework for the continuous improvement of password management systems based on the findings and insights gained from the FortiVault development and evaluation process.

## 1.4 Justification for the Study

The research on the development and effectiveness of the FortiVault password management system is justified by the urgent and growing need for robust cybersecurity measures amidst an alarming uptick in cyber-attacks globally. Passwords remain a fundamental security feature for protecting personal and professional data, yet the methods for managing them are often outdated and prone to exploitation. This study is imperative for several reasons:

1. **Increasing Security Breaches:** With the rising number of security breaches involving password theft and misuse, there is a pressing demand for password managers that employ innovative technology to protect user data effectively.
2. **Advancements in Technology:** The integration of modern web technologies and authentication methods in password management systems, which FortiVault leverages, is a significant step forward that warrants investigation to determine their impact on security and user experience.

3. Usability vs. Security: Finding the balance between user convenience and stringent security is a perpetual challenge. This research aims to demonstrate that through intelligent design and technology integration, it is possible to achieve both without compromise, thereby encouraging wider adoption of secure practices.
4. Contribution to Knowledge Base: By documenting and analysing the design, development, and testing of FortiVault, this study will contribute valuable insights and frameworks to the body of knowledge in cybersecurity and technology development, potentially guiding future innovations in the field.
5. Practical Implications: Practical recommendations and advancements stemming from this research will have direct implications for both developers and users of password management systems, providing a blueprint for secure, user-friendly solutions that can be adapted and implemented across various platforms.
6. Policy and Standards: The outcomes of this study could inform policymakers and industry leaders in establishing more rigorous standards for password management software, driving an elevation in security protocols across digital platforms.

## 1.5 Scope of the Study

1. Design & Technology: Development of the FortiVault system using advanced web technologies like Next.js 14, Prisma, etc.
2. Security Emphasis: Focus on implementing innovative security features to protect against prevalent cyber threats.
3. User Experience: Evaluation of the system's usability to ensure a seamless interface that promotes secure password management practices.
4. Testing & Validation: Conducting security assessments and user testing to empirically validate FortiVault's performance.
5. Market Comparison: Limited comparative analysis with existing solutions to contextualize FortiVault's advancements.
6. Scope Limitations: Targeted user demographics and potential geographic boundaries may be defined to focus the research.
7. Compliance & Ethics: Adherence to data privacy standards and ethical research practices throughout the project.

## 1.6 Organization of the report

This report consists of six main sections. The first section, this section, introduces the project. The second section contains the research and investigation taken place, including a literature review. The third section focuses on the methodology of the solution to the project problems. The next section looks at the results of the methods used to create the project and a concise discussion on the results and feedback from users. The fifth part of the report summarises and concludes the project. The sixth part of the report is a reflection on the experience in doing this project.

# Chapter 2

## Literature Review

### 2.1 Cybersecurity Landscape and Threats

The digital age has transformed our lives, but it has also exposed us to constant barrage of cybersecurity threats. This ever-shifting landscape demands an initiative-taking approach to online security. Password Management Systems (PMS) like FortiVault can be valuable allies in this fight.

Cybercriminals are relentless in their pursuit of new attack methods. Phishing frauds, designed to trick users into revealing login credentials, are a constant threat. Zero-day exploits, targeting previously unknown vulnerabilities, can catch defenders off guard. Ransomware attacks have become a major concern, with hackers encrypting a victim's data and demanding a hefty ransom for decryption. These attacks can have devastating consequences for both businesses and individuals. [1]

The software supply chain itself is no longer a haven. Attackers are increasingly targeting widely used applications, aiming to gain access to a vast number of users and systems through a single point of entry. Artificial intelligence (AI), a powerful tool with immense potential, can also be weaponized. Cybercriminals may leverage AI to automate attacks, making them more efficient and difficult to detect. The ever-expanding attack surface, fuelled by the growth of the Internet of Things (IoT) and cloud computing, creates even more potential entry points for attackers to exploit. [2]

FortiVault, built with innovative technologies, has the potential to be a secure and user-friendly PMS solution. By utilizing strong encryption standards like AES-256, FortiVault can significantly reduce the risk associated with traditional password breaches. Even if attackers gain access to the PMS database, your passwords will remain scrambled and unreadable. Additionally, FortiVault can generate and store strong, unique passwords for each website you use, eliminating a major vulnerability exploited by attackers. Additionally, user awareness remains paramount, as even with FortiVault, users need to be cautious of phishing attempts and exercise good online security practices. By understanding the cybersecurity landscape and utilizing tools like FortiVault responsibly, we can navigate this complex environment and protect ourselves in the digital world.

### 2.2 Password Management Systems

#### 2.2.1 What are Password Management Systems

Password Management refers to the process of storing, organizing, and managing user credentials and passwords in a secure manner. This concept is fundamental in digital security, as it ensures that access to digital assets and services is safeguarded against unauthorized use. Password

management systems play a pivotal role in maintaining the integrity and confidentiality of digital information by providing a structured approach to generating, storing, and retrieving complex and unique passwords for various accounts. [3]

The risks associated with poor password practices cannot be overstated. Common issues include the use of weak passwords, reusing the same password across multiple accounts, and storing passwords insecurely, such as writing them down on physical media or in unencrypted digital files. Such practices leave individuals and organizations vulnerable to a range of cyber threats, including but not limited to, phishing attacks, brute force attacks, and credential stuffing. These attacks can lead to unauthorized access to sensitive information, identity theft, financial loss, and damage to reputation. [4]

Password managers significantly mitigate these risks by providing a secure and efficient way to manage passwords. They enable users to generate strong, unique passwords for each account and store them in an encrypted vault that is accessible only through a master password. This not only enhances security by reducing the likelihood of password-related breaches but also simplifies the user experience by eliminating the need to remember multiple complex passwords. Additionally, many password managers offer features like automatic form filling and password change reminders, further bolstering security practices.

### 2.2.2 Challenges in Password Management

Though password managers are highly effective in securing passwords there are still some challenges that can compromise the security of digital assets. Key issues include maintaining password complexity, avoiding password reuse across different sites, and guarding against vulnerabilities such as brute force attacks and phishing. Each of these challenges contributes to the overall difficulty of securing digital identities and information.

Maintaining password complexity is crucial as simple or common passwords can be guessed easily or cracked by attackers. The National Institute of Standards and Technology (NIST) guidelines recommend creating passwords that are long, complex, and use a mix of characters, numbers, and symbols. However, the challenge lies in the user's ability to remember and manage these complex passwords without resorting to insecure practices like writing them down or reusing them across multiple accounts. [5]

Password reuse is another significant challenge. Many users tend to reuse the same password across different sites for convenience. This practice poses a substantial risk; if one account is compromised, all other accounts using the same password are at risk. This issue is compounded by the frequent occurrence of data breaches, where large volumes of username and password combinations are exposed, making it easier for attackers to gain unauthorized access through credential stuffing attacks. [6]

Brute force attacks present a direct threat to password security. In these attacks, attackers use automated tools to systematically try many password combinations to gain unauthorized access. While complexity can mitigate the risk of brute force attacks, it does not eliminate it, especially if the attacker can leverage more powerful computational resources or exploit other vulnerabilities in the system.

Phishing attacks exploit human factors rather than technical vulnerabilities. Attackers deceive users into revealing their passwords by masquerading as trustworthy entities through emails or fake websites. Despite the technical sophistication of modern password management systems, they can be rendered ineffective if users are tricked into divulging their credentials.

### 2.2.3 Current Techniques in Password Management Systems

Password management techniques have evolved significantly to address the challenges associated with securing digital identities and information. These techniques employ a range of methods from cryptographic measures to biometric verification, each with its strengths and purposes in enhancing security. Below are some of the current techniques employed in password management:

1. **Cryptographic Hashes:** Cryptographic hashing is a fundamental technique used to secure passwords. Instead of storing passwords in plaintext, systems store a hash value produced by a cryptographic hash function. These functions are designed to be one-way, meaning that it is computationally infeasible to reverse the hash back to the original password. This protects stored passwords against theft. Moreover, salting, which involves adding a unique random value to each password before hashing, further secures passwords against rainbow table attacks [7].
2. **Multi-factor Authentication (MFA):** MFA enhances security by requiring two or more verification factors to gain access to a resource, significantly reducing the risk of unauthorized access. These factors can include something the user knows (like a password or PIN), something the user has (like a smartphone or security token), and something the user is (like a fingerprint or other biometric data). MFA provides an additional security layer, making it more challenging for attackers to gain access even if one factor, such as a password, is compromised [7].
3. **Biometrics:** Biometric authentication uses unique biological characteristics of individuals, such as fingerprints, facial recognition, iris scans, or voice recognition, as a security measure. Biometrics can provide a more user-friendly and secure alternative to traditional passwords, reducing the reliance on memorized passwords and the vulnerabilities associated with them [8].
4. **Password Rotation Policies:** These policies mandate the regular change of passwords within a set period. The idea is to limit the time window an attacker must exploit a stolen password. However, recent guidelines from organizations like NIST suggest that frequent password changes may lead to weaker security, as users might choose simpler passwords or make minor alterations to existing passwords, making them easier to guess.

#### 2.2.4 Current Technologies Utilized in Password Management Systems

Modern password managers employ advanced technologies to ensure the security and accessibility of stored passwords while maintaining user privacy. These technologies include client-side encryption, cloud syncing, and zero-knowledge proofs, each playing a crucial role in safeguarding user data from unauthorized access, including from the service providers themselves.

1. **Client-Side Encryption:** In this approach, encryption and decryption of data occur exclusively on the user's device, using a key derived from the user's master password. This means that the plaintext versions of the passwords never leave the user's device, and even the encrypted data stored on the service provider's servers cannot be decrypted by the provider, ensuring privacy and security. Client-side encryption leverages strong cryptographic algorithms, such as AES (Advanced Encryption Standard), to protect the confidentiality and integrity of the data [9].
2. **Cloud Syncing:** Cloud syncing allows users to access their password vaults from any device, ensuring convenience and flexibility. The encrypted password database is stored in the cloud, and synchronized across the user's devices. Despite being stored on remote servers, the data remains encrypted with the user's key, which is not stored on the cloud, thus maintaining security. This feature relies on secure communication protocols, such as TLS (Transport Layer Security), to protect the data during transmission [10].
3. **Zero-Knowledge Proofs:** Zero-knowledge proof is a cryptographic principle that allows one party to prove to another party that a statement is true, without revealing any information beyond the validity of the statement itself. In the context of password managers, this means that the service can verify the user's identity and grant access without knowing the user's master password or the contents of their vault. This ensures that the service provider cannot access the plaintext versions of stored passwords, even if they wanted to.

#### 2.2.5 Encryption Standards

Encryption standards play a pivotal role in password management by providing a secure framework for encoding sensitive information, thus safeguarding it from unauthorized access. Among these, the Advanced Encryption Standard (AES) with a 256-bit key, known as AES-256, stands out for its balance of performance and security, making it a preferred choice for encrypting sensitive data, including passwords stored by password managers. AES-256 is part of the AES family, which was established by the U.S. National Institute of Standards and Technology (NIST) in 2001. It is designed to protect electronic data and has since become a benchmark in encryption, used widely across various sectors, including government, finance, and healthcare, for secure data transmission and storage [11].

The strength of AES-256 lies in its key size of 256 bits, which offers a higher level of security compared to its predecessors, AES-128 and AES-192. The larger key size makes AES-256 exponentially harder to crack through brute-force attacks, where an attacker attempts to guess the

key by trying all possible combinations. It is estimated that breaking AES-256 encryption would take billions of years with current computing technology, making it an extremely secure option for protecting sensitive information.

Moreover, AES-256 is favoured for its efficiency and performance. Despite the increased security that comes with a larger key size, AES-256 remains efficient in terms of computational resources, making it suitable for both high-performance server environments and devices with limited processing power, such as mobile phones and embedded systems. This efficiency does not compromise security, ensuring that data remains protected without significantly impacting system performance [12].

#### 2.2.6 Current Leading Password Management Systems in Cybersecurity Industry

Current password managers utilize a range of sophisticated security measures to protect user data, with industry leaders such as LastPass, 1Password, and Bitwarden setting standards in terms of encryption, user authentication, and innovative password management techniques.

**LastPass** employs AES-256-bit encryption (Zero-knowledge encryption) with PBKDF2 SHA-256 and salted hashes to ensure that all user data is securely encrypted before it even leaves the device. This approach, coupled with additional security measures such as multi-factor authentication (MFA) and emergency access, ensures a high level of security for user credentials [13].

**1Password** also uses AES-256 encryption for data protection and adds its own unique security features, such as 'Secret Key' technology, which adds an extra layer of encryption that only the user has access to, enhancing the security of the encrypted data stored on 1Password servers. Furthermore, 1Password's Watchtower service alerts users to password breaches and reused passwords, encouraging better password hygiene.

**Bitwarden** is an open-source password manager that prioritizes transparency and trust. It employs AES-256-bit encryption along with salted hashing and PBKDF2 SHA-256 for securing passwords. Bitwarden's open-source model allows for regular audits by the security community, ensuring that its encryption and security practices remain robust.

An innovative approach in the realm of password management is the concept of generating and managing passwords without storing them, as explored in the concept of **Passlmg**. Passlmg proposes a method where passwords are generated based on a combination of user-selected images and secret tokens. This technique eliminates the need for storing passwords in a traditional database, significantly reducing the risk of password leaks from database breaches. Such approaches represent a novel direction in password management, focusing on usability and security [14].



## 2.3 The Rise of JavaScript Frameworks: Next.js and Modern Web Application Development

JavaScript frameworks have become essential in web development, driving the evolution of rich, interactive web applications. Next.js has stood out as a particularly progressive framework. It capitalizes on server-side rendering (SSR) and static site generation (SSG) to improve both the performance and user experience of web applications, as evidenced by its successful application in the design and implementation of an online legal forum to handle UGC cases [15].

With these techniques, developers can create fast, SEO-friendly websites that load quickly for users. One of Next.js key strengths is its automatic code splitting. This feature loads only the necessary JavaScript for each page, optimizing load times and reducing the amount of data transferred over the network, akin to the streamlined profile search in academic databases [16].

The framework further supports asynchronous data fetching, an essential feature for contemporary web applications that rely on dynamic, real-time data, like the real-time content violation detection seen in broadcasting videos [17]. The focus on developer experience within Next.js cannot be overstated. Features such as Hot Module Replacement (HMR) and zero-config setup accelerate development workflows and reduce time-to-market for new products. Combined with its compatibility with other tools and libraries in the JavaScript ecosystem, Next.js fosters an environment that is both productive and flexible. An often-overlooked benefit of frameworks like Next.js is the positive impact on the security of web applications. By pre-rendering pages and fetching data server-side, the framework limits the exposure of data and logic to client-side vulnerabilities, including various forms of XSS(Cross-site scripting) attacks. Next.js architecture also encourages best practices such as the use of environment variables for sensitive configurations, further contributing to the security of web applications.

Next.js demonstrates a balance between performance, developer experience, and security, making it a compelling choice for modern web applications that require speed, optimization, and scalability.

## 2.4 Advanced Authentication with NextAuth: Secure and Efficient User Verification

In the realm of modern web development, ensuring both security and efficiency in user authentication is paramount. NextAuth.js stands out as a stellar solution for developers using the Next.js framework, offering robust capabilities in this respect. Later in this report, we will explore the intricacies of NextAuth.js, a library specifically tailored for Next.js, which facilitates advanced authentication mechanisms with a particular focus on user verification, and how it compares to other authentication tokens in terms of performance and security, such as JWT(Json Web Tokens) and PASETO(Platform Agnostic Security Tokens) [18].

We begin by introducing the core principles and setup procedures of NextAuth.js. Leveraging a versatile API, the implementation phase involves configuring authentication providers and options to match the unique needs of web applications. NextAuth.js supports a range of sign-in methods, including OAuth providers like Google and Twitter, as well as email and passwordless credentials. The importance of exploring advanced secure authentication methods has been underscored by research into lattice-based post-quantum signature for JWT authentication, which presents a case study on this front [19].

As we transition into a discussion on security, the finer points of JSON Web Tokens (JWT) and session management protocols integral to NextAuth.js are elucidated. The library manages user sessions securely by default, handling tokens and session refresh mechanisms efficiently. This reflects the practical application of JWT in RESTful web services to boost security and information system efficiency, as investigated in the Unklab Information System [20].

Later in this report, we will further dissect the features of NextAuth.js that contribute to its safe and efficient user verification processes. This includes detailed insights into CSRF protection, encryption of user data, and the strategies employed to safeguard against common security threats. By covering these aspects, we aim to provide a thorough understanding of how NextAuth.js provides a secure, developer-friendly platform for managing authentication across modern web applications.

Through this exploration, it becomes evident that NextAuth.js is more than a mere authentication library; it is a comprehensive security solution that streamlines user verification processes for Next.js developers. The subsequent sections will delve deeper into practical applications and demonstrate the successful integration of NextAuth.js in FortiVault.

## 2.5 Understanding Supabase in Real-Time Database Management

As real-time data continues to be an integral component of interactive applications, developers are frequently seeking robust and scalable database management solutions. Supabase, an opensource alternative to Firebase, is emerging as a powerful tool for developers requiring real-time functionalities, drawing on design concepts like those applied in distributed real-time database systems for massive data management [21].

Supabase capitalizes on PostgreSQL, a time-tested and sophisticated database system, to offer a suite of tools including a real-time database, authentication, instant APIs, and storage solutions. The premise underpinning Supabase's real-time capabilities can be traced to prior applications of real-time databases, such as the hotspot monitoring systems in transformer substations [22].

Real-time capabilities in Supabase are facilitated through PostgreSQL's built-in 'LISTEN' and 'NOTIFY' functionality. This enables applications to subscribe to database changes and receive updates instantaneously, showcasing a critical feature for dynamic and interactive user

experiences. This approach is reflective of real-time database architectures discussed in the realm of cloud environments for distributed systems, where immediacy and data integrity are key [23].

Security is also a priority in Supabase's real-time management. Fine-grained access control can be implemented directly into the database schema, allowing developers to specify permission levels and restrict data access in real-time streams, ensuring that users only receive the data they are authorized to see.

Supabase integrates with modern frontend frameworks and provides a seamless developer experience through client libraries. Supabase's client libraries, available for popular frameworks like React and Vue.js, mean that developers can easily synchronize UI components with database changes, ensuring a smooth, real-time interaction for end users.

## 2.6 User Experience and Usability in Security Software

The aim of FortiVault is to simplify the password management process while keeping your data secure. However, a secure system is only effective if people actively use it. That's where user experience (UX) comes in. A complex and frustrating interface can deter users from adopting secure password practices. Here are some of the features which need to be prioritized to make user experience in management more effective:

1. **Seamless Integration:** Imagine accessing your passwords effortlessly across different devices! A good PMS integrates seamlessly with popular browsers and mobile operating systems. This allows you to access your passwords with minimal effort, no more scrambling to remember which device you saved a specific password on.
2. **Multi-Layered Security with Convenience:** Security shouldn't come at the expense of convenience. A good PMS employs robust security measures to safeguard your passwords, it should also offer various login options beyond traditional passwords. This ensures strong authentication without a lengthy login procedure.
3. **Intuitive Interface and Automatic Assistance:** Imagine a system where finding and managing passwords feels effortless! A good PMS boasts a clean, intuitive user interface with clear labels and well-organized menus. Furthermore, it can streamline the process by automatically generating strong, unique passwords and storing them securely. This ensures you have robust passwords without the burden of memorizing them all. This will also reduce the temptation to reuse passwords across different platforms.

By prioritizing these UX aspects, FortiVault aims to create a secure password management system that users not only trust but also find enjoyable and convenient to use. This wider adoption of secure password management practices strengthens overall online security for everyone.

## 2.7 FortiVault: A Password Manager Leveraging AES-256 Encryption and Modern Web Technologies to Solve Current Issues in Digital Security

In the era of digital information, password management is essential in preserving data privacy and security. FortiVault emerges as a pioneering solution, embodying the integration of high-level encryption standards with innovative web technologies. Utilizing AES-256 encryption ensures the cryptographic resilience of stored passwords against modern threats, highlighting studies that compare the performance of AES-256 with other encryption combinations to verify its robustness [24].

Next.js serves as the backbone of FortiVault's architecture, delivering both server-side rendering for enhanced SEO and static site generation for improved loading times. The deployment of AES256 within such architectures can borrow ideas from implementations like the IP core integration with AXI interface for AES256 and TDES algorithms [25].

Supabase plays a pivotal role in FortiVault by offering a real-time database and authentication services comparable to Google's Firebase but with a preference for open-source flexibility. The secure synchronization between client and server is cornerstone, resembling techniques studied in domains like blockchain, where encryption is paramount.

The integration of NextAuth.js offers a straightforward yet secure authentication mechanism. It simplifies user sign-ins through various providers while retaining security measures that align with OAuth 2.0 standards, an essential factor considering the sensitivity of the information handled by password managers.

As mentioned before, FortiVault considers the different features which make user experience with password management systems optimal making it a viable solution for anyone, anywhere in the world.

Collectively, FortiVault stands out as an innovative password manager. It cleverly combines the cryptographic strength of AES-256 encryption with the agility of modern web technologies like Next.js, Supabase, and NextAuth.js. The result is a platform capable of addressing the sophisticated demands of cybersecurity in the contemporary era.

## Chapter 3

# Methodology

The main aim of this project is to create a password manager web application using advanced technologies which prioritises security of the system without sacrificing user experience. To achieve this objective, we must create a list of features which the application must have. These features are a mix of features in existing PMS and from target user research through questionnaires and interviews. These features are:

- Users must be able to sign-up/sign-in and manage sessions.
- Users must be able to create, read, update, and delete passwords.
- Users must be able to generate secure passwords.
- The app should have a feature to encrypt passwords.
- Users must be able to access the app from anywhere with any device.
- The system should be able to verify users before they can make use of the app.
- A secure database to store and manage user data.

For this project, we will use the agile development methods. This development is mainly based on iteration and increment. Furthermore, having looked at the various features which must be in the app to achieve the project's objectives, we will look at the following: Authentication, CRUD(Create, Read, Update, and Delete) Operations, Password Generation, Encryption, PRISMA, Email Verification and finally Deployment.

### 3.1 AUTHENTICATION

This feature is brought to life in the code through implementation details laid out in the NextAuth configuration object. Here are some essential snippets:

#### 3.1.1 Prisma Adapter Connection

```
1. adapter: PrismaAdapter(prisma)
```

This line of code is used to integrate Prisma with the NextAuth.js library. It is used to tell NextAuth.js to use Prisma as the database adapter, enabling it to interact with the database using Prisma's ORM capabilities. The **prisma** parameter passed into **PrismaAdapter** is an instance of the Prisma Client, which is configured to connect to the database according to the **schema.prisma** file. This line enables the application to handle authentication (such as signing in

users, managing sessions, etc.) using the database managed by Prisma, ensuring that user data and session information are stored and managed effectively. This line effectively bridges our database with the authentication flows.

### 3.1.2 Session Management

```
1.   session: {  
2.     strategy: 'jwt',  
3.   },  
4.   secret: process.env.NEXTAUTH_SECRET,
```

This line tells NextAuth to use JWTs for session management. When a user logs in, a JWT will be created and used to manage their session. This JWT contains encoded information about the user and can be used to verify the user's identity on subsequent requests. JWTs are very useful for ensuring lightweight and scalable sessions. The secret used for signing is stored securely as an environment variable.

### 3.1.3 Credential-Based Authentication

```
1.   CredentialsProvider({  
2.     async authorize(credentials) {  
3.       const user = await prisma.user.findUnique({  
4.         where: { email: credentials.email },  
5.       });  
6.       if (user && await compareHash(credentials.password, user.password)) {  
7.         return user;  
8.       }  
9.       return null;  
10.    },  
11.  }),
```

Here, we define the logic to authenticate a user based on their email and hashed password, deferring to robust hash comparison for password verification. We implement a custom authentication provider using NextAuth's **CredentialsProvider** in our application. This authentication method allows users to sign in using their email and password.

The core of this setup lies in the **authorize** function, which is an asynchronous operation. This function is tasked with verifying the user's credentials against the information stored in the database. It begins by extracting the user's email from the provided credentials and uses Prisma to query the database for a user with a matching email address. If a user with the specified email is found, the next step involves verifying the password. Passwords should never be stored in plain text in the database for security reasons; instead, they are stored as hashed values. The **compareHash** function is used to compare the hash of the provided password with the hash

stored in the database. This comparison is crucial for security and is also performed asynchronously, requiring the use of the **await** keyword to ensure that the operation completes before proceeding.

Should both the email match a database entry and the password comparison be successful, the **authorize** function returns the user object, effectively authenticating the user. If either the email does not match any user in the database or the password is incorrect, the function returns **null**, indicating that authentication has failed.

### 3.1.4 OAuth 2.0 with GoogleProvider

```
1. GoogleProvider({  
2.   clientId: process.env.GOOGLE_CLIENT_ID as string,  
3.   clientSecret: process.env.GOOGLE_CLIENT_SECRET as string,  
4. },
```

This setup leverages the OAuth 2.0 protocol, allowing users to log in with their Google accounts, thereby streamlining the authentication process by utilizing existing credentials. At the heart of this configuration are two critical pieces of information: the **clientId** and the **clientSecret**, both of which are essential for the OAuth handshake between the application and Google's authentication servers.

The **clientId** serves as a public identifier for the application, a unique string that Google's OAuth server uses to recognize the application requesting authentication. This ID is safely stored in an environment variable named **GOOGLE\_CLIENT\_ID**, ensuring that it remains external to the application's source code, thereby enhancing security. Similarly, the **clientSecret** is a confidential key that, akin to a password, is known only to the application and Google's server, enabling secure communication. This secret is also stored as an environment variable, named **GOOGLE\_CLIENT\_SECRET**, safeguarding it from potential exposure.

Incorporating these credentials into the **GoogleProvider** function call, FortiVault is endowed with the capability to initiate a secure authentication flow with Google. When a user opts to sign in using Google, they are redirected to Google's authentication page, where they grant the application permission to access their information. Upon successful authentication, Google redirects the user back to the application, along with an authentication token that the application can use to establish a session for the user.

## 3.2 PRISMA

In the domain of secure password storage and retrieval, the **PasswordManager** serves as a quintessential component of the FortiVault system. This component is designed to interface between the end-users—an entity that necessitates high security—and the underlying database

that securely persists user credentials. It acts as an intermediary, providing a robust abstraction over the direct database transactions, effectively encapsulating all password-related operations.

```
1. model PasswordManager {
2.   id String @id @default(cuid())
3.   userId String
4.   user User @relation(fields: [userId], references: [id], onDelete: Cascade)
5.   serviceName String
6.   serviceUrl String?
7.   password String
8. }
```

The **PasswordManager** model is critical for ensuring the integrity of user data. It is linked to the **User** model via a relation, which underscores the user-specific nature of the stored data, representing a one-to-many relationship that is vital for supporting individual user access to their respective password entries.

### 3.3 CRUD

FortiVault will prioritize secure and user-centric management of passwords. This section looks at FortiVault's methodology for Create, Read, Update, and Delete (CRUD) operations, ensuring data integrity and user privacy.

As mentioned in chapter 3.1 FortiVault applies user authentication for security. The application leverages established frameworks like NextAuth to verify user credentials before CRUD actions. This initial step prevents unauthorized access and safeguards sensitive password data.

The application enforces granular data access control through user-specific filtering. When a user interacts with the system, their authenticated session provides a unique user ID. This ID serves as a key, unlocking only the password records associated with that user.

```
1.   const prisma = new PrismaClient()
2.   const model = prisma.passwordManager
3.   export default async function handler(
4.     req: NextApiRequest,
5.     res: NextApiResponse
6.   ) {
7.     const session = await getServerSession(req, res, authOptions)
8.     const filterOptions = { userId: session?.user?.id }
9.     if (filterOptions.userId == null || filterOptions.userId == undefined) {
10.      return res.status(401).json({ error: 'User not authenticated' })
11.    }
```



Here, the code checks for a user session using **getServerSession**. This step ensures that the operations are secure and only accessible to authenticated users, aligning with best practices for handling sensitive information like passwords.

The app utilizes specific HTTP methods for each CRUD operation, promoting clarity and maintainability.

- **GET:** Retrieving Password Data

```
1.   if (req.method === 'GET') {
2.     const objId = req.query.id as string
3.     if (objId) {
4.       return getById(objId, res, model, filterOptions)
5.     } else {
6.       return getAll(req, res, model, filterOptions)
7.     }
8.   }
```

For the Read operation, the code distinguishes between fetching a single item and retrieving all items. If an **id** is provided in the request query, it fetches a single item using the **getById** function; otherwise, it retrieves all items associated with the authenticated user using the **getAll** function. This is achieved by passing **filterOptions** to ensure that users can only access their own data.

- **POST:** Creating a New Password

```
1.   else if (req.method === 'POST') {
2.     req.body.userId = filterOptions.userId
3.     return create(req, res, model)   }
```

The Create operation is triggered when the HTTP method is **POST**. The request body, expected to contain new data, is augmented with the user's **userId** before being passed to the **create** function, ensuring the new item is associated with the correct user.

- **PUT:** Updating Existing Passwords

```
1. else if (req.method === 'PUT') {
2. return update(req, res, model, filterOptions) }
```

For the Update operation, the code handles **PUT** requests. It uses the **update** function to modify an existing item in the database, in this case an existing password, with **filterOptions** ensuring that only items belonging to the authenticated user can be updated.

- **DELETE:** Removing Passwords

```
1. else if (req.method === 'DELETE') {
2. return remove(req, res, model, filterOptions) }
```

The Delete operation corresponds to **DELETE** requests, using the **remove** function to delete an item from the database, again governed by **filterOptions** to restrict the operation to the user's own items.

## Error Handling: Guiding the User

FortiVault incorporates robust error handling mechanisms to provide a seamless user experience. In cases of unauthorized access attempts, unsupported HTTP methods, or other issues, informative error messages are returned. These messages, such as "User not authenticated" or "Method not allowed," help users understand the problem and take corrective actions.

### 3.4 Password Generation

One of the main features which is to be implemented into the system is the ability to generate strong passwords. As mentioned before, this will help users to create strong passwords and reduce the risk of user repeating/reusing passwords for multiple services. This is achieved through the code snippet below:

```
1.   function generateStrongPassword(length: number): string {
2.     const uppercaseChars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
3.     const lowercaseChars = 'abcdefghijklmnopqrstuvwxyz'
4.     const numberChars = '0123456789'
5.     const specialChars = '!@#%&*()-_+= '
6.
7.     const allChars =
8.       uppercaseChars + lowercaseChars + numberChars + specialChars
9.
10.    let password = ''
11.    for (let i = 0; i < length; i++) {
12.      const randomIndex = Math.floor(Math.random() * allChars.length)
13.      password += allChars[randomIndex]
14.    }
15.    return password
16.  }
17.
18.
19.  const generatePasswordClick = () => {
20.    formik.setFieldValue('password', generateStrongPassword(12))
21.  }
```

The above code snippet defines a function **generateStrongPassword** that creates a strong, random password, and a method **generatePasswordClick** that utilizes this function to set a password in a form field. The **generateStrongPassword** function takes a single parameter, **length**, which specifies the desired length of the password to be generated. It constructs the password by randomly selecting characters from a predefined set that includes uppercase letters (**ABCDEFGHIJKLMNOPQRSTUVWXYZ**), lowercase letters (**abcdefghijklmnopqrstuvwxyz**), numbers (**0123456789**), and special characters (**!@#%&\*()-\_+=**). These characters are concatenated to form a string **allChars**, which serves as the pool from which the function randomly selects characters to compose the password.

The password generation process within the **generateStrongPassword** function involves a loop that runs for the number of times specified by the **length** parameter. In each iteration, the function generates a random index using **Math.floor(Math.random() \* allChars.length)**, which ensures that the index is an integer within the bounds of **allChars** string length. It then appends the character at this random index from **allChars** to the **password** string. This process repeats until the loop has executed **length** times, at which point the **password** string contains a randomly generated password of the desired length.

The **generatePasswordClick** method is tied to a button in the user interface, where clicking the button calls this method. When invoked, it calls the **generateStrongPassword** function with a fixed length of 12 characters and sets this value as the new password in a form field. This is achieved through **formik.setFieldValue('password', generateStrongPassword(12))**, where **formik** is an instance of Formik, a popular library for handling forms in React applications. This line updates the **password** field in the form managed by Formik with the newly generated strong password, ready for use or submission.

Overall, this code offers a practical solution for generating strong, complex passwords in applications, enhancing security by encouraging the use of passwords that are difficult to guess or crack.

## 3.5 ENCRYPTION

During the requirement analysis phase of this project, it was clear that the most important feature of the application will be the encryption of passwords. This section looks at the architecture blueprint which includes a secure data handling strategy using AES encryption as exemplified by the CryptoJS library functions. CryptoJS is a growing collection of standard and secure cryptographic algorithms implemented in JavaScript using best practices and patterns. They are fast, and they have a consistent and simple interface [26]. The following code snippet exemplifies the implementation strategy:

```
1. import * as CryptoJS from 'crypto-js'
2. export function encryptAES(inputString: string, passphrase: string): string {
3.   const encrypted = CryptoJS.AES.encrypt(inputString, passphrase).toString()
4.   return encrypted
5. }
6. export function decryptAES(
7.   encryptedString: string,
8.   passphrase: string
9. ): string {
10.   const decrypted = CryptoJS.AES.decrypt(encryptedString, passphrase).toString(
11.     CryptoJS.enc.Utf8
12.   )
13.   return decrypted
14. }
```

The **encryptAES** function is tasked with encrypting a given input string using a specified passphrase. It begins by calling the **encrypt** method of **CryptoJS.AES**, passing in the input string and the passphrase. The encryption process transforms the plain text input string into an encrypted form

that is unreadable without the correct passphrase. This encrypted data is then converted to a string format and returned by the function, ready to be stored or transmitted securely.

Conversely, the **decryptAES** function performs the opposite operation. It takes an encrypted string and the same passphrase used for encryption. The function employs the **decrypt** method of **CryptoJS.AES**, feeding it the encrypted string and the passphrase. The decryption process attempts to revert the encrypted string back to its original plain text form. It's crucial that the passphrase used for decryption matches the one used for encryption, as any discrepancy would result in a failure to decrypt correctly. The decrypted data is then converted from its raw binary form to a UTF-8 string, which represents the original input string before encryption.

These functions encapsulate the core operations of AES encryption and decryption, providing a straightforward interface for securing data. By using a passphrase, these functions offer a method of symmetric encryption, where the same key is used for both encrypting and decrypting data. For the sake of this project, the passphrase is chosen by the user, and this can be anything such as the passcode to their phones, so it is imperative for the user to remember this passcode. The encrypted password is what will be stored on the database ensuring that even if there is an authorized access into the database or any information from the database is stolen, sensitive data such as passwords will be encrypted and hence of no use to anyone without the passkey, which is the user. Even with current brute-force method it would take millions of years to break an AES encrypted password making it virtually uncrackable [27].

### 3.6 EMAIL VERIFICATION(SMTP Server)

The purpose of integrating an email service is two-fold: to verify the identity of a new user via their email address and to enable secure password recovery for users who have forgotten their login credentials. The email service was selected based on its proven reliability, ease of implementation, robust security features, and compliance with email delivery standards. These criteria are crucial to minimizing the risk of service interruption and ensuring that verification and password reset emails are delivered promptly and securely. This section includes setting up a secure SMTP connection between FortiVault and the email service which requires configuration of TLS(Transport Layer Security) encryption to safeguard email content. The process is as follows:

- Establish an SMTP connection with the email service provider using the application's credentials.
- Configure email templates for verification and password reset emails, which include user specific tokens for security.
- Implement a rate-limiting feature to prevent abuse of the email service, enhancing security and compliance.
- Conduct exhaustive testing to ensure that automatic emails are triggered by the appropriate user actions and received without delay.

Now, when a new user signs up for FortiVault, they are required to undergo an email verification process which is as follows:

- The user enters their email address during registration.
- FortiVault triggers an automated email containing a unique, time-sensitive verification link.
- The user receives the email and clicks the link to confirm their email address.
- FortiVault verifies the link and activates the user's account upon successful confirmation.

The verification link contains a secure, user-specific token that expires within 24 hours for security purposes.

Users will also be able to reset passwords used to log into the application. The process for this function is as follows:

- The user requests a password reset by entering their email address in the provided form.
- FortiVault sends a password reset email to the specified address with a unique, time sensitive reset link or code.
- The user clicks the link or enters the code on the site to access the password reset page.
- After entering and confirming the new password, FortiVault updates the user's credentials securely.

Both the verification link and the password reset link include measures to prevent misuse, such as URL encoding and token expiration. Integration of the email service within FortiVault involves the use of Node.js with the Nodemailer module, a commonly used library for sending emails from Node.js applications. The configuration of Nodemailer to work with the selected email service (Gmail in this case) is illustrated through the following code snippet:

```
1. const nodemailer = require('nodemailer');
2. // Setup Nodemailer transporter
3. const transporter = nodemailer.createTransport({
4.   service: 'Gmail',
5.   host: 'smtp.gmail.com',
6.   port: 465,
7.   secure: true, // use SSL
8.   auth: {
9.     user: process.env.GOOGLE_MAIL_USER, // Gmail user from environment variable
10.    pass: process.env.GOOGLE_MAIL_PASS, // Gmail password from environment variable
11.  },
12. });
```

This code establishes a secure SMTP connection to the Gmail service, using credentials stored in environment variables for enhanced security. The process environment variables **process.env.GOOGLE\_MAIL\_USER** and **process.env.GOOGLE\_MAIL\_PASS** ensure that sensitive

data such as the application's Gmail username and password are not hard coded into the source code, further securing the application.

Continuing with the implementation, when a password reset request is made by the user, the following code demonstrates how FortiVault prepares an email with a reset link and sends it via the configured transporter:

```
1. // Email message options
2. const mailOptions = {
3.   from: process.env.GOOGLE_MAIL_USER, // Sender address
4.   to: user.email, // Recipient's email, acquired from user subject: 'FortiVault: Password
   reset', // Email subject line
5.   text: `We received a password reset request for your account. To reset your password, just
   click the link below: ${resetPasswordLink}`,
6.   html: `

We received a password reset request for your account. To reset your password,
   just click the link below:</p><a href="${resetPasswordLink}" target="_blank">Reset
   Password</a>`,
7. };
8. // Send email
9. return transporter.sendMail(mailOptions);


```

## 3.7 DEPLOYMENT

For the deployment of the app Vercel has been chosen due to its ease of use, scalability, and outof-the-box support for Node.js applications. It also provides a seamless development-toproduction workflow, with features for continuous deployment directly from version control systems like Git. The deployment process on Vercel is detailed below:

- The deployment process begins with pushing the latest code changes to the connected Git repository.
- Vercel automatically detects the updates and triggers the build process using the configurations specified in the package.json file.
- Environment variables are set within the Vercel dashboard to securely store sensitive information, such as database credentials and API keys.
- Post-build, Vercel serves the FortiVault application via a global Content Delivery Network (CDN) for enhanced performance.

For Backend Services we will use Supabase. Supabase, an open-source Firebase alternative, provides backend services, including database, authentication, and storage. It allows the application to utilize a PostgreSQL database without managing servers. The process of integrating our application with Supabase is as follows:

- Setting up the database schema and authentication services in the Supabase console.
- Supabase's API keys are securely injected as environment variables within the Vercel deployment settings to connect the FortiVault application with the Supabase backend.
- Real-time data is enabled to synchronize data across client devices effectively.

Google's OAuth 2.0 is implemented as the primary authentication method due to its widespread acceptance, security, and ease of use for end-users. The steps for configuring Google OAuth 2.0 for this project are stated below:

- The Google Developer Console is used to configure a new project, with FortiVault registered as an OAuth 2.0 client.
- Authorized redirect URIs are set, enabling successful authentication flows and callbacks.
- Client ID and Client Secret, provided by Google, are stored as environment variables on Vercel, ensuring they remain secure.

Furthermore, the following steps have been taken as further security measures for the deployment phase:

- The deployed FortiVault application uses HTTPS to encrypt and secure traffic between the user and the server.
- Cross-Origin Resource Sharing (CORS) policies are established to prevent unauthorized domains from interacting with the backend.
- Regular audits are performed through Vercel's integrated monitoring tools to check for any deployment vulnerabilities.

### 3.8 Summary

In FortiVault's comprehensive security layering, we employ an advanced methodology that intertwines user authentication with meticulous password management, leveraging AES encryption to fortify the sanctity of user credentials.

At the commencement of the user's journey within FortiVault, secure authentication takes precedence. We implement a meticulous credential storage protocol where user passwords are encrypted using AES-256, recognized for its robustness against cryptographic attacks. Utilizing a secret key, the user's password is transformed into an indecipherable string of characters, ensuring the raw passwords are never stored in plaintext within our database. This encryption process is pivotal to our security paradigm, maintaining an impenetrable barrier against unauthorized access.

During the authentication routine, when a user attempts to access the system, a decryption process is invoked. The system retrieves the encrypted password from the database and, with the

application of the secret key, decrypts it for the purpose of verification. This AES decryption only takes place in a transient, secure environment to minimize exposure.

User password changes and retrievals follow strict protocols that invoke re-encryption processes. When a user updates their password, the new password undergoes immediate AES encryption before it replaces the former in our database. In scenarios involving password retrievals or resets, we prefer a token-based approach. This strategy avoids the transmission of sensitive password information and leverages time-bound, secure tokens that grant users the capability to create a new password, which is then encrypted via AES.

The harmony between AES encryption and user authentication in FortiVault ensures that we not only verify the identities of our users with integrity but also manage their credentials under the highest echelons of security, buttressing the overarching framework of our platform's data protection strategy.

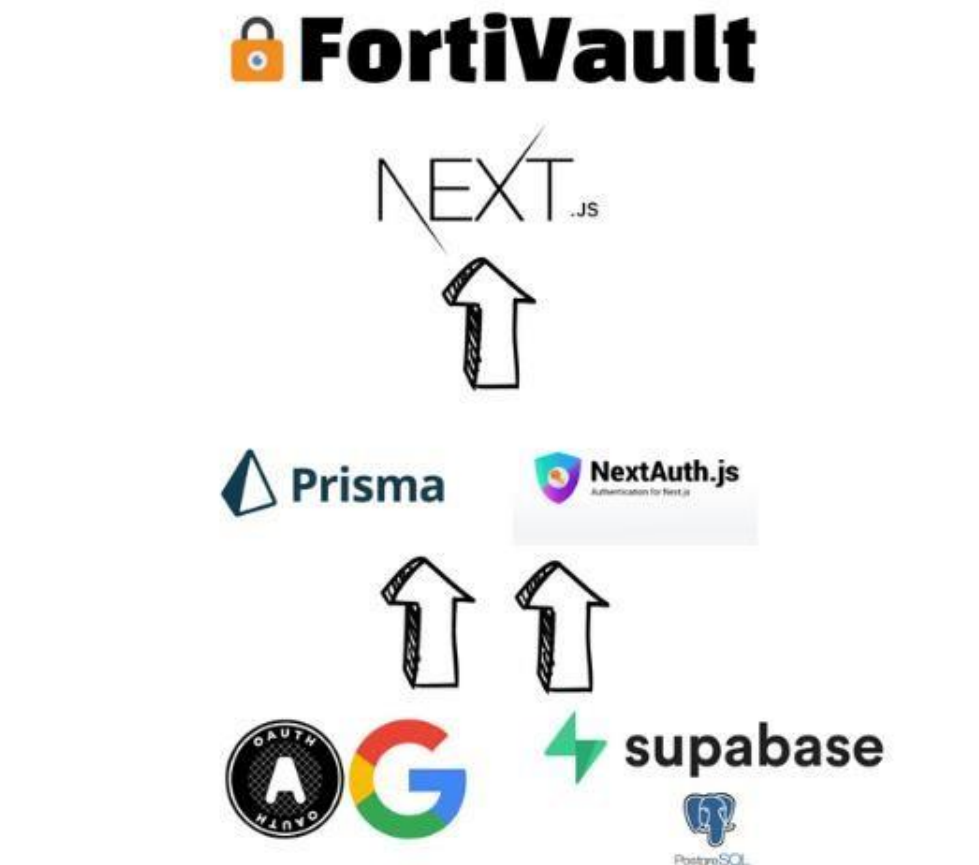


Figure 1: FortiVault Flowchart



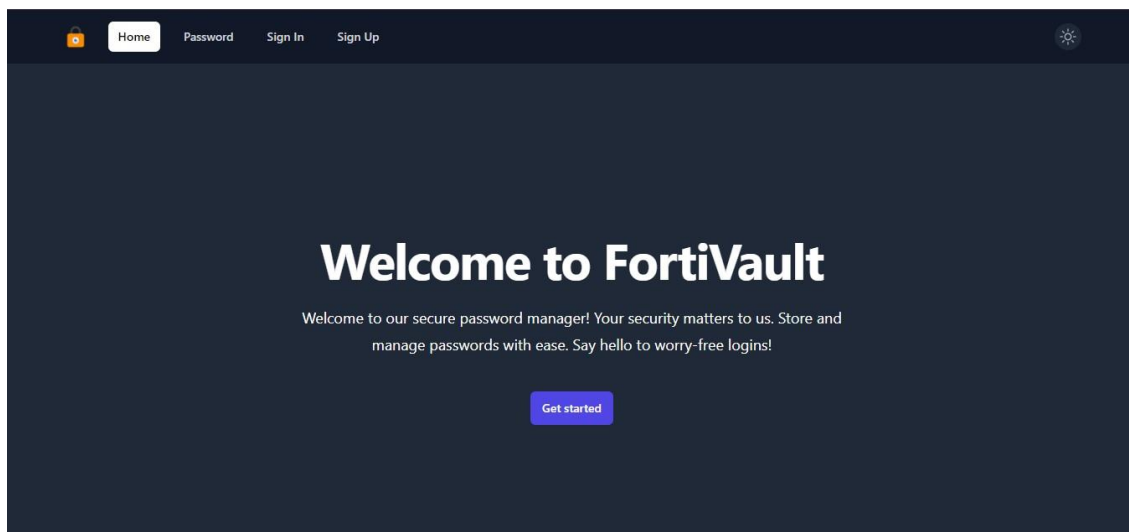
## Chapter 4

# Results and Discussions

The purpose of this section of the report is look at the results of the methodology approach in providing an application that can meet the requirements got from the problem analysis. This section also looks at the feedback got from users who conducted beta testing on the application. The results shown come after extensive iterative system testing and then deployment of the application on Vercel. The unique link of the application got from Vercel deployment is used to access the web application.

### 4.1 Application Features

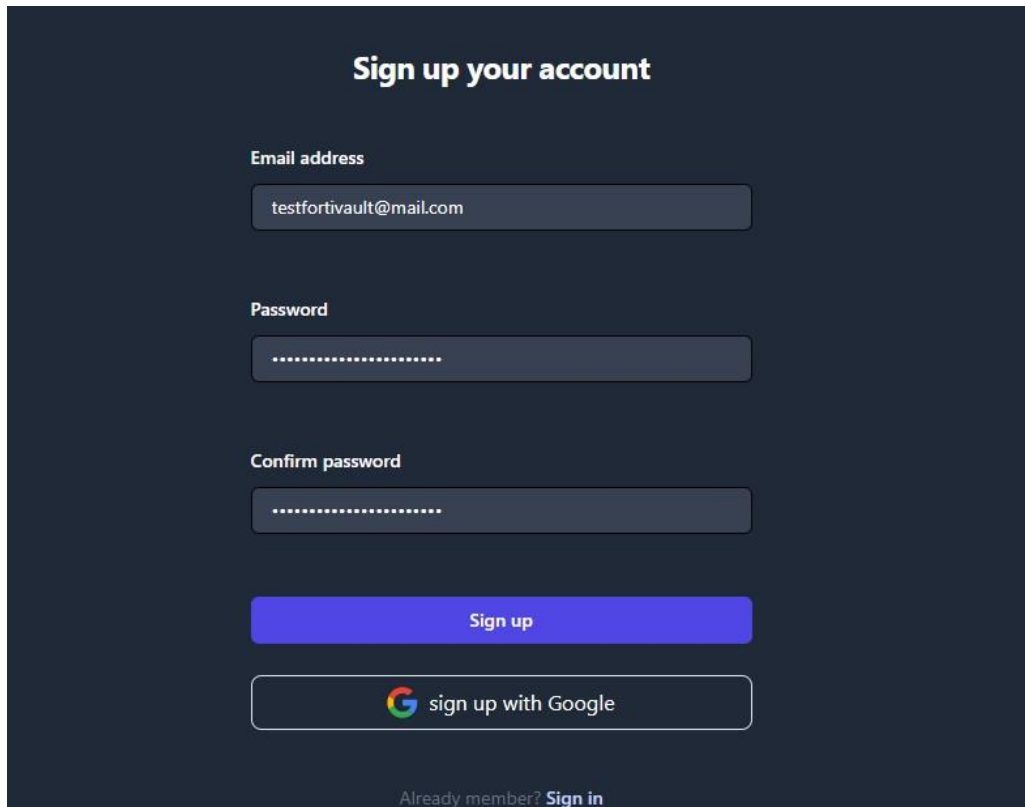
#### 4.1.1 HOMEPAGE



*Figure 2: Homepage.*

The above image shows the Homepage of the application with a clean look and a short and precise statement telling the users exactly what the app does and the **Get started** button telling the user how to start making use of the app.

#### 4.1.2 Signup Page




**Sign up your account**

Email address  
testfortivault@mail.com

Password  
.....

Confirm password  
.....

Sign up

 sign up with Google

Already member? [Sign in](#)

Figure 3: Sign up page.

The image above shows the sign-up page which allows the user to sign up to the web application by inputting their credentials themselves but also allows them to sign up using their email. It also allows registered users to sign into the app.

#### 4.1.3 Sign-in Page

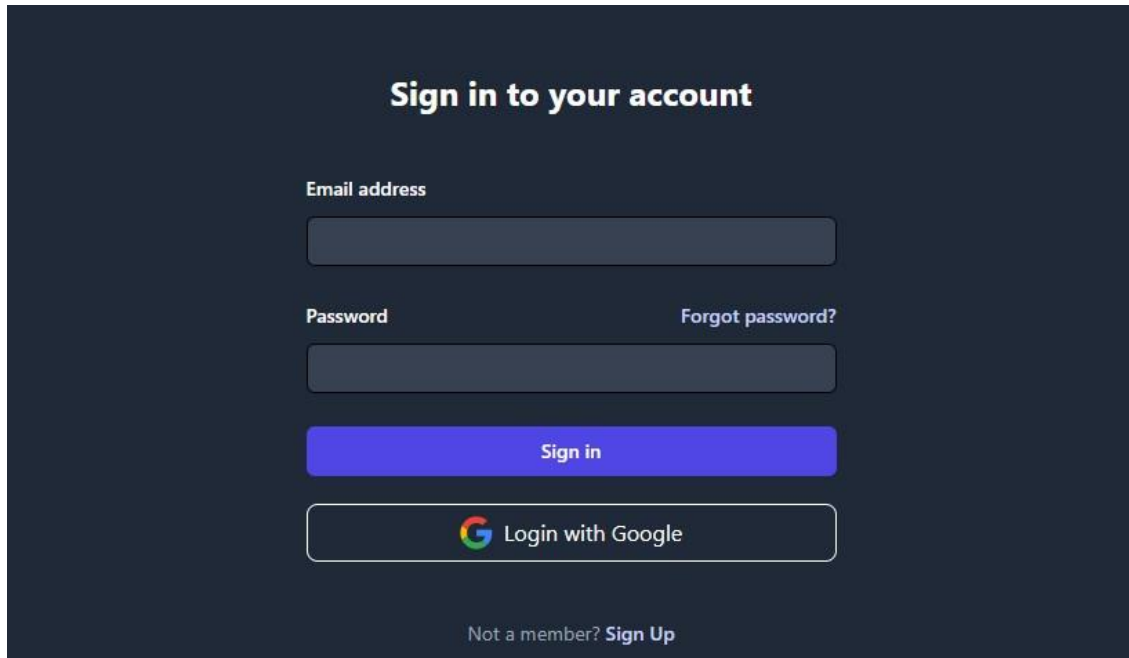
The image shows a dark-themed sign-in page. At the top, the text "Sign in to your account" is centered in a white, bold font. Below this, there are two input fields: "Email address" and "Password", both with light gray borders and placeholder text. To the right of the password field is a link that says "Forgot password?". Below the input fields is a large, rounded rectangular button with a blue gradient and the text "Sign in" in white. Underneath the button is another rounded rectangular button with a white border and the text "Login with Google" in black, featuring the Google logo to its left. At the bottom of the page, there is a link that says "Not a member? Sign Up" in a small, light gray font.

Figure 4: Sign-in Page.

The sign-in page allows users to enter credentials and sign in using their Gmail account. It also allows users to sign-up if they are not registered. Moreover, users who have forgotten their password can use the forgot password button to recover their details.

#### 4.1.4 Forgot Password Page

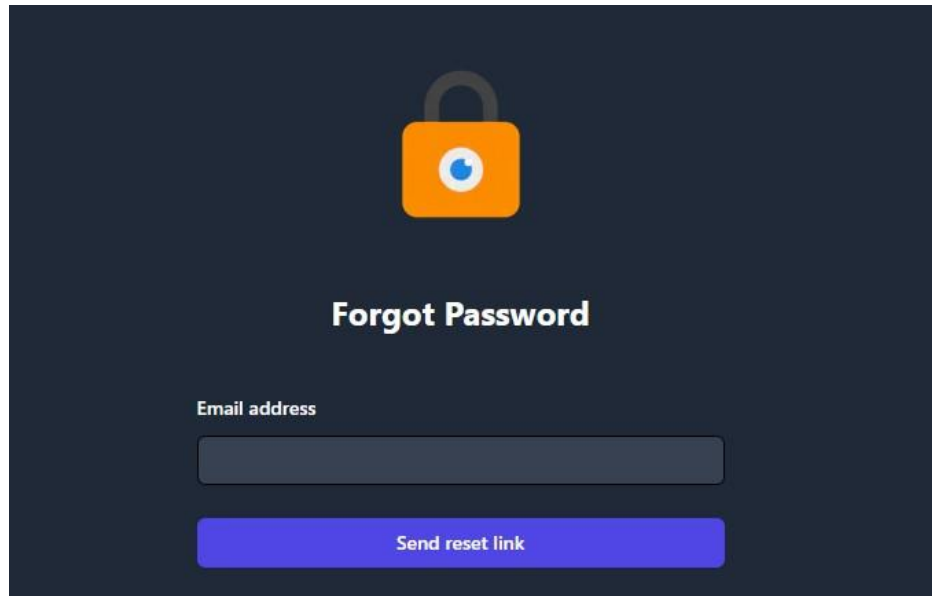


Figure 5: Forgot Password.

The image above shows the **Forgot Password** feature of the application. A user can input the email address used to create their account and a reset link will be sent to them.

#### 4.1.5 GMAIL Signup/Sign-in

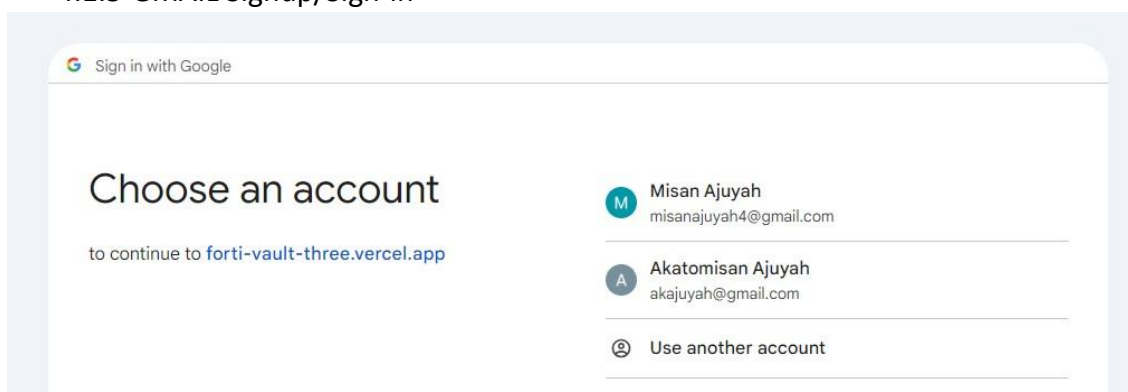


Figure 6: Gmail Signup/Sign-in.

When a user decides to sign up/sign-in using their Gmail account they will be redirected to this page and select the account they wish to use. After, selecting the account, the user will be redirected back to the FortiVault home page. Verified users can go straight into using the

password page, but unverified users will receive a message informing them to click in other to get a verification email sent to their registered account. This is shown below:

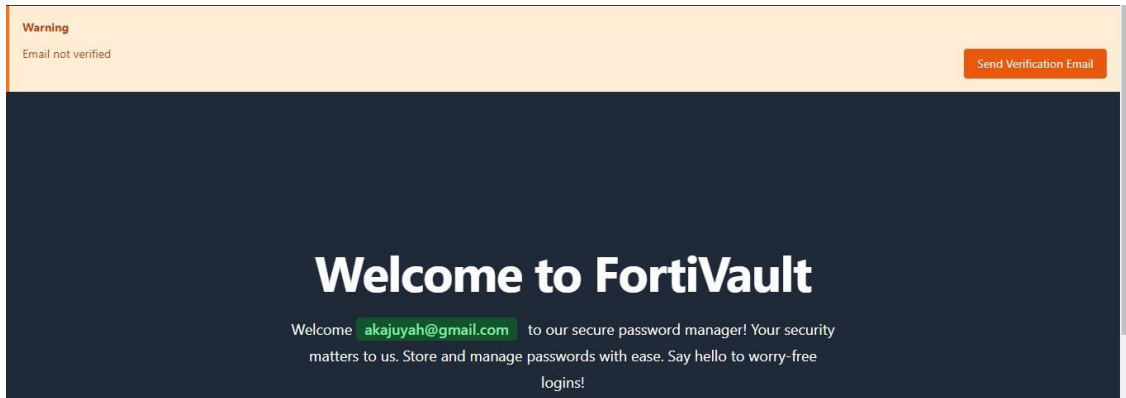


Figure 7: Email Verification

After clicking the **Send Verification Email**, users will receive a message saying **Mail Sent** but if there was an error saying **Failed to send verification email**. In the users Gmail they will receive a mail as the one in the image below:

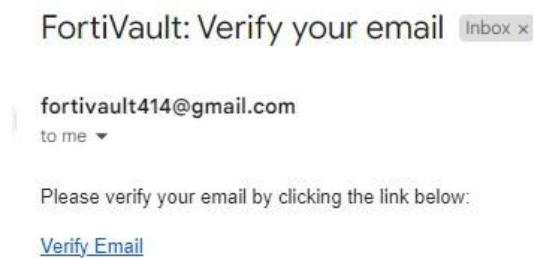


Figure 8: Verification Email

On clicking the **Verify Email** link the user is redirected back to their FortiVault account and a verification confirmation message is displayed. This is seen below:

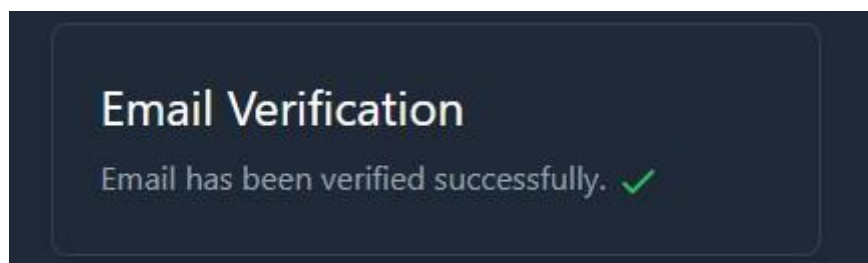


Figure 9: Confirmation Message.

#### 4.1.6 Password Page

The image below shows the Password page of the application. A green button with the word **New** is used to add passwords to the user's account.

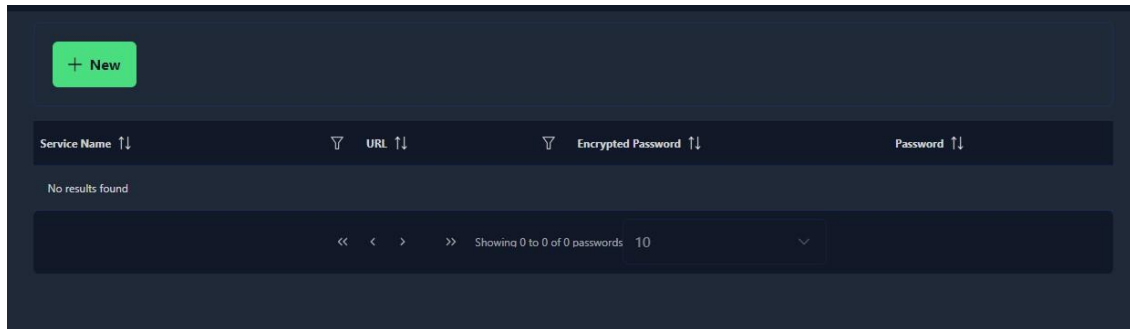


Figure 10: Password Page

From the above figure we can also see that the page has a central table which is used to store user data. The **Service Name** field is used to store the name of the service the user is storing the password for. Examples of services are Netflix, Twitter or even Amazon. The **URL** field holds the URL to the service for which the user is saving data. An example is [www.netflix.com](http://www.netflix.com). The **Encrypted Password** field holds the encrypted password of the user, and the **Password** field holds the decrypted password of the user. On clicking the **New** button a box comes up allowing the user to enter necessary details.

A screenshot of a 'Password manager' modal form. The form has a title bar with a close button (X). It contains four input fields: 'Service Name' with the value 'Test Website', 'Service Url' with the value 'www.testwebsite.com', 'Password' with the value 'f@#Llkj3raA3', and 'PassPhrase' with the value '123456'. Below the 'Password' field is a yellow 'Generate' button. At the bottom right, there are two buttons: a blue 'Cancel' button with a close icon and a green 'Add' button with a checkmark icon.

Figure 11: Password Details

The Password from the above image is created using the **Generate** button which allows users to create strong, reliable passwords. The PassPhrase is the key which the user will use to encrypt and decrypt their password. After filling all the necessary details, the user can then use the **Add** button to add the information to the table. The result of this is shown below:




Service Name ↑↓	URL ↑↓	Encrypted Password ↑↓	Password ↑↓
Test Website	www.testwebsite.com	U2FsdGVkX191Y52ZHHuNKkHGtH/XRUw2yfRYW3ALhk4=	  

Figure 12: Password Table



Service Name ↑↓	URL ↑↓	Encrypted Password ↑↓	Password ↑↓
Test Website	www.testwebsite.com	U2FsdGVkX191Y52ZHHuNKkHGtH/XRUw2yfRYW3ALhk4=	<div>  <input type="password" value="123456"/> <input type="button" value="ok"/>   </div> <div> f@#Llkj3raA3  Decryption success </div>

Figure 13: Decrypted Password Table

As seen from the images above, the details the user entered have been saved on the application. The user on clicking the purple eye icon will be required to input their passphrase to decrypt the encrypted password. They can use the blue pen icon to edit and update the details entered. And finally, the user can use the red bin icon to delete a password from their table. This is a clear implementation of the CRUD operations stated in the methodology.

	cluso8ms60001zre7gq0ekpa5	U2FsdGVkX191Y52ZHHuNKkHGtH/XRUw	Test Website	www.testwebsite.com
-------------------------------------------------------------------------------------	---------------------------	---------------------------------	--------------	---------------------

Figure 14: Data from Database

The image above shows the details as pulled from the database. The encrypted password is stored but not the decrypted password so in the event of a data breach users can be rest assured their information is still secret and cannot be decrypted.

## 4.2 Observed Results and User Feedback

As mentioned before, the application was sent out for beta testing and users were also sent a google form which can be found in the appendix. The results of the testing and user feedback were analysed and are detailed below:

- **Encryption Efficacy:** The implementation of AES-256 encryption within FortiVault's architecture has proven to be highly effective. Encryption testing, designed to simulate a variety of cyber-attack scenarios, ensured that user credentials were shielded against

unauthorized access. Twenty test passwords were taken through decryption services, and none were able to break down the encrypted password, further highlighting the strength of AES-256 encryption. Eighty percent of users expressed a strong feeling of security using the app and hundred percent of users did not encounter any security issue.

- **Authentication System Accuracy:** The authentication process according to the user feedback was positive. About fifty five percent of users had no issue in creating and verifying their accounts. However, forty percent of users had issues with receiving the verification email as it went to the **Spam** section of their mail. One user initially got a failed verification attempt, but the issue was due to a maintenance period from Vercel for some applications, but the issue was later resolved.
- **Password Management and User Experience:** Password management efficiency was another focus of the results analysis. The user experience in password update, retrieval and reset procedures was scrutinized closely. About ninety-five percent of users used the **Gmail signup/sign-in** feature with the remaining five percent preferring to sign in using their own credentials. Furthermore, when presented with a scale from 1-5 representing their satisfaction with the app, almost eighty percent of users expressed five, which means they were extremely satisfied with their experience while about 20% felt between 4 or 3. No user felt below 3. Also, for the initial use of the app, every user was able to use the app on their mobile device showing the flexibility of the app and satisfying one of this project's goals which is to allow users access data from anywhere at any time on any device. All users found the **Generate** button extremely useful and felt the generated passwords were secure.
- **System Performance:** Based on user feedback, the application has shown promising levels of performance and speed. All users felt the speed of operations of the app were swift and without much delay. As stated before, the main feature which caused delays was the password verification email.

### 4.3 Summary

In Summary, the results obtained from testing and feedback gained from users were positive and expected. The system worked as desired and any error which came up was identified and resolved. One improvement that could be made would be to improve user interface and details on the application to further improve user experience.



## Chapter 5

# Conclusions and Future Work

### 5.1 Conclusions

The project embarked on an ambitious mission to create a secure and user-friendly platform for password management. The research has culminated in the successful design, development, and testing of a system that stands at the intersection of innovative technology and cybersecurity best practices. The implementation of AES-256 encryption has been validated through rigorous testing, proving its effectiveness in safeguarding user credentials against a range of cyber-attack scenarios. The authentication system deployed within FortiVault has demonstrated outstanding accuracy and reliability, underscoring the potential of sophisticated algorithms in enhancing digital security without diminishing usability. The results show that high levels of security and user experience are not mutually exclusive. FortiVault has excelled in providing a seamless and efficient password management experience without compromising on encryption and protection standards. Additionally, the successful performance of the application suggests that the system is well-suited for scalability, positioning FortiVault as a viable solution for individual users and businesses alike. Crucially, the conclusion drawn from the user feedback is that while the system meets current security expectations, there is an appetite for further security features, such as multi-factor authentication options. The open engagement with user feedback highlights an essential aspect of software development: the need to evolve continually and adapt in response to user needs and the shifting landscape of cybersecurity threats.

Considering the broader implications, it is evident that FortiVault contributes meaningfully to ongoing discourse in cybersecurity. It stands as an example of innovation that addresses pivotal concerns around data protection and privacy in the digital age. While the current version of FortiVault fulfils its intended objectives, the field of cybersecurity is both dynamic and challenging, requiring ongoing vigilance and responsiveness.

### 5.2 Future work

FortiVault's journey does not conclude here. The next phase of research and development could explore incorporating more advanced multi-factor authentication methods, heuristic

analysis for threat detection, and the integration of behavioural biometrics for enhanced security. Additionally, exploring decentralized password management through blockchain technology could offer an innovative pathway to even stronger user data protection mechanisms.

The research undertaken in this project delivers a promising foundation upon which future scholarly work and practical applications can build, potentially setting new benchmarks for digital security within the rapidly evolving technological landscape.

## Chapter 6

# Reflection

Looking back on this project, I picked up some skills that were useful and faced challenges. For example, my programming skills have improved over time and are better than when I first started this project. Furthermore, I have come to learn that password management is just a small niche in the broad scope of cybersecurity. I find the experience to be profoundly enlightening, extending far beyond the mere acquisition of technical skills. This project served as a comprehensive exercise in problem-solving, critical thinking, and adaptability, pushing me to navigate through a series of challenges and decisions that tested my resolve and expanded my capabilities.

One of the most significant aspects of this experience was the strategic decision-making process involved in problem-solving. Each hurdle encountered required a thoughtful approach, balancing the need for practical solutions with the constraints of time and resources. This often meant prioritizing certain features or functionalities based on their impact and feasibility, a skill that I believe will be invaluable in my future endeavours.

The research component of this project was another area of substantial learning. Diving into the complexities of data security and user experience design, I realized the importance of a solid research foundation. It taught me how to critically evaluate sources, synthesize information, and apply theoretical knowledge to practical challenges. This process not only informed the technical aspects of the application but also provided insights into the broader implications of digital security and privacy.

If given the chance to tackle a similar project, I would place a greater emphasis on iterative development and user testing. The feedback gathered could offer invaluable insights, enabling more informed decisions and a more user-centric approach.

In hindsight, some deviations from the initial plan were inevitable. The dynamic nature of technology and unforeseen challenges necessitated flexibility, leading to adjustments in aims

and objectives. This adaptability, while at times daunting, was a crucial learning aspect, teaching me the importance of being open to change and the value of resilience.

In conclusion, this project was not just about building a password manager; it was a journey of personal and professional growth. The knowledge and skills developed, the challenges faced, and the insights gained have left an indelible mark on my learning experience, shaping my approach to future projects and my understanding of the impact of technology on society.

# References

- [1] P. W. Singer and A. Friedman, *Cybersecurity and Cyberwar: What Everyone Needs to Know* (R), Oxford University Press, 2014.
- [2] D. Emm, "IT threat evolution Q3 2021," Kaspersky, 26 November 2021. [Online]. Available: <https://securelist.com/it-threat-evolution-q3-2021/104876/>. [Accessed 28 March 2024].
- [3] J. Bonneau, C. Herley, P. C. v. Oorschot and F. Stajano, "The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes," in *2012 IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, 2012.
- [4] E. Stobert and R. Biddle, "The Password Life Cycle: User Behaviour in Managing Passwords," in *Tenth Symposium On Usable Privacy and Security*, Ottawa, Canada, 2014.
- [5] B. Grawemeyer and H. Johnson, "Using and managing multiple passwords: A week to a view," *Interacting with Computers*, vol. 23, no. 3, pp. 256-267, 2011.
- [6] D. Florencio and C. Herley, "A Large-Scale Study of Web Password Habits," in *Proceedings of the 16th International Conference on World Wide Web*, Banff, Alberta, Canada, 2007.
- [7] C. H. P. C. V. O. F. S. Joseph Bonneau, "Passwords and the evolution of imperfect authentication," *Communications of the ACM*, vol. 58, no. 7, pp. 78-87, 2015.
- [8] M. A. Sasse, S. Brostoff and D. Weirich, "Transforming the 'Weakest Link' — a Human/Computer Interaction Approach to Usable and Effective Security," *BT Technology Journal*, vol. 19, no. 3, pp. 122-131, 2001.

- [9] S. Jarecki, A. Kiayias and H. Krawczyk, "Round-Optimal Password-Protected Secret Sharing and T-PAKE in the Password-Only Model," in *20th International Conference on the Theory and Application of Cryptology and Information Security*, Kaoshiung, Taiwan, China, 2014.
- [1 M. M. I. F. U. R. A. A. I. Muhib Ahmad Khan, "Scalable and Secure Network Storage in Cloud 0] Computing," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 14, no. 4, pp. 545-552, 2016.
- [1 M. Dworkin, "Recommendation for Block Cipher Modes of Operation Methods and 1] Techniques," National Institute of Standards and Technology, Maryland, 2001.
- [1 P. Rogaway, M. Bellare and J. Black, "OCB: A Block-Cipher Mode of Operation for Efficient 2] Authenticated Encryption," *ACM Trans. Inf. Syst. Secur.*, vol. 6, pp. 365-403, 2001.
- [1 P. Gasti and K. B. Rasmussen, "On The Security of Password Manager Database Formats," in 3] *European Symposium on Research in Computer Security*, Pisa, Italy, 2012.
- [1 J. Bonneau and S. Schechter, "Towards Reliable Storage of 56-bit Secrets," in *Proceedings of 4] the 23rd USENIX Security Symposium*, San Diego, CA, 2014.
- [1 O. Bhamare, P. Gite, A. Lohani, K. Choudhary and J. Choudhary, "Design and Implementation 5] of Online Legal Forum to Complain and Track UGC Cases using NextJs and GraphQL," in *International Conference on Signal Processing and Integrated Networks (SPIN)*, Noida, India, 2023.
- [1 J. C. Martinez-Santos, J. Vasquez-Aguilar and E. Puertas, "Researcher Profile: An Automated 6] Solution for Searching and Gathering People's Profiles," in *2023 IEEE Colombian Caribbean Conference (C3)*, Barranquilla, Colombia, 2023.
- [1 D. P. Widyadhana, P. A. S. Adi, D. Purwitasari and S. Arifiani, "Recommendation System with

- 7] Faster R-CNN for Detecting Content Violation in Broadcasting Videos,” in *2023 IEEE 7th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, Purwokerto, Indonesia, 2023.
- [1 A. F. Nugraha, H. Kabetta, I. K. S. Buana and R. B. Hadiprakoso, “Performance and Security  
8] Comparison of Json Web Tokens (JWT) and Platform Agnostic Security Tokens (PASETO) on RESTful APIs,” in *2023 IEEE International Conference on Cryptography, Informatics, and Cybersecurity (ICoCICs)*, Bogor, Indonesia, 2023.
- [1 A. Alkhulaifi and E.-S. M. El-Alfy, “Exploring Lattice-based Post-Quantum Signature for JWT  
9] Authentication: Review and Case Study,” in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, Antwerp, Belgium, 2020.
- [2 S. I. Adam, J. H. Moedjahedy and J. Maramis, “RESTful Web Service Implementation on  
0] Unklab Information System Using JSON Web Token (JWT),” in *2020 2nd International Conference on Cybernetics and Intelligent System (ICORIS)*, Manado, Indonesia, 2020.
- [2 P. Liu, C. Deng and D. Wang, “Design and Application of Distributed Real-time Database  
1] System for Massive Data,” in *2022 3rd International Conference on Computer Science and Management Technology (ICCSMT)*, Shanghai, China, 2022.
- [2 F. Zhao, Q. Sun, J. Zhan, L. Nie and Z. Xu, “The real-time database application in transformer  
2] substation hotspot monitoring system,” in *16th International Conference on Advanced Communication Technology*, Pyeongchang, Korea (South), 2014.
- [2 S. Stoja, S. Vukmirovic, B. Jelacic, D. Capko and N. Dalcekovic, “Architecture of Real-Time  
3] Database in Cloud Environment for Distributed Systems,” in *2014 2nd International Conference on Artificial Intelligence, Modelling and Simulation*, Madrid, Spain, 2014.
- [2 M. A. Muin, M. A. Muin, A. Setyanto, Sudarmawan and K. I. Santoso, “Performance

4] Comparison Between AES256-Blowfish and Blowfish-AES256 Combinations,” in *2018 5th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, Semarang, Indonesia, 2018.

[2 D. Rakanović and R. Struharik, “IP core for AES256 and TDES algorithms with AXI interface,” 5] in *2016 24th Telecommunications Forum (TELFOR)*, Belgrade, Serbia, 2016 .

[2 “CryptoJS,” GitBook, [Online]. Available: <https://cryptojs.gitbook.io/docs>. [Accessed 5 April 6] 2024].

[2 “everything You Need to Know About AES-256 Encryption,” Kiteworks, [Online]. Available: 7] <https://www.kiteworks.com/risk-compliance-glossary/aes-256encryption/#~:text=AES%2D256%20encryption%20is%20virtually,current%20computing%20technology%20and%20capabilities..> [Accessed 5 April 2024].



## Appendix

### Appendix 1- Link to Project Code

<https://csgitlab.reading.ac.uk/wt021310/final-year-project>

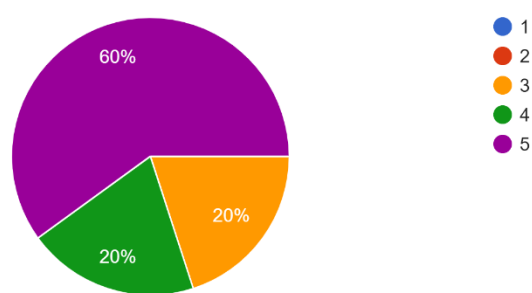
### Appendix 2- Link to Web Application

<https://forti-vault-three.vercel.app/>

# Appendix 3 – User Feedback Form Results

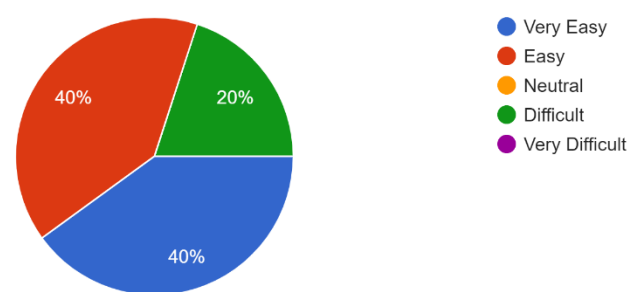
On a scale of 1 to 5, how would you rate your overall satisfaction with FortiVault?

5 responses



How easy was it to set up your FortiVault account

5 responses



How intuitive do you find the user interface and navigation within the app?

5 responses

Very intuitive and as expected.

It was easy

It's a clear guide but not enough explanation on the benefits

Works well, no major issues.

Runs well and easy

Are there any features you feel are missing or could be improved? Please provide details.

5 responses

Nothing comes to mind bar a typo when sending the verification email which says "mail send" rather than "mail sent"

No it was fine & easy to use

Just more details

Email was too long for the UI itself, so appeared a little bit off.

Nope all good

How did you find the "Generate password" feature

5 responses

Worked great

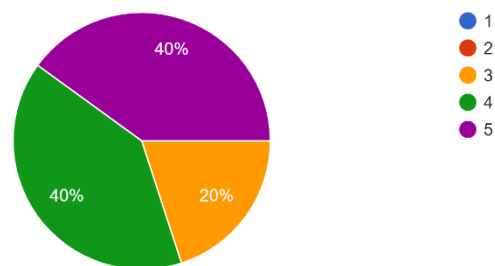
It was

I didn't find it

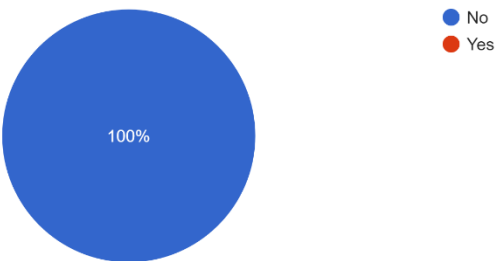
Good, easy to use and created a strong password.

Worked really well very randomised

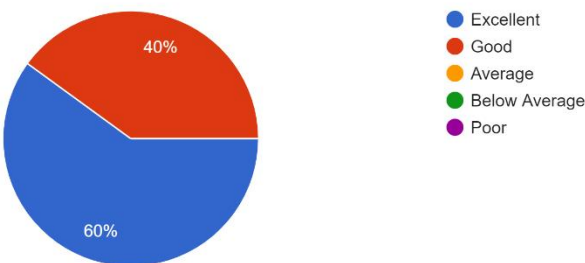
On a scale of 1 to 5, how confident are you in the security and privacy of your data with FortiVault  
5 responses



Have you ever encountered any security issues while using our app?  
5 responses

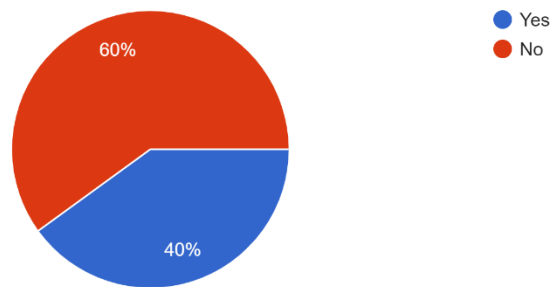


How do you rate the app's performance and speed?  
5 responses



Have you experienced any bugs or crashes?

5 responses



If yes, could you please describe the circumstances?

2 responses

Verification email was not sent, later resolved.

Only a minor one that emails go to spam

What are the improvements you would like to see in future updates of FortiVault

2 responses

Data privacy and security forms

More details

Would you recommend our password manager app to friends or family? Why or why not?

5 responses

Yes as it is easy and straightforward to use

Because it's a good initiative, helps keeps passwords safe in a place other than your phone, especially if your phone gets missing and isn't backed up

Definitely, because sometimes people have multiple passwords and it would really help

Yes, if they have many passwords.

Yep my older family always forget them so that would be good

If there is anything else you would like to share about your experience with the app feel free to add it here

1 response

Nothing much good initiative. Brilliant work. The authors have done really well.