كلية علوم الحاسب والمعلومات
قسم تقنية المعلومات

Secure Box

IT 497 Graduation Project Report
Product Release-2

Prepared by

Sarah K Juwied, 442201381
Suhad Ahmed Alhomaidhi, 442202332
Alanoud Al-Musallam, 441200824
Nouf Aldakheel, 442202526

Supervised by
Dr. Kholoud Saad Al-Saleh

Second Semester 1445
Spring 2024

Table of Contents

Table of Tables

Table of Figures

Secure Box

Sarah Juwied [1], Suhad Alhomaidhi [2], Alanoud Al-Musallam [3] Nouf Aldakheel [4]

[1]Information Technology Department, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia; 442201381 @student.ksu.edu.sa
[2]Information Technology Department, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia; 442202332@student.ksu.edu.sa
[3]Information Technology Department, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia; 441200824@student.ksu.edu.sa
[4]Information Technology Department, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia; 442202526@student.ksu.edu.sa

Abstract (English):

In this project, we address the need for a secure and efficient file storage and sharing system for mobile devices. The purpose of developing the Secure Box software system is to provide users with a reliable platform that overcomes the challenges of limited storage capacity and security risks associated with mobile file storage. The main methodology employed in the development process includes leveraging server-based storage, implementing advanced encryption techniques, and introducing a schema to prevent shoulder surfing attacks. Through an agile development approach, the project team has successfully designed and implemented Secure Box, focusing on data security, encryption, and user-friendly functionality, while also incorporating measures to prevent shoulder surfing attacks. The evaluation of the system has demonstrated its effectiveness in providing a secure environment for file storage and sharing, as well as preventing unauthorized access through shoulder surfing. The results show that Secure Box offers a seamless user experience, robust data protection, efficient file management capabilities, and enhanced privacy safeguards against visual eavesdropping. The conclusion drawn from this project is that Secure Box, with its comprehensive security measures, is a viable solution for individuals and organizations seeking a secure and convenient file storage system for mobile devices.

Abstract (Arabic):

في هذا المشروع، نعالج الحاجة إلى نظام آمن وفعال لتخزين ومشاركة الملفات على الأجهزة المحمولة. يهدف تطوير نظام البرمجيات "Secure Box" إلى توفير منصة موثوقة تتغلب على التحديات المتعلقة بسعة التخزين المحدودة ومخاطر الأمان المرتبطة بتخزين الملفات على الأجهزة المحمولة. تشمل المنهجية الرئيسية المعتمدة في عملية التطوير الاستفادة من التخزين القائم على الخوادم، وتنفيذ تقنيات التشفير المتقدمة، وتقديم حل لمنع التسلل من فوق الكتف. من خلال نهج التطوير الجريء، نجح فريق المشروع في تصميم وتنفيذ Secure Box"، مع التركيز على أمان البيانات والتشفير ووظائف سهلة الاستخدام، مع إدماج إجراءات لمنع التسلل من فوق الكتف. أظهرت تقييمات النظام فعاليته في توفير بيئة آمنة لتخزين ومشاركة الملفات، ومنع الوصول غير المصرح به من خلال متابعة الكتف. تشير النتائج إلى أن ”Secure Box"يوفر تجربة مستخدم سلسة، وحماية قوية للبيانات، وإمكانيات فعالة لإدارة الملفات، وتعزيزات للخصوصية للحماية من التجسس البصري. ويتوصل هذا المشروع إلى استنتاج أن "صندوق آمن"، مع تدابيره الأمنية الشاملة، هو حلاً قابلاً للتنفيذ للأفراد الباحثة عن نظام آمن وملائم لتخزين الملفات على الأجهزة المحمولة.

Keywords: secure file storage, mobile devices, encryption, data security, file sharing, agile development, Biometric Authentication, factor authentication, file management, shoulder surfing attacks, privacy protection.

Introduction

1 Introduction

Nowadays, mobile phones have become an essential part of our lives in the current age of technology, acting as our personal assistants, entertainment centers of activity, and communication engines.

The limited storage of smartphones frequently causes a problem for users who gather multimedia files and application data. Furthermore, the constant security risks linked to file storage on mobile devices, such as the potential for hacking or damage, intensify the necessity for a dependable and robust solution.

We propose a file storage and sharing system that makes use of server-based storage and encryption to mitigate these concerns. Our solution seeks to reduce the risk of data loss or unauthorized access while providing users with an effortless and secure platform for storing and accessing their files.

Therefore, our proposal intends to provide customers with a secure, practical, and effective file-storing and sharing experience by merging these advanced technologies. To meet the evolving needs of mobile users in a world that is becoming more connected, we set a high priority on data security, encryption, and user-friendly functionality.

We present this document which contains several parts including the problem it intends to solve and its solution Then we will determine our product vision and the road map of the product. Continually, we will illustrate product objectives and the scope of the problem we will be solving. Afterwards, we'll discuss the software and hardware requirements for this project. We will also introduce the scrum team's capabilities and responsibilities. In addition, we will describe our background and perform a literature review. Finally, we'll provide an overview of our system description and a list of our product backlog.

1.1 The Problem

The limited storage of smartphones frequently causes a problem for users who gather multimedia files and application data. This limitation can restrict the number and size of files you can store on your device. Running out of storage space can lead to difficulties in saving new files or installing updates. Furthermore, storing sensitive or confidential files on a smartphone can pose security risks. Smartphones can be lost, stolen, hacked or damaged, potentially exposing the stored files to unauthorized access.

Additionally, these days cyberthreats have grown. One of the common security attacks mobile phone users' encounters is shoulder surfing attacks. A Shoulder surfing attack is "using direct observation techniques, such as looking over someone's shoulder, to get information" [1]. According to 2023 Eye-opening Shoulder surfing statistics which shows There have been real-life incidents highlighting the risks of shoulder surfing. In the UK, a victim lost £70,000 of their personal and business funds after visiting a busy pub. Similarly, in France, a gang was arrested for stealing over 153,000 euros by observing victims' PIN codes [2].

1.2 Objective

The main goal of this project is to create Secure Box, an app designed to enhance file storage and sharing experience on mobile devices. This initiative aims to build a robust and efficient software system that tackles the common issues of constrained storage space and security vulnerabilities in mobile file management. By developing Secure Box, we aim to offer users a dependable platform that effectively addresses these critical challenges. Specifically, Secure Box should be able to:

- Users can register, log in, and log out.
- Users can store their files in an encrypted form on a server.
- Users can view all the files they have uploaded.
- Users can update their information.
- Users can delete their files.
- Users can rename their files.
- Users can download their files.
- Users can create folders.
- Users can delete their folders.
- Users can reset their password.
- Users can search their files.
- Users could share their files through encrypted messages.
- Users could view files that were sent to them.
- Users could mark files as favorite.
- Users can unmark favorite files.
- Users can filter their files.
- Users can get notified whether another user shares file with them.
- Users can contact the support team.

- Admin can log in and log out.
- Admin can start and stop the server.
- Admin can reset their password.

1.3 Scope

Secure Box is a mobile application designed for Android smartphones that supports the English language only. Its primary goal is to provide a secure platform for users to store and exchange files using strong encryption protocols to encrypt files during transit and storage. Users of this application can share their files using encrypted messages. Moreover, this application utilizes two-factor authentication and uses fingerprint biometrics and text-based graphical passwords to provide a very high level of security.

1.4 Product Vision

For individuals seeking secure and effortless file sharing on mobile devices **Who value** the Confidentiality of their data and require a robust solution for sharing and protecting files against threats. The Secure Box **is a** secure file-sharing platform **That** facilitates storing and sharing encrypted files while using strong authentication and avoiding common shoulder surfing attacks **Unlike** common file transfer methods that rely on Text-based passwords and do not utilize biometrics, **our** product features advanced encryption protocols and combines the use of text-based graphical passwords and biometrics for authentication, while also allowing for easy file sharing

1.5 Approach

The selected software development methodology for this project is Agile, which emphasizes iterative development and the continuous delivery of small, functional segments of software to enhance customer satisfaction. Initially, we engaged with potential users through interviews and a questionnaire to gather insights and define user requirements. These findings helped us create a comprehensive product backlog, which we then segmented into various sprints and releases.

Throughout the project, we adopted an incremental development approach. At the end of each sprint, we produced a functional portion of the software, which was then reviewed and assessed by our supervisor (acting as the product owner) and the scrum master. This iterative cycle ensured that feedback was integrated promptly and effectively.

Before the final release, the software underwent User Acceptance Testing (UAT) and Non-Functional Requirements (NFR) testing. These tests were crucial in identifying areas for enhancement and in shaping the considerations for future iterations of the product.

1.6 The Solution

In response to the challenges we've identified, we have developed a server-based file storage application. Our solution addresses the limitations of device storage, enabling users to store large files without concern for device capacity.

Our application stands out with an exceptional level of security and confidentiality. Unlike conventional methods, our approach employs advanced security measures that safeguard files against a range of threats, including shoulder surfing and other types of attacks. Users access the application through a combination of text-based graphical passwords and fingerprint biometric authentication. Furthermore, we plan to implement it and make use of the fingerprint sensor on recent Android phones, ensuring a robust and convenient login process.

Moreover, we've integrated a unique feature to simplify file sharing between app users: encrypted messages. This feature ensures secure communication and exchange of files among application users. As for file encryption, we will use a secure encryption algorithm to protect the confidentiality of the files.

1.7 Summary

In the subsequent sections, we will begin by presenting our background and conducting a literature review. Following this, we will detail our system design and development process. We will then proceed to evaluate the system's performance. Finally, we will conclude with a summary of our findings, outline potential future work, and extend our acknowledgments.

Background

2 Background

One of the most vital assets for individuals and organizations in today's interconnected world is information. Information is more vulnerable than ever due to the speed at which technology is developing and the widespread use of digital systems.

Additionally, this interconnectivity gives attackers the chance to take advantage of weaknesses and perform numerous attacks. And this could lead to serious problems and financial losses. Attacks may spread through a variety of techniques, such as ransomware attacks, phishing, viruses, and hacking. Since these attacks could come from anywhere in the world, it is difficult to identify and capture those who are behind them.

In this section we will address the need for information security and how important it is in today's interconnected world. We will go over some of the most frequent information security attacks that affect both individuals and organizations, as well as the main security concepts that will be used to reduce these risks. Lastly, we will be discussing the APIs that will be used in this project.

2.1 Types Of Attacks

There are various types of attacks that pose significant risks to information security, including social engineering and malware. Malware is software that penetrates a computer system without the user's awareness or permission and proceeds to carry out an undesired, typically harmful action. Malware can be classified based on its primary goal. For instance, worms and versus are types of malwares that rapidly propagate their infections. Furthermore, Trojan horse, logic bomb, and rootkit are types of malicious software that attaches itself to legitimate programs or files and can replicate and spread by infecting other files or systems which aims to conceal their malicious activities [3]. Another classification of malware includes ransomware, botnet and spyware which aims to generate illegal profits [4].

Another type of attack that poses risks to information security is social engineering, which involves the manipulation of individuals to obtain unauthorized access or sensitive information [3]. Social engineering attacks can use both psychological and physical techniques.

- **Psychological Techniques**

1. Impersonation: meaning to develop a false persona and then play out the character on a victim [3].
2. Phishing: involves sending an e-mail or displaying a Web notice that fraudulently claims to be from a real business to trick the user into revealing personal information.
3. Spam: refers to random or undesired messages sent to many people.

- **Physical Techniques**

1. Tailgating attacks: also called piggybacking or physical access, consist of accessing an area or building by following someone who has the security clearance to that place [3].
2. Dumpster Diving: entails searching through dumpsters for information that can be used in an attack [3].
3. Shoulder surfing: Shoulder surfing is the technique that involves an attacker glancing over the target's shoulder in public places such as cafes, airports, or offices to observe their mouse clicks, screen content, or other while entering their confidential data. This method allows the attacker to get usernames, passwords, and other sensitive information. Real-life incidents have highlighted the significant risks associated with shoulder surfing. In the UK, a victim fell prey to this attack while visiting a busy pub and lost a staggering £70,000 from their personal and business funds. Similarly, in France, a gang was apprehended after stealing over 153,000 euros by carefully observing victims' PIN codes [2]. These incidents highlight the real-world impact of shoulder surfing, emphasizing the critical need for information security.

2.2 Information Security

Information security creates a defense that attempts to ward off attacks and prevents the collapse of the system when a successful attack occurs. Thus, information security is protection [6].

Information security can be achieved through a combination of the basic trio Confidentiality, Integrity, Availability (CIA) in addition to Identification, Authentication and Authorization.

Confidentiality ensures that only authorized individuals can access sensitive information, and measures like strong passwords can be implemented to prevent unauthorized access. By maintaining confidentiality, the information remains accessible only to those with authorization [3].

Integrity ensures the accuracy and unaltered state of the information. Unauthorized modifications by individuals or malicious software are prevented, ensuring the integrity of the data. For example, altering file content by an attacker would violate the integrity of the information [3].

Availability guarantees that authorized users can access the data when needed. Implementing redundant systems, infrastructure, and load balancing techniques contribute to achieving availability. These measures ensure that data remains available to authorized users, even in the face of disruptions or failures [3].

Authentication plays a critical role in access control. It is the initial step in verifying the identity of individuals seeking access to protected resources. Multi-factor authentication (MFA) further enhances security by requiring individuals to provide multiple forms of authentication [3].

2.2.1 Authentication Types

- **Something You Have**

This aspect involves possessing a physical item or token as proof of identity such as tokens, smart cards, etc. These devices often generate one-time passwords that are synchronized with the authentication server. When the user presents or uses the device during authentication, the system verifies the validity and uniqueness of the code to grant access. This can add an extra layer of security.

- **Something You Know**

The most common type in this aspect is password and there are different types of it, for example, alphanumeric passwords which is a combination of letters and numbers, complex password which consist of uppercase, lowercase letter, numbers, and special character.

However, there have been security concerns that these textual passwords are vulnerable to shoulder surfing attacks. To mitigate this risk, one effective solution is the use of password concealing or graphical tricks for obfuscation. For instance, something you know could be text-based graphical passwords and according to Manjunath G et al. [5] text-based graphical password provides a viable defense against shoulder surfing attacks. Rather than using textual passwords, users can identify themselves in a way that makes them less vulnerable to observation or being captured by someone else. In 2014 Manjunath G et al. [5] proposed the Text-Based Shoulder

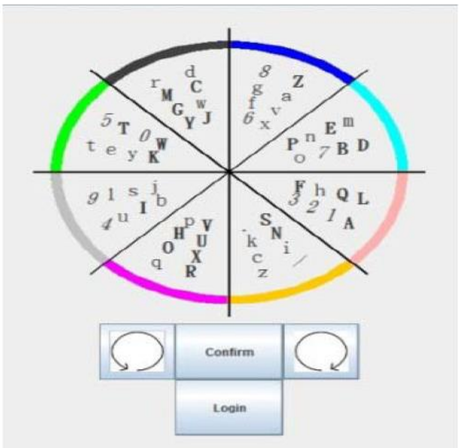
Surfing Resistant Graphical Password Scheme as shown in Fig 1. The proposed scheme involves two phases, the registration phase, and the login phase, which can be described as in the following.

1. Registration phase

The user must enter his textual password K of length L between 8 to 15 characters and select one of the eight colors supplied by the system as his pass color. The remaining seven colors that the user did not select are his deceptive colors.

2. Login phase

When the user decides to log in to the system, the system presents a circle with eight equal-sized sectors. The arcs of the eight sectors have different colors, and each sector is distinguished by the color of its arc, for example, the blue sector is the sector of the blue arc. First, 64 characters are distributed evenly and randomly among these sectors. All the shown characters can be rotated into the adjacent sector clockwise by clicking the "clockwise" button once or the adjacent sector counterclockwise by clicking the "counterclockwise" button once, and the rotation action is also possible by scrolling the mouse wheel. The login screen of the proposed scheme can be illustrated by an example shown in Fig 1.



To log into the system, the user must first complete the following steps:

Figure 1 (text-based graphical passwords 1 [5] *)*

Step 1: A user asks for access to the system.

Step 2: The system displays a circle with 8 equally sized sectors and distributes 64 characters evenly and randomly among the 8 sectors, such that each sector has 8 characters. The 64 characters are formatted in three ways: the upper-case letters are in bold font, the lower-case letters and the two symbols "." and "/" are in normal font, and the ten decimal digits are in italic style.

Additionally, the "Confirm" and "Login" buttons are shown on the login screen, along with the buttons for spinning clockwise and counterclockwise. One-click on the "clockwise" or "counterclockwise" buttons will rotate all the displayed characters simultaneously into the adjacent sector in either a clockwise or counterclockwise direction. The rotation operations can also be carried out by scrolling the mouse wheel. Let I = 1. The rotation operation can be illustrated by an example shown in Fig. 2.

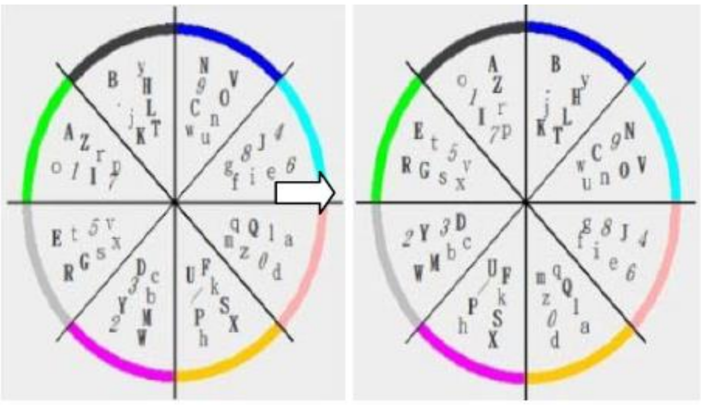


Figure 2 (text-based graphical passwords 2** [5] **)

Step 3: The user must rotate the sector holding his password's i-th pass-character, designated by Ki, into his pass-color sector before selecting the "Confirm" button. Let i = i + 1.

Step 4: If i < L, the system randomly shuffles all 64 characters that are shown before moving on to Step 3. If not, the user must press the "Login" button to finish the login procedure. The system will send an email with a secret link to the user's registered email address once the account is disabled if it cannot be successfully authenticated three times in a row. The legitimate user can use this link to reactivate the account. The login process of the proposed scheme can be illustrated by an example shown in Fig. 3.

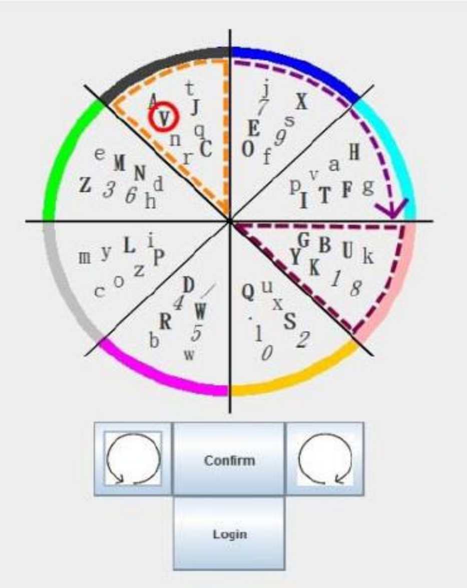


Figure 3 (text-based graphical passwords 3** [5] **)

The user must rotate the sector (marked with an orange dotted line for illustration only) containing Ki (marked with a small red circle for illustration only) into his pass-color sector (marked with a brown dotted line for illustration)

- **Something You Are**

Biometric data such as fingerprints, iris patterns, facial recognition, and voiceprints are commonly used. Biometric data is captured and stored for comparison during the authentication process. Despite its advantages, it's important that we consider the potential disadvantages and challenges associated with biometric authentication and the need for specialized hardware or software components to capture and process biometric data. These may include iris scanners, facial recognition cameras, or fingerprint scanners. These technologies need an investment from organizations, which could include paying in advance for resources.

By combining the text-based graphical password (something you know) with fingerprint authentication (something you are). We can achieve a powerful access control with two factor authentication approach which will greatly minimize the possibility of unauthorized access since an attacker would have to compromise many factors to gain access, ensuring user’s confidentiality.

2.3 Cryptography

"Cryptography, a word with Greek origins, means "secret writing." However, we use the term to refer to the science and art of transforming messages to make them secure and immune to attacks. " [6].

Cryptography provides confidentiality, integrity, authentication, and non-repudiation to ensure the privacy and security of data.

Cryptography plays an essential part in many areas, including secure communication over the Internet, secure storage, digital signatures, and authentication protocols. It is a vital component in current information security, contributing to risk mitigation and guaranteeing the confidentiality and integrity of sensitive data. Furthermore, there are two primary types of cryptography which are:

2.3.1 Asymmetric Key Cryptography

There are two keys used in this type of cryptography which are a private key and a public key. For example, Alice wants to send a secure message to Bob, Alice first encrypts the message using Bob's public key. To decrypt the message, Bob uses his private key an illustration of asymmetric cryptography is shown in figure 4 [6].

Asymmetric cryptography provides secure key exchange and digital signatures using a pair of mathematically related keys. It scales well for large networks and offers enhanced security. However, asymmetric algorithms are computationally intensive and slower, requiring longer key lengths and more complex key management. Examples of asymmetric key algorithms include DSA (Digital Signature Algorithm) and RSA (Rivest-Shamir-Adleman).

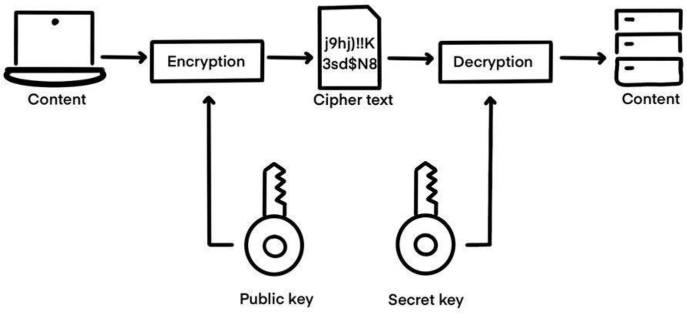


Figure 4 (Asymmetric encryption)

2.3.2 Symmetric Key Cryptography

Which is basically shared-key cryptography, symmetric cryptography uses a single secret key for both encryption and decryption. Both the sender and the receiver use the same key to convert plaintext into ciphertext and vice versa. For instance, Alice encrypts the message using the shared secret key; Bob decrypts the message with the same shared key and reads the message an illustration of symmetric cryptography is shown in figure 5 [6].

Symmetric key cryptography offers efficiency and simplicity, with fast encryption and decryption of large amounts of data using a single shared key. Common symmetric key algorithms include Advanced Encryption Standard (AES), Data Encryption Standard (DES), and Triple DES (3DES).

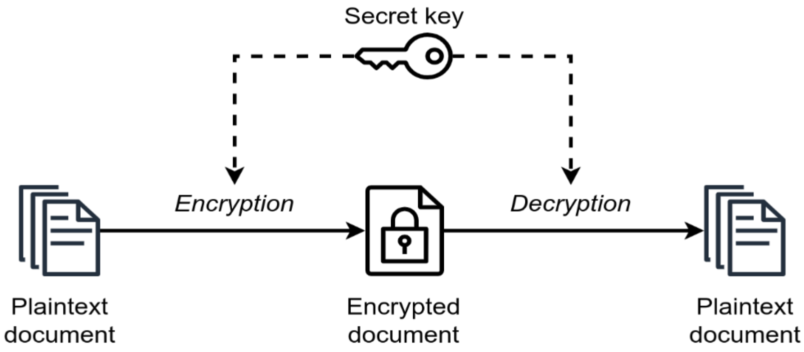


Figure 5 (Symmetric encryption)

We have decided to use a combination of symmetric key cryptography and asymmetric encryption for our data security measures. For file encryption, we employ symmetric key cryptography, specifically utilizing the AES algorithm. This choice is based on its fast encryption and decryption capabilities, making it suitable for handling large amounts of data while ensuring performance efficiency. AES is widely recognized for its robust safety measures, having undergone extensive examination by the cryptographic community without any real-world attacks being discovered. It provides a dependable option for protecting sensitive data and works effectively on modern computer systems, allowing for quick encryption and decryption rates without significant performance degradation [7].

In addition to symmetric key cryptography, we have implemented asymmetric encryption, also known as public-key cryptography, to secure users' passwords and SMS messages. Asymmetric encryption offers unique advantages in specific scenarios, notably the convenience of eliminating the need for a shared secret key between the sender and receiver. By utilizing both symmetric key cryptography and asymmetric encryption, we can achieve a comprehensive approach to data security, combining the efficiency and ease of use of symmetric key cryptography with the specific benefits of asymmetric encryption.

2.4 API

API stands for Application Programming Interface. It is a set of rules and protocols that allows different software applications to communicate and interact with each other. APIs define how different software components should interact, specifying the methods, data formats, and protocols that applications can use to access and exchange information [8].

In our application we will use two APIs which are:

2.4.1 Messages API

Messages are a valuable component for our app, as it provides convenient messaging functionalities and simplifies the integration of sharing features. And by leveraging the Messages API, we can incorporate this functionality without having to build it from scratch. Additionally, the notification functionality offered by the Messages API is another advantage. Real-time notifications are essential for keeping users engaged and ensuring they receive important updates. By utilizing this feature, our application can deliver notifications to users as new messages occur, helping to maintain their interest and prevent them from missing critical information. Messages API offers a standardized and efficient solution for implementing file sharing and notification functionalities in our app. By leveraging these capabilities, we can enhance the user experience and the development process.

2.4.2 Android Biometric API

Our application also uses the Android Biometric API to enhance security through biometric authentication. We enable users to authenticate using their fingerprints by using the Biometric API. Users must first enroll their fingerprint on their device to use the biometric authentication feature. The enrolling process is straightforward and may be accessible via the device settings. Users can register their fingerprints by going to the "Security" section and following the procedures. After enrolling, users can register in our application successfully.

We prioritize confidentiality of user data, especially biometric information. We guarantee that all biometric data entered into our application is kept secret. It is never shared or saved anywhere other than the user's device.

By incorporating the Android Biometric API, our application provides a secure and convenient login experience. Combination of biometric authentication and text-based graphical password as a two-factor authentication approach adds an extra layer of protection, ensuring that only authorized users can access sensitive data within our application.

Literature Review

3 Literature Review

In this section we will first outline some of the research that tried to address and prevent shoulder surfing attacks. Then we will introduce a competitive product analysis.

3.1 Research on Prevention Shoulder Surfing Attack

Among the first was the research conducted by Sobrado and Birget [9] in 2002 the Movable Frame scheme, the Intersection scheme, and the Triangle scheme were three shoulder surfing-resistant graphical password schemes. The Intersection scheme and the Movable Frame scheme, however, both have high failure rates. In 2006, Wiedenbeck et al. [10] introduced the Convex Hull Click technique (CHC) as an enhanced version of the Triangle technique with more security and usability. The user must properly answer numerous tasks before being allowed to log in. Every challenge requires the user to locate any three of the three pass-icons that are presented on the login page, then click within the invisible convex hull that is created by every pass-icon that is displayed. Convex-Hull Click scheme's login time, however, can be excessively long. A shoulder-surfing-resistant graphical password scheme called Color Login was proposed by Gao et al. in 2009 [11], where the background color can be used to speed up login. However, Color Login's risk of accidental login is too great, and there isn't enough password space. In 2014 Manjunath G et al. [5] proposed the Text-Based Shoulder Surfing Resistant Graphical Password Scheme. This scheme is suitable for everyone due to its simplicity, speed, and effectiveness. By implementing it, we can streamline our processes and achieve our goals efficiently. The alphabet used in the suggested scheme has 64 characters total, including 26 capital letters, 26 lowercase letters, 10 decimal digits, and the symbols "." and "/" shown in Fig. 4. The proposed scheme involves two phases, the registration phase, and the login phase, as we mentioned in the background.

3.2 Competitive Product Analysis

Here we will introduce our competitors because it will help us to understand what features, design elements, and user experiences competitors offer, allowing us to identify opportunities for improvement. By analyzing other apps in the same category, we will refine our own app's user interface, and functionality. We selected useful applications that share some functionalities such as OneDrive, Google Drive, Dropbox, Sync.com, and Nord Locker.

3.2.1 The Competitors

1- Google Drive

Google Drive is a cloud storage and file-sharing service provided by Google. It offers a secure platform for storing files and allows users to share files with others via shareable links.[1]

Main Features:

- Google Drive provides a platform to store files and folders in the cloud, offering ample storage space for users.
- Google Drive synchronizes files across multiple devices, allowing you to access your files from anywhere, whether it's on a computer, smartphone, or tablet.
- Google Drive offers various tools to help you organize your files and folders. You can create folders and subfolders to categorize your files.

Drawbacks:

- Google Drive provides 15GB of free storage, which is shared between Google Photos, Gmail, and Google Drive and this might not be sufficient for people with plenty of data.
- Since Google Drive is a cloud-based service, an internet connection is required to view your files.
- Uploading files that were created in other software. For example, Microsoft Word documents, PowerPoint presentations, or Excel spreadsheets may not always look the same when opened in Google Docs, Slides, or Sheets.
- According to Google's terms of service, although everything that you create and upload to Google Drive is yours to be managed you also grant Google a worldwide license to perform various actions with your content.

[1] https://www.google.com/drive/

2- OneDrive

OneDrive is a cloud storage service provided by Microsoft that allows users to store, share, and access files from anywhere with an internet connection.[2]

Main Features:

- OneDrive makes it easy to share files and folders with others. You can generate shareable links with varying levels of access permissions.
- OneDrive employs security features such as encryption, two-factor authentication, and data loss prevention to help protect your files and data.
- OneDrive offers a secure area within your storage called Personal Vault, which requires an additional layer of identity verification (such as fingerprint or PIN) to access. This is useful for storing highly sensitive files.

Drawbacks:

- OneDrive only offers 5GB of free cloud storage space; if you want more, you must pay. It offers various types of chargeable options.
- OneDrive doesn't allow data sharing between non-OneDrive users.
- To use OneDrive, you need a Microsoft account. If you prefer not to use Microsoft services, this could be a drawback.

[2] https://www.microsoft.com/en-us/microsoft-365/onedrive/online-cloud-storage

3- Dropbox

Dropbox is a cloud storage service that allows users to store and share files and folders online.[3]

Main Features:

- Dropbox employs security measures like encryption, two-factor authentication (2FA), and advanced sharing permissions to help protect your data.
- Dropbox makes it easy to share files and folders with others. You can generate shareable links with varying access permissions.
- Dropbox Transfer allows you to send large files and folders to others without sharing the entire Dropbox folder. It's useful for sending large multimedia files or project deliverables.

Drawbacks:

- Storing data on a third-party cloud service may raise privacy and security concerns for some users.
- The cost of Dropbox's paid plans can be higher compared to some competing cloud storage services.
- Dropbox has file size limits for uploads and downloads. While these limits have increased over time, they can still be a limitation for users dealing with very large files.

[3] https://www.dropbox.com/

4- Sync.com

Sync.com is a cloud-based file storage and synchronization service that focuses on security and privacy.[4]

Main Features:

- All files and data are encrypted on your device before they are uploaded to Sync.com's servers, and they remain encrypted during transit and storage. Only you have the encryption keys, ensuring that Sync.com cannot access your data.
- You can collaborate on files with others by sharing folders. All files in shared folders are protected with the same security features, including encryption.
- Sync.com complies with HIPAA and GDPR regulations, making it suitable for businesses and organizations with stringent data privacy requirements.
- Sync.com employs a zero-knowledge security model, which means that even Sync.com employees cannot access your data because they do not have access to your encryption keys.

Drawbacks:

- Sync.com does not offer a personal vault or secure folder feature for extra protection of sensitive files.
- Sync.com offers a limited amount of free storage space, which may be less than what is provided by some other cloud storage providers.
- Some users have reported slower upload and download speeds compared to other cloud storage providers.

[4] https://www.sync.com/

5- NordLocker

NordLocker is a file encryption software integrated with end-to-end encrypted cloud storage.[5]

Main Features:

- NordLocker allows you to encrypt individual files or folders, ensuring that only you can access their contents with the encryption key.
- NordLocker uses a zero-knowledge approach.
- NordLocker uses strong encryption algorithms like AES-256 for file protection, which is considered highly secure.

Drawbacks:

- While NordLocker emphasizes security, it's crucial to manage your encryption keys properly.
- Sharing encrypted files with others can be more complex compared to traditional cloud storage services. Your recipients also need to have NordLocker installed to decrypt and access the shared files.
- NordLocker does not provide file previews within the app. To view a file, you need to decrypt it first.

[5] https://nordlocker.com/

3.2.2 Comparison Between Similar Application

Table 1 (Comparison Between Similar Application)

			OneDrive	Dropbox		NordLocker
two-factor authentication	✓	✓	✓	✓	✓	
Personal Vault	✓		✓	✓		✓
Easy to share	✓	✓	✓	✓	✓	
User friendly	✓	✓	✓	✓	✓	✓
text-based graphical passwords	✓					
biometric authentication	✓	✓	✓	✓		✓
Encrypt files	✓	✓	✓	✓	✓	✓

3.2.3 Summary

After conducting the analysis, it became apparent that none of the existing solutions have the text-based graphical passwords feature, and clearly lack strong security measures. For example, in addition to the presence of the text-based graphical password, we will also add biometric authentication, and this will give our application another level of security, which we consider a critical core feature of our application. Additionally, we observed that most of these solutions are cloud-based for data storage. While cloud-based storage offers advantages, it also comes with potential security risks, dependency on internet connectivity, and the possibility of service outages. However, our application will employ server-based storage due to its advantages, which include enhanced data security, scalability, and improved performance, aligning with our strong focus on security. In terms of user interfaces, we've prioritized creating user-friendly designs and have opted to follow a general design approach to enhance the overall usability of our application.

System Design and Development

4 System Design and Development

In this section, we explore the methodology and system requirements crucial for the project. We begin by outlining the development approach. Next, we detail the system requirements, including user roles, requirements gathering, user interactions, and a roadmap with a product backlog to guide the development phases efficiently. This ensures a thorough foundation for successful system development.

4.1 Methodology

The Agile software development methodology was chosen for this project due to the dynamic nature of the requirements. Agile emphasizes flexibility, continuous communication, and rapid iterations, making it an ideal choice for projects where requirements evolve over time. This method utilizes a continuous feedback loop, enabling regular updates and refinements based on feedback.

A key framework for implementing Agile is Scrum, distinguished by its focus on communication, teamwork, and collaboration. In the Scrum framework used in our project, three primary roles were defined:

Product Owner: In our case, the supervisor, who is responsible for defining and prioritizing the product backlog according to business value.

Scrum Master: Facilitates the Scrum process, ensuring adherence to its principles and removing impediments to progress. For our project, Dr. Kholoud Alyahya and Dr. Nora Al-Hammad served in this role.

Development Team: Comprises individuals (us) tasked with delivering product increments.

The Scrum framework includes five key events:

Sprint: A fixed duration (typically 2-4 weeks) during which the team works to deliver a product increment. Our project included five sprints.

Sprint Planning: Conducted at the start of each sprint, where the team plans the sprint's activities, selecting tasks from the product backlog. This planning session typically lasted one hour with the supervisor.

Daily Scrum: A brief, 15-minute meeting to discuss daily progress and plan the day's activities.

Sprint Review: Held at the end of each sprint to present the product increment to stakeholders and gather feedback.

Sprint Retrospective: A meeting after each sprint where the team reflects on what went well and what could be improved.

Additionally, Scrum defines three key artifacts:

Product Backlog: A comprehensive list of all needed features, enhancements, and bug fixes.

Sprint Backlog: The set of tasks selected for completion during the current sprint.

Increment: The cumulative result of all product backlog items completed during the sprint.

By employing Agile and Scrum, we successfully adapted to changing requirements, frequently delivering functional software. Our approach prioritized simplicity and efficiency, focusing on immediate needs with a self-organizing team that actively communicated and collaborated. This environment fostered high-quality product development conducive to excellent architecture, requirements, and design.

The use of tools such as Jira and GitHub significantly enhanced our process. Jira facilitated sprint planning, meeting note recording, and backlog management, while GitHub was instrumental in tracking code changes and collaboration among team members.

4.2 System requirements

4.2.1 System Users

Our application (Secure Box) targets users who gather multimedia files and application data and want an exceptional level of security and confidentiality. Their characteristics are as follows: their age should be above 18 years, regardless of their education level. Our app requires users to know average to advanced levels of information about storing and sharing files. They should know how to read/write and know how to use the app so they can interact with it. Users range from basic computer users to sophisticated programmers who must have Wi-Fi and know how to use the cellphone.

4.2.2 Requirements Elicitation and Analysis

For requirements elicitation, we decided that stakeholders and similar systems such as Google Drive, Dropbox, and other applications would be our main sources of information and requirements. We have mentioned some of them above. You can refer to the Competitive Product Analysis section.

For requirements discovery methods we used two methods: a questionnaire and interviews. Because of their cost-effectiveness, questionnaires were an effective way to quickly engage with a large, diverse audience. This approach allowed us to collect data from many participants in different settings and in a relatively short time frame. On the other hand, interviews offered a unique advantage by allowing us to address potential misinterpretations in real-time. The ability to instantly seek clarification through follow-up questions can be invaluable in ensuring accurate data collection.

This dual approach significantly contributed to our ability to gather comprehensive information about our users, including their knowledge and requirements. For more detailed information about both the interviews and questionnaires, please refer to the appendices.

Based on the answers we received, we noticed that 92.2% of people faced the problem of limited storage space on their mobile phones, and when asked if they were using any storage software such as cloud storage or server-based file storage applications, most of them said that they would start or continue using them but we discovered that the majority of users were using cloud storage applications instead of server-based file storage applications, so we thought that server-based file storage applications were scarce and that the spread of this type of software would make a security difference and a new experience for users.

Also, when we asked if they had sensitive files and wanted to store them, what would they do? A large portion of them said that they use applications that encrypt files and keep them in a safe place. Therefore, we believe that there is an urgent need for such applications that help store confidential files on mobile phones.

As for shoulder-surfing attacks and other types of hacks, we found that there is a fair number of people who have been exposed to this type of attack. and when asked about the improvements they would like to see in storage programs from a security standpoint and other aspects, they answered that they wanted:

- Better encryption methods than traditional approaches
- Secure login that is difficult to hack

- Ensured security and confidentiality when sending files

- Increased storage capacity 24 - Simplified file sharing

- Protection from various types of attacks, such as shoulder surfing, they confirmed that adding many layers of security when logging in will make their accounts and files more secure.

In addition, the majority welcomed the idea of sharing sensitive files via messages and thought it was a good idea to ensure the transfer of files.

4.2.3 Users Interaction

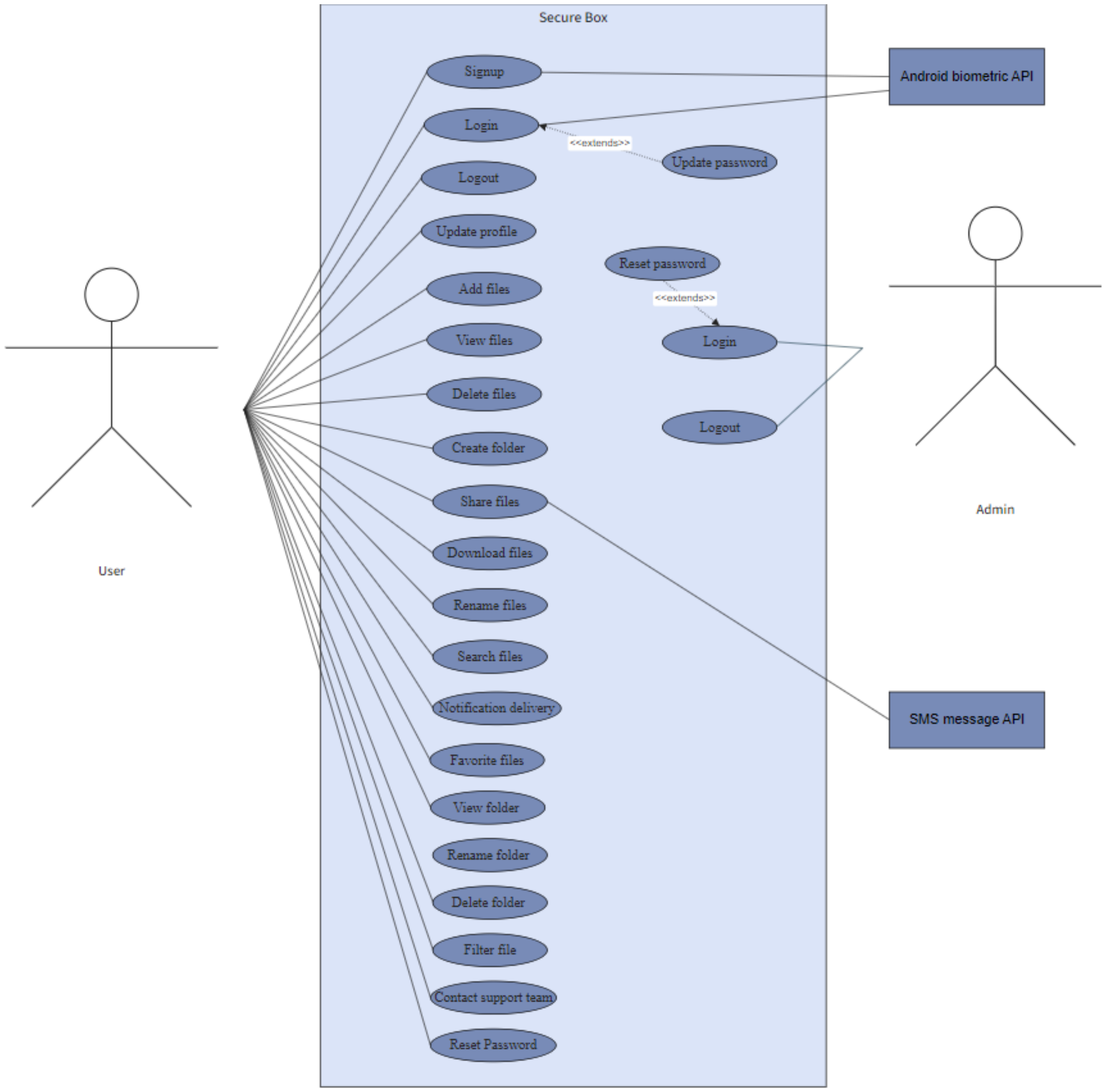
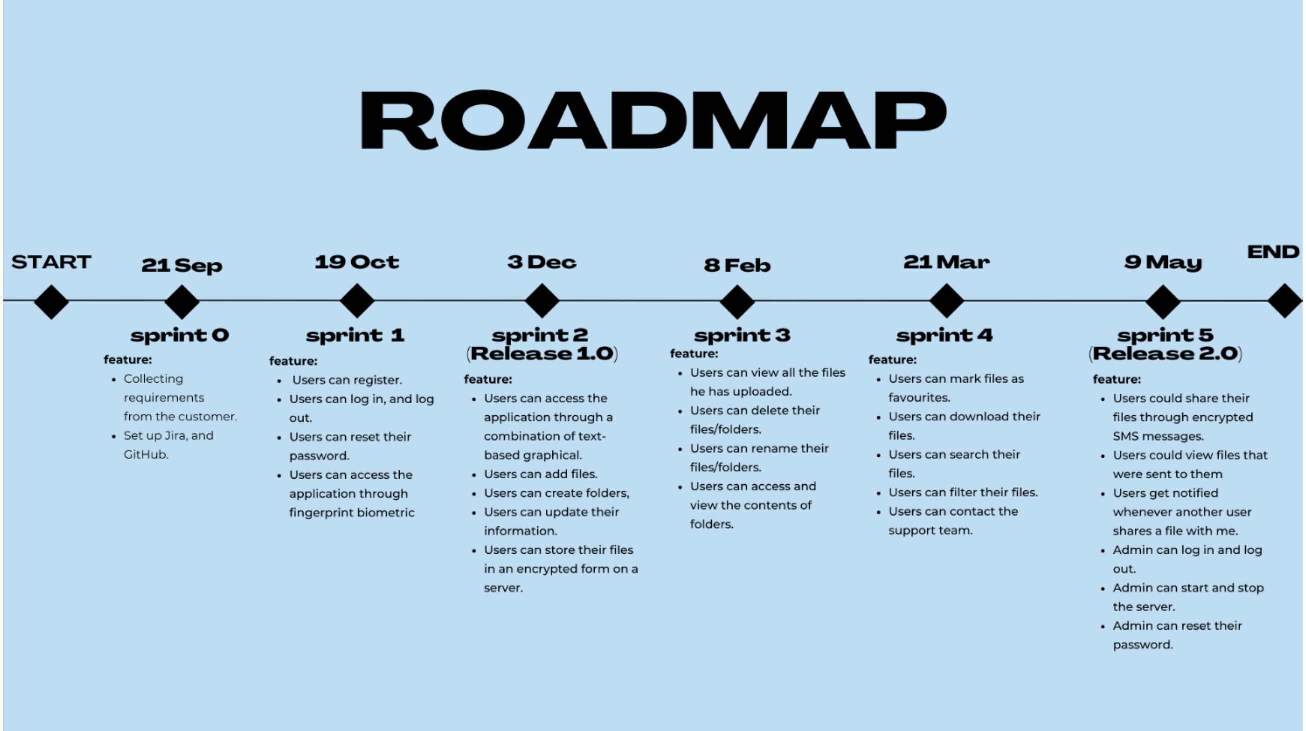


Figure 6 (Use Case)

4.2.4 Roadmap and Product Backlog

4.2.4.1 Roadmap



ROADMAP

START

21 Sep

sprint 0

feature:

- Collecting requirements from the customer.
- Set up Jira, and GitHub.

19 Oct

sprint 1

feature:

- Users can register.
- Users can log in, and log out.
- Users can reset their password.
- Users can access the application through fingerprint biometric

3 Dec

sprint 2 (Release 1.0)

feature:

- Users can access the application through a combination of text-based graphical.
- Users can add files.
- Users can create folders,
- Users can update their information.
- Users can store their files in an encrypted form on a server.

8 Feb

sprint 3

feature:

- Users can view all the files he has uploaded.
- Users can delete their files/folders.
- Users can rename their files/folders.
- Users can access and view the contents of folders.

21 Mar

sprint 4

feature:

- Users can mark files as favourites.
- Users can download their files.
- Users can search their files.
- Users can filter their files.
- Users can contact the support team.

9 May

sprint 5 (Release 2.0)

feature:

- Users could share their files through encrypted SMS messages.
- Users could view files that were sent to them
- Users get notified whenever another user shares a file with me.
- Admin can log in and log out.
- Admin can start and stop the server.
- Admin can reset their password.

END

Figure 7 (Roadmap)

4.2.4.2 Definition Of Ready

Table 2 (Definition of Ready)

	Definition Of Ready
□	Well-defined user story.
□	A team can ‘demo’ the user story.
□	Acceptance criteria are clear and testable.
□	Estimated and small enough to be completed during the sprint.
□	User stories have business value.
□	User story dependencies have been identified.
□	Performance criteria are defined.

4.2.4.3 Product Backlog Table

Table 3 (Product Backlog)

PBI (user story)	Sprint No.	Size (Story points)	Status (To do, in progress, or done)	Type (Feature, defect, technical work, knowledge acquisition)	Acceptance Criteria The conditions of satisfaction that must be met for that item to be accepted.
1. As a user, I want to be able to sign up so that I can have an account in the application that gives me access to all features.	1	2	Done	Feature	-As a user, if I go to the sign-up page and enter my name, email, phone number, password, and choose a color, then I should have an account to get access to every feature. -As a user, if I go to the sign-up page and leave an empty blank or invalid information then the sign-up will not be completed, and an appropriate message will appear that indicates what's the error. ○ Email should be in the format "example@example.com". ○ Number should start with 0 and 10 digits. ○ The password should contain either a "." or a "/" character.
2. As a user, I want to be able to log in to my account so that I can access my account and my files.	1	2	Done	Feature	-As a user, if I go to the login page and enter my username, and text-based graphical password, and apply my fingerprint correctly then I can access my account. -As a user, if I go to the login page and enter an incorrect name or password, then log-in fails and an error message appears that indicates what was wrong.

3. As a user, I want to be able to log out so that I can exit my account and prevent unauthorized access.	1	2	Done	Feature	-As a user, if I click on logout, and confirm it, then I should have no access to my documents.
4. As a user, I want to be able to add files so that I can protect them on a secure server.	2	2	Done	Feature	-As a user, if I click on the add button and choose my file, then my file should be uploaded.
5. As a user, I want to be able to reset my password so that I can retrieve my account in-case I forgot my password.	1	2	Done	Feature	-As a user, if I click on forgot password on the login page and enter my email and received a link and entered new password, then this will be my new password.
6. As a user, I want to be able to update my information so that my information is up to date	2	2	Done	Feature	-As a user, if I go to my profile and click on ‘update’ and update my information, then my updated information should be displayed. -As a user, if I go to my profile click on’ update’ and enter invalid information then an appropriate message should appear. (-the number should start with +966 -phone number exists).

7. As a user, I want to be able to delete my files so that I can remove useless files.	3	3	Done	Feature	-As a user, if I click on a file and click the Delete button then a confirmation message should appear, and deletion is completed only if I confirm it. -As a user, if I click on a file and confirm deletion, then the file should no longer exist.
8. As a user, I want to be able to delete my folders so that I can remove useless folders.	4	4	Done	Feature	-As a user, if I click on a folder and click the Delete button then a confirmation message should appear, and deletion is completed only if I confirm it. -As a user, if I click on a folder and confirm deletion, then the folder should no longer exist.
9. As a user, I want to be able to download files so that I can have access to them on my phone.	3	3	Done	Feature	-As a user, if I click on a file and click on download, then my file will be downloaded on my phone.
10. As a user, I want to share my files so that I can easily exchange information with others.	5	4	Done	Feature	As a user, when I click on a file and select 'Share with,' I should have the option to share it with another user who is already a member of the app. Once the file is shared. The recipient of the file will be able to access its content within the app.
11. As a user, I want the ability to create a folder so that I can	2	4	Done	Feature	- As a user if I click on the create folder button and a dialog appears asking me to enter the name of the folder -As a user if I click on create folder without entering a name, then the app should display an

organize my files.					error message indicates the folder name is required.
12. As a user, I want the ability to view a folder so that I can see folder details.	3	4	Done	Feature	-As a user, if I click on a folder, I should be able to view its contents.
13. As a user, I want to be able to rename my file so that I can update files information if needed.	3	1	Done	Feature	-As a user, if I go and click on ‘rename’ and edit my file name, then updated file name should be immediately reflected in the user interface. - As a user, if the name of the file contains invalid characters like ([,], {, }, (/, \), : , ? , “ , < , > ,| , or (*)) then the system should notify me that the file name contains invalid characters.
14. As a user, I want to be able to rename my folder so that I can update folder information if needed.	4	1	Done	Feature	-As a user, if I go and click on ‘rename’ and edit my folder name, then updated folder name should be immediately reflected in the user interface. - As a user, if the name of the file contains invalid characters like ([,], {, }, (,), (/, \), : , ? , “ , < , > ,| , or (*)) then the system should notify me that the folder name contains invalid characters.
15. As a user, I want the capability to mark files as favorites so that I can easily locate and access them	4	2	Done	Feature	-As a user, selecting the "mark as favorite" option for a file ensures that the file is designated as a favorite for easy access. - As a user, if I mark a file as favorite, then it should appear in the list of my favorite files.

when needed.					
16. As a user, I want the capability to unmark favorite files so that I can manage my favorite files more efficiently and remove files that are no longer relevant.	4	1	Done	Feature	-As a user, if I select a file from the list of my favorite files, then I should have the option to unmark the file. -As a user, if I choose to unmark a file, then it should no longer be designated as a favorite. -As a user, if I unmark a file, then it should no longer appear in the list of my favorite files.
17. As a user, I want to be able to search so I can find my files or folder if needed.	4	2	Done	Feature	-As a user, if I input the correct file or folder name into the search box, the application should promptly present the corresponding file or folder. -As a user, if I perform a search with file or folder that doesn’t exist, the application should continue to display the empty file of folder list.
18. As a user, I want the ability to filter and view specific types of files, specifically PDF and DOC files,	5	3	Done	Feature	- As a user, if I choose to filter files, then I should be able to select the desired file format options, specifically PDF and DOC. -As a user, if I select the PDF format option, then only files with the ".pdf" extension should be displayed in the file list.

so that I can easily access them.					-As a user, if I select the DOC format option, then only files with the “.docx" extensions should be displayed in the file list.
19. As a user, I want the ability to easily contact the support team so that I can seek assistance and resolve any issues or inquiries I may have.	5	1	Done	Feature	-As a user, if I click on the "Contact Support" option, then I should be directed to a support contact email.
20. As a user, I would like to be notified whenever another user shares a file with me, so that I can stay updated about any new files that are shared with me.	5	4	Done	Feature	-As user, if another user shares a file with the me, the system should promptly generate a notification, ensuring it is delivered in real-time. - As a user, if another user shares a file with the me, notifications should be delivered through multiple channels, including in-app notifications and messages, as per the user. - As user, if another user shares a file with the me, the notification content should include pertinent details, such as the email of the user who shared the file, a name of the shared file.
21. As a user, I want to be able to download files shared with me so that I can access them	5	4	Done	Feature	-As a user, if I click on a file and click on download, then my file will be downloaded on my phone.

on my phone.					
22. As an admin, I want to be able to start the server by logging in using the admin credentials.	5	3	Done	Feature	- As an admin, I want to be able to start the server by logging in using the admin credentials. The system should verify my credentials and provide access to server controls upon successful verification. - As an admin, if another admin tries to start the server, the server will not start because the verification will fail.
23. As an admin, I want to be able to reset my password if I forget it.	5	3	Done	Feature	- As a user, if I forget my password, there should be a clearly visible option for resetting it on the screen. The system must verify my identity using my username and password before allowing me to set a new password.
24. As an admin, I want to be able to stop the server by logging out.	5	2	Done		- As an admin, I want to be able to stop the server by logging out when I type "logout."
25. As a user, I want the application to be well-designed for navigation so that I can complete my task in less than 1 minute.	-	NA	Done	Feature	-As a user, if I want to complete a specific task, I should be able to easily and quickly access the services that will assist me in doing so in less than 1 minute.
26. As a user, I want my account information to be secured by encryption	-	NA	Done	Feature	-As a user, I can create an account, fill out my information, and be assured that it will be stored securely.

so that I can protect my information.					
27. As a user, I want the app to be available 90% of the time I try to access it, so I don't become upset and go to another app.	-	NA	Done	Feature	-As a user, If I try to use the app, it needs to be available to me at least 90% of the time.
28. As a user, I want the homepage of the application to be opened within at most 5 seconds to use the application as quickly as possible.	-	NA	Done	Feature	-As a user, if I am in a hurry and don’t have time to wait, then the application home page loading should not exceed 5 seconds.

4.3 System Design

In this section, we explore the design components shaping our system. We begin with an overview of the system's structure, move on to understand relationships and data flow, and then delve into the design of critical functionalities. Data Design and Interface Design are key focuses, ensuring a well-organized and user-friendly system.

4.3.1 Architecture Diagram

"The Client-server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters called clients.

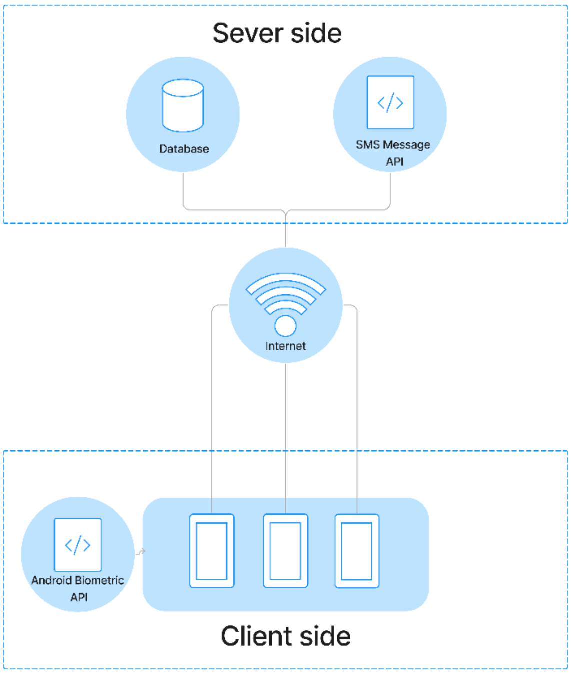


Figure 8 (Client-server model)

In the client-server architecture, when the client computer sends a request for data to the server through the internet, the server accepts the requested process and delivers the data packets requested back to the client." [12].

By using this architecture shown in Fig 8, we will divide the application into two main sides:

Client-side:

Contains SecureBox users and Android Biometric API.

- Users: who want to store and share files, as well as the ability to view, rename, and delete them.
- Android biometric API: will allow our application to incorporate the Android Biometric API to enhance security through biometric authentication.

Server-side:

Contains SecureBox dataset, SMS message.

- SecureBox dataset server: will be used to store users' data and the file in encrypted format.
- SMS message API: will allow our application to integrate SMS (Short Message Service) messaging into our software platform.

4.3.2 Class Diagram/ DFD

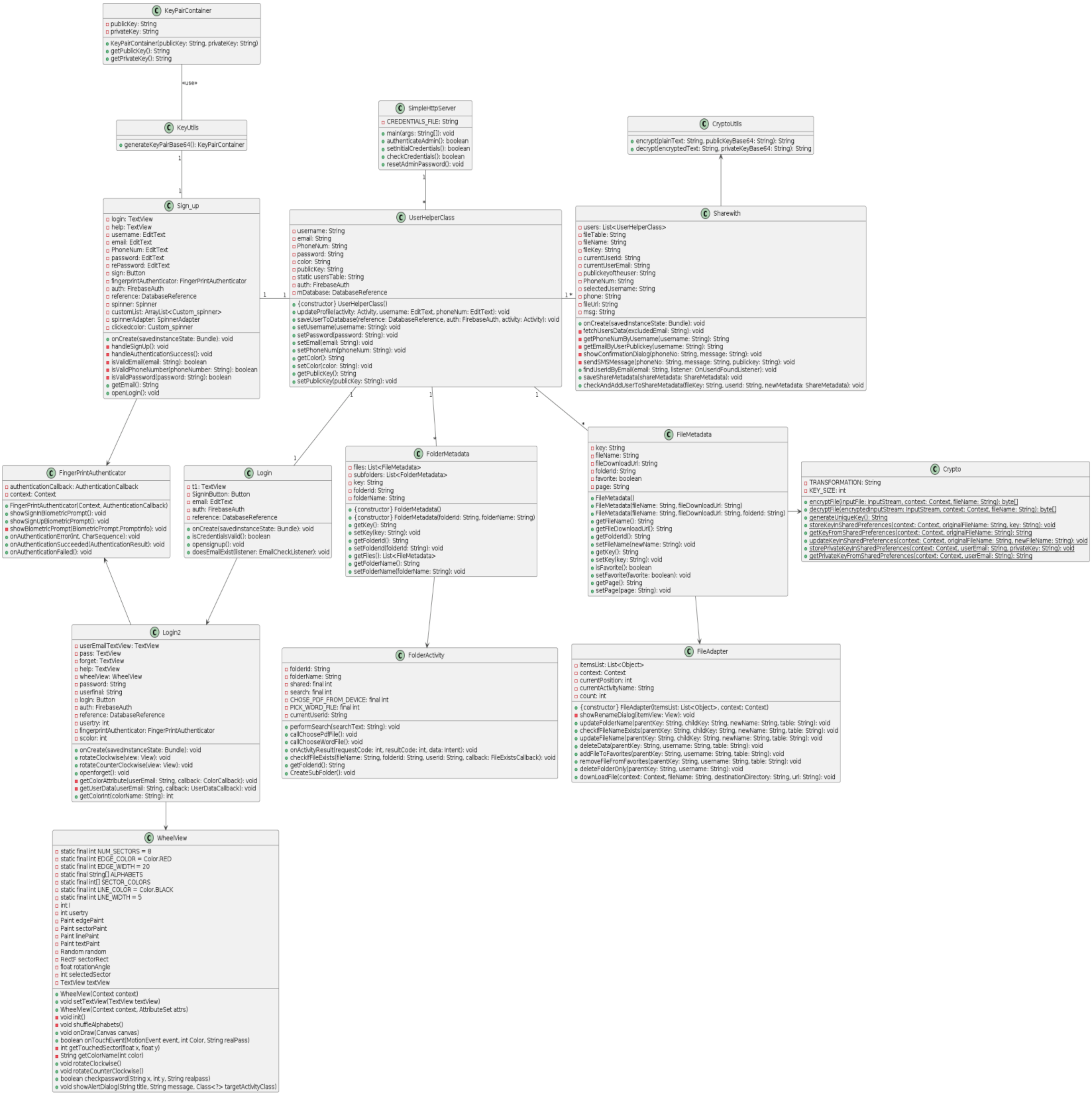

KeyPairContainer
publicKey: String
privateKey: String
KeyPairContainer(publicKey: String, privateKey: String)
getPublicKey(): String
getPrivateKey(): String
«use»
KeyUtils
generateKeyPairBase64(): KeyPairContainer
1
1
Sign_up
login: TextView
help: TextView
username: EditText
email: EditText
PhoneNum: EditText
password: EditText
rePassword: EditText
sign: Button
fingerprintAuthenticator: FingerPrintAuthenticator
auth: FirebaseAuth
reference: DatabaseReference
spinner: Spinner
customList: ArrayList<Custom_spinner>
spinnerAdapter: SpinnerAdapter
clickedcolor: Custom_spinner
onCreate(savedInstanceState: Bundle): void
handleSignUp(): void
handleAuthenticationSuccess(): void
isValidEmail(email: String): boolean
isValidPhoneNumber(phoneNumber: String): boolean
isValidPassword(password: String): boolean
getEmail(): String
openLogin(): void
1
1
SimpleHttpServer
CREDENTIALS_FILE: String
main(args: String[]): void
authenticateAdmin(): boolean
setInitialCredentials(): boolean
checkCredentials(): boolean
resetAdminPassword(): void
1
*
UserHelperClass
username: String
email: String
PhoneNum: String
password: String
color: String
publicKey: String
static usersTable: String
auth: FirebaseAuth
mDatabase: DatabaseReference
{constructor} UserHelperClass()
updateProfile(activity: Activity, username: EditText, phoneNum: EditText): void
saveUserToDatabase(reference: DatabaseReference, auth: FirebaseAuth, activity: Activity): void
setUsername(username: String): void
setPassword(password: String): void
setEmail(email: String): void
setPhoneNum(phoneNum: String): void
getColor(): String
setColor(color: String): void
getPublicKey(): String
setPublicKey(publicKey: String): void
1 *
CryptoUtils
encrypt(plainText: String, publicKeyBase64: String): String
decrypt(encryptedText: String, privateKeyBase64: String): String
Sharewith
users: List<UserHelperClass>
fileTable: String
fileName: String
fileKey: String
currentUserId: String
currentUserEmail: String
publickeyoftheuser: String
PhoneNum: String
selectedUsername: String
phone: String
fileUrl: String
msg: String
onCreate(savedInstanceState: Bundle): void
fetchUsersData(excludedEmail: String): void
getPhoneNumByUsername(username: String): String
getEmailByUserPublickey(username: String): String
showConfirmationDialog(phoneNo: String, message: String): void
sendSMSMessage(phoneNo: String, message: String, publickey: String): void
findUserIdByEmail(email: String, listener: OnUserIdFoundListener): void
saveShareMetadata(shareMetadata: ShareMetadata): void
checkAndAddUserToShareMetadata(fileKey: String, userId: String, newMetadata: ShareMetadata): void
1
1
1
1
*
*
FingerPrintAuthenticator
authenticationCallback: AuthenticationCallback
context: Context
FingerPrintAuthenticator(Context, AuthenticationCallback)
showSignInBiometricPrompt(): void
showSignUpBiometricPrompt(): void
showBiometricPrompt(BiometricPrompt.PromptInfo): void
onAuthenticationError(int, CharSequence): void
onAuthenticationSucceeded(AuthenticationResult): void
onAuthenticationFailed(): void
Login
t1: TextView
SignInButton: Button
email: EditText
auth: FirebaseAuth
reference: DatabaseReference
onCreate(savedInstanceState: Bundle): void
isCredentialsValid(): boolean
opensignup(): void
doesEmailExist(listener: EmailCheckListener): void
FolderMetadata
files: List<FileMetadata>
subfolders: List<FolderMetadata>
key: String
folderId: String
folderName: String
{constructor} FolderMetadata()
{constructor} FolderMetadata(folderId: String, folderName: String)
getKey(): String
setKey(key: String): void
getFolderId(): String
setFolderId(folderId: String): void
getFiles(): List<FileMetadata>
getFolderName(): String
setFolderName(folderName: String): void
FileMetadata
key: String
fileName: String
fileDownloadUrl: String
folderId: String
favorite: boolean
page: String
FileMetadata()
FileMetadata(fileName: String, fileDownloadUrl: String)
FileMetadata(fileName: String, fileDownloadUrl: String, folderId: String)
getFileName(): String
getFileDownloadUrl(): String
getFolderId(): String
setFileName(newName: String): void
getKey(): String
setKey(key: String): void
isFavorite(): boolean
setFavorite(favorite: boolean): void
getPage(): String
setPage(page: String): void
Crypto
TRANSFORMATION: String
KEY_SIZE: int
encryptFile(inputFile: InputStream, context: Context, fileName: String): byte[]
decryptFile(encryptedInputStream: InputStream, context: Context, fileName: String): byte[]
generateUniqueKey(): String
storeKeyInSharedPreferences(context: Context, originalFileName: String, key: String): void
getKeyFromSharedPreferences(context: Context, originalFileName: String): String
updateKeyInSharedPreferences(context: Context, originalFileName: String, newFileName: String): void
storePrivateKeyInSharedPreferences(context: Context, userEmail: String, privateKey: String): void
getPrivateKeyFromSharedPreferences(context: Context, userEmail: String): String
Login2
userEmailTextView: TextView
pass: TextView
forget: TextView
help: TextView
wheelView: WheelView
password: String
userfinal: String
login: Button
auth: FirebaseAuth
reference: DatabaseReference
usertry: int
fingerprintAuthenticator: FingerPrintAuthenticator
scolor: int
onCreate(savedInstanceState: Bundle): void
rotateClockwise(view: View): void
rotateCounterClockwise(view: View): void
openforget(): void
getColorAttribute(userEmail: String, callback: ColorCallback): void
getUserData(userEmail: String, callback: UserDataCallback): void
getColorInt(colorName: String): int
FolderActivity
folderId: String
folderName: String
shared: final int
search: final int
CHOSE_PDF_FROM_DEVICE: final int
PICK_WORD_FILE: final int
currentUserId: String
performSearch(searchText: String): void
callChoosePdfFile(): void
callChooseWordFile(): void
onActivityResult(requestCode: int, resultCode: int, data: Intent): void
checkIfFileExists(fileName: String, folderId: String, userId: String, callback: FileExistsCallback): void
getFolderId(): String
CreateSubFolder(): void
FileAdapter
itemsList: List<Object>
context: Context
currentPosition: int
currentActivityName: String
count: int
{constructor} FileAdapter(itemsList: List<Object>, context: Context)
showRenameDialog(itemView: View): void
updateFolderName(parentKey: String, childKey: String, newName: String, table: String): void
checkIfFileNameExists(parentKey: String, childKey: String, newName: String, table: String): void
updateFileName(parentKey: String, childKey: String, newName: String, table: String): void
deleteData(parentKey: String, username: String, table: String): void
addFileToFavorites(parentKey: String, username: String, table: String): void
removeFileFromFavorites(parentKey: String, username: String, table: String): void
deleteFolderOnly(parentKey: String, username: String): void
downLoadFile(context: Context, fileName: String, destinationDirectory: String, url: String): void
WheelView
static final int NUM_SECTORS = 8
static final int EDGE_COLOR = Color.RED
static final int EDGE_WIDTH = 20
static final String[] ALPHABETS
static final int[] SECTOR_COLORS
static final int LINE_COLOR = Color.BLACK
static final int LINE_WIDTH = 5
int I
int usertry
Paint edgePaint
Paint sectorPaint
Paint linePaint
Paint textPaint
Random random
RectF sectorRect
float rotationAngle
int selectedSector
TextView textView
WheelView(Context context)
void setTextView(TextView textView)
WheelView(Context context, AttributeSet attrs)
void init()
void shuffleAlphabets()
void onDraw(Canvas canvas)
boolean onTouchEvent(MotionEvent event, int Color, String realPass)
int getTouchedSector(float x, float y)
String getColorName(int color)
void rotateClockwise()
void rotateCounterClockwise()
boolean checkpassword(String x, int y, String realpass)
void showAlertDialog(String title, String message, Class<?> targetActivityClass)

Figure 9 (Class Diagram)

4.3.3 Component Level Design

4.3.3.1 *Add Files*

- Flowchart

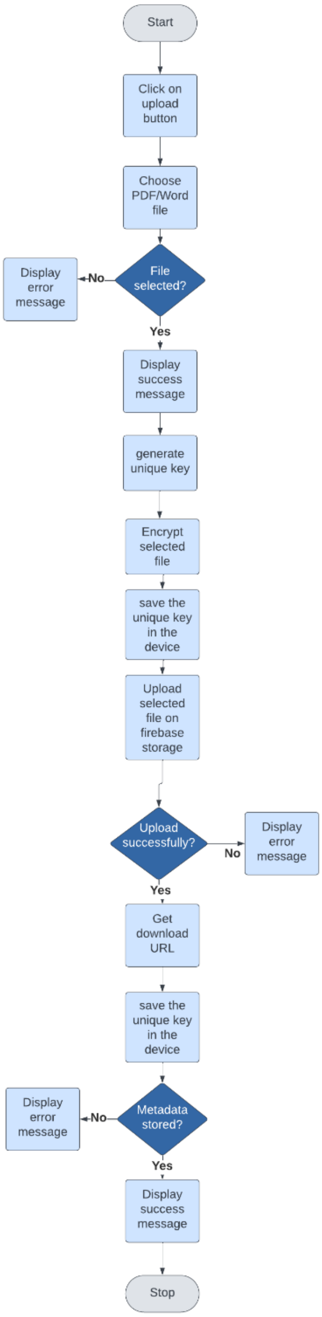


Figure 10 (Add files flowchart)

- Pseudocode

```
Function callChoosePdfFile():
    Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT)
    intent.addCategory(Intent.CATEGORY_OPENABLE)
    intent.setType("application/pdf")
    startActivityForResult(intent, CHOSE_PDF_FROM_DEVICE)
Function onActivityResult(requestCode, resultCode, resultData):
    If requestCode is CHOSE_PDF_FROM_DEVICE and resultCode is Activity.RESULT_OK:
        If resultData is not null:
            Uri fileUri = resultData.getData()
            If fileUri is not null:
                String fileName = getFileNameFromUri(fileUri) // Get the original file name
                // Encrypt the file
                InputStream inputStream = getContentResolver().openInputStream(fileUri);
                      UniqueKey= generateUniqueKey
                         storeKeyInSharedPreferences(context, fileName, uniqueKey);
                byte[] encryptedBytes = Crypto.encryptFile(inputStream, UniqueKey);
                If encryptedBytes is not null:
                    // Upload the encrypted file to Firebase Storage
                    String encryptedFileName = fileName
                    StorageReference fileReference = storageReference.child(currentUserId + "/" + encryptedFileName)
                    UploadTask uploadTask = fileReference.putBytes(encryptedBytes)
                    uploadTask.addOnSuccessListener(taskSnapshot -> {
                        DisplaySuccessMessage("PDF uploaded successfully!")
                        // Get the download URL of the uploaded file
                        fileReference.getDownloadUrl().addOnSuccessListener(uri -> {
                            String fileDownloadUrl = uri.toString()
                            // Store file metadata in the Realtime Database
                            FileMetadata fileMetadata = new FileMetadata(encryptedFileName,
fileDownloadUrl)

databaseReference.child("files").child(currentUserId).push().setValue(fileMetadata)
                        })
                    }).addOnFailureListener(e -> {
                        DisplayErrorMessage("Failed to upload encrypted PDF: " + e.getMessage())
                    }).addOnProgressListener(snapshot12 -> progressBar.setVisibility(View.VISIBLE))
                Else:
                    DisplayErrorMessage("Encryption failed.")
            End If
        End If
    ElseIf requestCode is PICK_WORD_FILE and resultCode is Activity.RESULT_OK:
        // Handle picking Word file similarly as above
    End If


Function getFileNameFromUri(Uri uri):
    String fileName = "unknown"
    Cursor cursor = null
    Try:
        cursor = getContentResolver().query(uri, null, null, null, null)
        If cursor is not null and cursor.moveToFirst():
```

```
        int displayNameIndex = cursor.getColumnIndex(OpenableColumns.DISPLAY_NAME)
        If displayNameIndex is not -1:
            fileName = cursor.getString(displayNameIndex)
Finally:
    If cursor is not null:
        cursor.close()
Return fileName
```

4.3.3.2 Fingerprint Authentication

- Flowchart

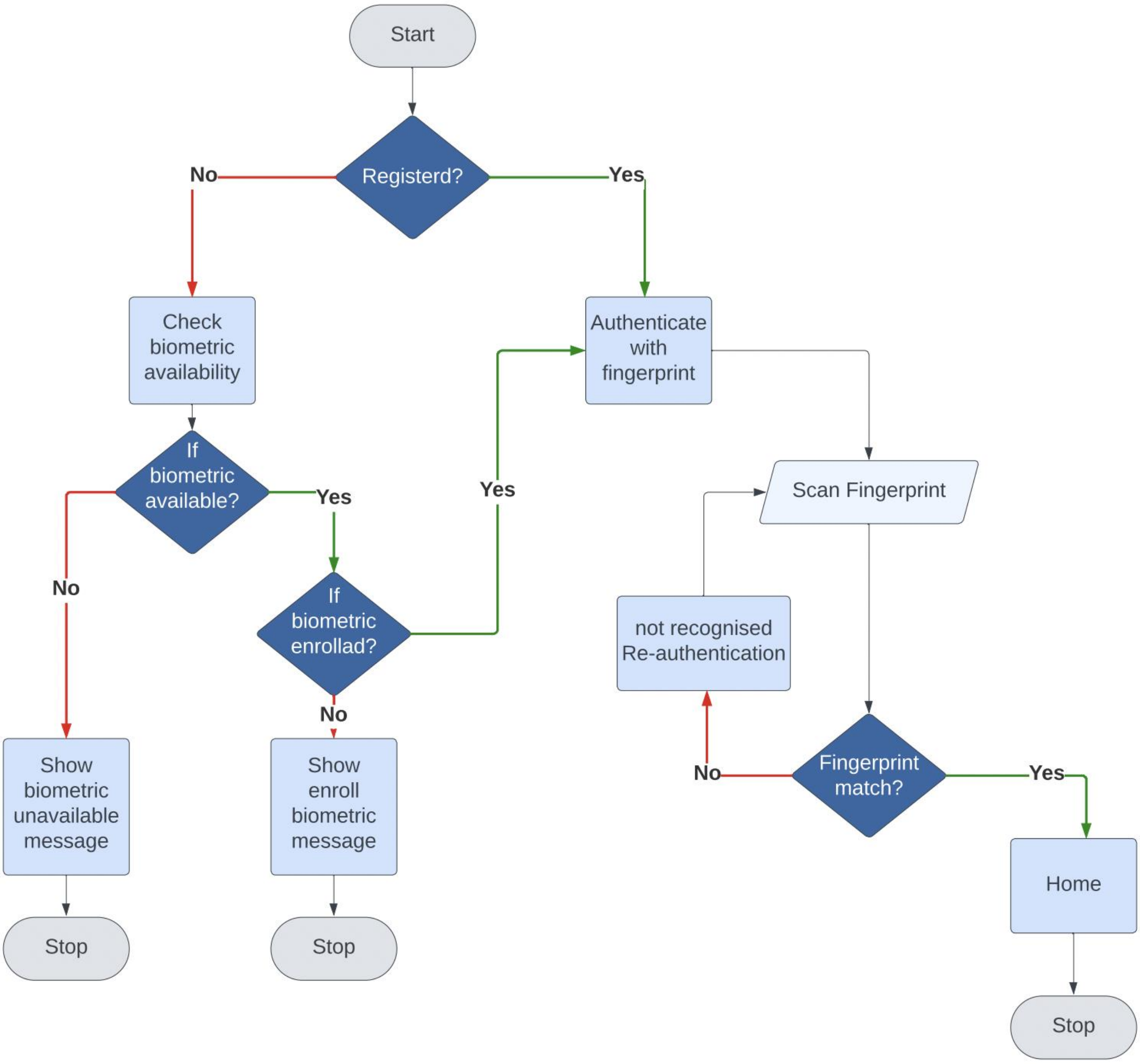

Start
Registerd?
No
Yes
Check biometric availability
Authenticate with fingerprint
If biometric available?
No
Yes
Yes
Scan Fingerprint
If biometric enrollad?
No
not recognised Re-authentication
Show biometric unavailable message
Show enroll biometric message
No
Fingerprint match?
Yes
Home
Stop
Stop
Stop

Figure 11 (Fingerprint authentication flowchart)

4.3.3.3 Text-based graphical Password

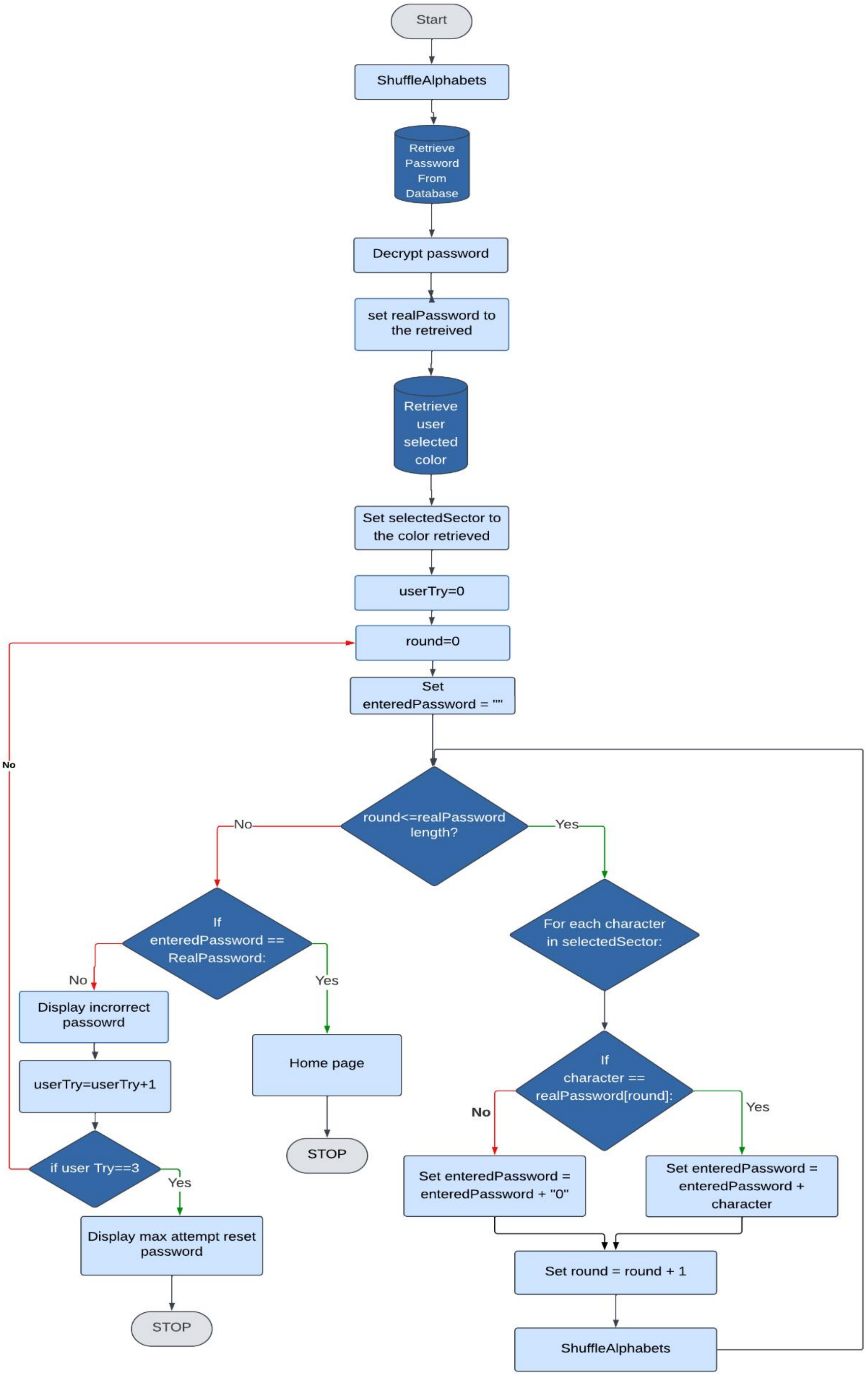

Start
ShuffleAlphabets
Retrieve Password From Database
Decrypt password
set realPassword to the retreived
Retrieve user selected color
Set selectedSector to the color retrieved
userTry=0
round=0
Set enteredPassword = ""
No
round<=realPassword length?
No
Yes
If enteredPassword == RealPassword:
No
Yes
Display incrorrect passowrd
userTry=userTry+1
if user Try==3
Yes
Display max attempt reset password
STOP
Home page
STOP
For each character in selectedSector:
If character == realPassword[round]:
No
Yes
Set enteredPassword = enteredPassword + "0"
Set enteredPassword = enteredPassword + character
Set round = round + 1
ShuffleAlphabets

Figure 12 (Text-based graphical password flowchart)

- Pseudocode

```
//Retrieve stored password from the database
RealPassword = RetrievePasswordFromDatabase()
sendPostRequest("http://192.168.100.46:8000/test", password, response -> {
    if (response != null) {
        responseNoSpaces = removeSpaces(response)
        runOnUiThread(() -> {
            logMessage("Response from server: " + response)
            decryptpass = responseNoSpaces
            logMessage("password: " + decryptpass)
            wheelView.setOnTouchListener((v, event) -> {
                wheelView.onTouchEvent(event, scolor, decryptpass)
                return true
            })
        })
    }
})
// Variable to track the number of attempts
userTry = 0
// Retrieve selected color from the user input
selectedSector = UserSelectedColor()
// Set round counter
round = 0
// Variable to store the entered password
enteredPassword = ""

//Loop through each round until round <= N (length of password)
while round <= length(decryptpass):
    //Shuffle the alphabets
    shuffleAlphabets()
    //Iterate through characters in the selected sector
    for each character in selectedSector:
        // If the character matches the i-th character of the password
        if character == decryptpass [round]:
            enteredPassword = enteredPassword + character
            round = round + 1
        else:
            enteredPassword = passwordInput + "0"
            round = round + 1
    // Check if the entered password matches the stored password
    if enteredPassword == decryptpass:
        // Password verification successful
        displayHomepage()
        exit()
    else:
        // Incorrect password, increment login attempts
        userTry = userTry + 1

    // Check if maximum attempts reached
    if userTry >= 3:
        displayMaxAttempts()
        resetPassword()
        exit()
```

```
function shuffleAlphabets():
alphabetList = createList(ALPHABETS) // Create a new list with the alphabets
shuffle(alphabetList) // Shuffle the elements in the list
ALPHABETS = toArray(alphabetList) // Convert the shuffled list back to an array

end function.
```

4.4 Data Design

In data design, we will express our data model using ER diagrams and non-relational data models.

4.4.1 Data Models

4.4.1.1 ER Diagram

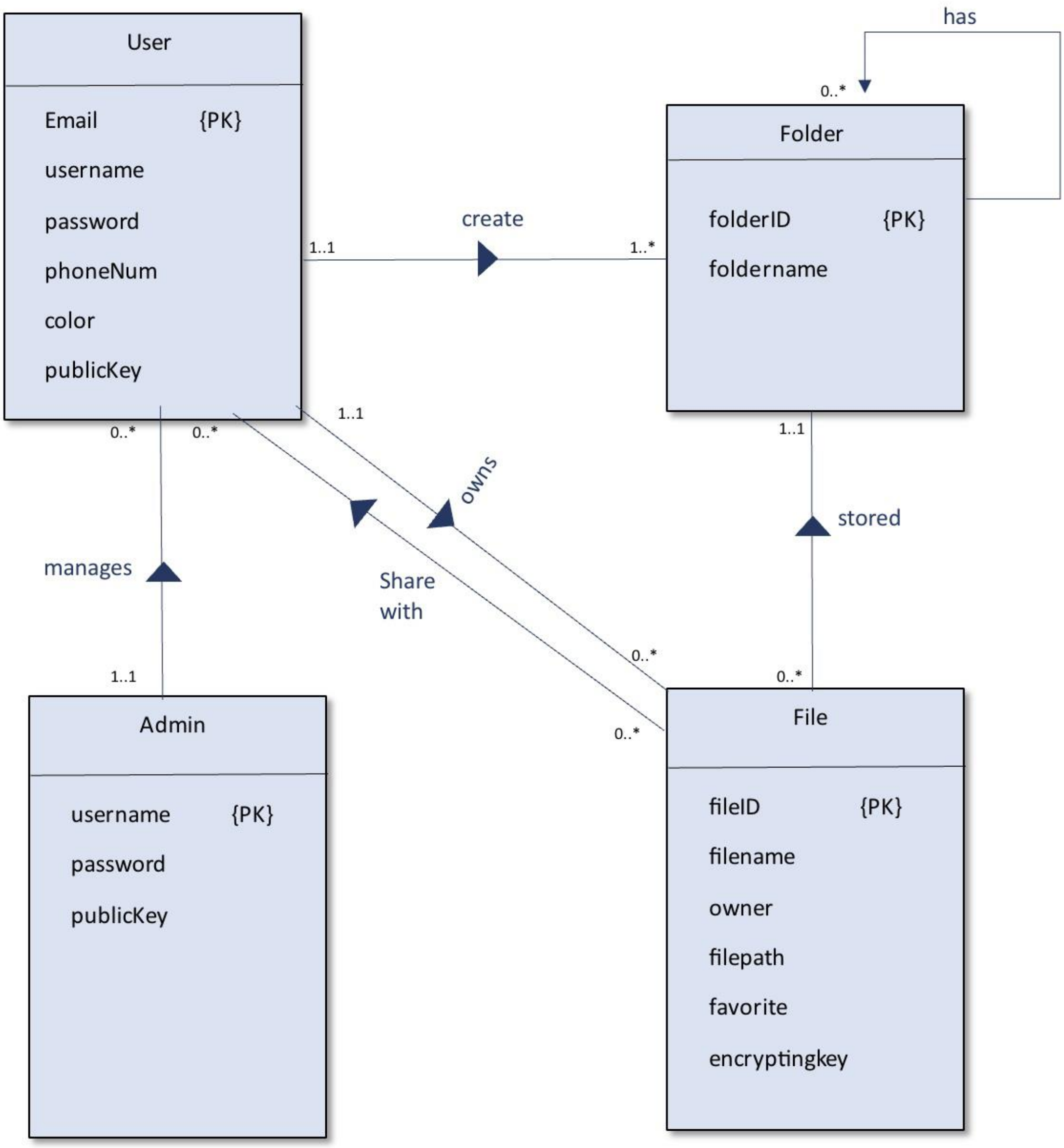
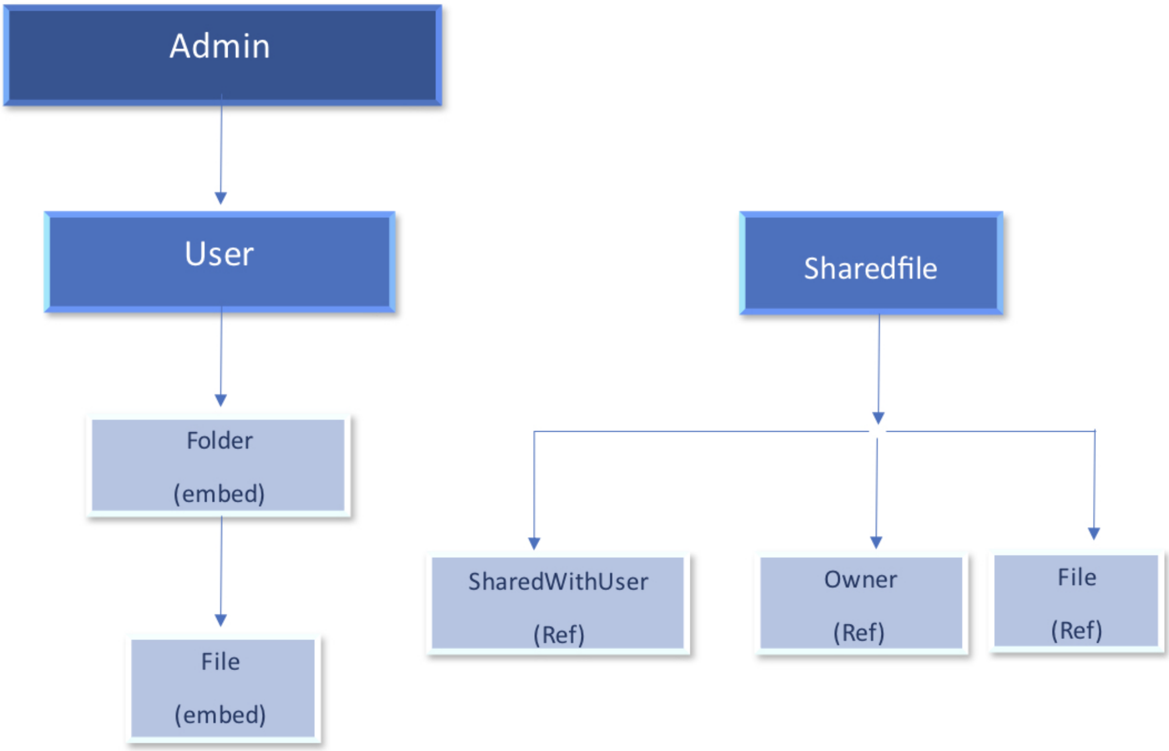
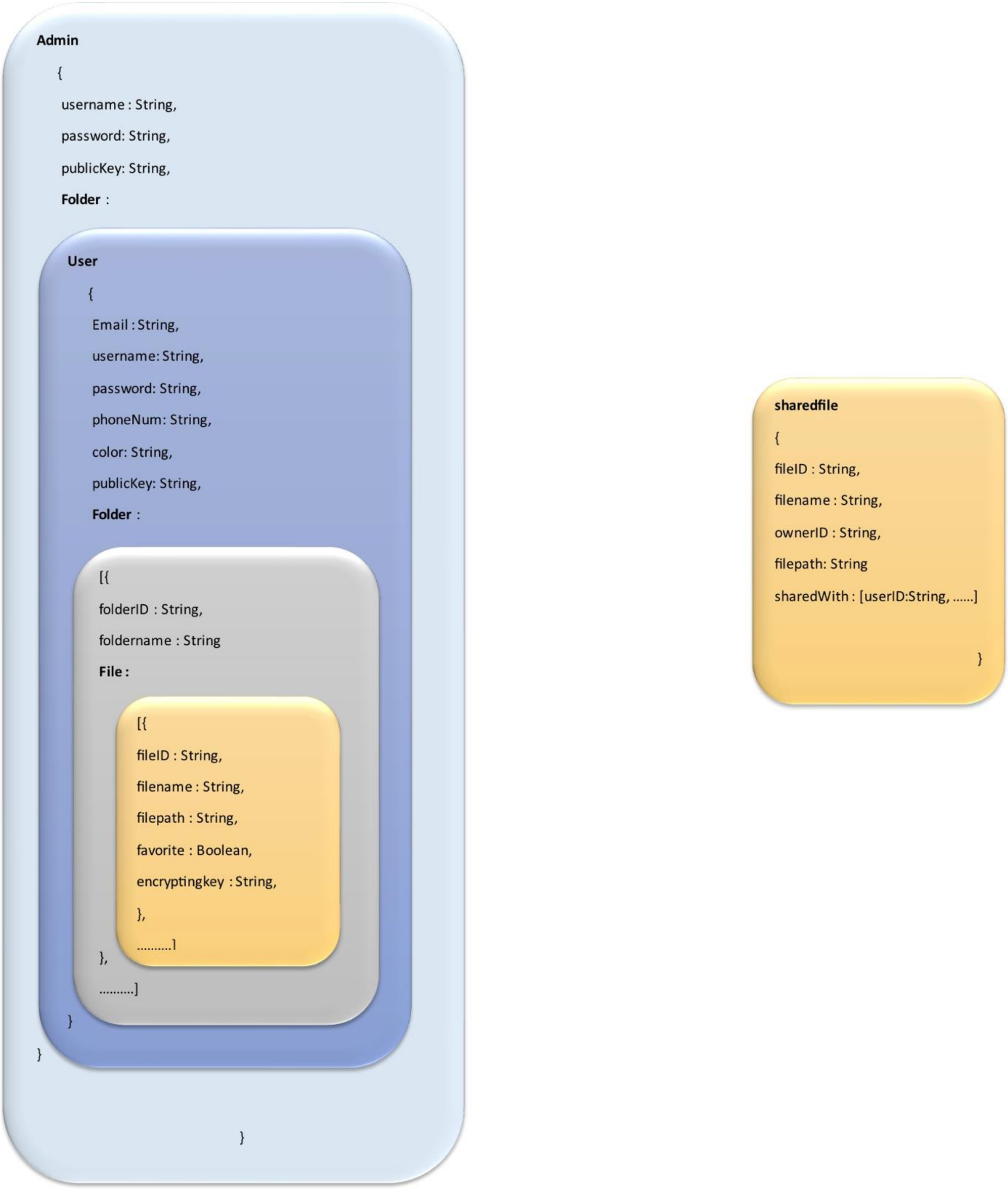


Figure 13 (ER diagram)

4.4.1.2 Non-Relational Data Model.





```
Admin
    {
    username : String,
    password: String,
    publicKey: String,
    Folder :
        User
            {
            Email : String,
            username: String,
            password: String,
            phoneNum: String,
            color: String,
            publicKey: String,
            Folder :
                [{
                folderID : String,
                foldername : String
                File :
                    [{
                    fileID : String,
                    filename : String,
                    filepath : String,
                    favorite : Boolean,
                    encryptingkey : String,
                    },
                    ……….]
                },
                ……….]
        }
}
                    }

sharedfile
{
fileID : String,
filename : String,
ownerID : String,
filepath: String
sharedWith : [userID:String, ……]
                        }
```

Figure 14 (Non-Relational Data Model)

4.5 Interface Design

4.5.1 Site Map

4.5.1.1 User Site Map

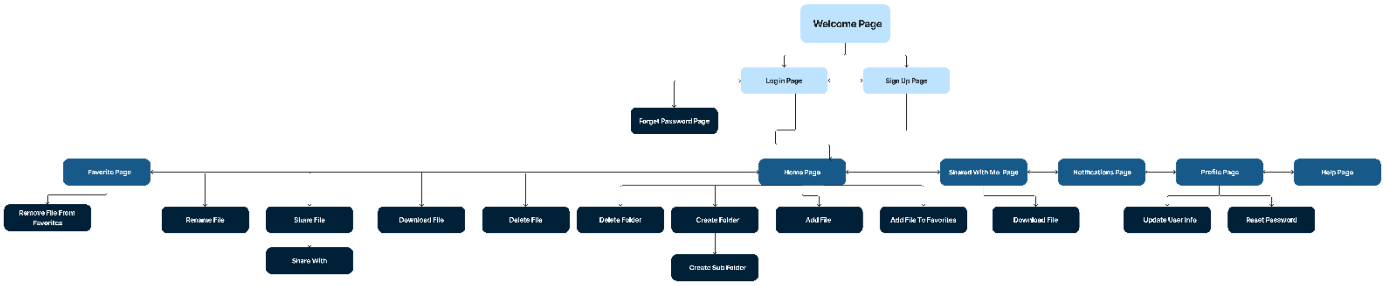


Figure 15 (User Sitemap)

4.5.1.2 Admin Site Map

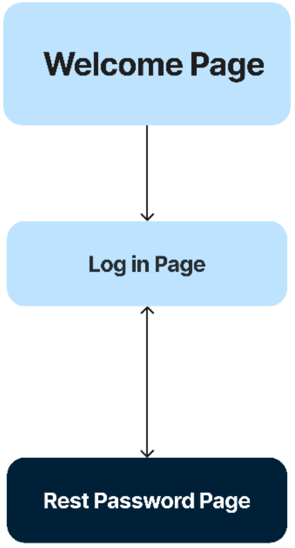


Figure 16 (Admin Sitemap)

4.5.2 UX Guidelines

- Error handling:

  Our app excels in error handling, validating user inputs seamlessly. It provides clear and prompt feedback to guide users in accurate data entry, enhancing the overall user experience.

- Security:

  Our app prioritizes security by using a combination of fingerprint authentication for quick and secure access and text-based graphical passwords for traditional security measures. This multi-layered approach ensures convenience and resilience.

- Navigation:

  Navigation in our app is designed to be simple and quick, aiming to provide users with an easy and efficient way to move between pages. The user-friendly navigation system ensures a seamless experience, allowing users to effortlessly access different sections with minimal effort.

- Consistency:

  Our app uses a uniform color scheme across all pages, promoting a consistent and recognizable visual style. Each page shares similarities, making navigation easy and fostering a sense of familiarity.

- Feedback and Confirmation:

  When users interact with our app, we prioritize providing clear feedback and confirmation to keep them informed about their actions. This ensures users have a seamless and confident experience, knowing exactly what they've accomplished within the app.

4.6 Implementation

4.6.1 Software Tools

Table 4 (Software Tools)

Software type	Version	Description
Android Studio	2023.2.1 patch 2	Android Studio is the official integrated development environment (IDE) for Google's Android operating system. Built on JetBrains' IntelliJ IDEA software, it provides a comprehensive set of tools for developing Android apps.
Firebase	-	Cloud database service used to store data.
GitHub	3.12	A code hosting platform used to manage and integrate source code.
Apache NetBeans	2.0.	NetBeans is an integrated development environment for Java. NetBeans allows applications to be developed from a set of modular software components called modules. NetBeans runs on Windows, macOS, Linux and Solaris.

4.6.2 Software Implementation

The SecureBox application was developed using Android Studio and Java, placing a strong emphasis on user authentication and data security. Key features include the successful implementation of two-factor authentication and fingerprint authentication, along with the development of a text-based graphical password system. Integration with Firebase ensures secure storage of user information. Additionally, users can securely create, manage, and share files within the application. On the server side, we utilized NetBeans and Java to enhance the application's security, particularly in assisting with user password encryption. The system is designed to ensure that all administrative actions are authenticated using admin credentials.

4.6.3 Major Implementation

Our primary goal during the major implementation phase was to seamlessly integrate two-factor authentication into our application. This involved combining fingerprint authentication with a Text-based Graphical Password system for enhanced security. Concurrently, we optimized data management using Firebase, ensuring a robust environment for user interactions. Furthermore, we developed a unique Text-based Graphical Password system, providing users with a distinct and secure method of authentication.

Importantly, for files added by users, we implemented robust encryption practices. Each file undergoes symmetric encryption, with a unique key generated upon uploading to our app, enhancing security. This key is stored on the user's phone, ensuring the confidentiality and integrity of user data.

Moreover, our development efforts have enabled users to effortlessly add and manage files within the application, thus enhancing both data security and user-friendliness.

Additionally, we constructed our own server-side app to ensure that user passwords are encrypted using asymmetric encryption and stored securely in the database, further enhancing security.

- ### Text-based Graphical Password system:

1. **Getting user email**
2. **Retrieving user data (password, password color)**

```
getUserData(userEmail, (userpass, userColor) -> {
    if (userpass != null && userColor != null) {
        scolor = getColorInt(userColor);
        password = userpass;
        // Perform actions with the retrieved user data
    } else {
        // Handle the case where user data is not available & display an error message or take
appropriate action
    }
});
```

3. **Setting onTouchListener for the WheelView.**

   This code sets an onTouchListener for the wheelView custom view. When a touch event occurs, it calls the onTouchEvent() method of the wheelView and passes the event, color (scolor), and password.

```
wheelView.setOnTouchListener((v, event) -> {
// Call onTouchEvent() method of the wheelView and pass the event, color, and password
wheelView.onTouchEvent(event, scolor, password);
return true.
});
```

4. **on the on-Touch Event:**

   A StringBuilder named selectedLetters is used to store the selected letters from the touched sector.

```
1.StringBuilder selectedLetters = new StringBuilder();
// Loop through 8 letters in the sector
for (int i = 0; i < 8; i++) {
   int letterIndex = (sectorStartIndex + i) % ALPHABETS.length;
  // Append the letter and a hyphen to the selectedLetters StringBuilder
selectedLetters.append(ALPHABETS[letterIndex] + "-");
}
```

The checkpassword()function checks if the entered password contains the character at the specified index of the real password and returns a boolean value accordingly.

```
2. // Check if the entered password matches the real password
if (checkpassword(selectedLetters, I, realPass)) {
    // Append the character at index I from the real password to the current text
    currentText += realPass.charAt(I);
    // Update the textView with the new current text
    textView.setText(currentText);
// If it's the last character of the real password, display a "Done" toast message
    if (realPass.length() - 1 <= I) {
        Toast.makeText(getContext(), "Done", Toast.LENGTH_SHORT).show();
    } else {
        // Otherwise, increment the index I, shuffle the alphabets, and invalidate
the view
        I++;
        shuffleAlphabets();
        invalidate();
    }
} else {
    // If the entered password does not match, reset the index I to 0
    I = 0;
    // Append the first character of the real password to the current text
    currentText += realPass.charAt(I);
    // Update the textView with the new current text
    textView.setText(currentText);
    // Shuffle the alphabets and invalidate the view
    shuffleAlphabets();
    invalidate();
}
```

- **File Encryption algorithm.**

This code demonstrates file encryption and decryption using the AES algorithm. It provides methods for encrypting and decrypting files, generating unique keys, storing keys in Shared Preferences, and retrieving keys for decryption.

1.encryptFile Method:

```
public static byte [] encryptFile(InputStream inputFile, Context context, String fileName)
```

This method encrypts the input file using the AES algorithm and returns the encrypted data along with a unique key. It generates a new unique key for each file, stores it in SharedPreferences, and encrypts the file using the generated key.

Inside the method, a new unique key is generated using the generateUniqueKey method. This key is then stored in SharedPreferences using the storeKeyInSharedPreferences method, associating it with the provided file name.

The method initializes a Cipher object with the AES transformation and the generated key. It reads the contents of the input file in chunks and writes the encrypted data to a ByteArrayOutputStream using a CipherOutputStream. Finally, it returns the encrypted data as a byte array.

2.decryptFile Method:

```
public static byte [] decryptFile (InputStream encryptedInputStream, Context context, String fileName)
```

This method decrypts the encrypted file using the associated key stored in SharedPreferences. It retrieves the key based on the original file name and uses it to decrypt the file, returning the decrypted data.

Inside the method, the key is retrieved from SharedPreferences using the getKeyFromSharedPreferences method. If the key is found, a Cipher object is initialized with the AES transformation and the retrieved key.

The method reads the encrypted input file in chunks and decrypts the data using a CipherInputStream. The decrypted data is then returned as a byte array.

3.generateUniqueKey Method:

```
private static String generateUniqueKey()
```

This method generates a new unique key for each file. It creates a 128-bit random key and encodes it to a Base64 string representation. The generated key is returned as a string.

4.6.4 API Implementation

Implementing the **Android Biometric API** for biometric authentication in our application, we follow a straightforward process. First, we check the availability of biometric authentication on the user's device using the Biometric API. This ensures that the device supports biometric authentication, and that the user has enrolled their biometric credentials.

Once biometric availability is confirmed, we prompt the user to authenticate using their biometric credentials. This is accomplished by invoking the appropriate API method, which triggers a system-provided biometric prompt. The prompt interface guides the user through the authentication process, ensuring a seamless and intuitive experience.

To handle the authentication result, we implement the necessary callbacks or listeners provided by the Biometric API. These callbacks inform us of the authentication outcome, whether it is a successful authentication, a failure, or if the user cancels the process.

To further strengthen the security of our application, we combined the Android Biometric API with our existing text-based graphical password system. This creates a two-factor authentication approach, requiring both biometric authentication and the correct graphical password for successful login and access to sensitive data. This additional layer of authentication adds an extra level of protection and ensures that only authorized users can access the application.

Throughout the entire implementation process, we prioritize the confidentiality of user data, particularly biometric information. We assure our users that all biometric data entered into our application remains secret and is never shared or saved anywhere other than their own device. By adhering to the best practices and guidelines provided by the Android Biometric API, we ensure that our application offers a secure and convenient login experience while maintaining the privacy of our users' biometric data.

- **Implementing fingerprint Biometric API into our app:**

For connecting the fingerprint Biometric API into our application, we did as follow:

1. Adding the androidx biometric library in our dependencies.

```
implementation 'androidx.biometric:biometric:1.1.0'
```

2. Then import the necessary Android classes and libraries.

```
import androidx.biometric.BiometricPrompt;
import androidx.biometric.BiometricManager;
import androidx.core.content.ContextCompat;
import androidx.fragment.app.FragmentActivity;

import java.util.concurrent.Executor;
```

3. Implement the method to show the biometric prompt for sign-in & sign-up.

```
public void showSignInBiometricPrompt() {
   // Build the prompt info for sign-in
   BiometricPrompt.PromptInfo promptInfo = new BiometricPrompt.PromptInfo.Builder()
         .setTitle("Sign in")
         .setSubtitle("Touch the fingerprint sensor to sign in")
         .setNegativeButtonText("Cancel")
         .build();
   // Show the biometric prompt
   showBiometricPrompt(promptInfo);
}
```

4. Implementing the method to handle the biometric prompt.

```
private void showBiometricPrompt(BiometricPrompt.PromptInfo promptInfo) {
   // Check if the device supports biometric authentication
   BiometricManager biometricManager = BiometricManager.from(context);
   switch (biometricManager.canAuthenticate(BiometricManager.Authenticators.BIOMETRIC_STRONG)) {
      case BiometricManager.BIOMETRIC_SUCCESS:
         Log.d("MY_APP_TAG", "App can authenticate using biometrics." );
         break;
      case BiometricManager.BIOMETRIC_ERROR_NO_HARDWARE:
         Log.e("MY_APP_TAG", "No biometric features available on this device.");
         break;
      case BiometricManager.BIOMETRIC_ERROR_HW_UNAVAILABLE:
         Log.e("MY_APP_TAG", "Biometric features are currently unavailable.");
         break;
      case BiometricManager.BIOMETRIC_ERROR_NONE_ENROLLED:
         Log.e("MY_APP_TAG", "Please enroll your fingerprint in the settings.");
         break;
   }
   // Create an executor
   Executor executor = ContextCompat.getMainExecutor(context);
   BiometricPrompt biometricPrompt = new BiometricPrompt((FragmentActivity) context, executor, this);
   // Show the biometric authentication prompt
   biometricPrompt.authenticate(promptInfo);  }
```

5. Call the appropriate method to show the biometric prompt. For example, to show the sign-in prompt.

```
authenticator.showSignInBiometricPrompt();
```

In addition to the mentioned implementation, we have also incorporated the use of **SMS messages** to securely share the necessary information with the intended recipient. When sharing a file with a shared user, we send an SMS message containing the file name, username, and the file key required for decrypting the file content. This ensures that only the authorized recipient can access and decrypt the file.

To implement this functionality, the following steps are taken:

1. **Importance**: The use of SMS messages adds an extra layer of security to our application by securely sharing file details and encryption keys between users. It ensures that sensitive information is transmitted directly to the intended recipient.
2. **Permissions**: To send SMS messages, our application requires the SEND_SMS permission. This permission allows the application to send SMS messages without user intervention. We request this permission from the user if it has not been granted already.

```
if (ContextCompat.checkSelfPermission(this, android.Manifest.permission.SEND_SMS)
        != PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions(this,
            new String[]{android.Manifest.permission.SEND_SMS},
            MY_PERMISSIONS_REQUEST_SEND_SMS);
}
```

To retrieve SMS messages containing secure file details and encryption keys, our application utilizes the SMS inbox. By extracting the relevant information from these messages, we can securely store and process the received data.

The following highlights the key aspects:

1. **Importance**: Reading SMS messages allows our application to retrieve encrypted file details and encryption keys shared by other users. This functionality ensures that the recipient can access and decrypt files securely.
2. **Permissions**: Reading SMS messages requires the READ_SMS permission. This permission allows our application to query and retrieve SMS messages from the device's SMS inbox.

```
ContentResolver contentResolver = getContentResolver();
String selection = Telephony.Sms.READ + " = 0 AND " + Telephony.Sms.BODY + " LIKE
?";
String[] selectionArgs = new String[]{"%SECUREBOX%"};
Cursor cursor = contentResolver.query(
        Telephony.Sms.CONTENT_URI,
        null,
        selection,
        selectionArgs,
        null);
```

4.6.5 Implementation Challenges and Difficulties

The implementation of the Text-based Graphical Password system presented several challenges that required innovative solutions and resourceful thinking. One significant difficulty we faced related to the visibility of user passwords within the Text-based Graphical Password algorithm. Unlike traditional password hashing methods, this algorithm required direct access to the user's password to match the selected sector with the characters of the password, raising concerns about password security without compromising the functionality of the algorithm.

To address the challenge of potential password visibility within the Text-based Graphical Password algorithm, we developed a solution involving the incorporation of an additional server-side application. This application is responsible for managing the encryption and decryption processes using asymmetric encryption and securely overseeing the secret key crucial for decrypting user passwords. The introduction of this server-side application established a robust framework that not only ensures the confidentiality of user passwords but also enables the Text-based Graphical Password algorithm to operate seamlessly and securely.

Another challenge we faced was the decryption of files when users download them to their phones; the files must be decrypted so the users can view them. It was difficult to manage decryption upon download because each file has its own key due to our use of symmetric encryption.

In conclusion, during the implementation of the Text-based Graphical Password algorithm, we encountered and addressed critical challenges related to the visibility of user passwords. Through the innovative use of asymmetric encryption, we enhanced the security of the system. We are committed to continuously improving the Text-based Graphical Password system to ensure it remains secure and effective.

GitHub Repository Link:
https://github.com/Sarahkhj/2023-GP1-16.git

System Evaluation

5 System Evaluation

In this section, we evaluate the system through user acceptance testing and quality assessments. This includes analyzing demographics of participants and their feedback from questionnaires and interviews to assess user satisfaction. Additionally, we test the system against non-functional requirements like performance and usability. The results are discussed to highlight the system’s effectiveness and areas for improvement, offering a comprehensive view of its performance and user alignment.

5.1 User Acceptance Testing

In this section, we embark on a comprehensive evaluation of our system through User Acceptance Testing (UAT). Our method involved assembling a testing team comprising end users. This team, comprised of 20 participants meeting specific criteria outlined in section 4.2.1: System users, actively participated in assessing various facets of our application. The evaluation encompassed gauging satisfaction levels with the user interface, assessing technical functionality, pinpointing major weaknesses, and providing an overall evaluation of the application.

To gather insightful data from our target audience, we deployed a meticulously crafted questionnaire with a series of pertinent questions. These questions were tailored to extract valuable insights into user experiences and preferences.

The subsequent sections meticulously detail the UAT testing design process, elucidating our methodology and approach. Additionally, we present the outcomes of this critical evaluation, shedding light on areas of strength and areas requiring improvement.

5.1.1 Demographics of Participants

Our UAT participants exhibited diverse demographics, with a majority being female users (12) compared to males (7). Most participants were between 18-30 years old (12), followed by those aged 31-40 (5) and 41-50 (2), with one participant aged 50 or older. The educational background varied, with a higher representation of bachelor’s degree holders (10), followed by those with a master’s degree (4), diploma (2), and high school (4). Regarding technical proficiency, the majority of participants were experts (8), followed by intermediate (9) and beginners (3), ensuring a comprehensive evaluation of our application.

Table 5 (Demographics of Participants)

Variable	Classification	Value
Gender	Female	12
	Male	7
Age	18 - 30	12
	31- 40	5
	41 - 50	2
	50 < years	1
Education level	High school	4
	Diploma	2
	Bachelor	10
	Master	4
	PhD	0
Technical Proficiency	Beginner	3
	Medium	9
	Expert	8

5.1.2 Questionnaire/Interview Results

In this section, we provide a detailed overview of the outcomes derived from our questionnaire. The questionnaire incorporated a mix of rating scale questions, utilizing a scale from one to five, where one signifies Very Satisfied or Very easy and five indicates Very Dissatisfied or Very hard. Additionally, we included yes/no questions to gather specific insights from the participants. For the complete set of questionnaire questions, refer to the provided appendix

The evaluation covered key aspects of our application, starting with the overall user interface design. (13) participants rated it as 1, (5) as 2, and (2) as 3, showcasing a varied yet generally positive response.

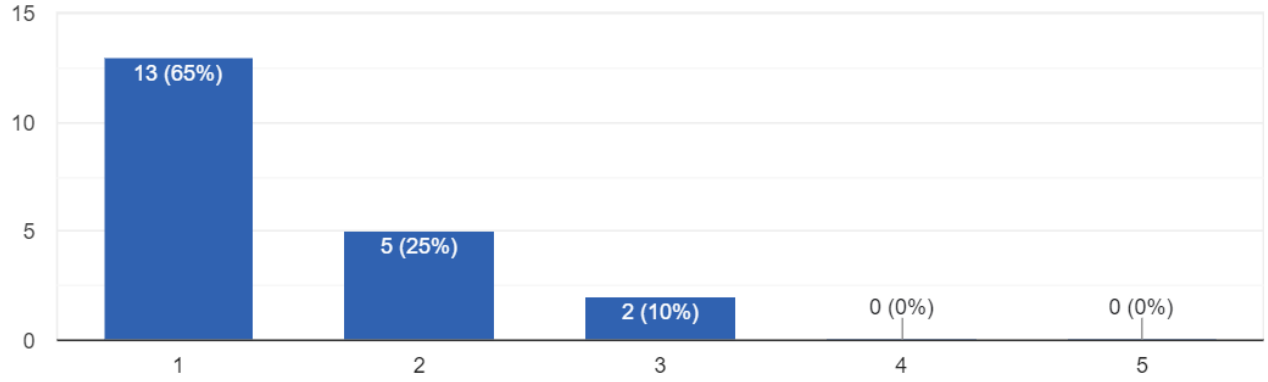


Figure 17 (Questionnaire Results)

All participants provided feedback on their experience with deleting files/folders, with all 20 participants rating it as 1. This indicates a high level of satisfaction with the deletion process, as all participants found it very easy to delete files/folders.

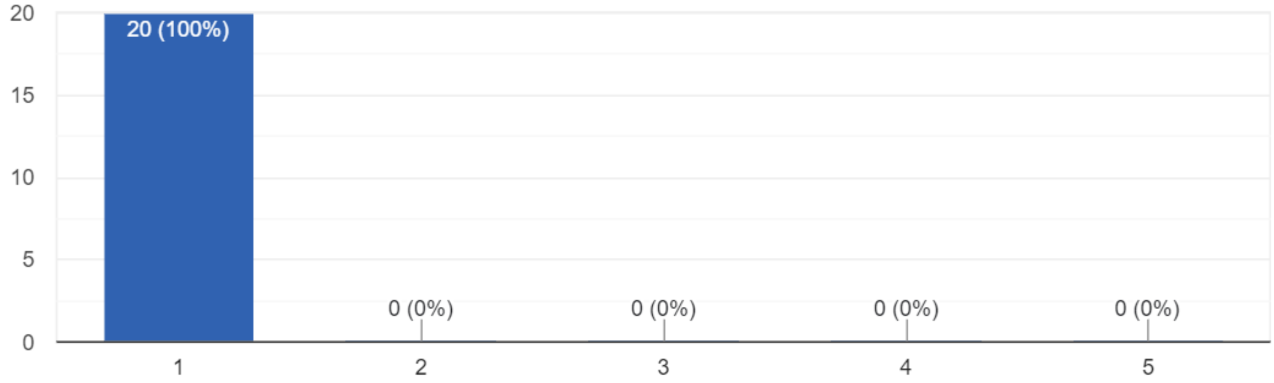


Figure 18 (Questionnaire Results)

As an extension of the previous inquiry, participants were asked whether they were able to successfully delete their files or folders. All 20 respondents confirmed their ability to do so, highlighting a positive user experience in this aspect of our application's functionality.

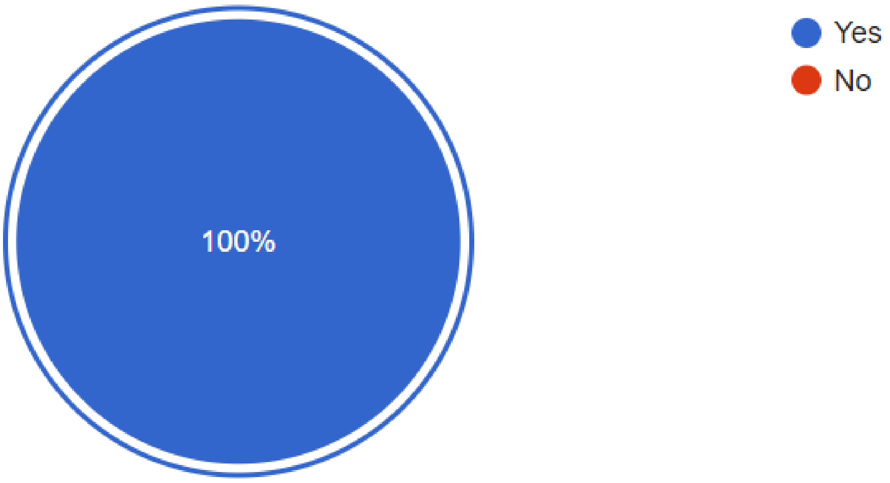


Figure 19 (Questionnaire Results)

Continuing with the assessment, participants were queried regarding their ability to rename files or folders within the application. The entirety of the 20 participants reported success in this task, indicating a smooth and effective renaming functionality within the system.

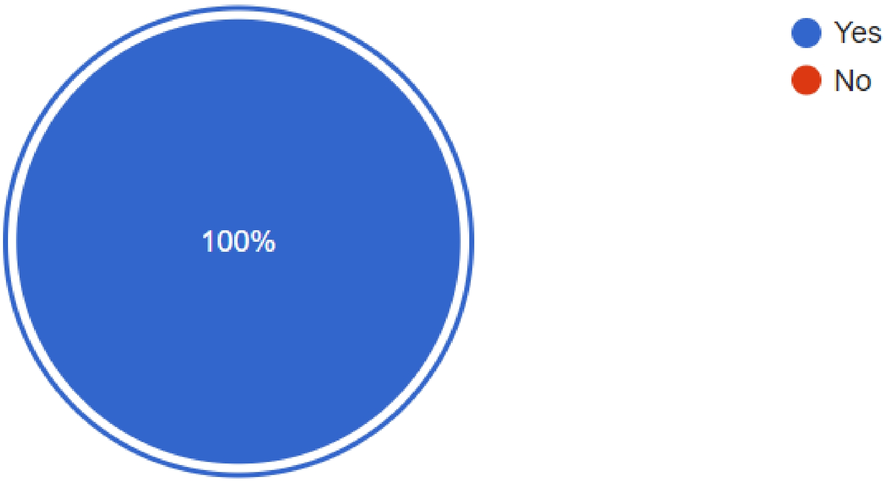


Figure 20 (Questionnaire Results)

The evaluation of accessing and viewing the contents of folders revealed that 18 participants (90%) rated their experience as 1, indicating a high level of satisfaction. Additionally, 2 participants (10%) rated it as 2, suggesting a generally positive experience for the majority of users.

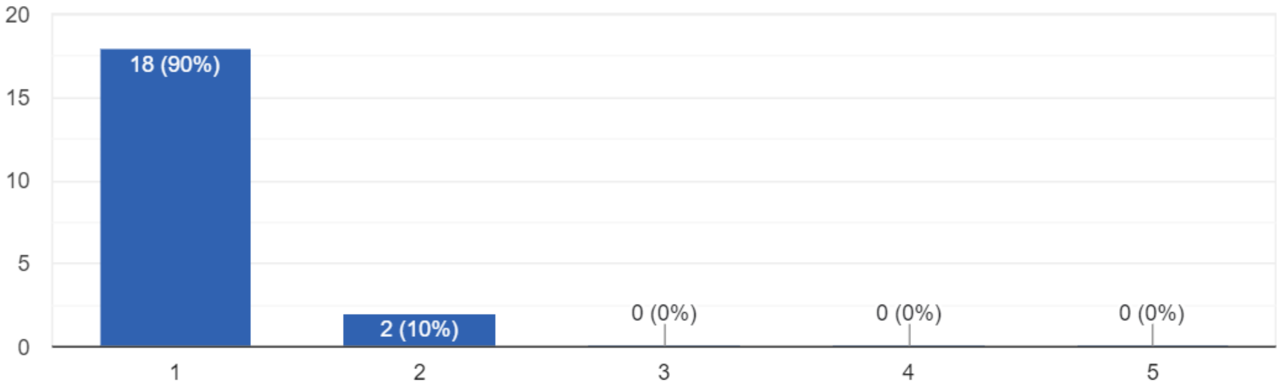


Figure 21 (Questionnaire Results)

Regarding the ability to successfully mark files as favorites, 19 participants (95%) responded affirmatively. However, one participant (5%) indicated they were unable to do so but did not provide any specific reasons for this difficulty.

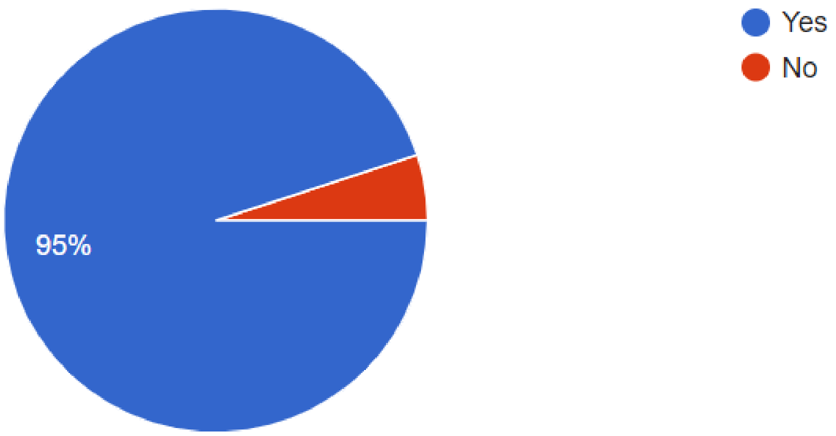


Figure 22 (Questionnaire Results)

In terms of the experience with downloading files, the responses were predominantly positive. Specifically, 18 participants (90%) rated their experience as 1, indicating a high level of satisfaction, while 2 participants (10%) rated it as 2.

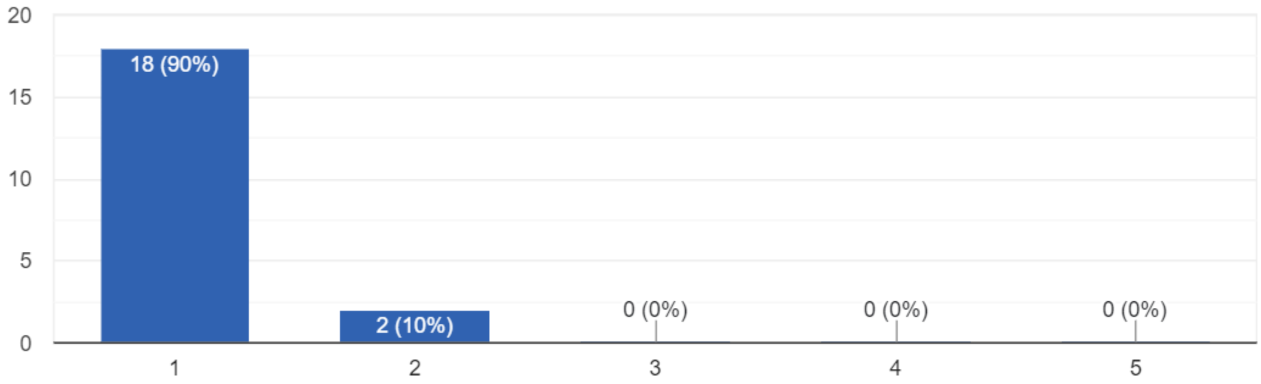


Figure 23 (Questionnaire Results)

All 20 participants reported being able to successfully search for their files, indicating a high level of success in this aspect of the application.

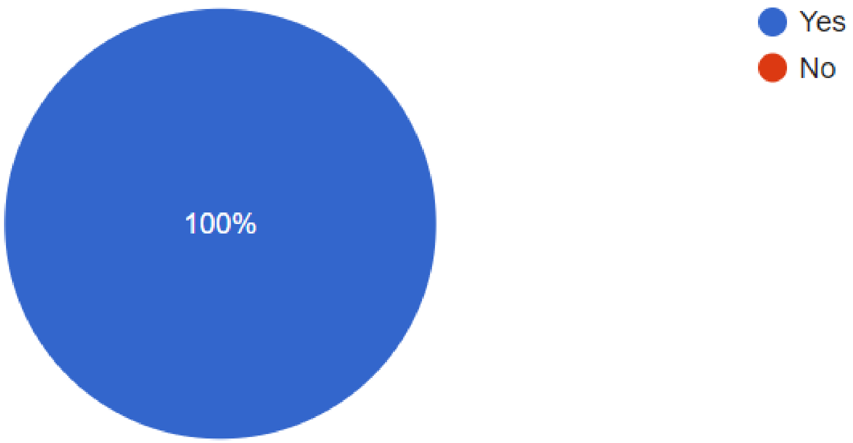


Figure 24 (Questionnaire Results)

All 20 participants confirmed that they were able to successfully filter their files, demonstrating a consistent and positive experience with this functionality.

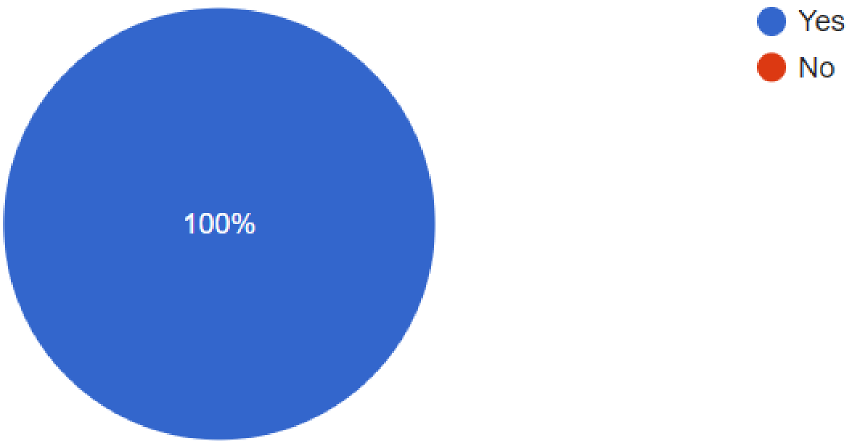


Figure 25 (Questionnaire Results)

All participants affirmed their ability to share files securely through encrypted SMS messages, highlighting the effectiveness of our encryption measures in ensuring data safety during transmission.

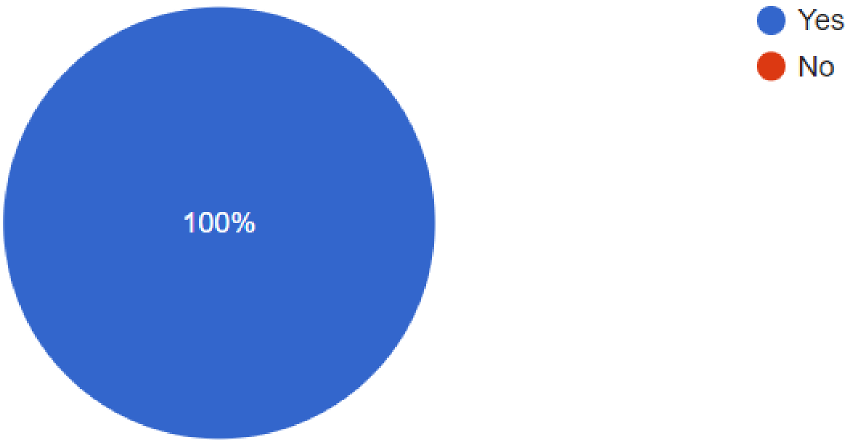


Figure 26 (Questionnaire Results)

All participants reported successful viewing of files shared by other users, indicating seamless functionality in accessing shared content within the application.

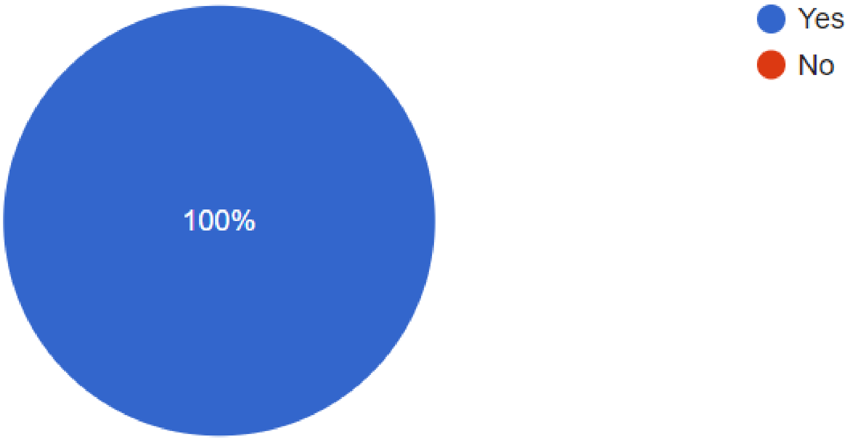


Figure 27 (Questionnaire Results)

All participants confirmed receiving notifications upon file sharing by other users, underscoring the effective notification system implemented within the application.

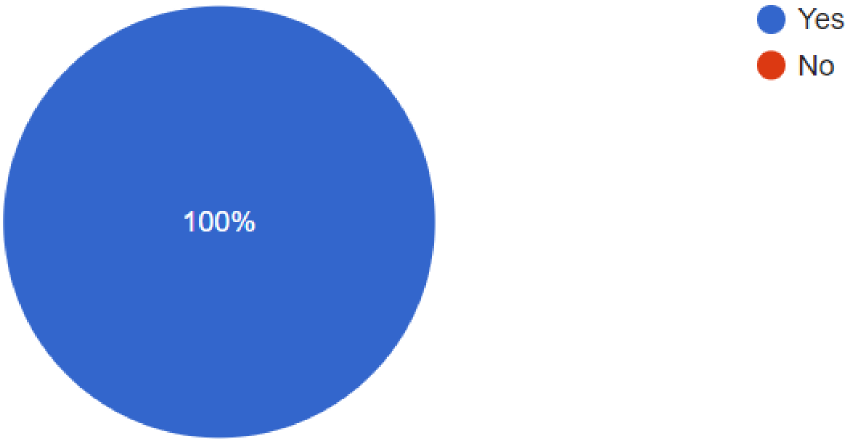


Figure 28 (Questionnaire Results)

Among the participants, 70% reported no major weaknesses or areas requiring improvement. However, 30% identified areas necessitating enhancement. Common concerns included difficulties in dragging files into folders and challenges with the text-based Graphical Password system on the login page, particularly due to small font sizes. Furthermore, participants expressed a desire for additional security measures during password changes or resets, suggesting the inclusion of personal questions to verify the account holder's identity effectively.

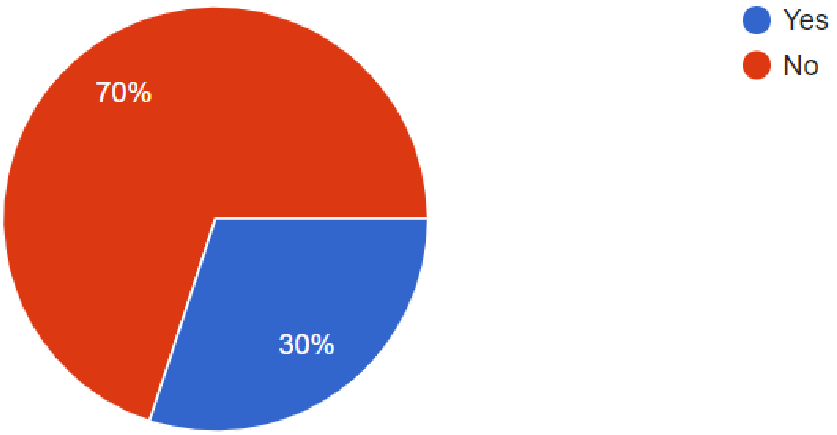


Figure 29 (Questionnaire Results)

While the majority of participants (75%) did not encounter any technical issues while using the application, 25% reported facing some challenges. Although specific problems were not consistently mentioned, some participants highlighted issues related to the application's speed or responsiveness.

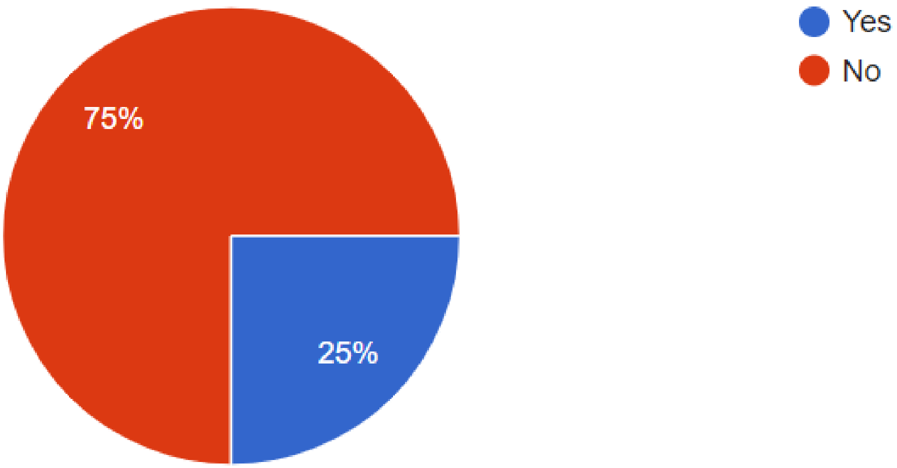


Figure 30 (Questionnaire Results)

Regarding their experience with contacting the support team, responses varied. A significant majority, 14 participants (70%), rated their experience as 1, indicating a high level of satisfaction. Meanwhile, 4 participants (20%) rated their experience as 2, and 2 participants (10%) rated it as 3, suggesting there is room for improvement in this area.

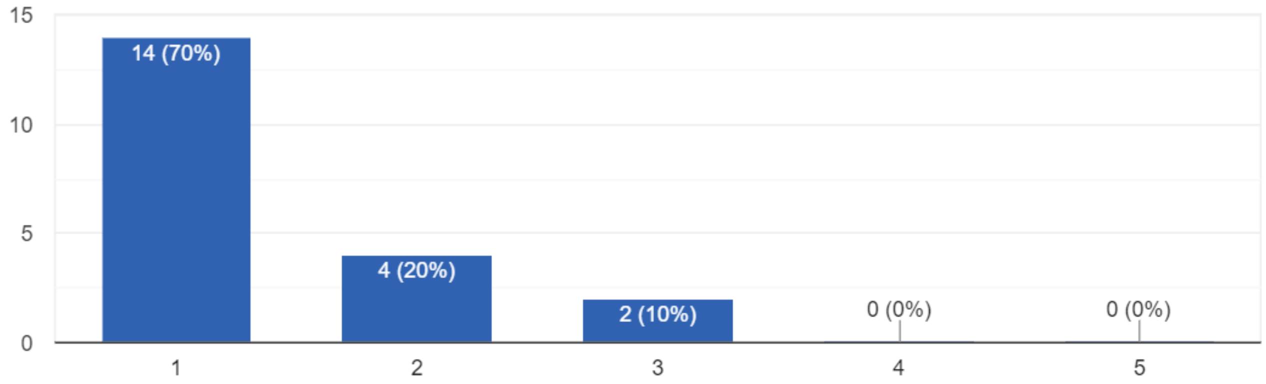


Figure 31 (Questionnaire Results)

Regarding overall satisfaction with the application, responses were predominantly positive. 13 participants (65%) rated their overall satisfaction as 1, indicating they were very satisfied. 6 participants (30%) rated it as 2, and one participant (5%) rated it as 3, highlighting a generally positive reception with some areas for potential enhancement.

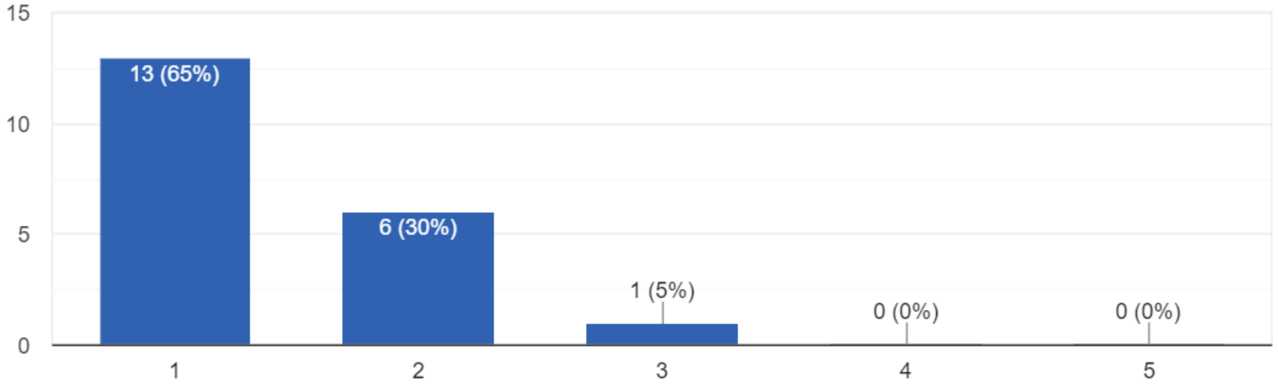


Figure 32 (Questionnaire Results)

Overall, the UAT results demonstrate positive user feedback, valuable insights for enhancement, and acknowledgment of the application's notable strengths. This information serves as a guide for ongoing improvements and aligning the application with user expectations.

5.2 Quality Attributes (NFR testing)

Table 6 (NFR testing)

User story	Quality Attribute	Measure	Results
As a user, I want the application to be well-designed for navigation so that I can complete my task in less than 1 minute [13].	Usability: how the application is easy to learn and user-friendly.	The measure used to assess the usability of the application's navigation design for file uploading and favoriting is the task completion time. [13].	We tested 4 individuals and the average task completion time for the file uploading and mark as favorite task was 14 seconds which means it is well below the desired threshold of 1 minute, indicating efficient navigation design in facilitating users to complete the task quickly.
As a user, I want my account information to be secured by encryption so that I can protect my information.	Security: how authorized access to protected data is granted and unauthorized access is restricted in the application.	Only authenticated users are allowed to access the software.	We implemented robust security measures which are Encryption sensitive data, text-based graphical pass adds extra layer of security, and Firebase Authentication ensures reliable authentication methods. User account information remains

			secure, fostering trust in application's security.
As a user, I want the app to be available 90% of the time I try to access it, so I don't become upset and go to another app.	Availability: how likely it is that a user will be able to access the system.	Compute the percentage of application availability.	Securebox relies on Firebase Realtime Database and Storage, which significantly impact its availability. Firebase guarantees a minimum availability of 99.95% [14] for its services. Therefore, based on Firebase's availability SLA, our application's availability is also expected to be at least 99.95%.
As a user, I want the homepage of the application to be opened within at most 5 seconds to use the application as quickly as possible.	Performance: how the speed, and responsiveness of an application holds up under a given workload	Compute the response time for opening the homepage. The application should be open the homepage in less than 5 seconds.	We tested 20 users individually, and timer was used to compute the response time. The average response time was 0.651 seconds which is less than 5 seconds.

5.3 Discussion

The User Acceptance Testing (UAT) results revealed positive feedback from participants on various aspects of our application. Overall, participants found the user interface design intuitive and easy to navigate. Specific functionalities such as deleting files/folders, renaming files, and accessing/viewing contents of folders were met with high satisfaction, with all participants reporting positive experiences. Additionally, participants were able to successfully perform tasks like marking files as favorites, downloading files, searching, filtering, and sharing files, indicating the effectiveness of these features.

While the majority of participants did not encounter technical issues, some reported challenges related to speed or responsiveness. Feedback on contacting the support team varied, suggesting opportunities for improvement in communication channels. However, overall satisfaction with the application was positive, with most participants expressing high levels of contentment.

Participants highlighted the need for smoother functionality when dragging files into folders, suggesting that this process could be made more intuitive and user-friendly. Additionally, they indicated a desire for improvements to the text-based Graphical Password system, particularly in terms of accessibility, such as adjusting font sizes. Moreover, participants expressed interest in additional security measures during password changes or resets. They suggested incorporating personal questions or additional verification steps to ensure that only authorized users can access and modify account information. Strengthening security protocols in this manner would enhance user confidence in the application's security features.

In addition to the UAT feedback, the results from assessing various user requirements and corresponding quality attributes of the application through Non-Functional Requirements (NFR) testing were equally positive. The evaluations indicate strong outcomes across usability, security, availability, and performance. The application demonstrated efficient navigation design, secure handling of user account information, high availability based on Firebase's SLA, and satisfactory performance in terms of response time. These findings highlight the application's ability to meet user needs effectively, ensuring a positive user experience.

In conclusion, the insights from both UAT and NFR testing provide a comprehensive understanding of the application's strengths and areas for enhancement. User feedback remains a crucial guide in this ongoing process. Leveraging these insights and implementing targeted improvements will contribute to an even more positive user experience.

Conclusions and Future Work

6 Conclusions and Future Work

This document represents our journey with SecureBox. It starts with an introduction that explains the problem we aimed to solve and introduces SecureBox. Then, we have a background chapter that helps readers understand the details of the SecureBox application. It gives a brief explanation of important concepts like information security and text-based graphical passwords.

In the literature review chapter, we looked at existing research on preventing shoulder surfing attacks. This helped us find gaps in the market and decide what features SecureBox should have. Once we knew what we wanted SecureBox to do, we moved on to the system analysis and design chapter. We figured out how to implement SecureBox and how its different parts work together.

After that, we tested our system by developing it using Android Studio. We made sure there were no bugs, and everything worked smoothly.

6.1 Global and local impact

Our application, designed for secure file storage and sharing and featuring two-factor authentication, Text-based Graphical Passwords, and fingerprint biometric authentication, creates a substantial impact both locally and globally. Locally, the integration of advanced authentication methods enhances individual user security, addressing concerns related to unauthorized access during secure file storage and sharing. The dual-layered security system ensures a robust defense against potential threats, contributing to heightened data protection and privacy in local digital interactions.

On a global scale, our application's commitment to two-factor authentication promotes a culture of secure digital practices. The utilization of Text-based Graphical Passwords provides an innovative solution to combating shoulder surfing attacks. This not only aligns with global efforts to strengthen cybersecurity measures but also sets a standard for responsible authentication practices in the broader digital landscape.

Additionally, our files are encrypted before being sent to the Firebase database, ensuring a secure storage mechanism that allows users exclusive control over their data. This reinforces our commitment to maintaining the privacy and integrity of user files globally, adding an extra layer of security to our comprehensive approach. The encryption process guarantees that only users have the ability to access and securely store their files. Furthermore, the encryption of SMS messages when sharing files with other users contributes to a heightened level of data protection within our application's user-centric framework on a global scale.

6.2 Problems and challenges encountered during software development.

The most challenging aspect of our project centered around the deployment of text-based graphical passwords and the integration of Firebase, a platform previously unfamiliar to our team. This integration demanded a deep dive into new technology and adapting to Firebase, which brought about a steep learning curve. Specifically, mastering the functionalities of Firebase, including file encryption and decryption, and implementing the biometric fingerprint and SMS APIs, posed significant challenges. Additionally, the tight project timelines required us to quickly grasp and implement these technologies, necessitating on-the-go learning. This learning curve extended to acquiring skills necessary for encrypting files before their transmission to Firebase and decrypting them when users download the files for viewing. Despite these hurdles, our team's commitment to rapid learning and adaptive problem-solving was pivotal in not only adopting text-based graphical passwords but also in achieving seamless integration of Firebase into our application. This integration supported the implementation of advanced features like biometric fingerprint authentication and the robust encryption and decryption of files and SMS messages.

6.3 Limitations of the system

Our SecureBox application, a file storage and sharing service, has several limitations. Firstly, it is only supported on Android devices and is available exclusively in English. Additionally, SecureBox is restricted to users in Saudi Arabia, thereby preventing access from other countries. Currently, the app supports only PDF and DOC files and does not support a wide range of other file types, including images and video files. Furthermore, when users reset their password via Firebase, they are required to enter their new password only once, without needing to confirm it. Notably, there are no restrictions on the complexity or format of the new password, which could potentially increase security risks.

6.4 The Main Contribution of The Project

SecureBox is designed to transform the way individuals securely store their files. Our user-friendly application empowers users to have complete control over their files while prioritizing their privacy and strengthening security. By encrypting data, we store files securely in the Firebase database, ensuring the utmost confidentiality.

Throughout our journey, we have gained valuable experience and knowledge. We applied the skills we acquired during our university studies to become experts in information security. We implemented features like Text-based graphical password and measures to prevent shoulder surfing attacks. As a result, SecureBox provides an innovative and secure solution for file storage that significantly improves user security. We empower users with complete control of their data and encourage users to responsibly protect the privacy of their information. By prioritizing data privacy and advocating for secure practices, SecureBox has a positive impact on information security.

6.5 Future work

Like many applications on the market today, no software is perfect since new features and updates are continually rolled out to users. This is also true for Secure Box. To enhance the app, it would be beneficial to address the previously mentioned limitations. These improvements could include adding support for additional languages like Arabic, expanding compatibility to other operating systems such as iOS, extending availability to more countries, and supporting a wider variety of file types.

Acknowledgements

7 Acknowledgements

We would like to express our deepest appreciation to everyone who played a role in the successful completion of this project. Special thanks go to our supervisor, Dr. Kholoud Saad Al-Saleh, whose invaluable advice and steadfast guidance were instrumental throughout the development of the project. We are also immensely grateful to our loving parents and friends for their unwavering support and encouragement during every stage of this journey.

we extend our gratitude to the dedicated testers who participated in the project’s user acceptance testing. Their insightful feedback was crucial and greatly contributed to the enhancements of the project.

References

8 References

[1] K. T. Hanna, "What is shoulder surfing? how do you protect yourself from it?," techtarget, 29 10 2021. [Online]. Available: https://www.techtarget.com/searchsecurity/definition/shoulder-surfing. [Accessed 30 8 2023].

[2] D. Cvetnarevic, "11 eye-opening shoulder surfing statistics to know," Security Escape, 27 6 2023. [Online]. Available: https://securityescape.com/shoulder-surfing-statistics/. [Accessed 29 8 2023].

[3] M. Ciampa, CompTIA® Security+ Guide to Network Security Fundamentals, Cengage Learning, 2015.

[4] NCSC, "A guide to ransomware," NCSC, [Online]. Available: https://www.ncsc.gov.uk/ransomware/home. [Accessed 2 august 2023].

[5] G. Manjunath, "An Improved Text-Based Shoulder Surfing Resistant Graphical Password Scheme by Using Colors," *International Journal of Computer Science and Information Technologies,* Vols. ,Vol. 5 , no. 2277-2280 , 2014.

[6] W. Stallings, Introduction to Cryptography and Network Security, 2017: Pearson Education.

[7] S. Ranjitha Kumari and B. Padmavathi, "Survey on Performance Analysis of DES, AES," *International Journal of Science and Research (IJSR),* no. 2319-7064, pp. 170-174, 2013.

[8] Amazon, "What is an API," Amazon, [Online]. Available: https://aws.amazon.com/what-is/api/#:~:text=API%20stands%20for%20Application%20Programming,other%20using%20requests%20and%20responses.. [Accessed 8 august 2023].

[9] L. Sobrado and J. C. Birget, "Graphical passwords," The Rutgers Scholar, An Electronic Bulletin for Undergraduate Research, 2002.

[10] S. Wiedenbeck, J. Waters, L. Sobrado and C. Birget, "Design and evaluation of a shoulder-surfing resistant graphical password scheme," *Proc. of Working Conf. on Advanced Visual Interfaces,* pp. 177-184, 2006.

[11] H.Gao, X. Liu, S. Wang, H. Liu and R. Dai, Design and analysis of a graphical password scheme, Conf. on Innovative Computing, Information and Control, 2009.

[12] s. assirali, "Client-Server Model," geeksforgeeks, 22 12 2022. [Online]. Available: https://www.geeksforgeeks.org/client-server-model/. [Accessed 1 9 2023].

[13] J. Mifsud, "Usability Geek," 13 September 2019. [Online]. Available: https://usabilitygeek.com/usability-metrics-a-guide-to-quantify-system-usability/. [Accessed May 15 2024].

[14] Firebase, "Google," 9 April 2020. [Online]. Available: https://firebase.google.com/terms/service-level-agreement. [Accessed 18 May 2024].

[15] Glossary, "Man-in-the-middle," ENISA, 29 november 2022. [Online]. Available: https://www.enisa.europa.eu/topics/incident-response/glossary/man-in-the-middle. [Accessed 1 august 2023].

[16] F. Salahdine and N. Kaabouch, "Social Engineering Attacks: A Survey," *Future Internet*, vol. 11, p. 89, 2019.

Appendix

9 Appendix

9.1 Appendix A: Requirements Elicitation: interviews

Interview Questions:
1- What do you do if you want to store some files on your device, but you do not have enough space?

2- When you have an important file containing sensitive information that you need to store and share, describe what you do. Which type of application do you usually use for sending such files?

3- When using cloud or server-based file storage applications, have you encountered any specific difficulties or pain points? What do you suggest dealing with these difficulties?

4- In your experience, what difficulties have you faced when sharing sensitive files with others? Do you believe that sharing files via messages offers better security and usability compared to other methods?

5- Among the features offered by cloud or server-based file storage applications, which ones do you find most valuable or appealing? Are there any particular features that you wish these apps had?

6- How do you currently protect your passwords from potential shoulder-surfing attacks? Could you provide specific examples of how you secure your sensitive files against various types of attacks?

7- In your opinion, what constitutes a useful and robust login method for encryption systems and confidential file protection? Do you think implementing multiple layers of login enhances security like using graphical passwords or biometrics?

8- Do you have anything else to add?

Interviews transcriptions:

Table 7 (interviews 1)

Interview 1	
Date: 14/9/2023 Location: in the University Duration: 15 minutes	
Interviewee: Raya Alsuhaim	Interviewer: Nouf Aldakheel
Reminders: -The interviewee is a student in the cybersecurity track and is a frequent user of storage applications -He has a high knowledge of technology and encryption methods	
Questions	Answers
1- What do you do if you want to store some files on your device, but you do not have enough space?	Mostly, I use OneDrive for storage
2- When you have an important file containing sensitive information that you need to store and share, describe what you do. Which type of application do you usually use for sending such files?	I encrypt files with any encryption program available. As for sharing, I have never shared sensitive files before, but I certainly will not send and share files via social media applications because they are a completely unreliable place and the possibility of them being hacked is high.
3- When using cloud or server-based file storage applications, have you encountered any specific difficulties or pain points? What do you suggest dealing with these difficulties?	Yes, synchronization, that is, when I work on the file, I am not sure whether it will save my work continuously or not. Also, sometimes it takes a long time when saving.
4- In your experience, what difficulties have you faced when sharing sensitive files with others? Do you believe that sharing files via messages offers better security and usability compared to other methods?	I don't think sharing via messages is very safe because I haven't seen any applications use it before, so I have a little doubt.
5- Among the features offered by cloud or server-based file storage applications, which ones do you find most valuable or appealing? Are there any particular features that you wish these apps had?	I believe that the most valuable feature is that I can access files from anywhere, any device, and at any time. As for the features that I hope to have, they are expanding the storage space more and facilitating the process of sharing files between users.

6- How do you currently protect your passwords from potential shoulder-surfing attacks? Could you provide specific examples of how you secure your sensitive files against various types of attacks?	I paste a dark-colored screen on my phone's screen and reduce the brightness of the phone's light to make sure that no one can see my passwords while I write them. In some cases, I change the names of the files to fake ones so that no one knows what's inside them.
7- In your opinion, what constitutes a useful and robust login method for encryption systems and confidential file protection? Do you think implementing multiple layers of login enhances security like using graphical passwords or biometrics?	Yes, I think that adding multiple layers of login enhances security. It would also be better if the feature of monitoring unusual activity and showing an alert when there is any suspected movement or strange entry was added.
8- Do you have anything else to add?	Yes, and I hope that your application will solve the problem of limited storage space that most people suffer from

Table 8 (interviews 2)

Interview 2	
Date: 14/9/2023 Location: in the University Duration: 15 minutes	
Interviewee: Siham Al-Mashuh	Interviewer:Nouf Aldakheel
Reminder: -The interviewee is a student in the cybersecurity track and a frequent user of storage applications -Interested in cyber security and protection applications	
Questions	Answers
1- What do you do if you want to store some files on your device, but you do not have enough space?	I use google drive
2- When you have an important file containing sensitive information that you need to store and share, describe what	I can encrypt it, or if I am going to put a strong password for my account in Google Drive, I can suffice with it and make sure that I do not

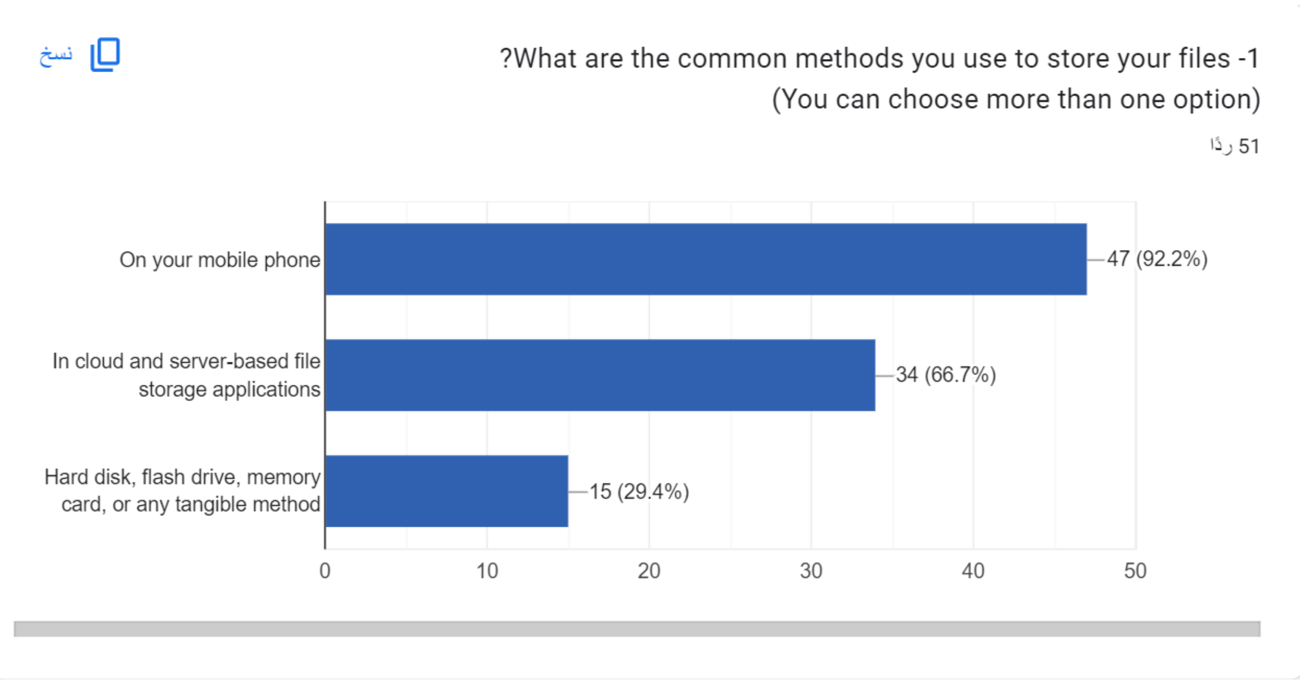
you do. Which type of application do you usually use for sending such files?	upload very sensitive files, or I can send the file via WhatsApp and delete the message immediately if the other party receives it.
3- When using cloud or server-based file storage applications, have you encountered any specific difficulties or pain points? What do you suggest dealing with these difficulties?	Yes, storage space is very limited. I wish I could download more files
4- In your experience, what difficulties have you faced when sharing sensitive files with others? Do you believe that sharing files via messages offers better security and usability compared to other methods?	Yes, I think that sharing files via messages is secured because it is linked to my phone number, unlike social media applications, which are an unsecured option for sharing
5- Among the features offered by cloud or server-based file storage applications, which ones do you find most valuable or appealing? Are there any particular features that you wish these apps had?	Yes, the file arrangement feature is very useful, as it helps me reach the file I need quickly
6- How do you currently protect your passwords from potential shoulder-surfing attacks? Could you provide specific examples of how you secure your sensitive files against various types of attacks?	I secure my sensitive files by setting passwords and reducing the brightness of the mobile phone so that no one can see my files from behind me.
7- In your opinion, what constitutes a useful and robust login method for encryption systems and confidential file protection? Do you think implementing multiple layers of login enhances security like using graphical passwords or biometrics?	I think using Face ID will make the login process more secure
8- Do you have anything else to add?	I think that having a monthly subscription for a small amount of money that opens up additional features is a great idea

Table 9 (interviews 3)

Interview 3	
Date: 14/9/2023 Location: interviewee's office Duration: 15 minutes	
Interviewee: Mohammed Ahmed	Interviewer: Suhad Alhomaidhi
Reminder: -The interviewee is a student in the scientific track at the university -Interested in technical fields and cyber security and protection applications	
Questions	Answers
1- What do you do if you want to store some files on your device, but you do not have enough space?	I delete some files that I no longer need, or I send them to myself by email, or upload them to Google Drive.
2- When you have an important file containing sensitive information that you need to store and share, describe what you do. Which type of application do you usually use for sending such files?	I use applications that put a password on files to protect them
3- When using cloud or server-based file storage applications, have you encountered any specific difficulties or pain points? What do you suggest dealing with these difficulties?	Yes, when it tells me that the storage space is full, but I did not upload many files and I do not know what files filled this space
4- In your experience, what difficulties have you faced when sharing sensitive files with others? Do you believe that sharing files via messages offers better security and usability compared to other methods?	Yes, I think that sharing files via messages is more secure because it is not linked to the Internet but rather to the mobile number
5- Among the features offered by cloud or server-based file storage applications, which ones do you find most valuable or appealing? Are there any particular features that you wish these apps had?	I think the feature of seeing the last time you modified the file is very good. I also wished there was a double protection feature for some files.
6- How do you currently protect your passwords from potential shoulder-surfing attacks? Could you provide specific	When I have to open my personal accounts and set my passwords, I make sure to be in a safe place and lower the brightness of my

examples of how you secure your sensitive files against various types of attacks?	mobile phone so that no one sees my password. I am also careful not to use the Wi-Fi of cafes and public places.
7- In your opinion, what constitutes a useful and robust login method for encryption systems and confidential file protection? Do you think implementing multiple layers of login enhances security like using graphical passwords or biometrics?	Yes, of course. I support the use of Face ID, fingerprint, etc., and I think they are very effective in terms of security.
8- Do you have anything else to add?	No, thank you

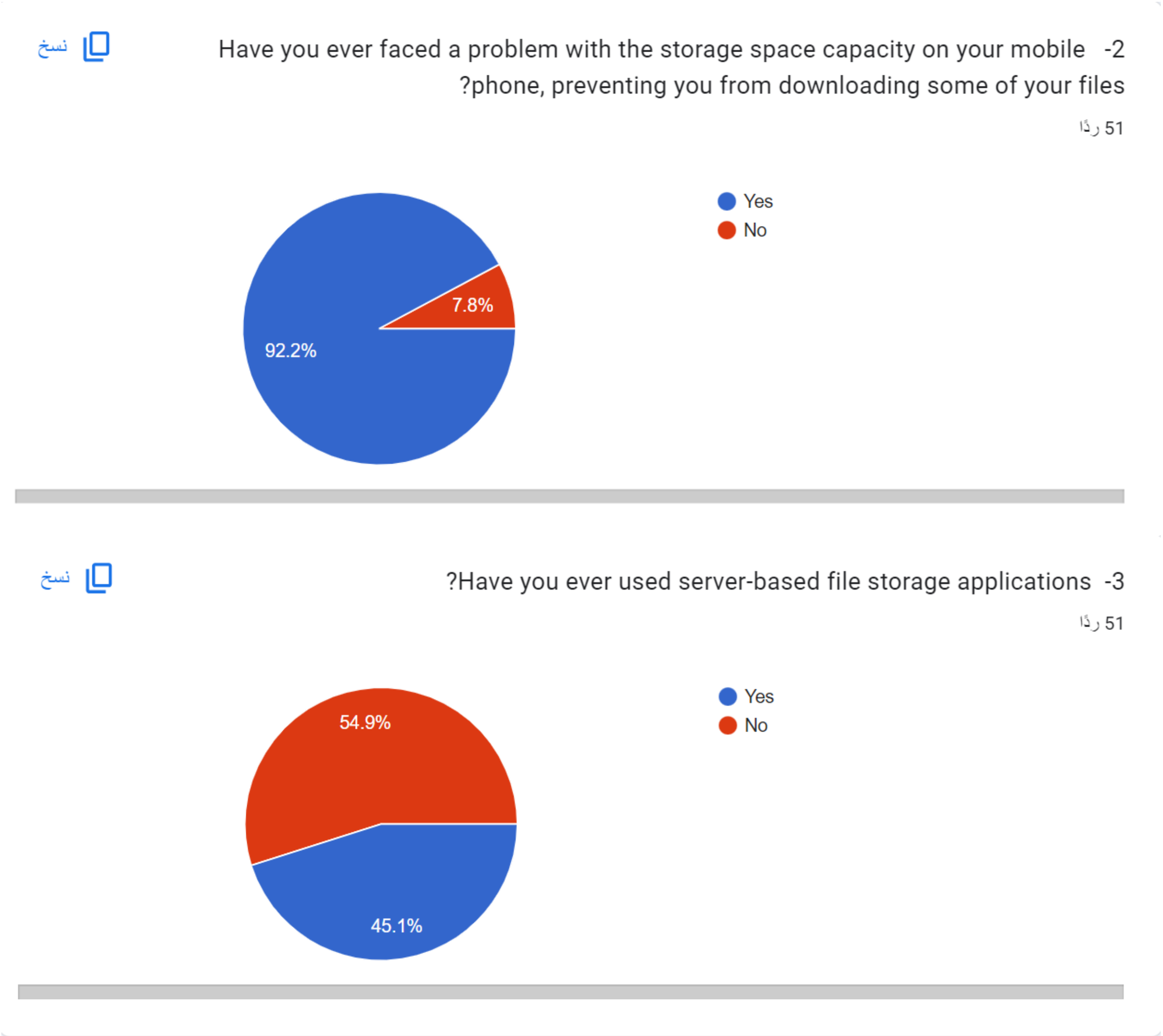
9.2 Appendix B: Requirements Elicitation: questionnaires



?What are the common methods you use to store your files -1
(You can choose more than one option)

51 ردًا

نسخ
On your mobile phone
In cloud and server-based file storage applications
Hard disk, flash drive, memory card, or any tangible method
47 (92.2%)
34 (66.7%)
15 (29.4%)
0
10
20
30
40
50

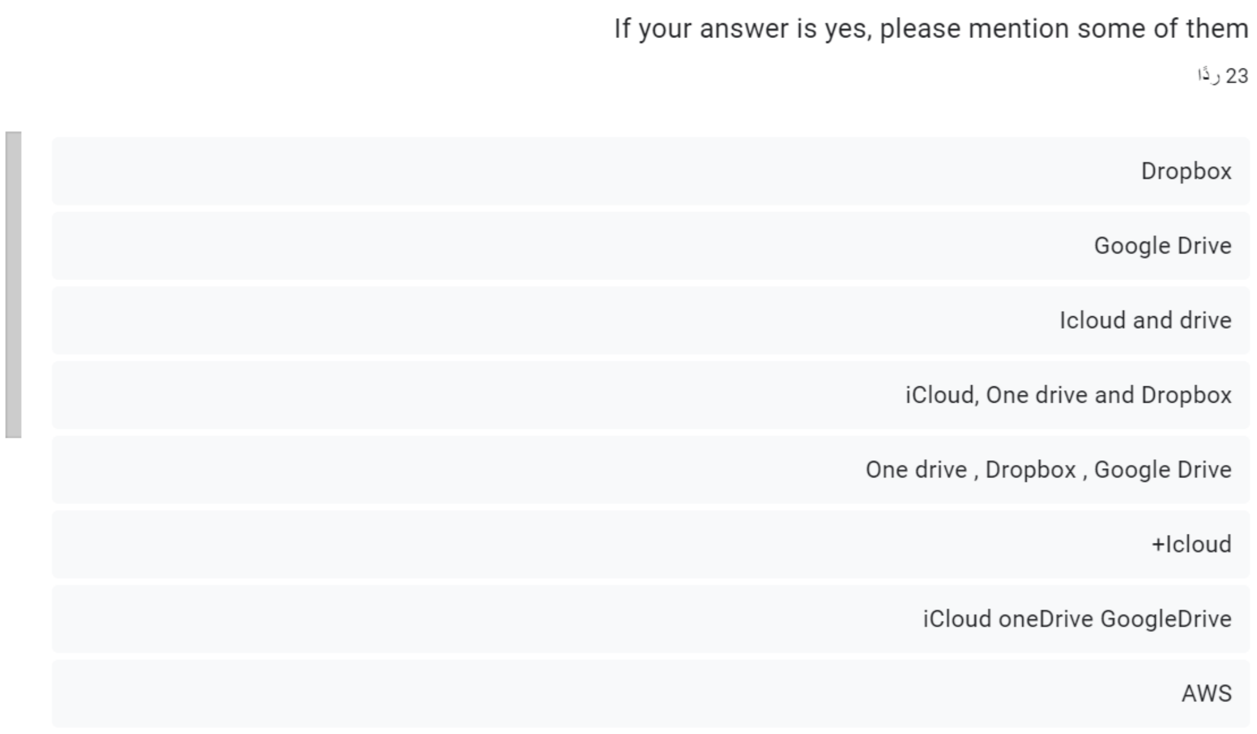


Have you ever faced a problem with the storage space capacity on your mobile -2
?phone, preventing you from downloading some of your files

51 ردًا

?Have you ever used server-based file storage applications -3

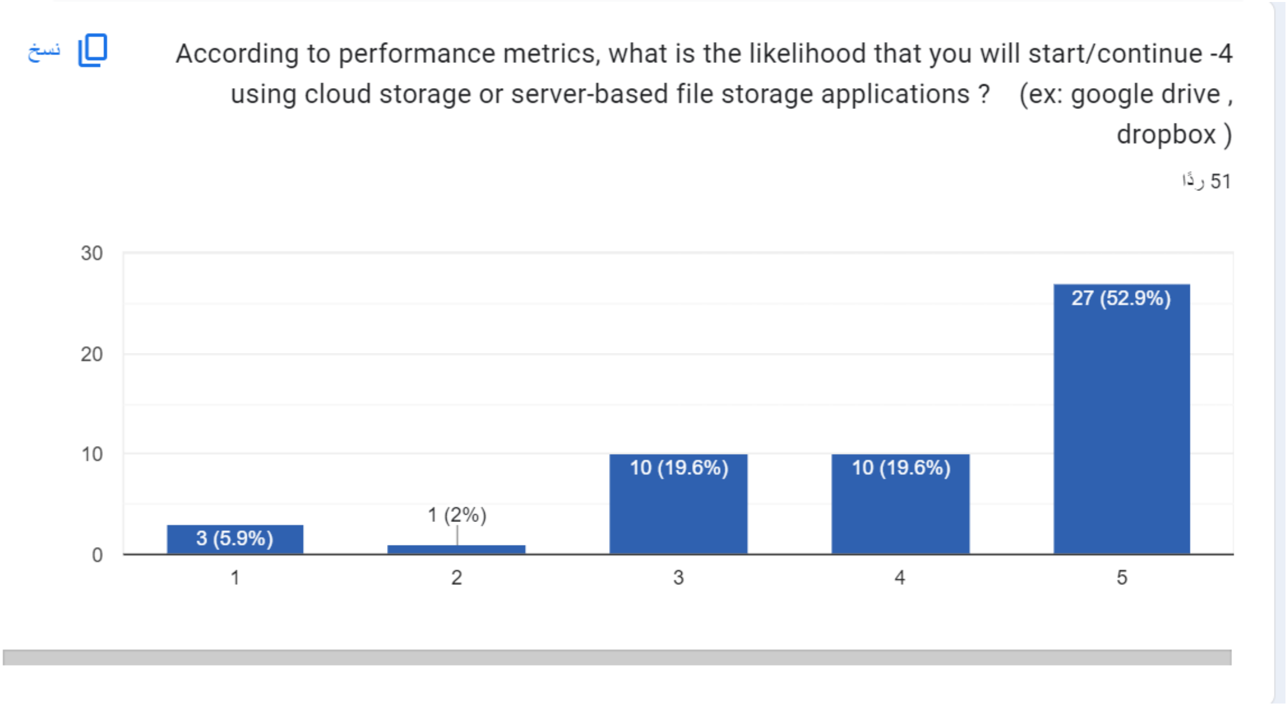
51 ردًا



If your answer is yes, please mention some of them

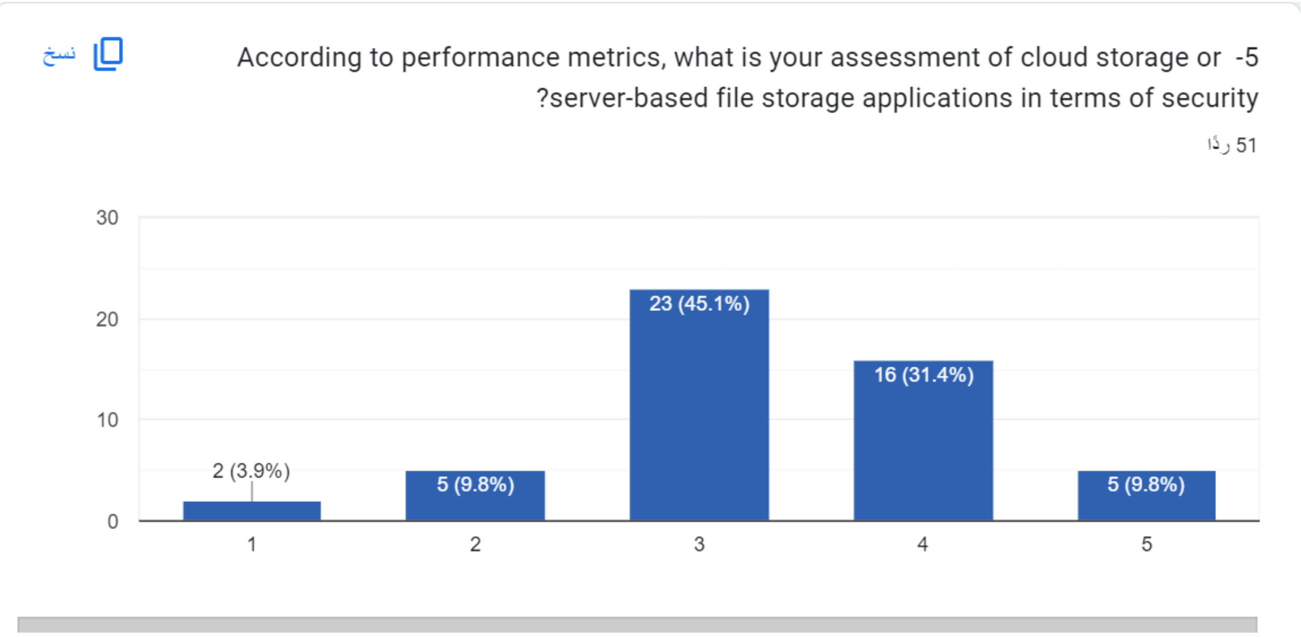
23 ردًا

- Dropbox
- Google Drive
- Icloud and drive
- iCloud, One drive and Dropbox
- One drive , Dropbox , Google Drive
- +Icloud
- iCloud oneDrive GoogleDrive
- AWS



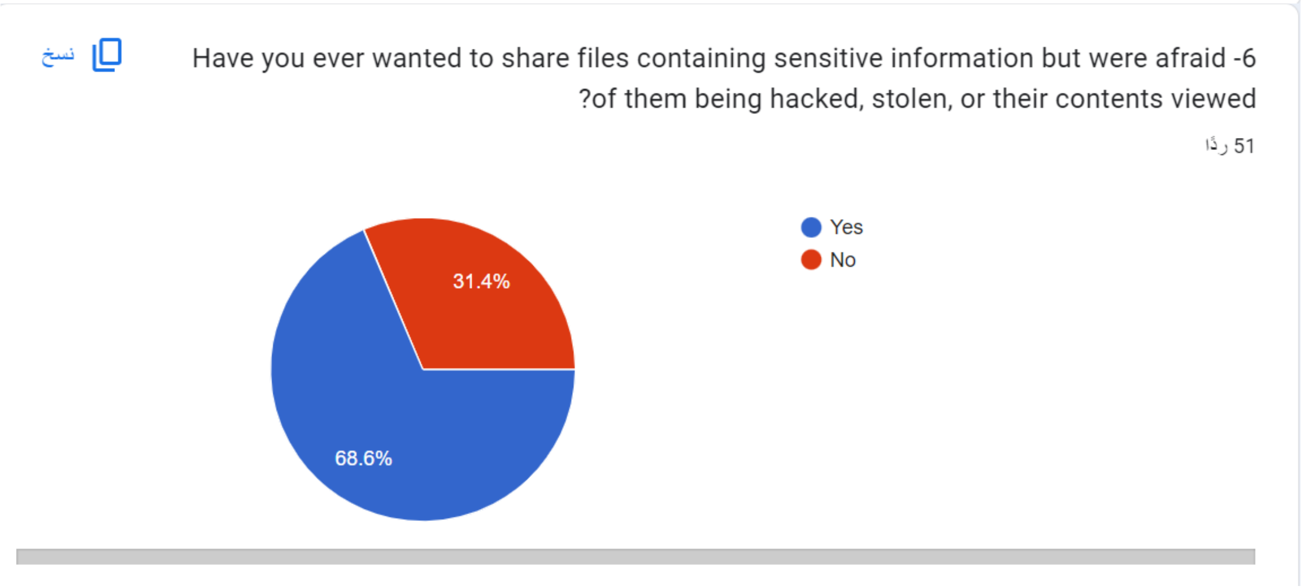
According to performance metrics, what is the likelihood that you will start/continue -4 using cloud storage or server-based file storage applications ? (ex: google drive , dropbox)

51 ردًا



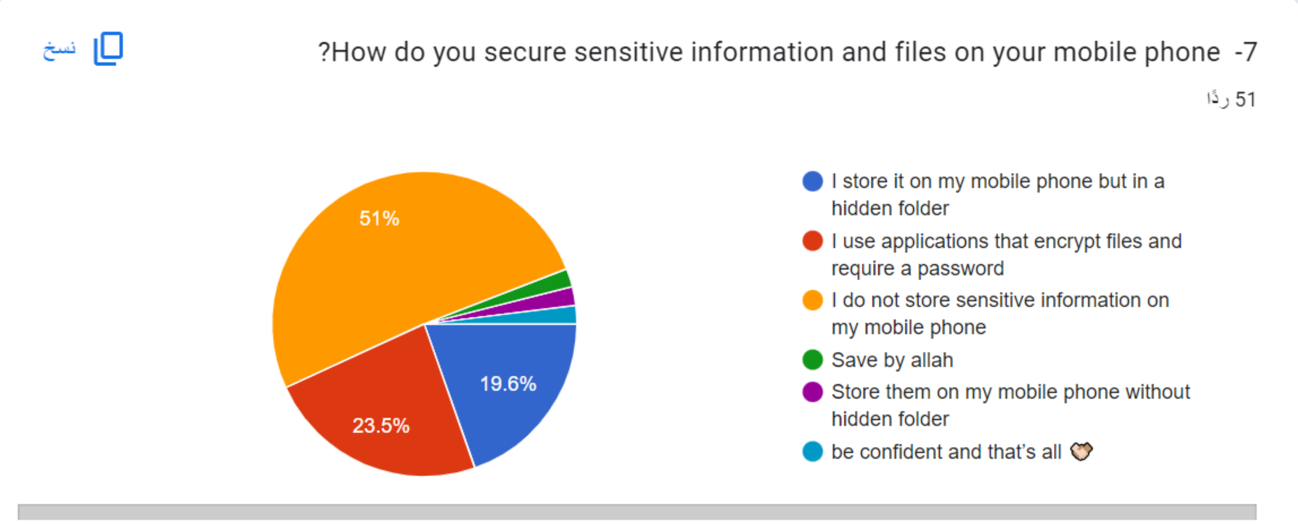
According to performance metrics, what is your assessment of cloud storage or -5
?server-based file storage applications in terms of security

51 ردًا



Have you ever wanted to share files containing sensitive information but were afraid -6
?of them being hacked, stolen, or their contents viewed

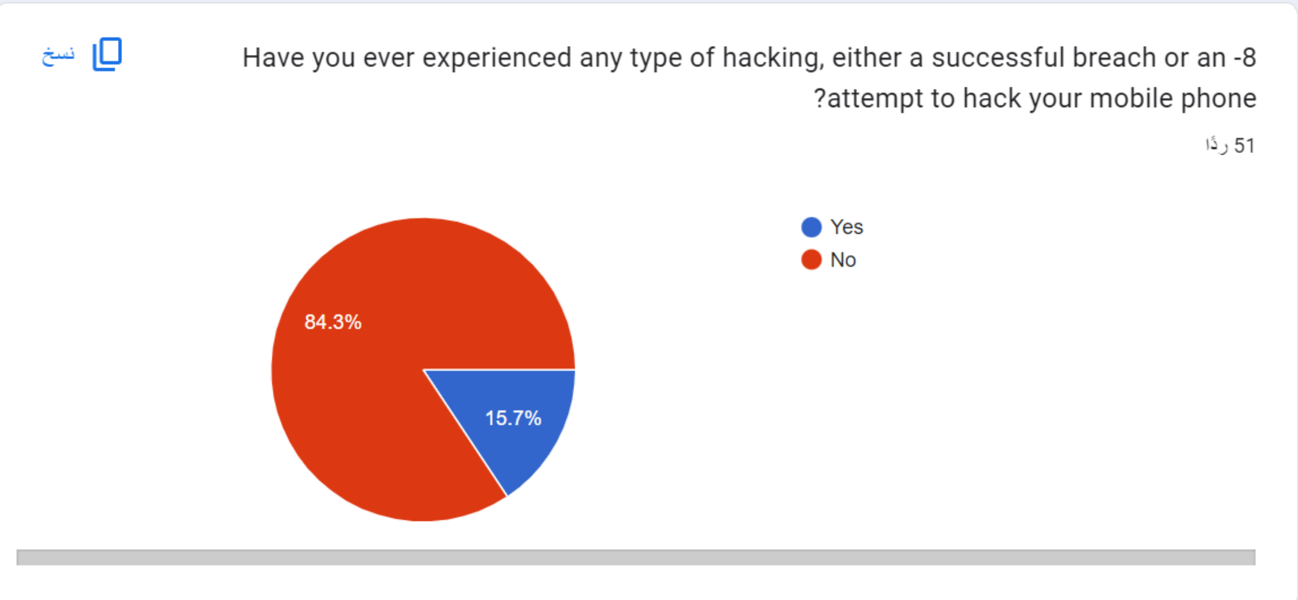
51 ردًا



?How do you secure sensitive information and files on your mobile phone -7

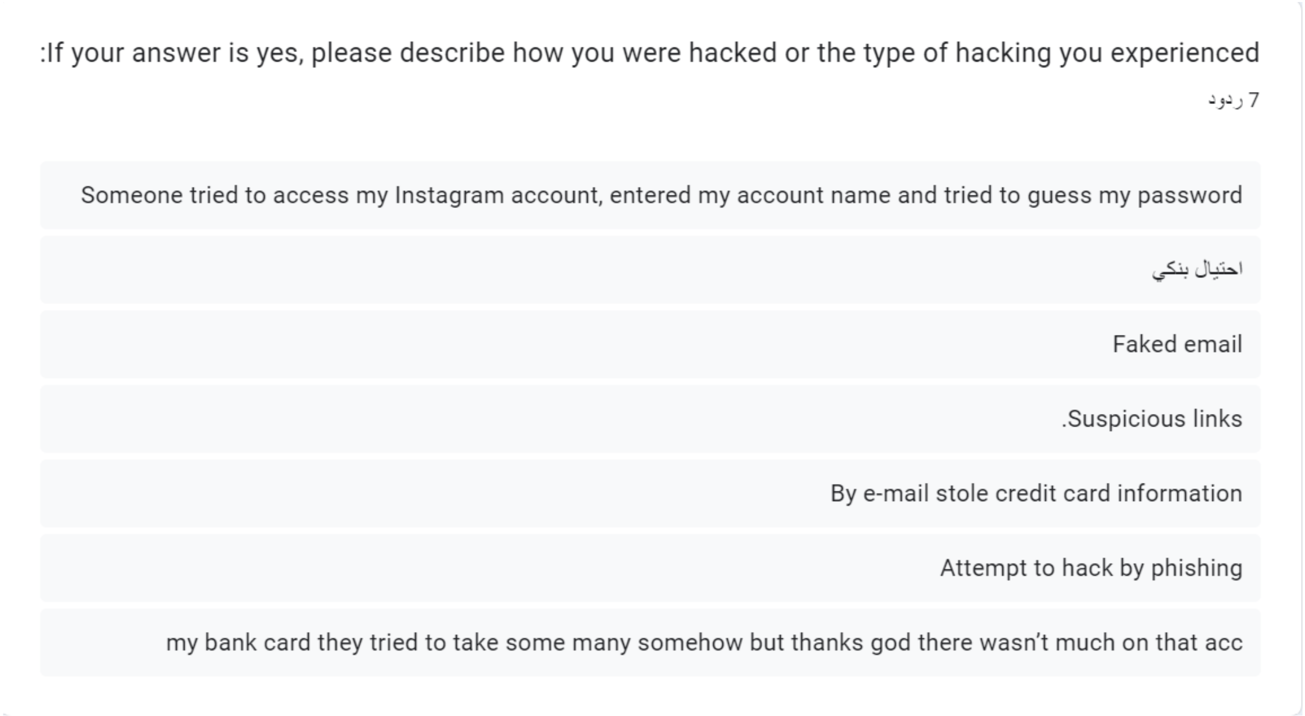
51 ردًا

نسخ
51%
19.6%
23.5%
● I store it on my mobile phone but in a hidden folder
● I use applications that encrypt files and require a password
● I do not store sensitive information on my mobile phone
● Save by allah
● Store them on my mobile phone without hidden folder
● be confident and that's all 🤝



Have you ever experienced any type of hacking, either a successful breach or an -8 ?attempt to hack your mobile phone

51 ردًا



:If your answer is yes, please describe how you were hacked or the type of hacking you experienced

7 ردود

Someone tried to access my Instagram account, entered my account name and tried to guess my password

احتيال بنكي

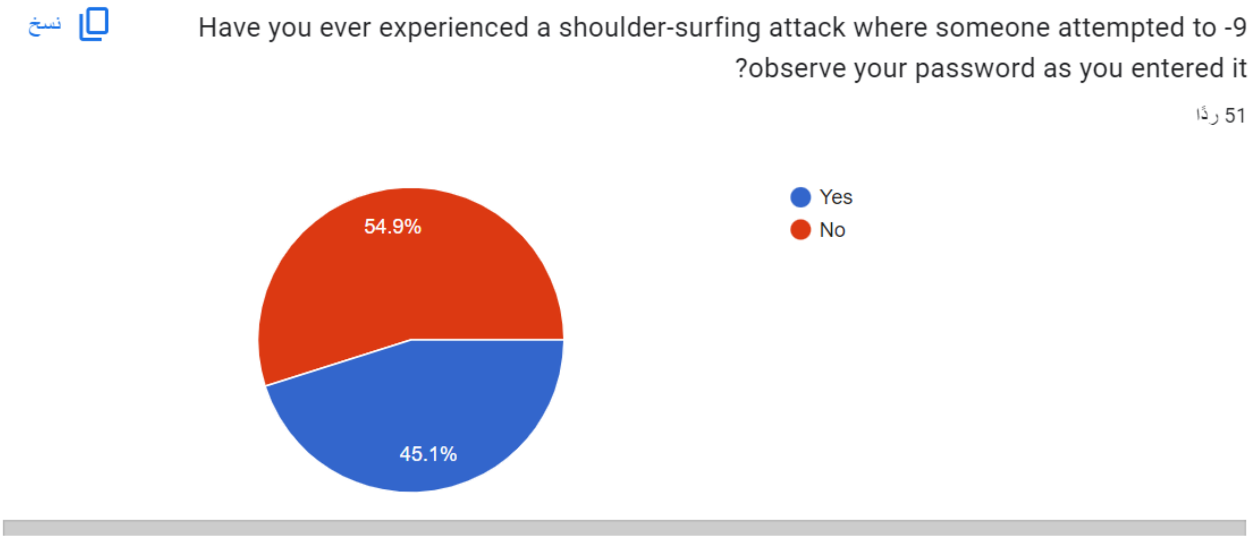
Faked email

.Suspicious links

By e-mail stole credit card information

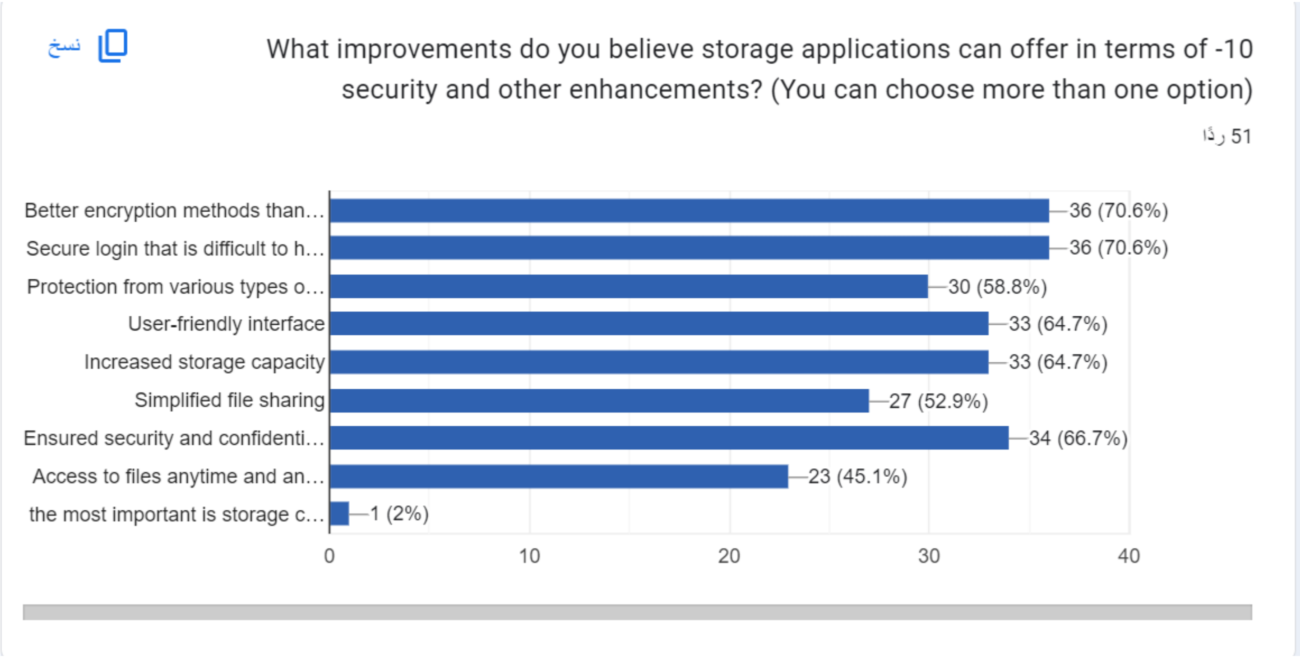
Attempt to hack by phishing

my bank card they tried to take some many somehow but thanks god there wasn't much on that acc



Have you ever experienced a shoulder-surfing attack where someone attempted to -9 ?observe your password as you entered it

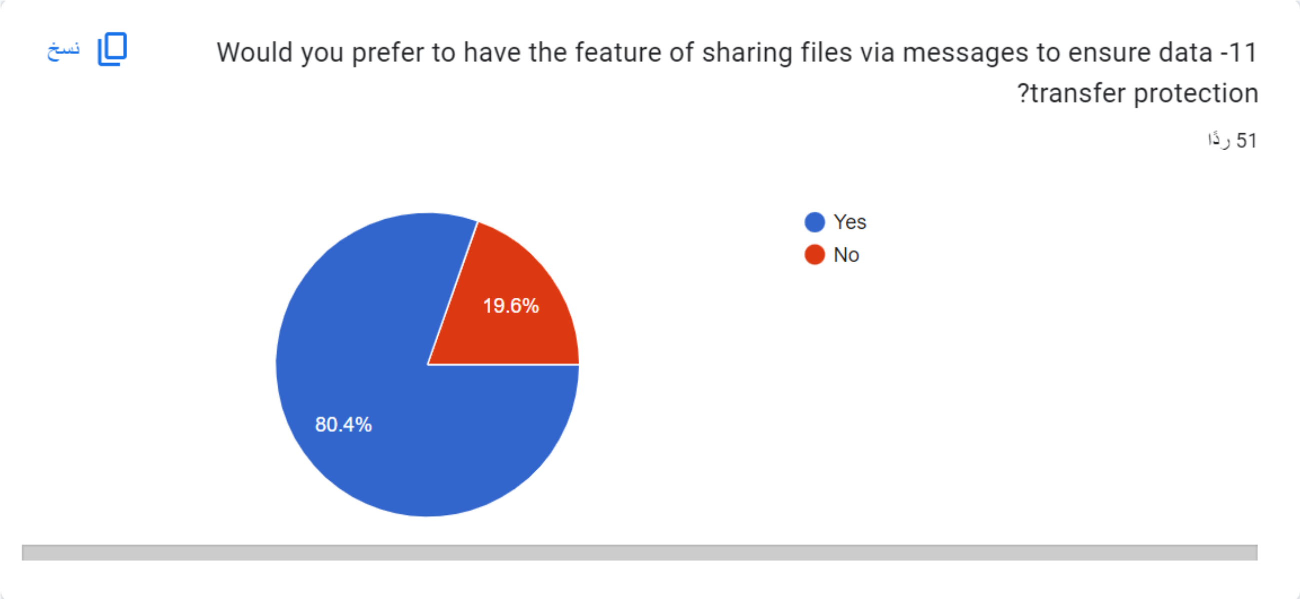
51 ردًا



نسخ

What improvements do you believe storage applications can offer in terms of -10 security and other enhancements? (You can choose more than one option)

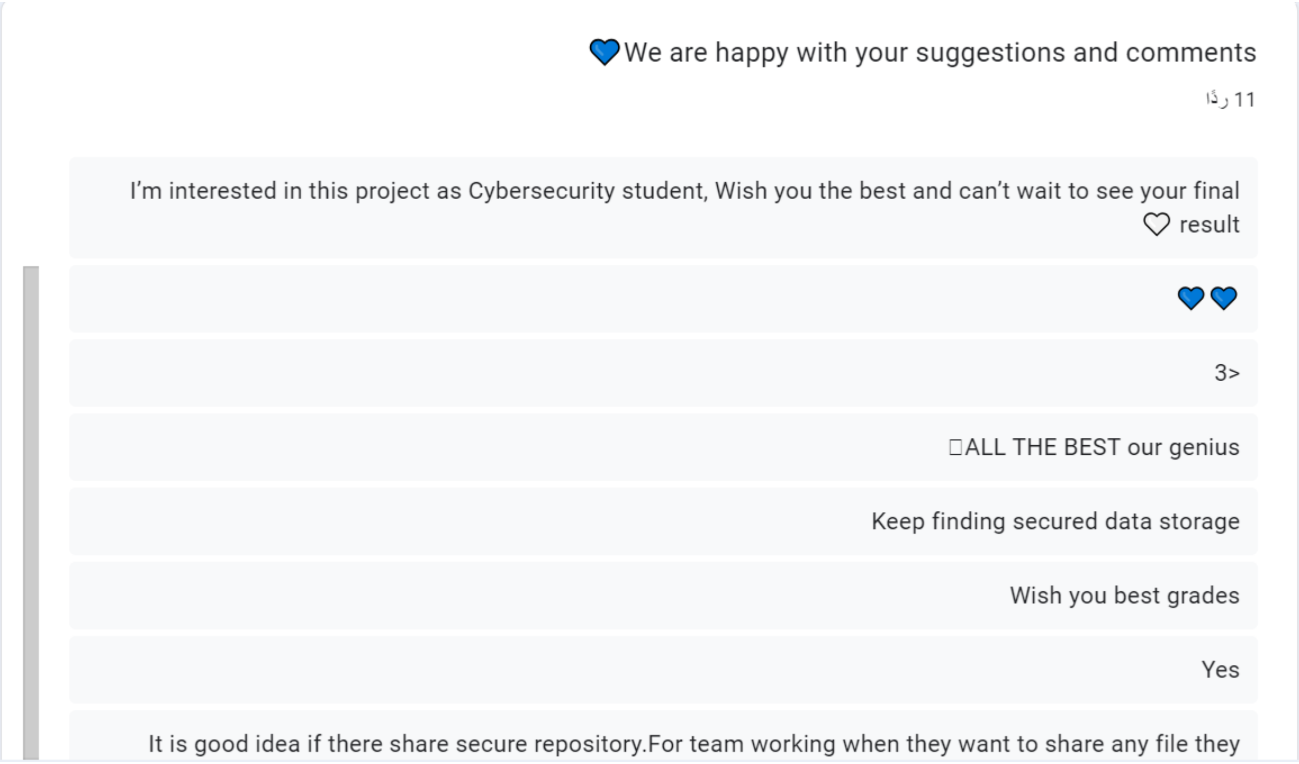
51 ردًا



نسخ

Would you prefer to have the feature of sharing files via messages to ensure data -11 ?transfer protection

51 ردًا



💙We are happy with your suggestions and comments

11 ردًا

I’m interested in this project as Cybersecurity student, Wish you the best and can’t wait to see your final ♡ result

💙💙

3>

□ALL THE BEST our genius

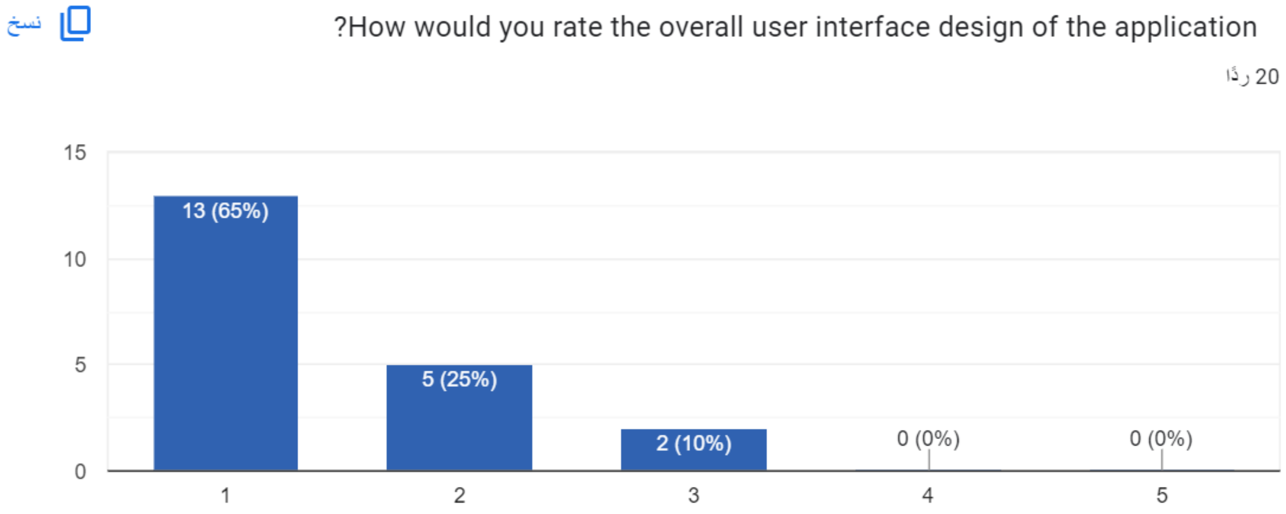
Keep finding secured data storage

Wish you best grades

Yes

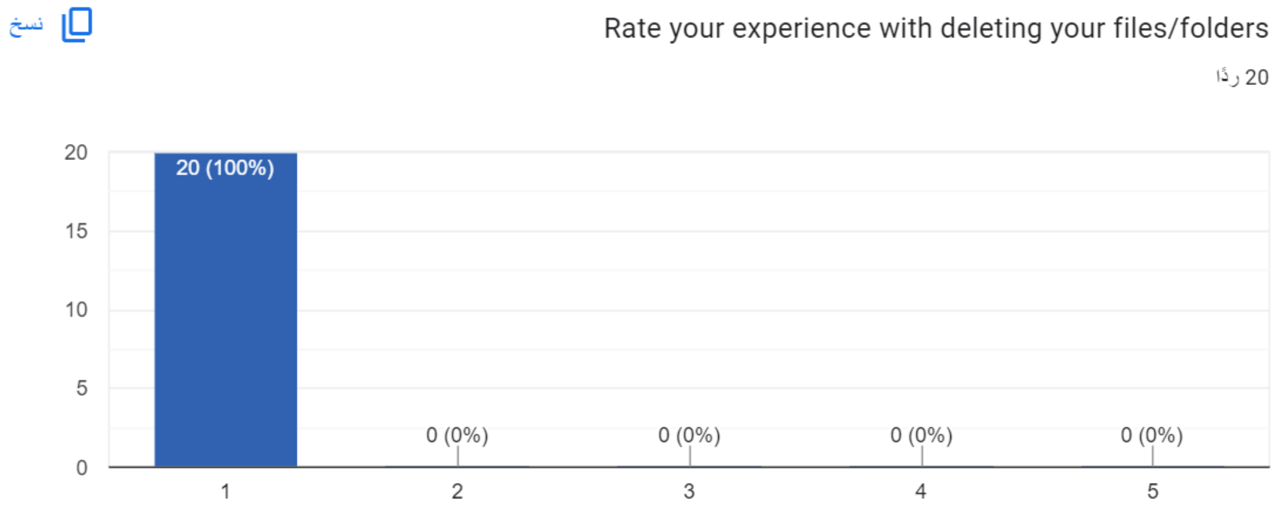
It is good idea if there share secure repository.For team working when they want to share any file they

9.3 Appendix C: User Acceptance Testing: Questionnaire



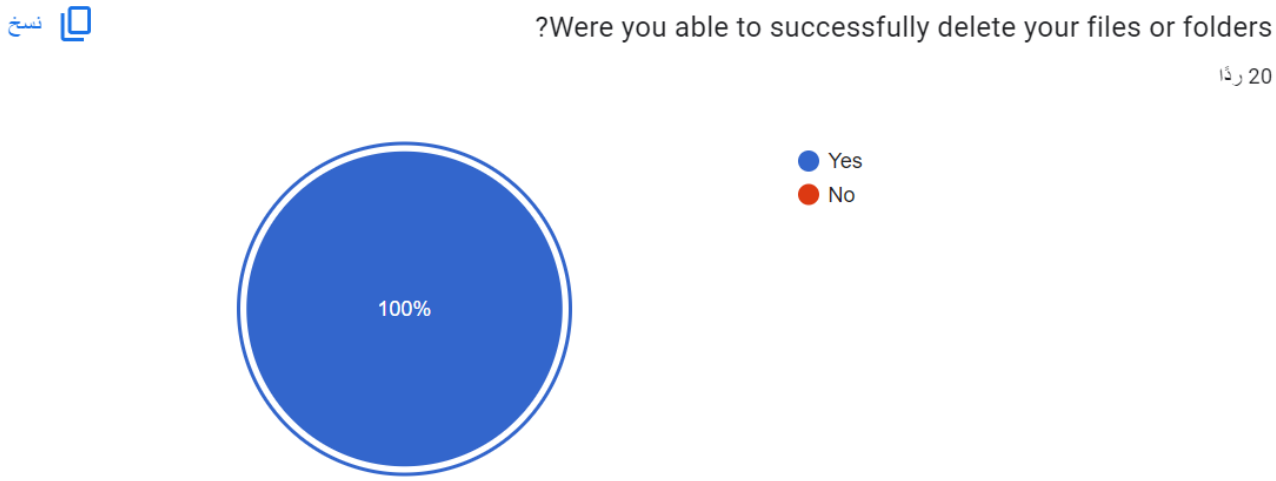
?How would you rate the overall user interface design of the application

20 ردًا



Rate your experience with deleting your files/folders

20 ردًا



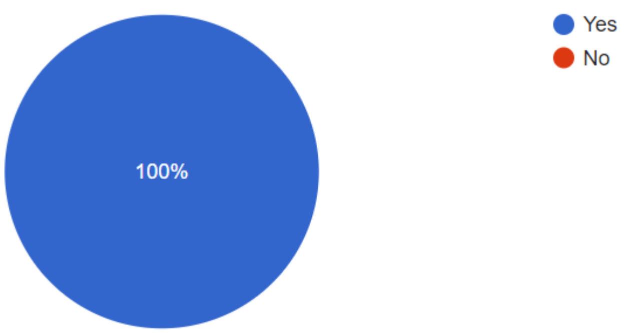
?Were you able to successfully delete your files or folders

20 ردًا

نسخ

?Were you able to successfully rename your files or folders

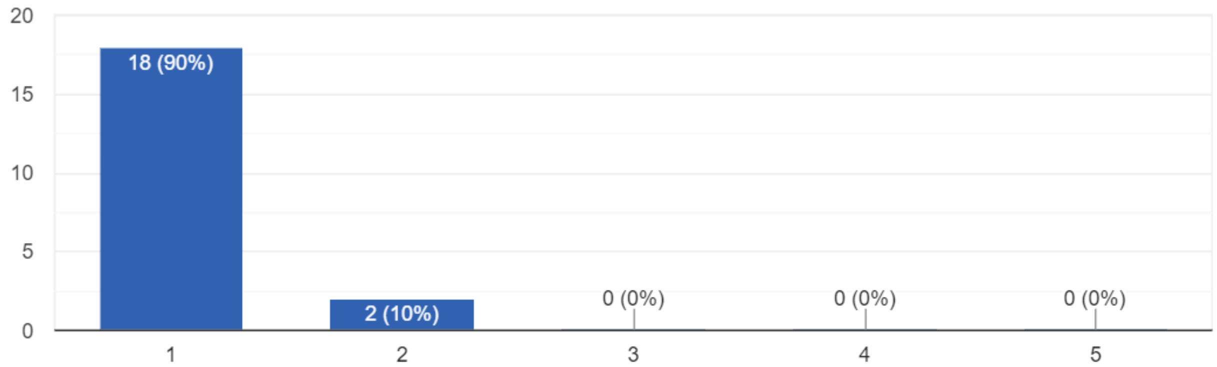
20 ردًا



نسخ

Rate your experience with accessing and viewing the contents of folders

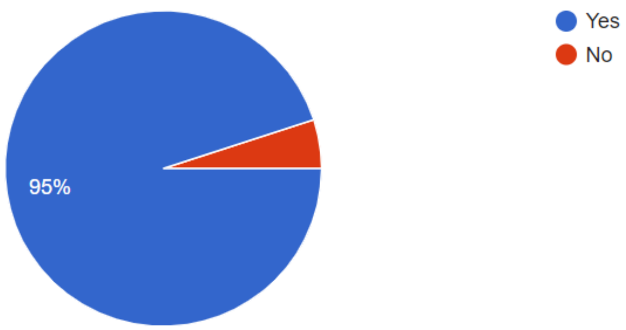
20 ردًا



نسخ

?Were you able to successfully mark files as favorites

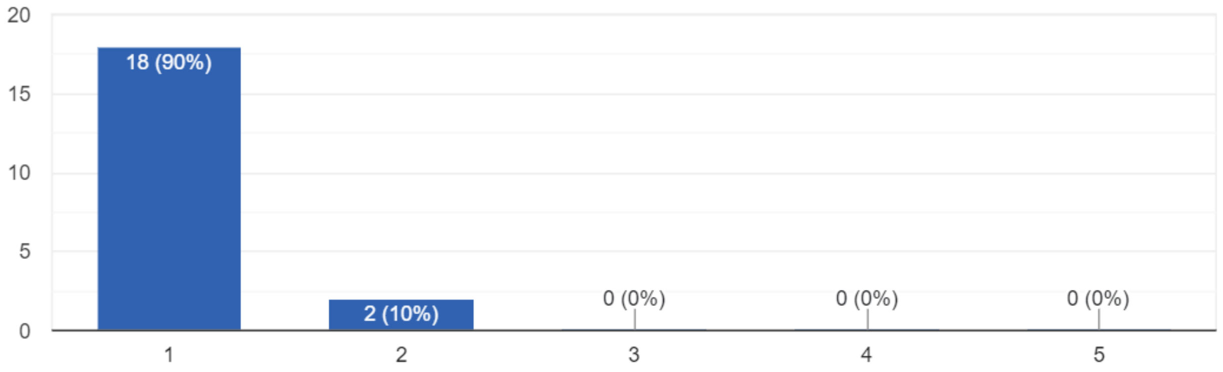
20 ردًا



نسخ

Rate your experience with downloading your files

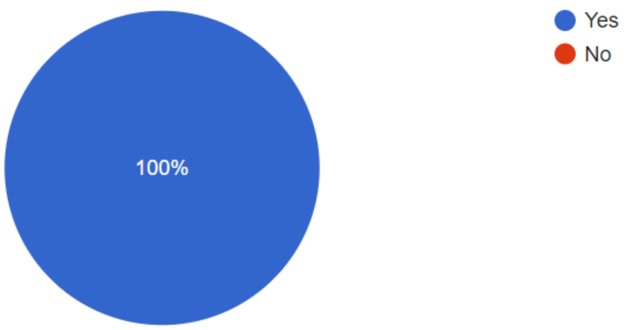
20 ردًا



نسخ

?Were you able to successfully search for your files

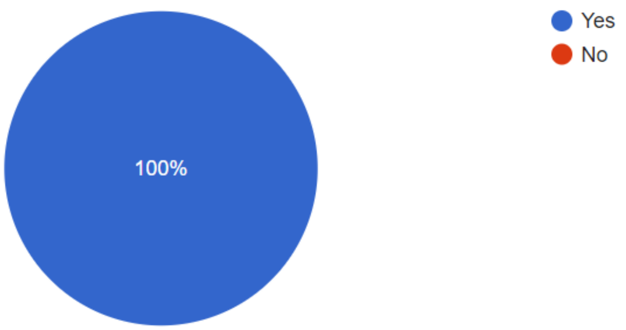
20 ردًا

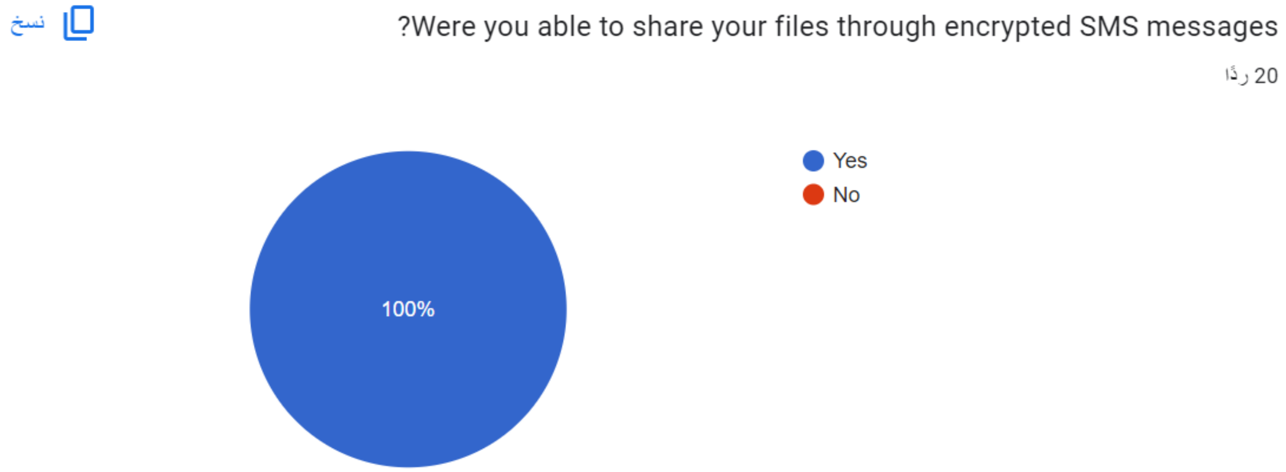


نسخ

?Were you able to successfully filter your files

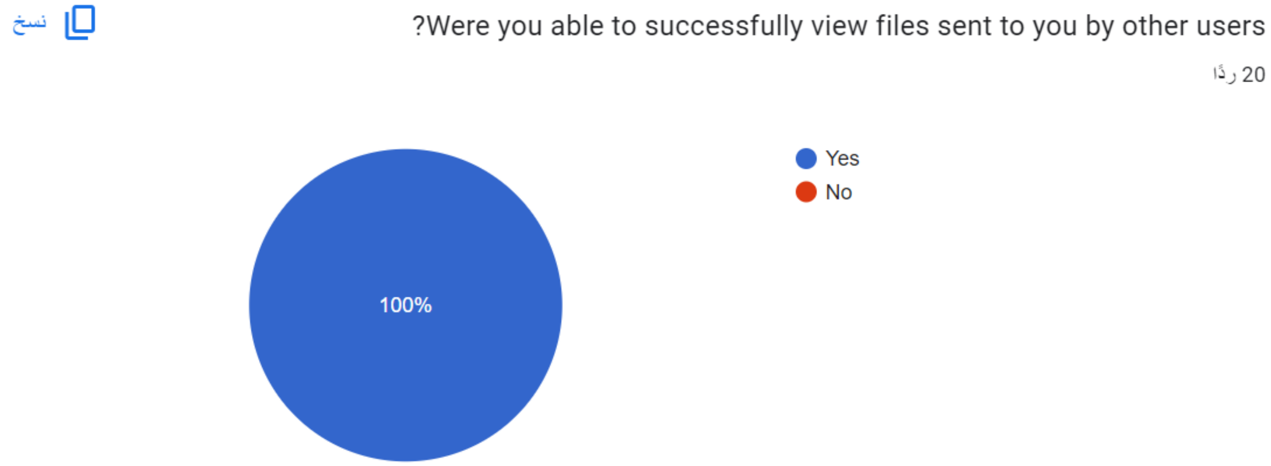
20 ردًا





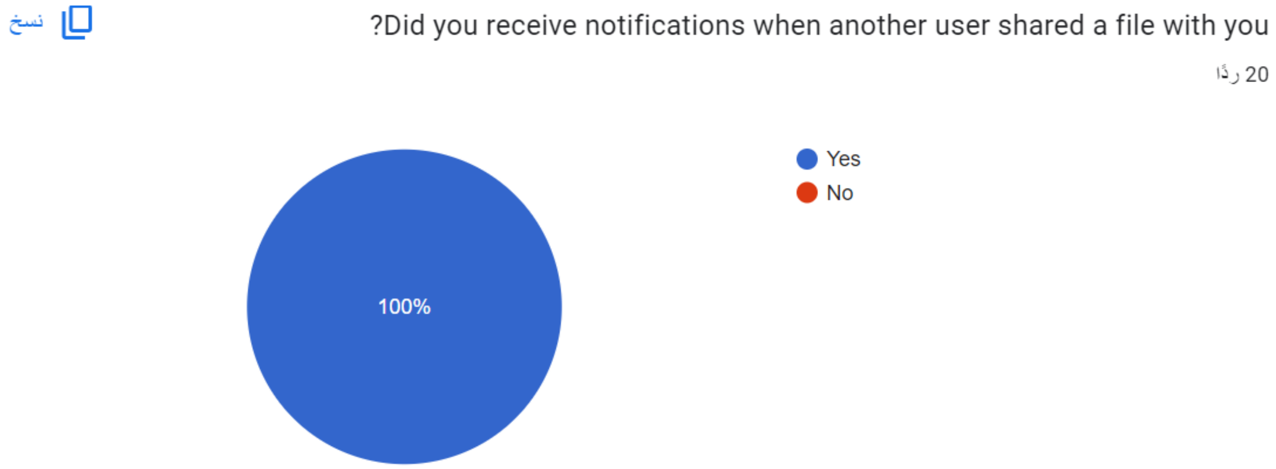
?Were you able to share your files through encrypted SMS messages

20 ردًا



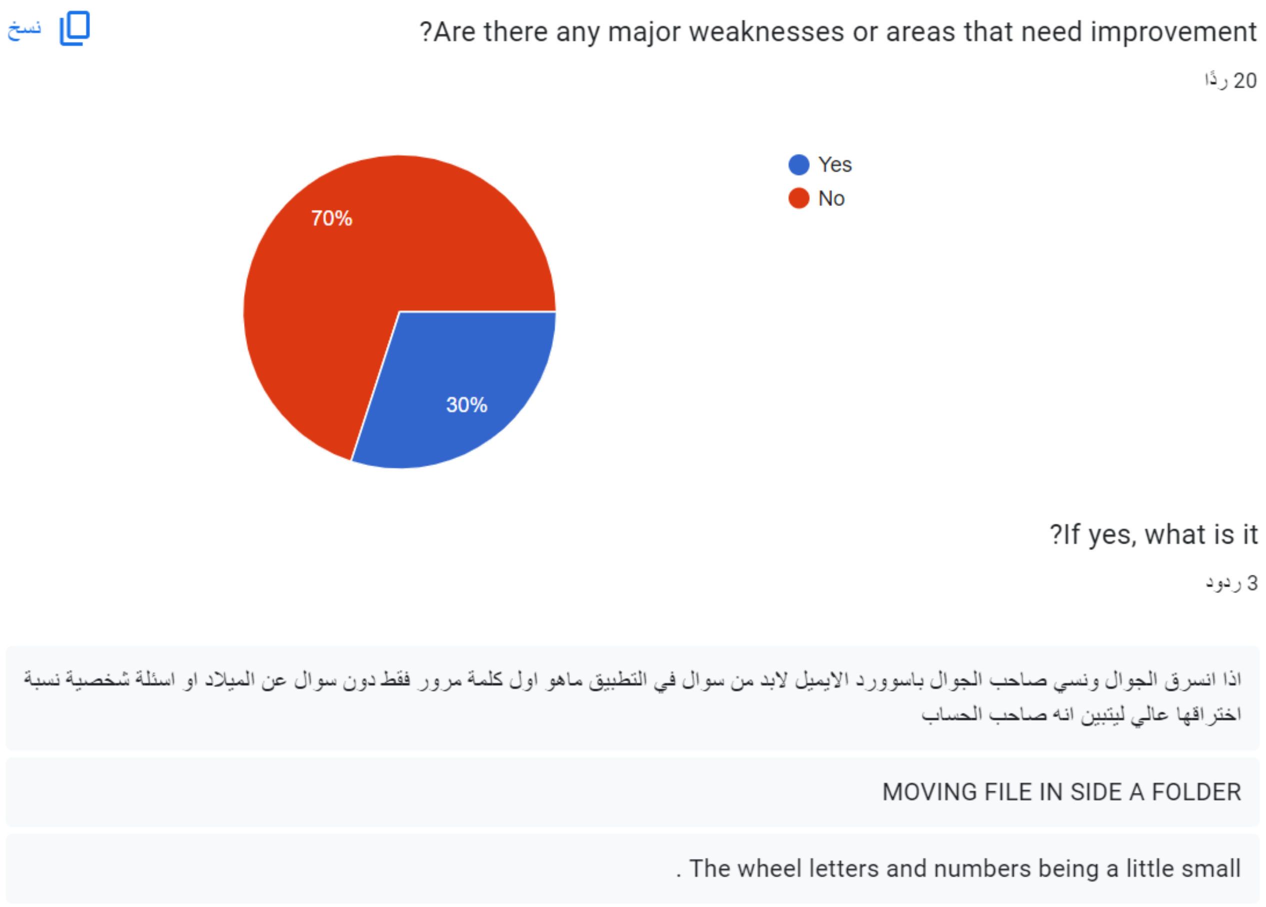
?Were you able to successfully view files sent to you by other users

20 ردًا



?Did you receive notifications when another user shared a file with you

20 ردًا



?Are there any major weaknesses or areas that need improvement

20 ردًا

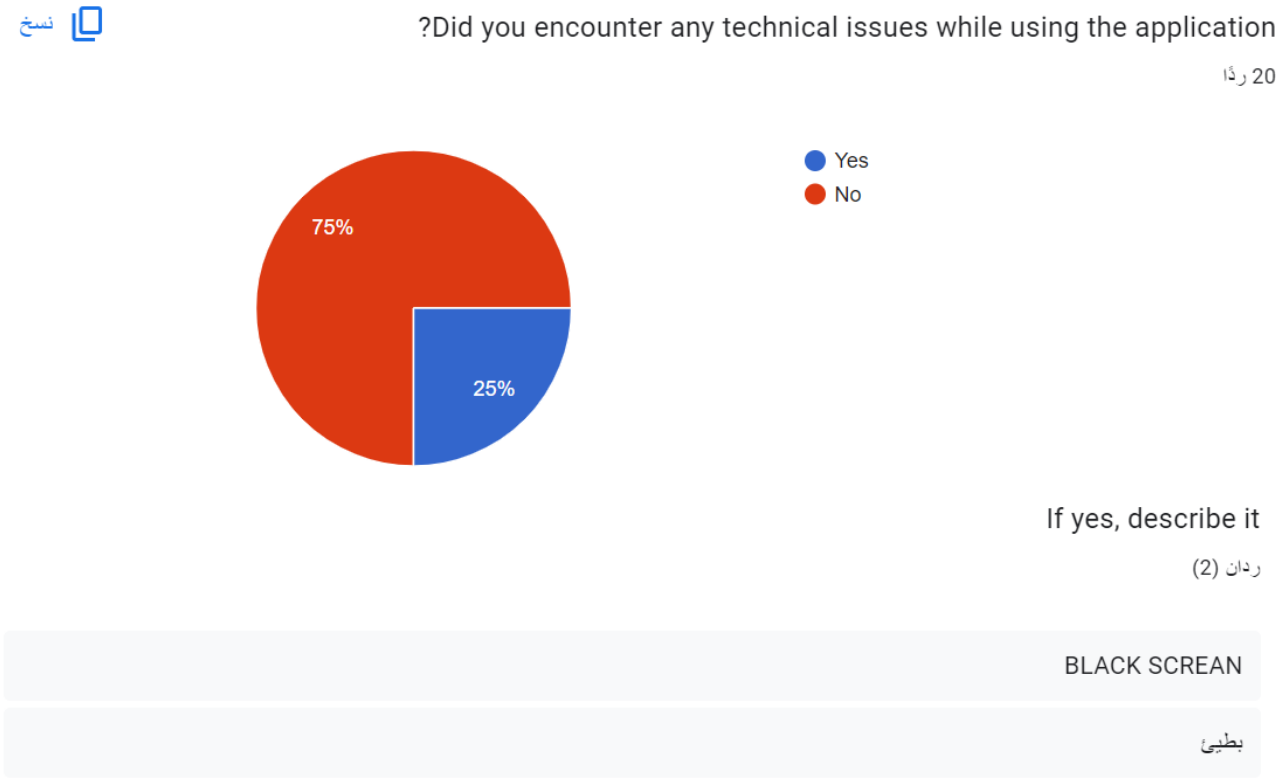
?If yes, what is it

3 ردود

اذا انسرق الجوال ونسي صاحب الجوال باسوورد الايميل لابد من سوال في التطبيق ماهو اول كلمة مرور فقط دون سوال عن الميلاد او اسئلة شخصية نسبة اختراقها عالي ليتبين انه صاحب الحساب

MOVING FILE IN SIDE A FOLDER

. The wheel letters and numbers being a little small



?Did you encounter any technical issues while using the application

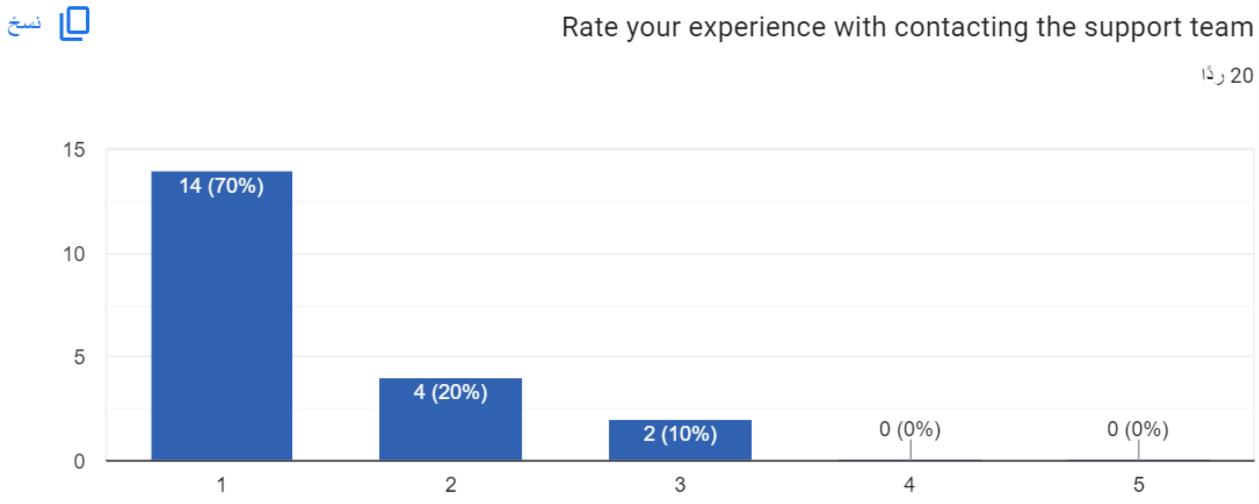
20 ردًا

If yes, describe it

ردان (2)

BLACK SCREAN

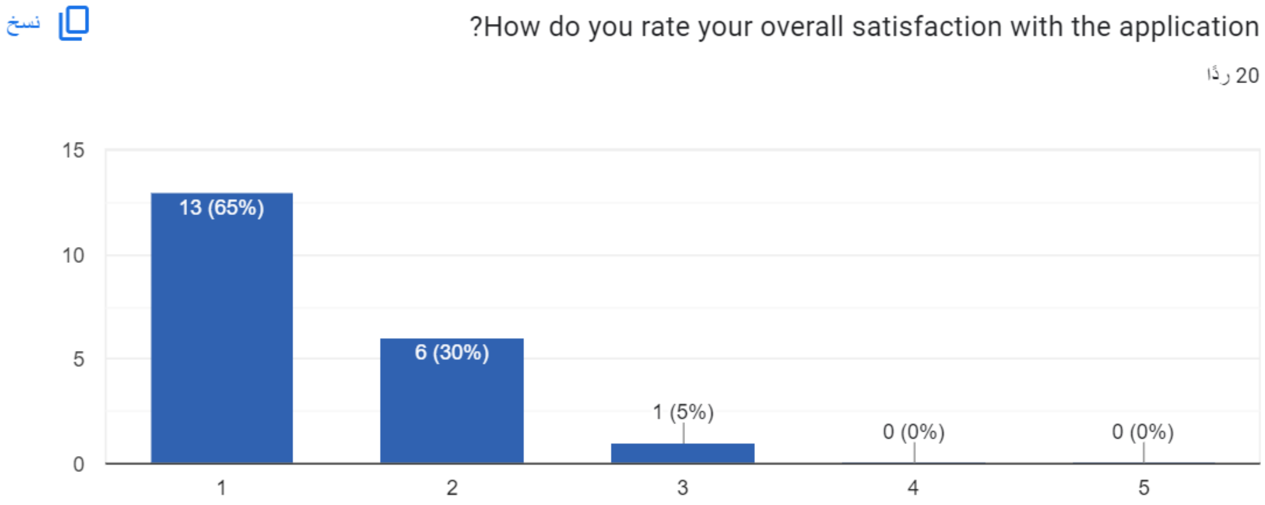
بطيئ



Rate your experience with contacting the support team

20 ردًا

نسخ



?How do you rate your overall satisfaction with the application

20 ردًا

نسخ

9.4 Appendix D: Jira

https://2023-1st-gp16.atlassian.net/jira/software/projects/SB/boards/2/backlog