Ain Shams University
Faculty of Computer & Information Sciences
Computer Science Department

Distributed Mobile Cloud

July 2022

Ain Shams University
Faculty of Computer & Information Sciences
Computer Science Department

Distributed Mobile Cloud

By:

Kirlos Melad Lamey [CSystem]
Azza Abdalla Samy [CSystem]
Marihan Hany Samir [CSystem]
Alyaa Saeed Mohamed [CSystem]
Hisham Sameh Kenawy [CSystem]

Under Supervision of:

Dr.Mahmoud Fayez
Computer Systems Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

TA Rokaya Safwat
Teaching Assistant,
Computer Systems Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

Abstract

Most people nowadays are getting more concerned about protecting their data, so backup crucial to save your important files from inevitable data loss situations due to common events such as system crash, malware infection, hard drive corruption and failure, etc. That comes with the fear of compromising their privacy and having someone or an organization get their hands on the backed-up data and use it for malicious reasons.

Our application - Distributed Mobile Cloud for android devices - provides that layer of privacy by keeping any 3rd-party hands away from your data. Using it will allow you to share your data with your family, friends, coworkers or any group of people safely and securely, as all your files will be encrypted before sharing them. Also, as another layer of security and privacy, you can use File Stripping mode to share your files, which prevents all the members in your group from accessing or viewing it unless everyone is online. You can have as many groups, members and files as you like.

Table of Content

List of Figures

1.Introduction

1.1 Motivation

While online users increasingly rely on the use of mobile applications for their everyday activities and needs, the processing of personal data through such tools poses significant risks to user's privacy. Such risks come mainly from the variety of data and sensors held in mobile devices, the use of different types of identifiers and extended possibility of users' tracking, the complex mobile app ecosystem and limitations of app developers, as well as the extended use of third-party software and services.

Sharing data with group of people may be vulnerable and have not enough privacy, data can be used by 3rd-parties for marketing, spamming or data statistics. Trusting your network to outside companies is not without risk. Putting your critical information out there can be just as dangerous. When you entrust your data and applications to 3rd-parties, you have no real assurances that they will be safe. Everything will be outside your physical control, your information will be managed by others, and you will be susceptible to the changing fortunes of a broadly shared IT environment.

1.2 Problem Definition

Building personal cloud with any group of people to preserve their privacy while sharing and/or backing up their personal files.
Controlling backups and keeping users' privacy by preventing 3rd-parties from using, modifying, selling or keeping a copy of the data after deleting it.

1.3 Objective

The major objectives of Distributed Mobile Cloud: -

- Providing high level of privacy to data.
- Ensuring that operational and sensitive data are not accessible to 3rd-parties' services providers.
- Sharing photos, videos, records and files with a group of people, which also enables backing up personal data to them.
- Helping users recovering their data in case of any type of data loss.

2.Background

2.1 Algorithms

2.1.1. Encryption

Encryption is a form of data security in which information is converted to ciphertext. Only authorized people who have the key can decipher the code and access the original plaintext information. This serves to thwart cybercriminals to find out that the data is unreadable and therefore useless. It can prevent data breaches. Even if an attacker maliciously gains access to a network, if a device is encrypted, the device will still be secure, rendering attempts by the attacker to consume the data useless. Encryption ensures no one can read communications or data except the intended recipient or data owner. This prevents attackers from intercepting and accessing sensitive data.

And we used **Advanced Encryption Standard (AES)**: Developed in 1997 by the National Institute of Standards and Technology (NIST) as an alternative to the Data Encryption Standard, the Advanced Encryption Standard is a cipher chosen by the U.S. government to protect sensitive information. AES has three different key lengths to encrypt and decrypt a block of messages: 128-bit, 192-bit, and 256-bit. AES is widely used for protecting data at rest in such applications as databases and hard drives. We used AES in our app to encrypt Files before we upload them to firebase to secure them from attacking.

2.1.2. Hashing

Hashing is the process of transforming any given key or a string of characters into another value. This is usually represented by a shorter, fixed-length value or key that represents and makes it easier to find or employ the original string. The most popular use for hashing is the implementation of hash tables. A hash table stores key and value pairs in a list that is accessible through its index. Because key and value pairs are unlimited, the hash function will map the keys to the table size. A hash value then becomes the index for a specific element.

SHA-256 is used for this project: SHA-256 stands for Secure Hash Algorithm 256-bit and it's used for cryptographic security. The SHA-256 algorithm is one flavor of SHA-2 (Secure Hash Algorithm 2), which was created by the National Security Agency in 2001 as a successor to SHA-1. SHA-256 is a patented cryptographic hash function that outputs a value that is 256 bits long. It produces irreversible and unique hashes. The larger the number of possible hashes, the smaller the chance that two values will create the same hash. It is one of the most secure hashing functions.

2.2 Techniques

2.2.1. The SOLID Principles

The SOLID Principles are five principles of Object-Oriented class design. They are a set of rules and best practices to follow while designing a class structure. These five principles help us understand the need for certain design patterns and software architecture in general.

It's first introduced by the famous Computer Scientist Robert J. Martin (A.K.A. Uncle Bob) in his paper in 2000. But the SOLID acronym was introduced later by Michael Feathers.

We used The **Single Responsibility Principle**:

It states that a class should do one thing and therefore it should have only a single reason to change.

To state this principle more technically: Only one potential change (database logic, logging logic, and so on.) in the software's specification should be able to affect the specification of the class.

2.2.2 Architecture Pattern

An Architectural Pattern expresses a fundamental structural organization or schema for software systems.

It provides a set of predefined subsystems, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them. It is a general, reusable solution to a commonly occurring problem in software architecture within a given context.

MVC Architecture Pattern is used for this project: The Model View Controller (MVC) architecture pattern specifies that an application consist of a data model, presentation information, and control information. The pattern requires that each of these be separated into different objects.

- The Model contains only the pure application data, it contains no logic describing how to present the data to a user.
- The View presents the model's data to the user. The view knows how to access the model's data, but it does not know what this data means or what the user can do to manipulate it.
- The Controller exists between the view and the model. It listens to events triggered by the view (or another external source) and executes the appropriate reaction to these events. In most cases, the reaction is to call a method on the model. Since the view and the model are connected through a notification mechanism, the result of this action is then automatically reflected in the view.

Advantages:

- Multiple developers can work simultaneously on the model, controller and views.
- MVC enables logical grouping of related actions on a controller together. The views for a specific model are also grouped together.
- Models can have multiple views.

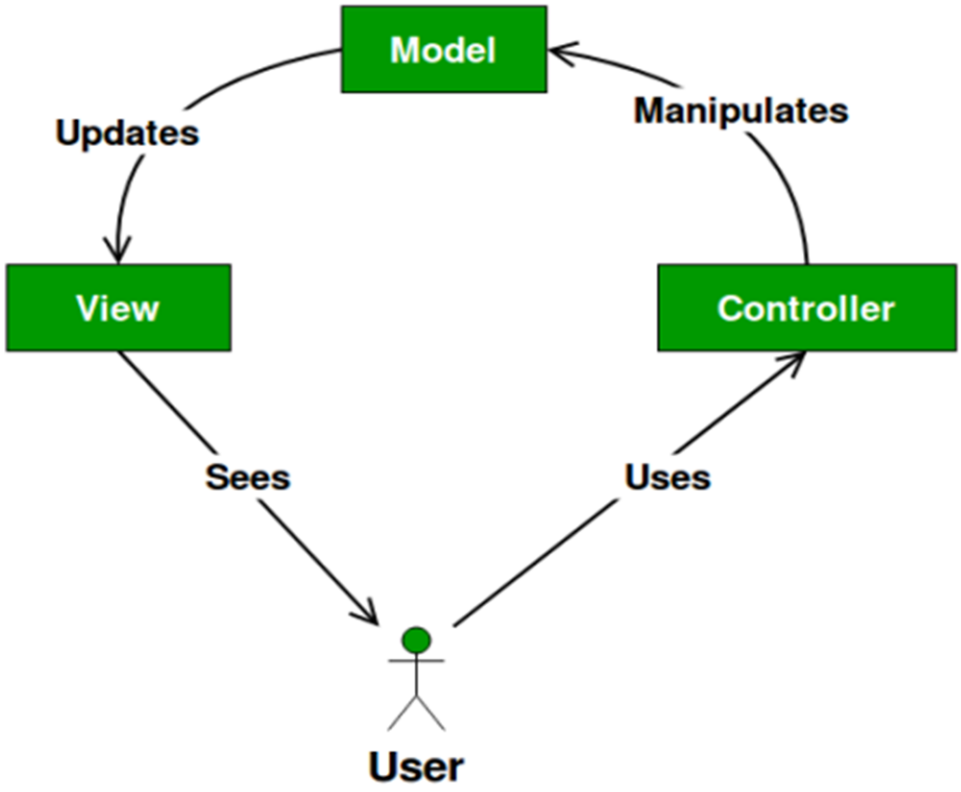


Figure 1. MVC Architecture Pattern

2.2.3. Design Patterns

A design pattern is a general repeatable solution to a commonly occurring problem in software design. A design pattern isn't a finished design that can be transformed directly into code. It is a description or template for how to solve a problem that can be used in many different situations.

Design patterns can speed up the development process by providing tested, proven development paradigms. Effective software design requires considering issues that may not become visible until later in the implementation. Reusing design patterns helps to prevent subtle issues that can cause major problems and improves code readability for coders and architects familiar with the patterns.

Often, people only understand how to apply certain software design techniques to certain problems. These techniques are difficult to apply to a broader range of problems. Design patterns provide general solutions, documented in a format that doesn't require specifics tied to a particular problem.

In addition, patterns allow developers to communicate using well-known, well understood names for software interactions. Common design patterns can be improved over time, making them more robust than ad-hoc designs.

The design patterns used within the application are:

Singleton Design Pattern

- Ensure a class has only one instance, and provide a global point of access to it.
- Encapsulated "just-in-time initialization" or "initialization on first use".

Make the class of the single instance object responsible for creation, initialization, access, and enforcement. Declare the instance as a private static data member. Provide a public static member function that encapsulates all initialization code, and provides access to the instance.

Mediator Design Pattern

- Define an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently.
- Design an intermediary to decouple many peers.
- Promote the many-to-many relationships between interacting peers to "full object status".

Observer Design Pattern

- Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.
- Encapsulate the core (or common or engine) components in a Subject abstraction, and the variable (or optional or user interface) components in an Observer hierarchy.
- The "View" part of Model-View-Controller.

2.2.4. Multithreading

Is the ability of a central processing unit (CPU) to provide multiple threads of execution concurrently, supported by the operating system. This approach differs from multiprocessing. In a multithreaded application, the threads share the resources of a single or multiple cores, which include the computing units and the CPU caches. Where multiprocessing systems include multiple complete processing units in one or more cores, multithreading aims to increase utilization of a single core by using thread-level parallelism, as well as instruction-level parallelism. As the two techniques are complementary, they are combined in nearly all modern systems architectures with multiple multithreading CPUs and with CPUs with multiple multithreading cores.

2.2.5. Data View

It is a feature that allows you to more easily write code that interacts with views. Data Binding Library is a support library that allows you to bind UI components in your layouts to data sources in your app using a declarative format rather than programmatically. Once view binding is enabled in a module, it generates a binding class for each XML layout file present in that module. An instance of a binding class contains direct references to all views that have an ID in the corresponding layout.

Advantages:

- Faster compilation: View binding requires no annotation processing, so compile times are faster.
- Ease of use: View binding does not require specially-tagged XML layout files, so it is faster to adopt in your apps. Once you enable view binding in a module, it applies to all of that module's layouts automatically.

2.3 Tools

2.3.1. Firebase

Firebase is an app development platform that helps up build and grow apps and games for users. Backed by google and trusted by millions of businesses around the world.

It provide many important products like:

- **Firebase Realtime Database:** an API that synchronizes application data across iOS, Android, and Web devices, and stores it on Firebase's cloud. The product assists software developers in building real-time, collaborative applications.
- **Firebase Authentication:** provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more. Firebase Authentication integrates tightly with other Firebase services, and it leverages industry standards like OAuth 2.0 and OpenID Connect, so it can be easily integrated with your custom backend.
- **Cloud Storage:** is a powerful, simple, and cost-effective object storage service built for Google scale. The Firebase SDKs for Cloud Storage add Google security to file uploads and downloads for your Firebase apps, regardless of network quality.

2.3.2. Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development.[8] It is available for download on Windows, macOS and Linux based operating systems or as a subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development. Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. Android Studio supports all the same programming languages of IntelliJ (and CLion) e.g. Java, C++, and more with extensions, such as Go; and Android Studio 3.0 or later supports Kotlin and "all Java 7 language features and a subset of Java 8 language features that vary by platform version." External projects backport some Java 9 features. While IntelliJ states that Android Studio is built on supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8 support). At least some new language features up to Java 12 are usable in Android.

2.3.3. Java Programming Language

Java is a set of computer software and specifications developed by James Gosling at Sun Microsystems, which was later acquired by the Oracle Corporation, that provides a system for developing application software and deploying it in a cross-platform computing environment. Java is used in a wide variety of computing platforms from embedded devices and mobile phones to enterprise servers and supercomputers. Java applets, which are less common than standalone Java applications, were commonly run in secure, sandboxed environments to provide many features of native applications through being embedded in HTML pages.

3. Analysis & Design

3.1 System Overview

3.1.1. System Architecture

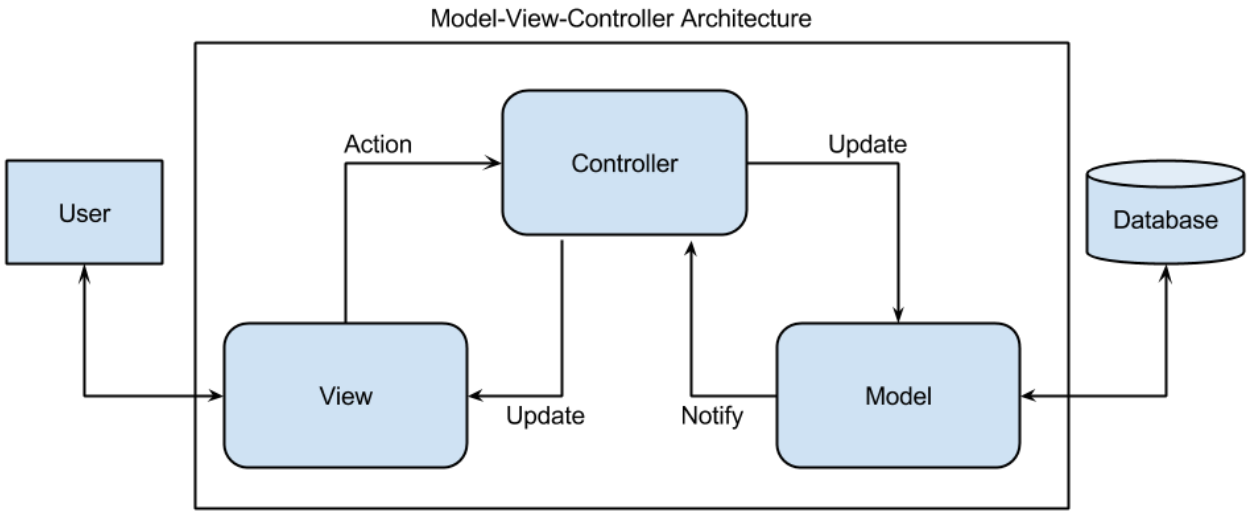


Figure 2. System Architecture

3.1.2. DFD Diagram:

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled

Level-1 DFD:

A picture of the movement of data between external entities and the processes and data stores within a system.

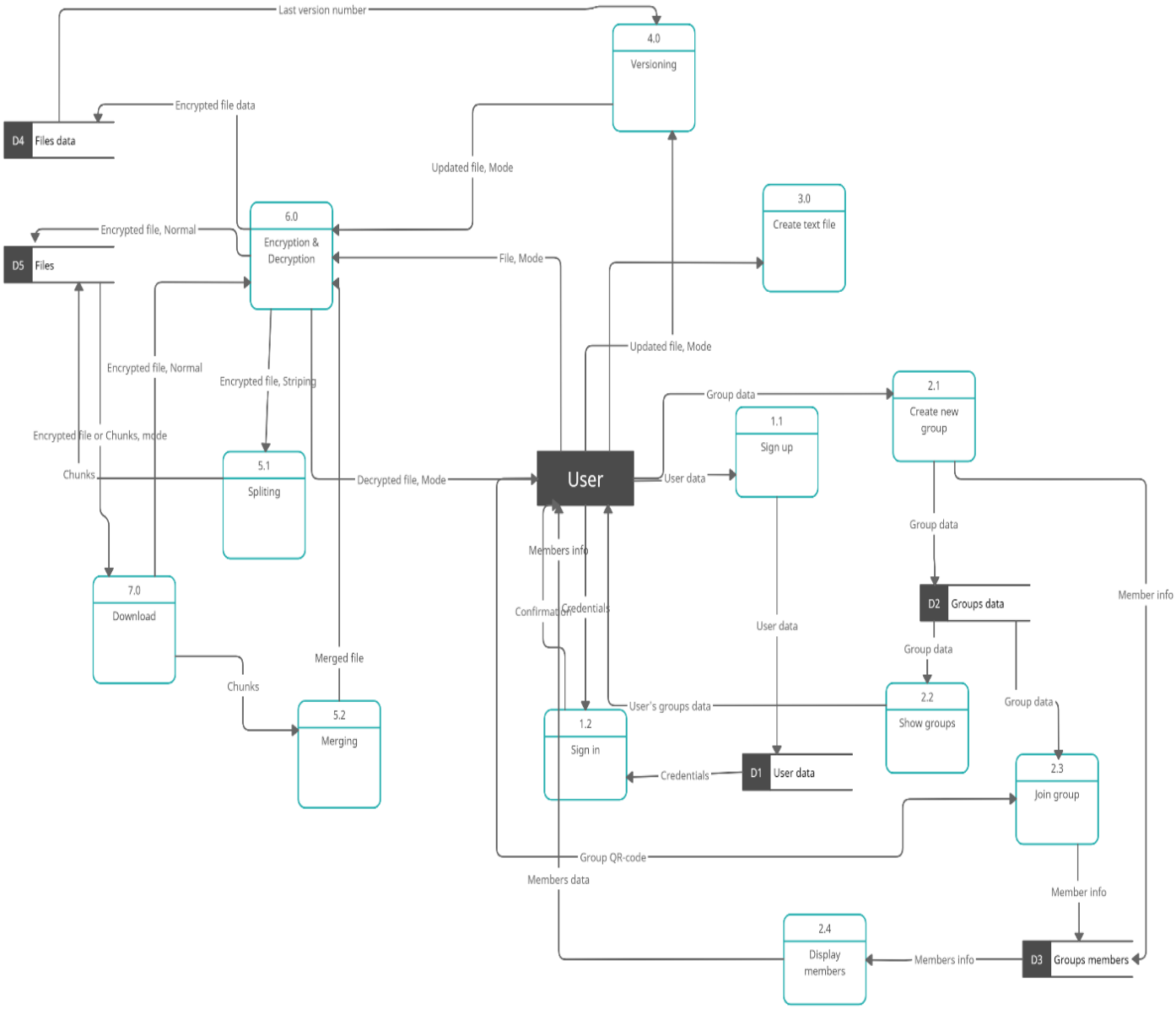


Figure 3. Level-1 DFD Diagram

3.1.3. System Users

Intended Users:

The application is built to general audience, all ages and kinds of people can use it to share their data with family and friends easily, directly and safely.

The user that intends to share a file need only to have the distributed mobile cloud application via his phone, stable connection to internet and to create or join a group then he can start sharing with the group members.

User Characteristics

It’s not required from the user to have an advanced experience with mobile phone, cloud or internet. The user need only to have abstract knowledge on using his mobile phone to be apple to use the application to create, join or view groups or to share or view the files.

3.2. System Design

3.2.1. Business Process Model

Upload Process:

The upload process may start or be invoked by two different events:

- User shares a new file
  - Here the user needs to select the sharing mode of the file before handing it to the system for the upload.
  - After choosing the preferred mode the system encrypts the file then sends it to the upload task.
- User modifies an existing file
  - Here the system knows that the file already exists and synchronized so it asks for the shared mode.
  - After retrieving the mode from firebase it encrypts the file and uploads it again.

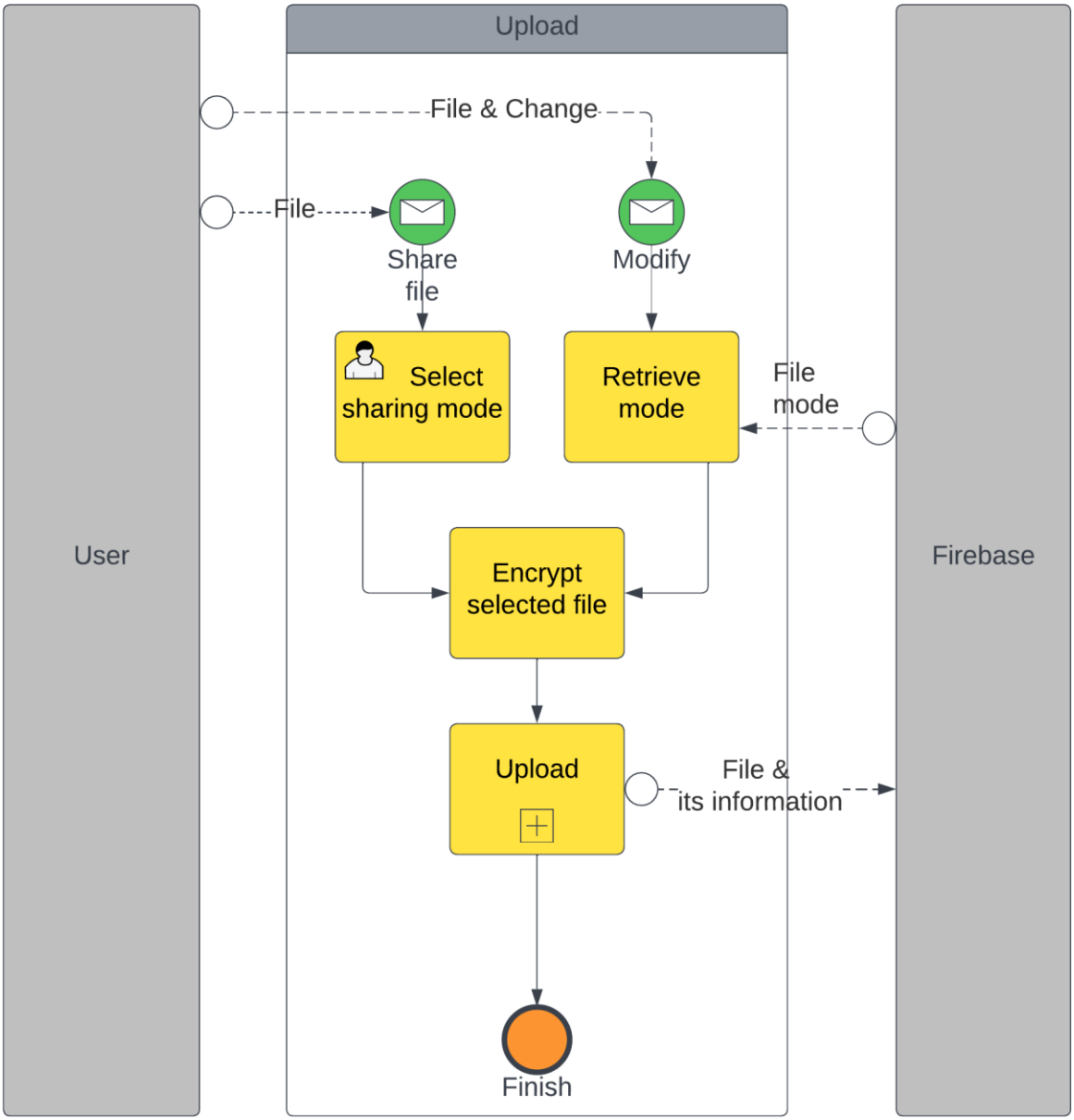


Figure 4. BPM Upload

Upload Sub-process

The system must check for the sharing mode before starting its upload task.

The system has two sharing modes that defines the flow of the upload sub-process:

- Normal mode:
  The system starts uploading the encrypted file right away then it invokes the versioning task for the file.
- Striping mode:
  The system needs to split or divide the encrypted file into smaller parts called chunks which will be equal to the number of the group members. Then the chunks are uploaded to firebase after that the system invokes the versioning task for the file.

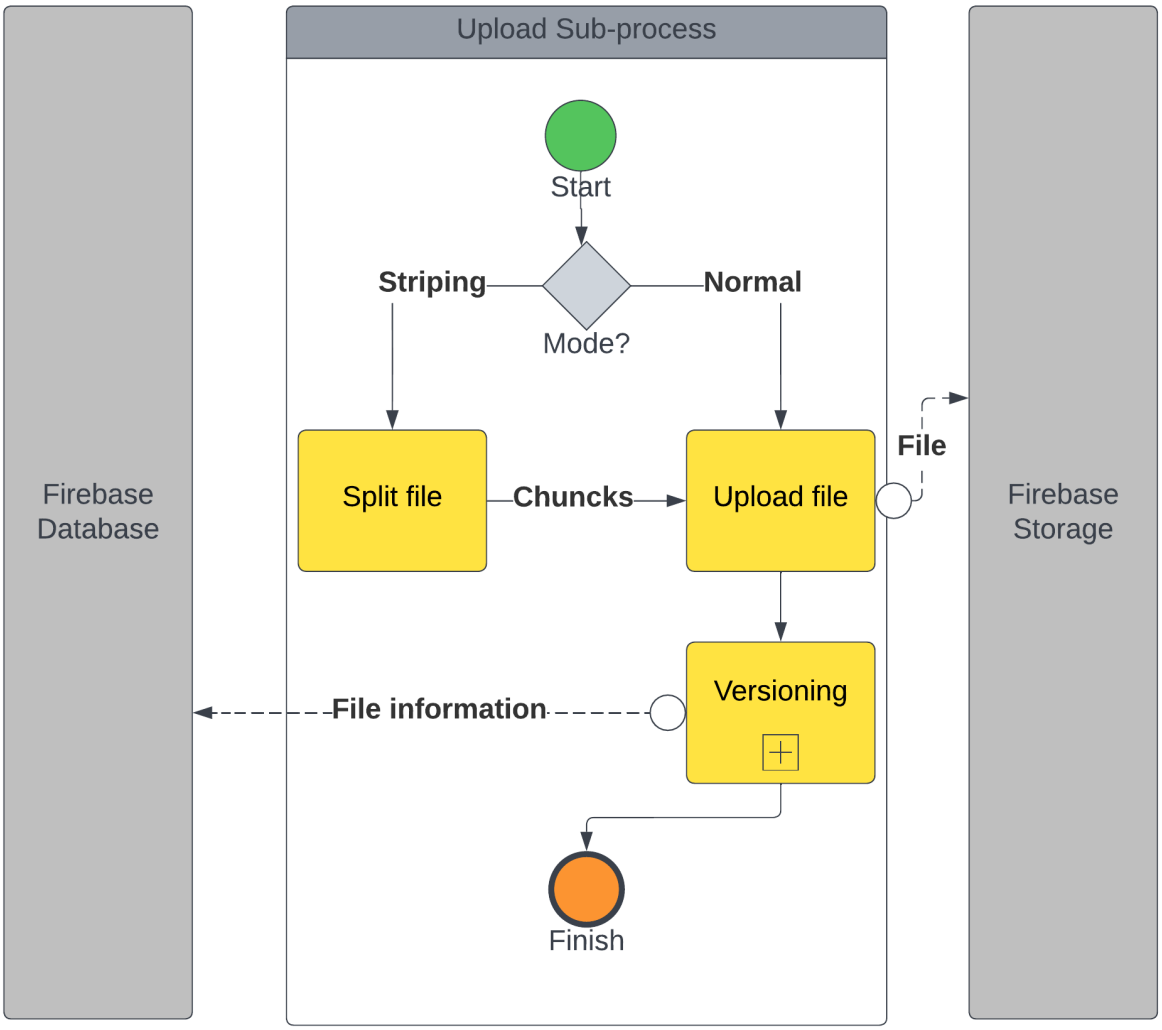


Figure 5. PBM Upload Sub-process

Versioning Sub-process

The system keeps track of the files and its changes across its lifetime or its deletion to do so the system needs some information about the file beforehand then it determines how to handle it according to the type of change of the file.

We have two types of information:

- General information:
  Like its location in the storage and the sharing mode which will be kept under a unique id.
- Variable info:
  Like its name and upload date which will be kept under the new version number.

The system has different types of changes yet when dealing with versioning it only needs to know if the file is new or not:

- If the file is new all the file extracted data will be added as a new row in the database with version number ZERO
- Else only variable information is needed which will be added under the incremented version number .

Conflict Scenario:

We have 2 users (A) and (B) in a group and synchronized file (X.1), if user (A) is offline and he attempts to modify the file he has to wait for the connection to be restored, however user (B) attempts to modify this file after user (A) and due to having an internet connection the file is upload as (X.2). Upon connecting to the internet user (A) will upload the file as (X.3) although it should be (X.2). User (B) version can be restored but has to be merged manually which will create (X.4).

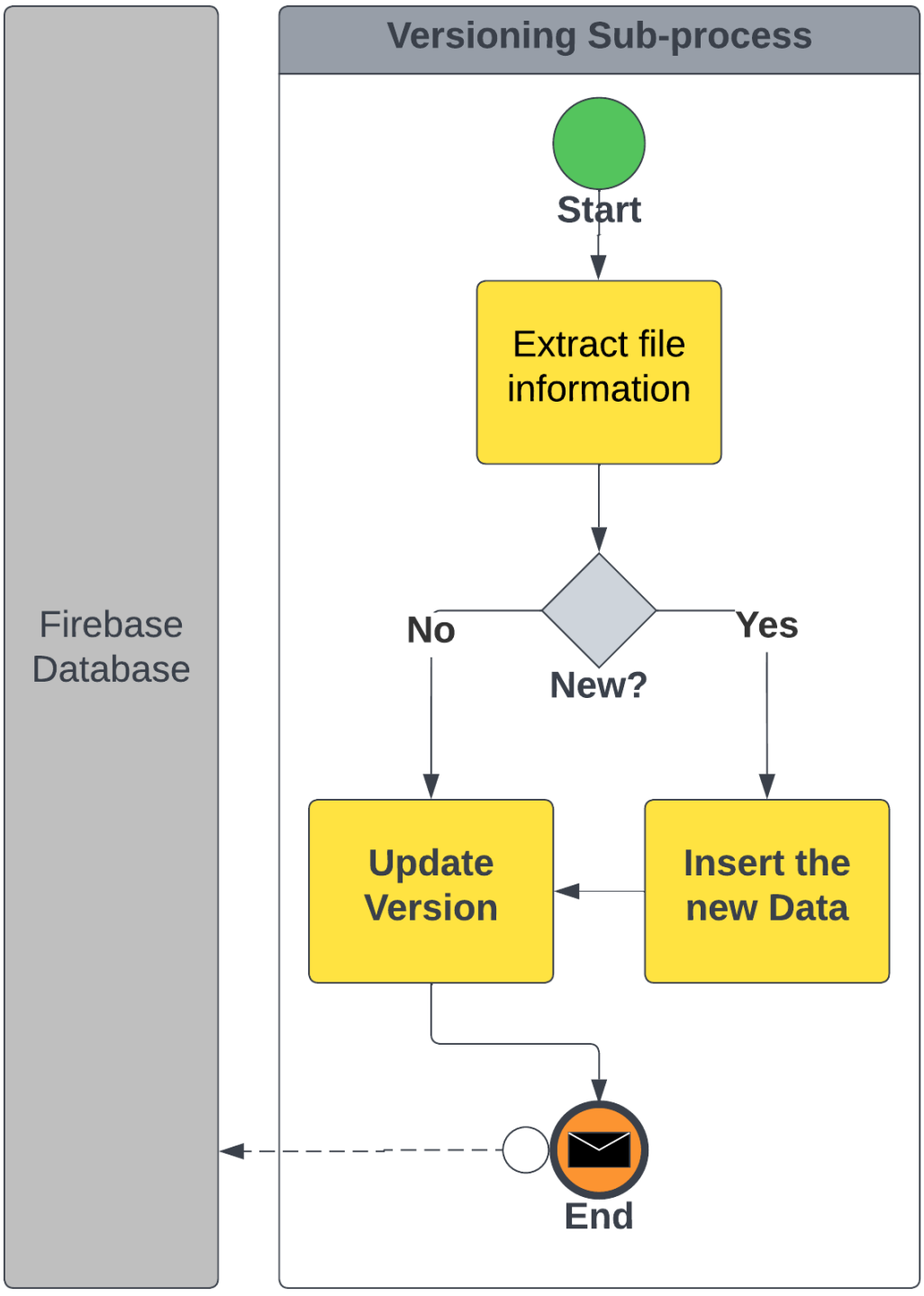


Figure 6. BPM Versioning Sub-process

Download Process

The download process may start or be invoked by two different events:

- A new file is uploaded to firebase which invokes the download process with the file information.
    - Here we have all the needed information ready upon start so the system downloads the file right away.
    - After downloading is complete the system checks for its mode if it's normal then the file is ready to be used so it is decrypted and stored in the user's device storage.
    - However if it's shared in striped mode the file is just a chunk which is useless by itself so it doesn't get decrypted for now and will be just kept in the user's device storage.
- All the group members become online
    - Upon having all the members come online firebase invokes the process that starts collecting the information of each shared file in striping mode to collect the rest of its chunks.
    - After retrieving the needed information the system downloads all the chunks needed by each file, get merged and decrypted then it becomes available for the user to use. Although it will be deleted again if at least one member goes offline.

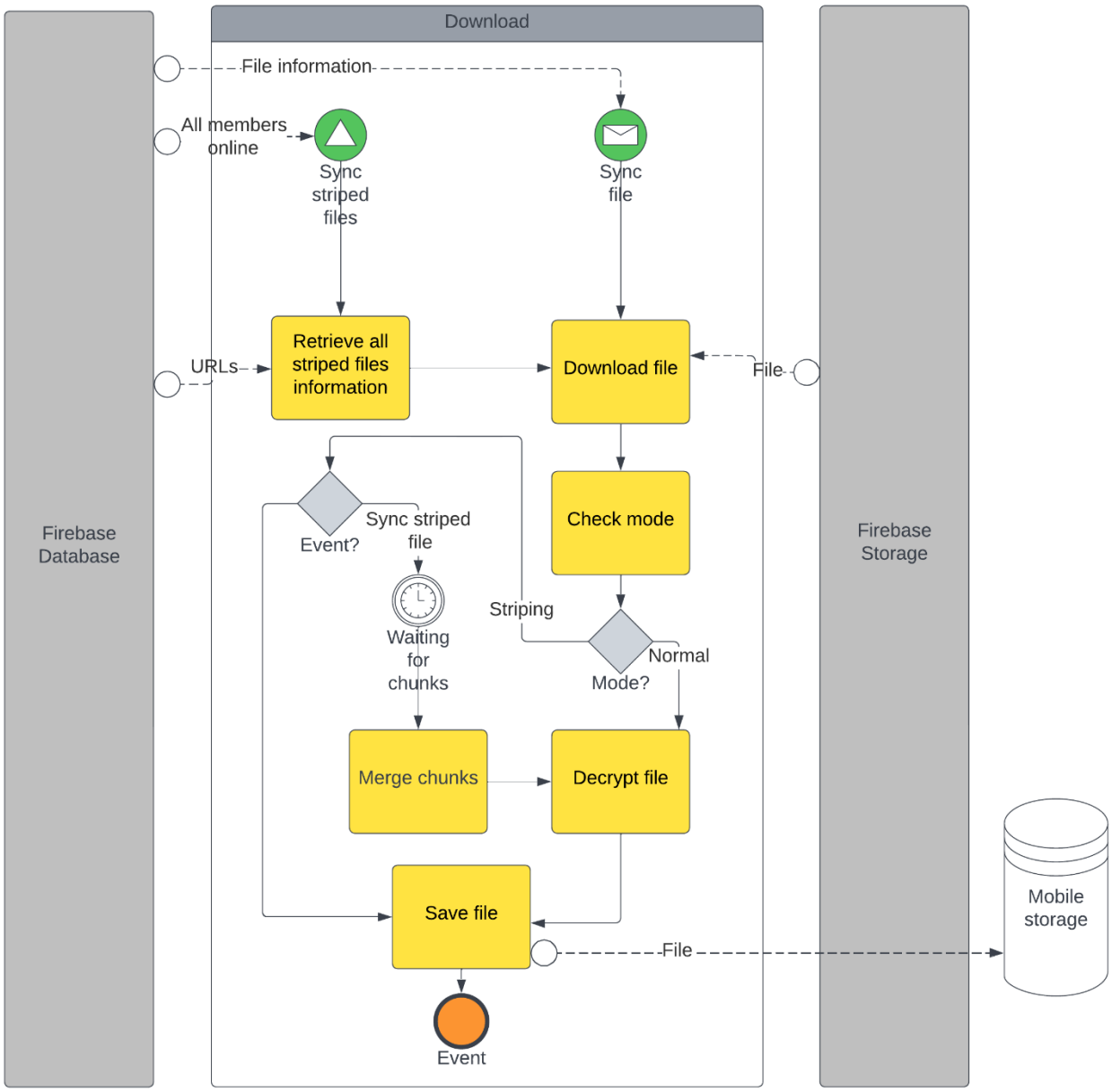


Figure 7. BPM Download Process

3.2.2. State Machine Diagram

State Machine diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. State Machine diagrams are often used in business process modeling. They can also describe the steps in a use case diagram. Activities modeled can be sequential and concurrent.

Here the diagram initially represent the authentication steps of the user like following:

- If the user is signed in, home page open directly.
- If not, the user directed to sign in activity, if he is not authenticated he can choose to sign up and email verification used to ensure the email validity.
- If user forget his password he can reset it.
- After going to home page, the user can join group, create group, open existing group or view his profile.
- When selecting specific group The user have many options like:
    - sharing file
    - add text file
    - show QR code
    - edit file
    - delete file
    - rename file
    - get older version of a file

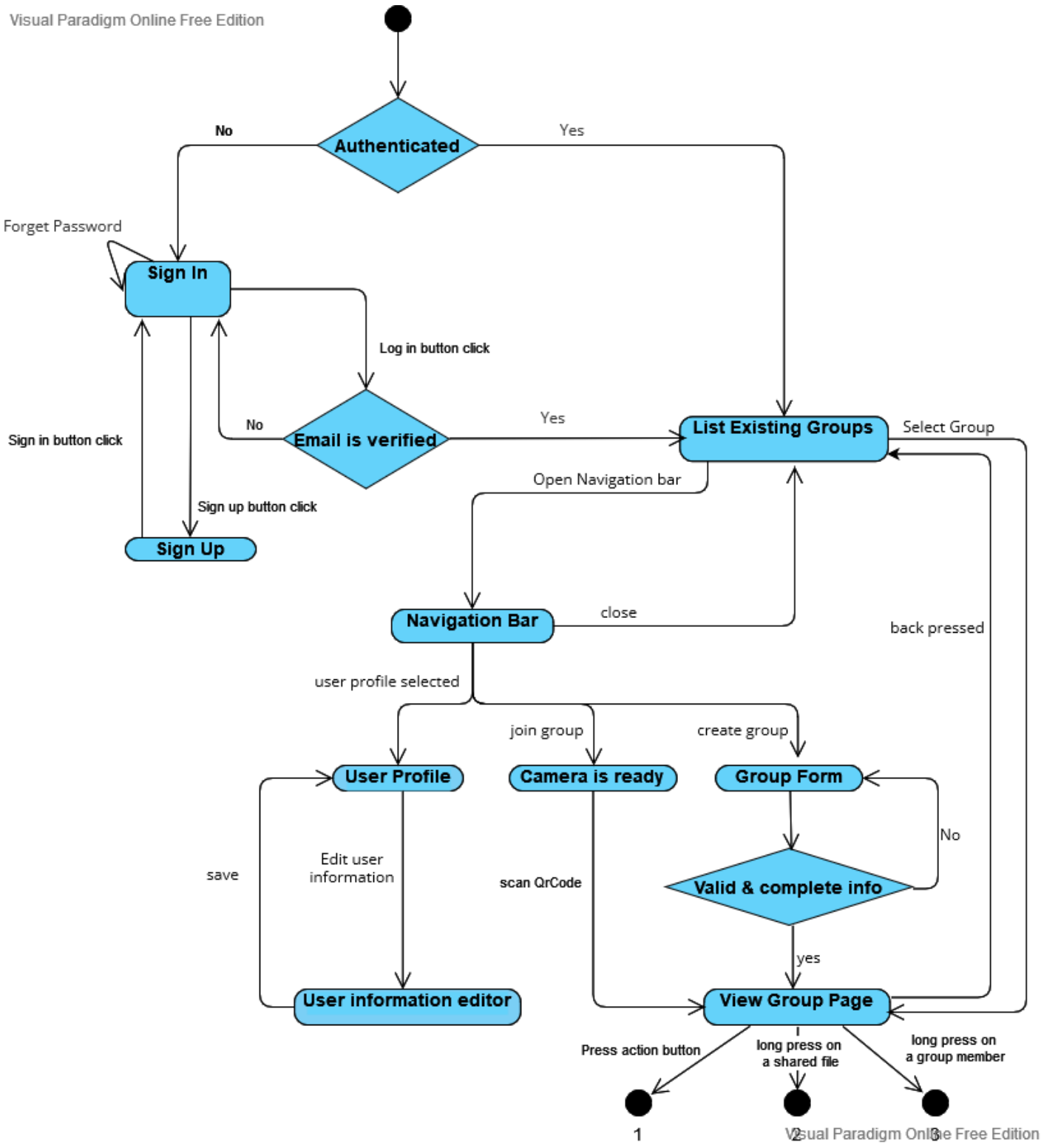


Figure 8. State Machine Diagram

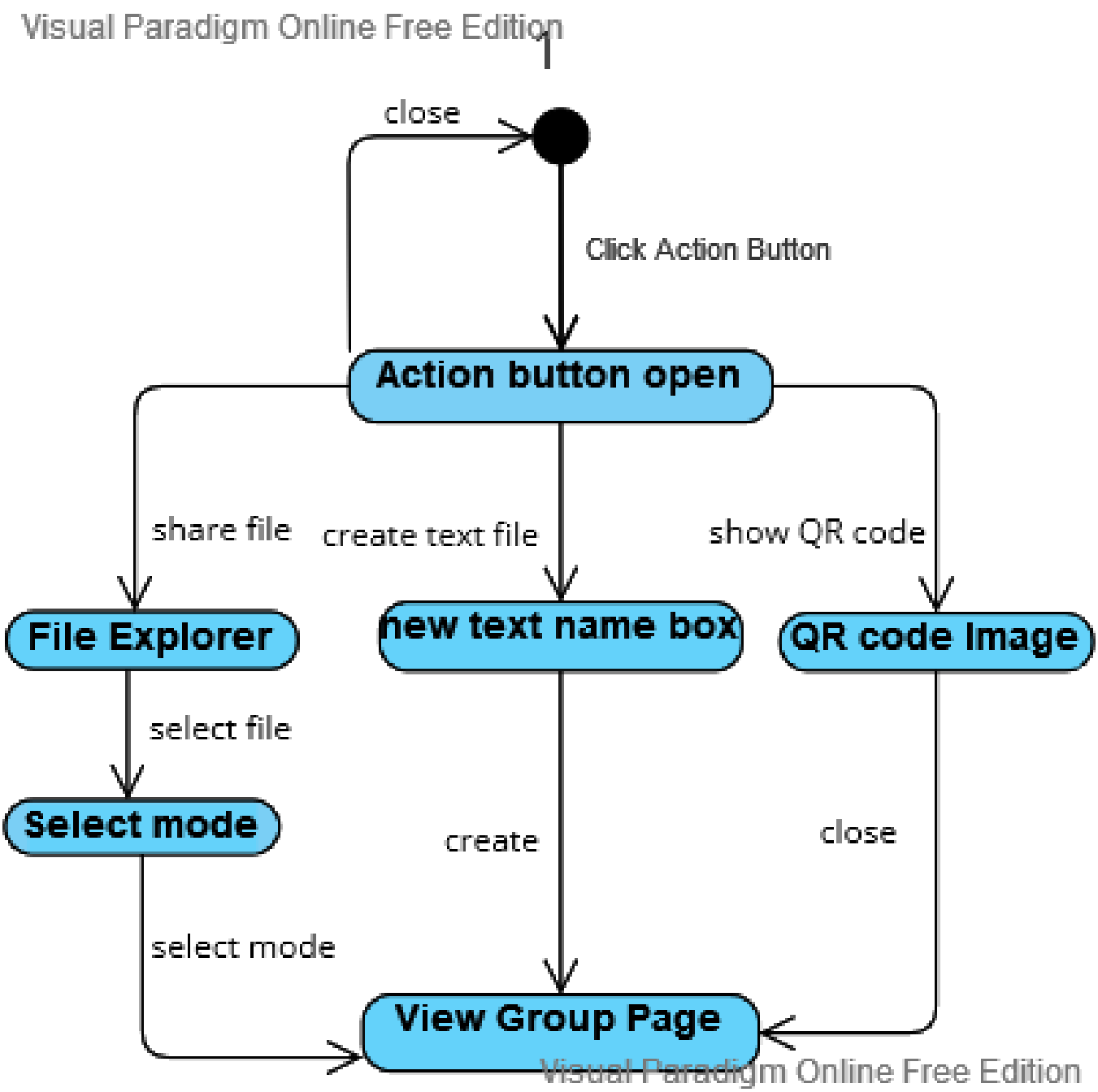


Figure 9. State Machine Diagram1

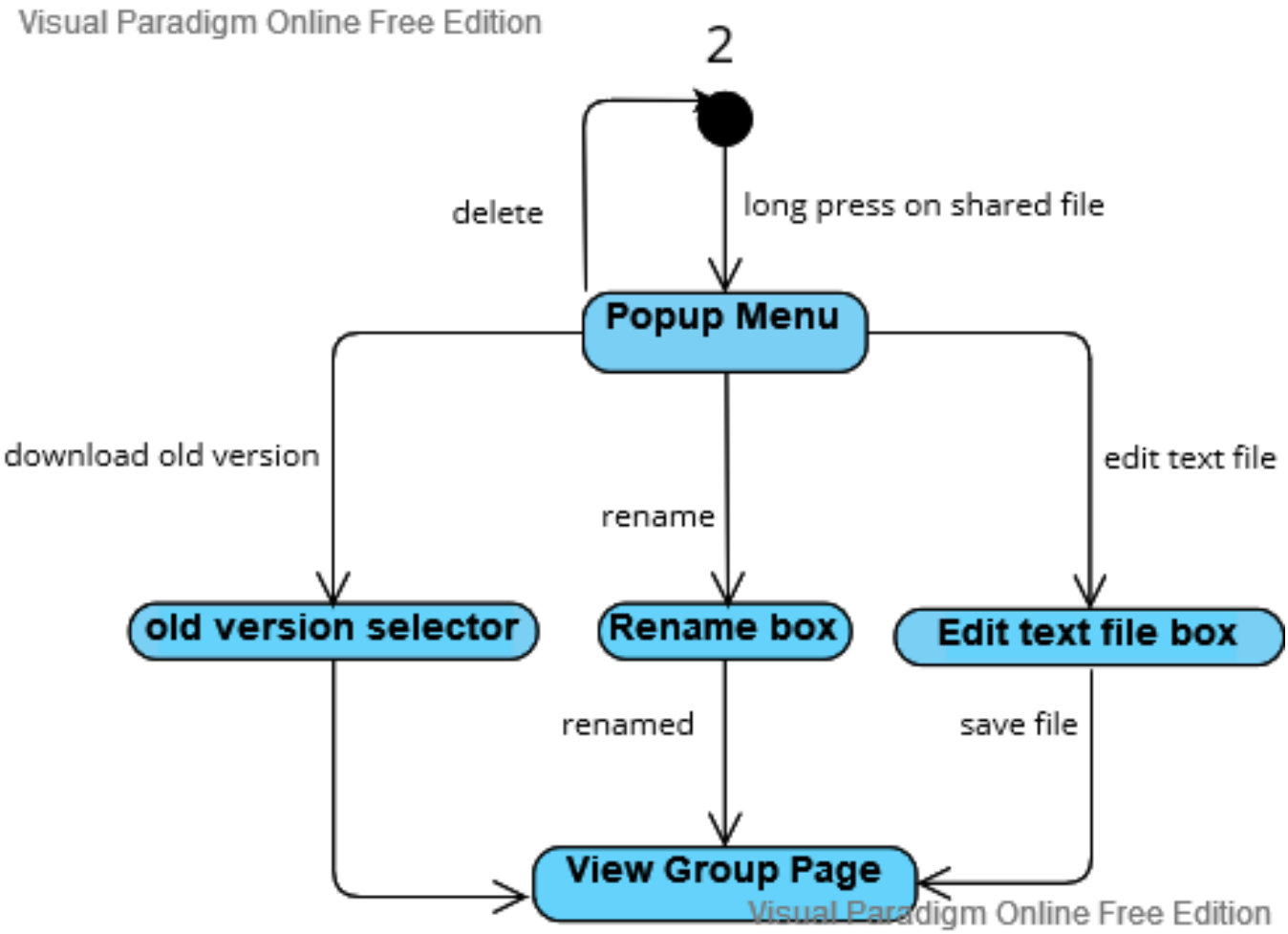


Figure 10. State Machine Diagram2

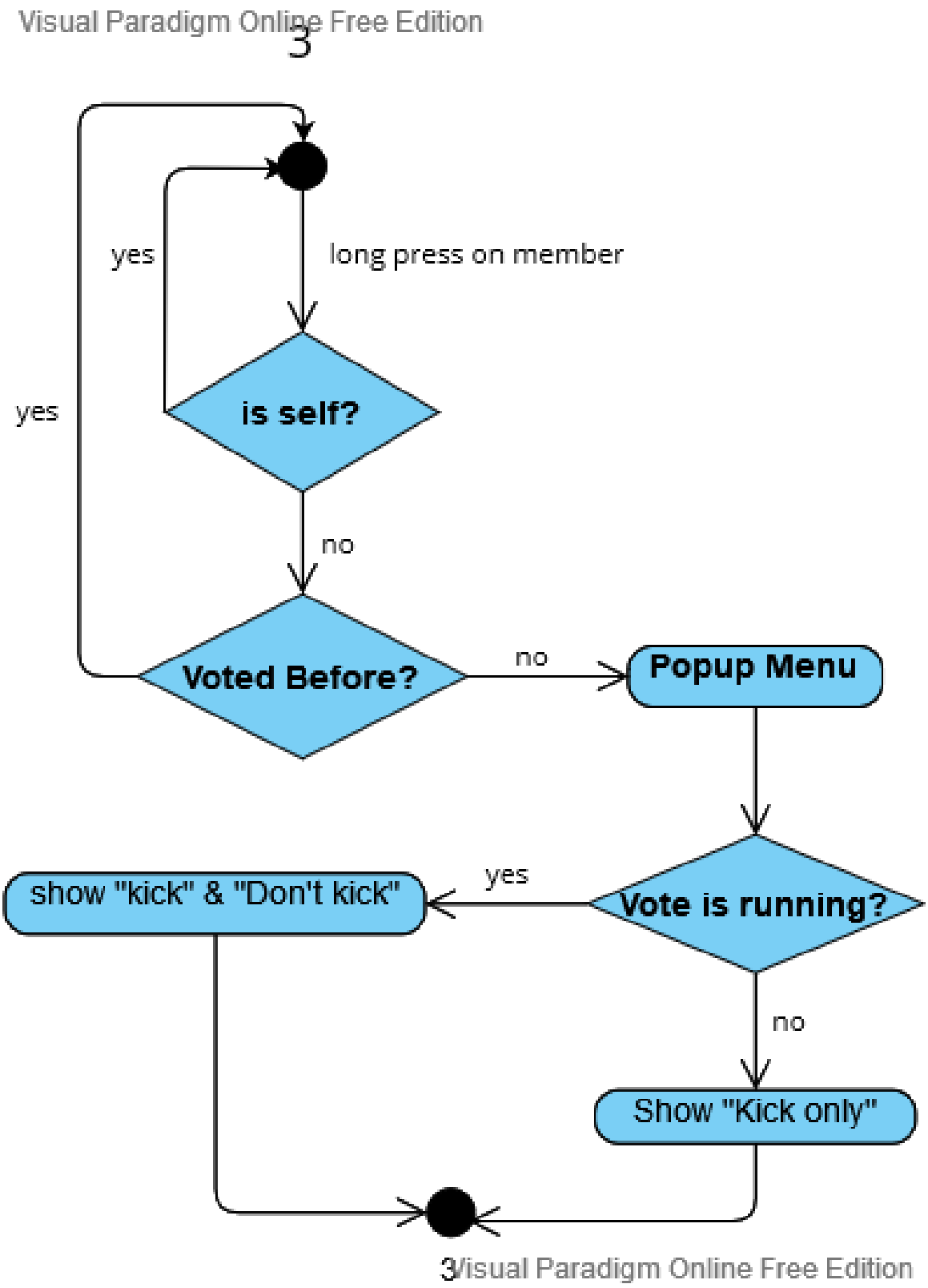


Figure 11. State Machine Diagram3

3.2.3. Class Diagram

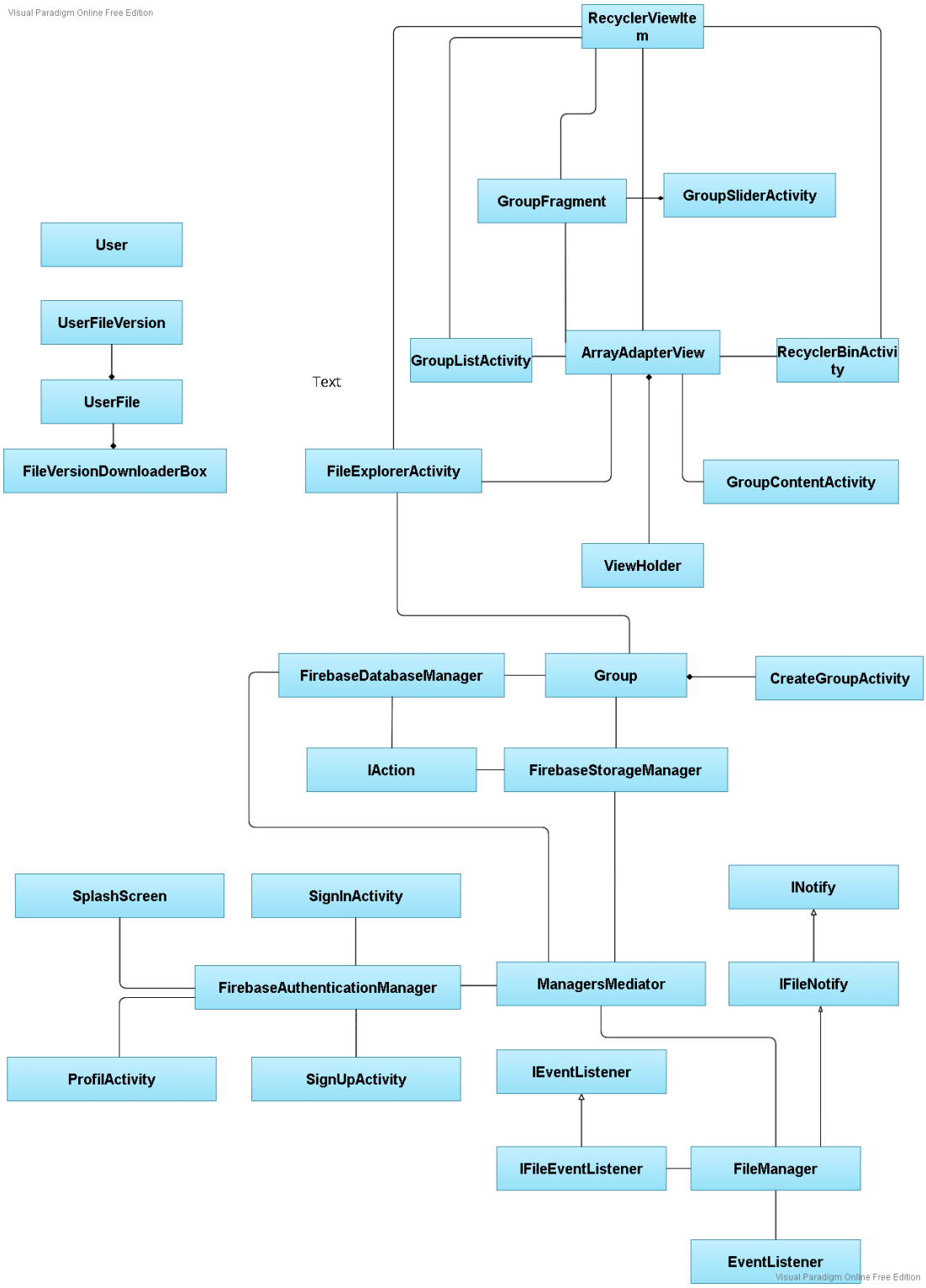


Figure 12. Overall Class Diagram

The Classes in Details:

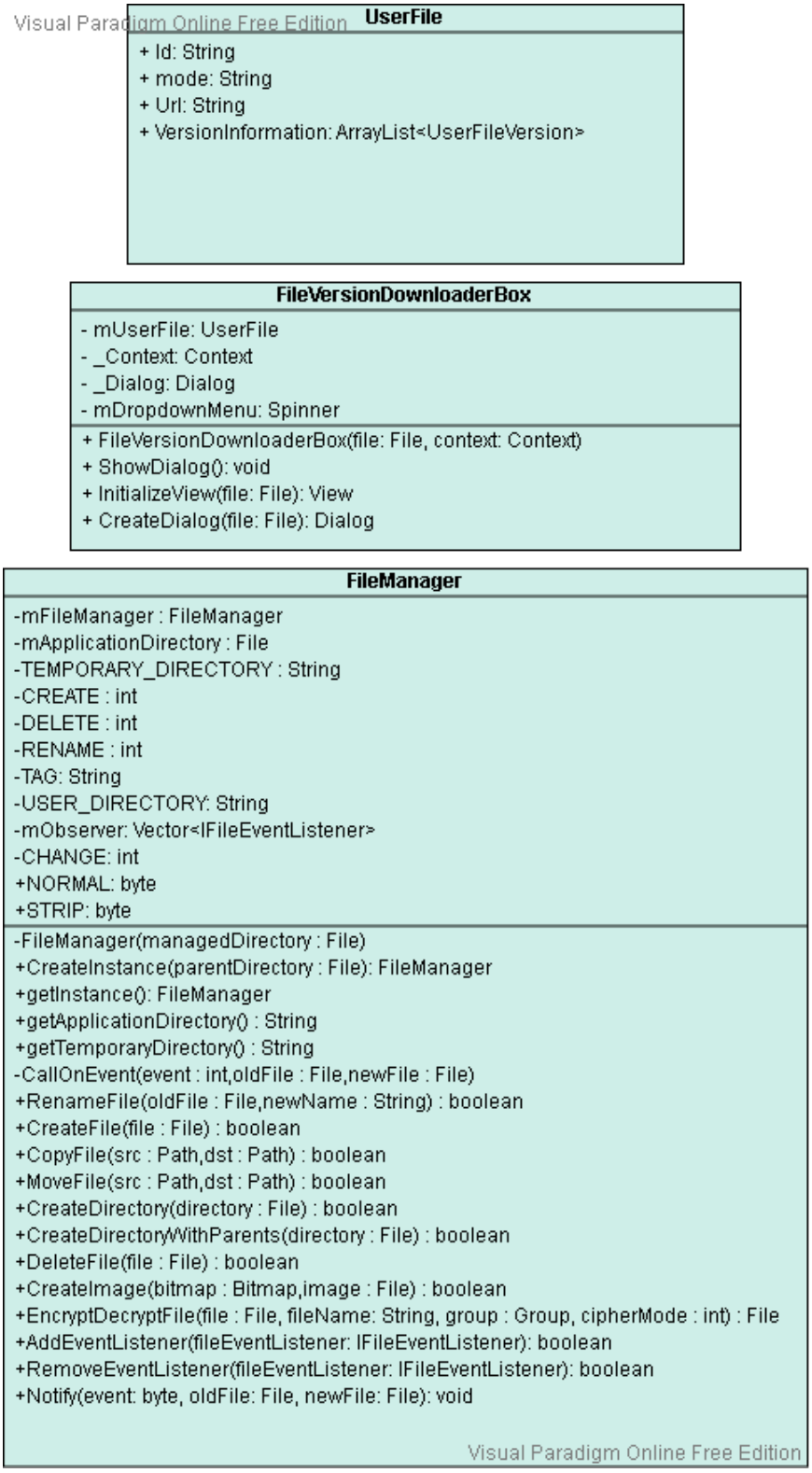


Figure 13. Set 1 of Classes

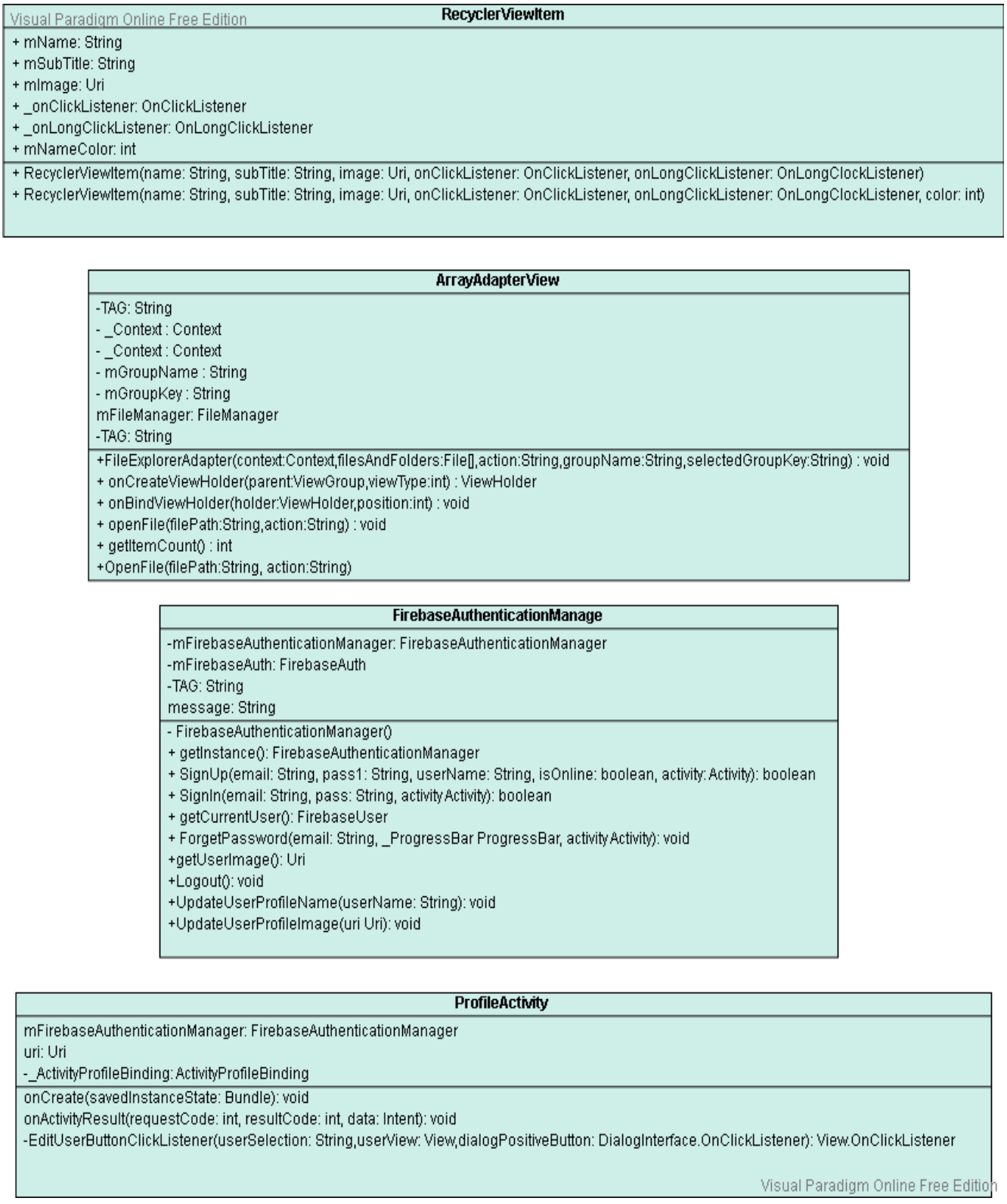


Figure 14. Set 2 of Classes

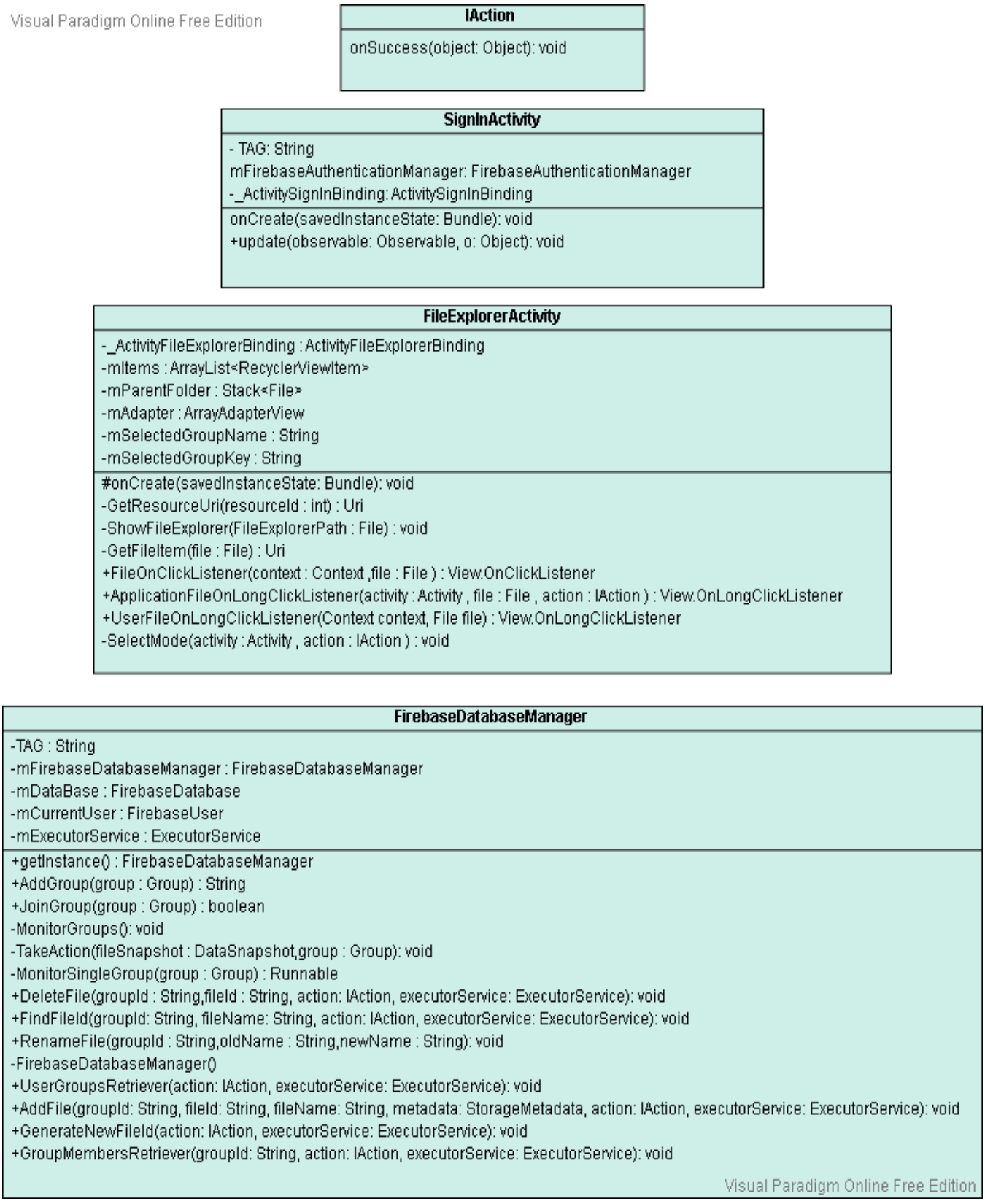


Figure 15. Set 3 of Classes

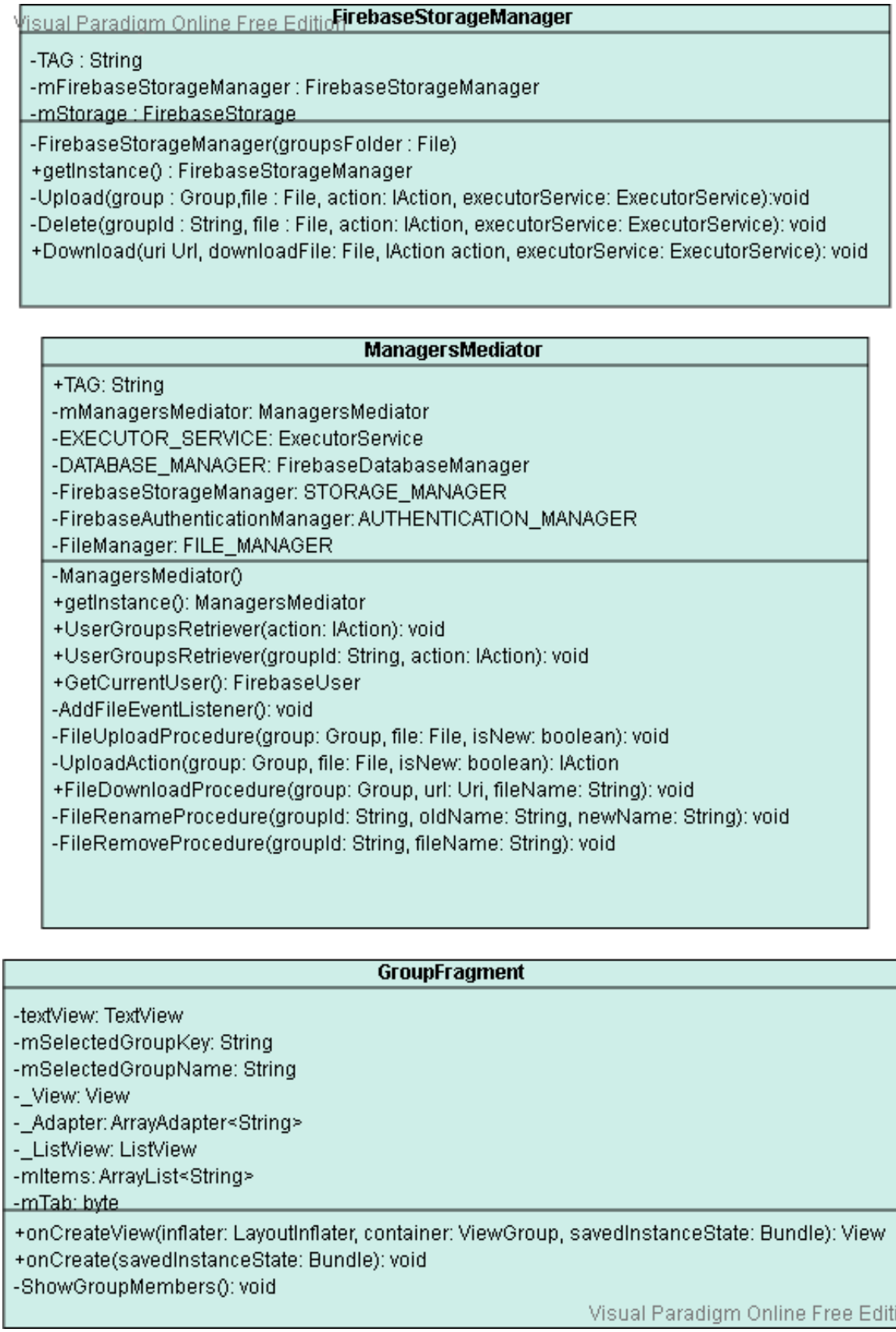


Figure 16. Set 4 of Classes

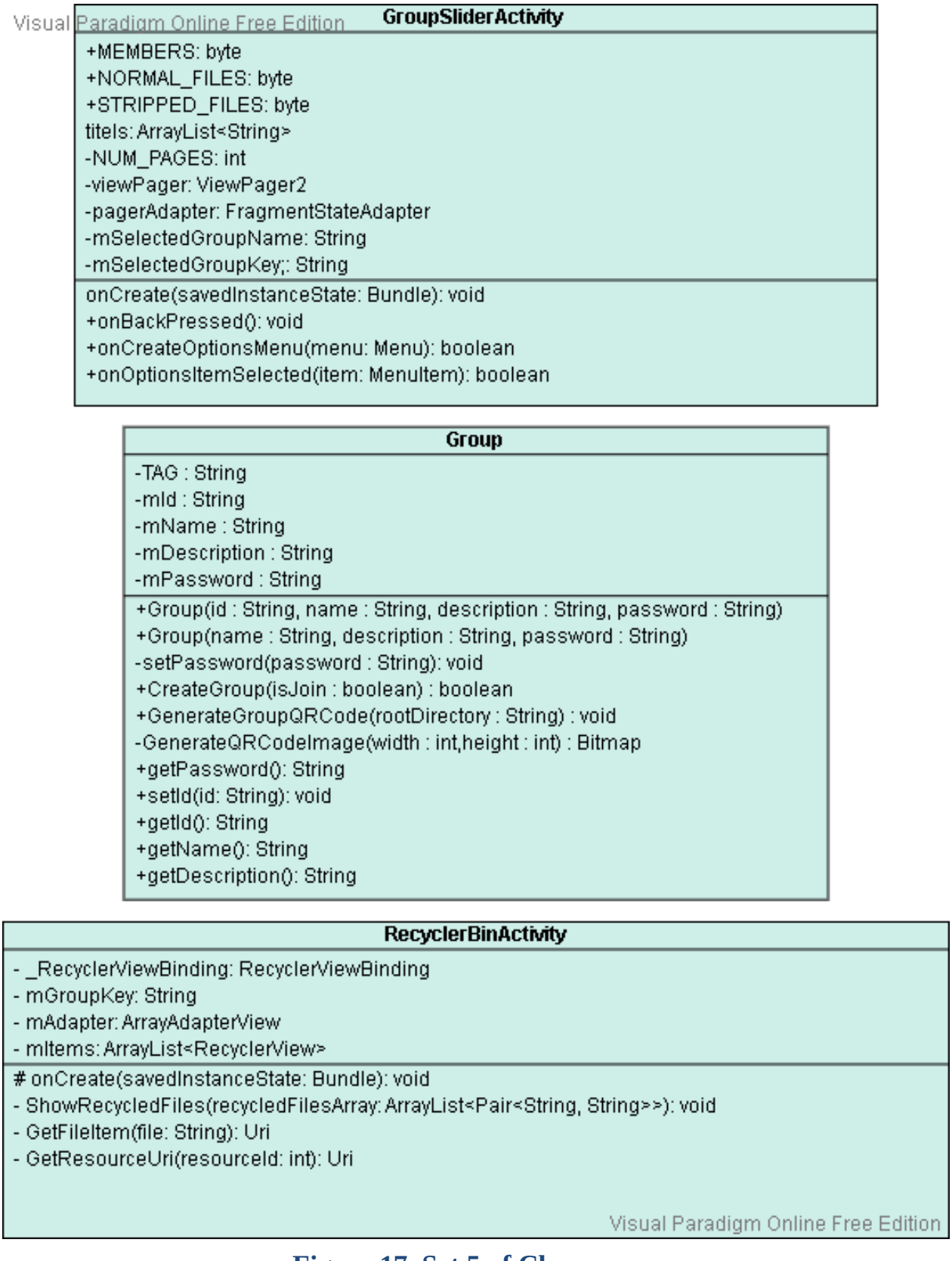


Figure 17. Set 5 of Classes

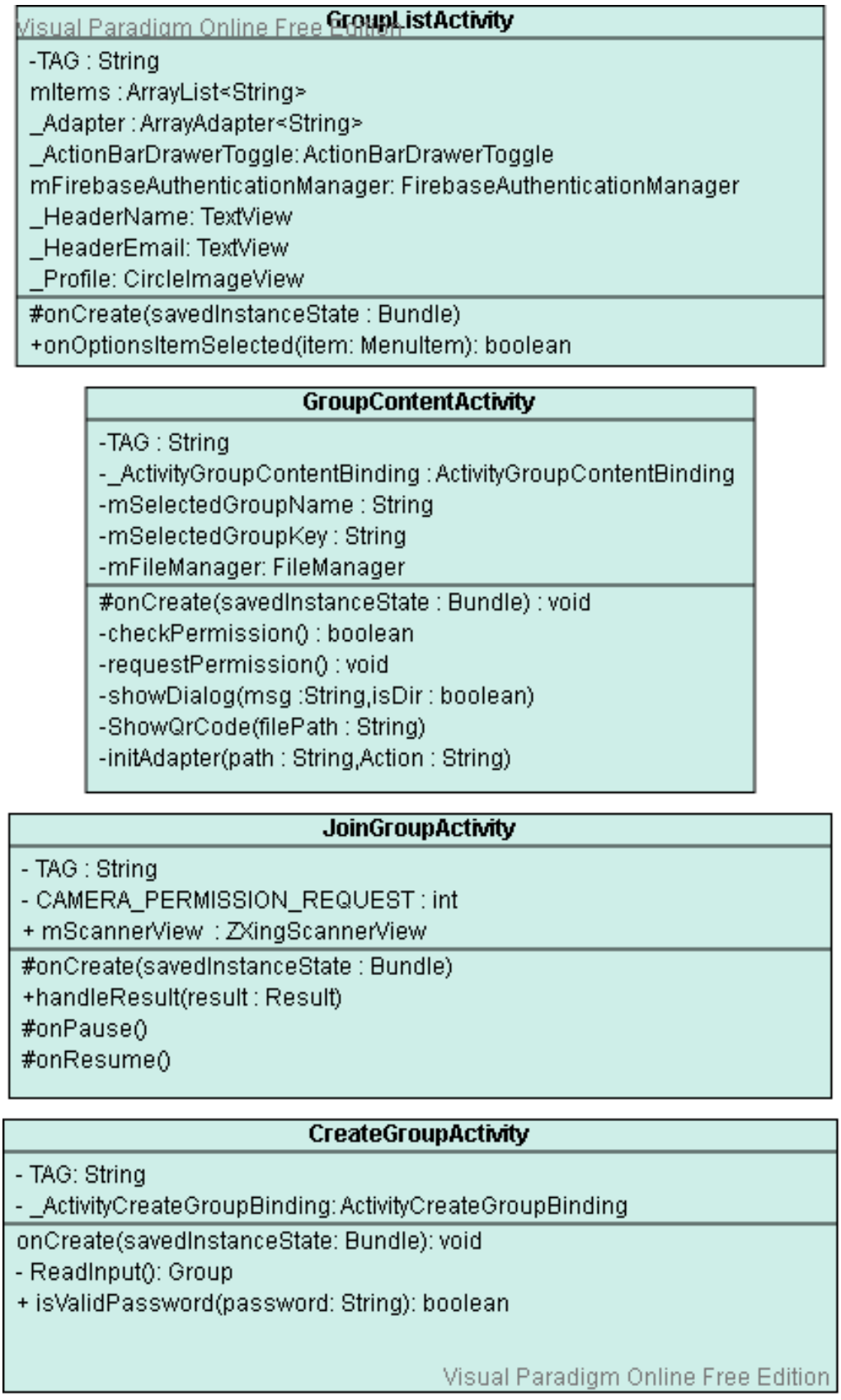


Figure 18. Set 6 of Classes

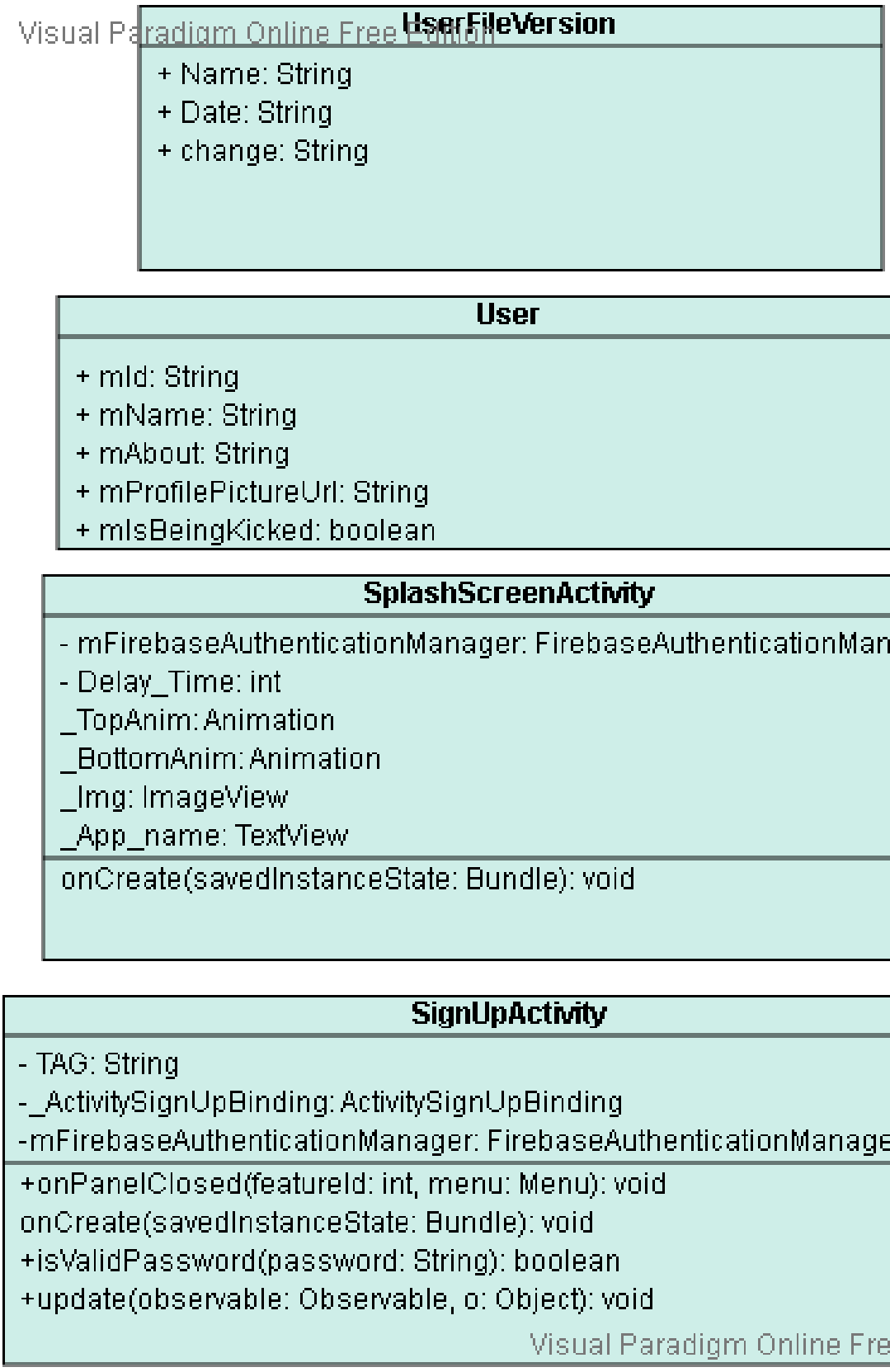


Figure 19. Set 7 of Classes

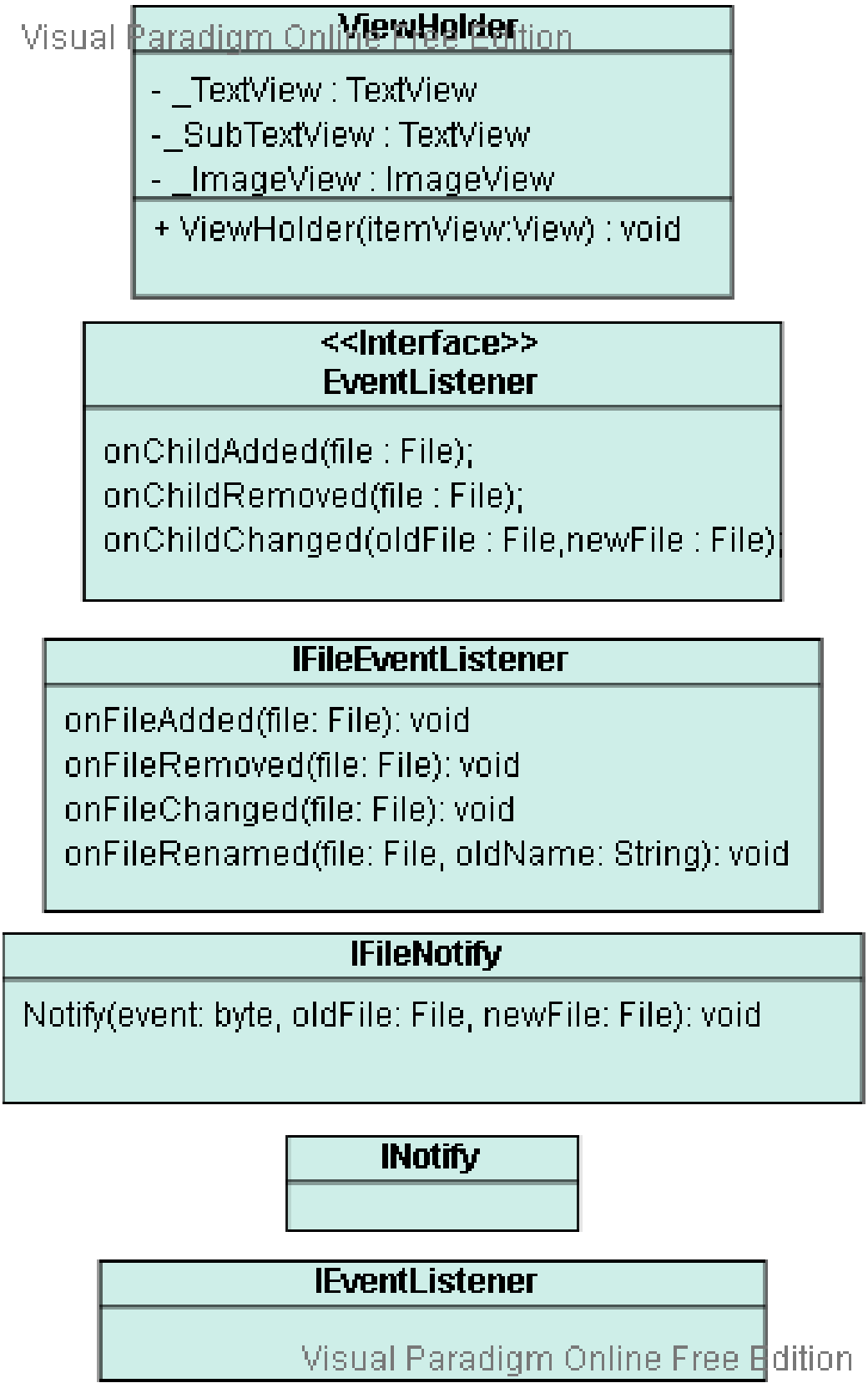


Figure 20. Set 8 of Classes

4. Implementation & User Manual

4.1 Implementation

Our application depends on these features mainly:

- File manager
- File-Sharing
- File striping
- Versioning
- Event Systems

4.1.1. File manager

As a start, we built our File manager/explorer from scratch to facilitate browsing and exploring the files for the user.
The advantage of building it from scratch is that we can easily access the chosen file's path on the user's phone, customize it with any feature we want to add later on and give him the option to share his files with the group in normal or striping mode.

4.1.2. File-Sharing

File-sharing is the practice of sharing your files or giving access to it to someone. Including all types of files, documents, images, multimedia (audio/video), e-books, text files, and many more. File-sharing can be done using several methods. But we chose to use Firebase to aid file-sharing between group members and improved it by synchronizing it among all users in a group. Once you share a file, the other group members detect that and

download it instantly and can open it directly on their device using the suitable application for that shared file.
For security and privacy, before uploading any file, we encrypt it using the Advanced Encryption Standard (AES), also known by its original name Rijndael.

And that takes us to the second main features:

4.1.3. File-Striping

File-Striping is essentially the segmenting of logically sequential data, like Files. And those consecutive segments are stored on different physical storage devices. So how did we use that to our advantage? We used the idea and implemented it with a few tweaks.
We encrypt the file, and then we segment the file into several chunks. Each member downloads a chunk that is specifically assigned to him to download. So no one can open the file independently as he has just a chunk of the file; however, he has no access to it. But once they are all online, each member can download other users' chunks, assemble the file, and open it. But also, once one of them is gone offline, the assembled file is deleted, and the user now has no access to it till they're all online again.

4.1.4. Versioning

Versioning is the "creation" and management of multiple product releases, but all have the same function but are altered or customized.

But here we are talking about file versioning. In its most basic form, file versioning is when you store multiple versions of the same file, upon modification or any change affecting it, in an inventory that you can view or restore later as needed. Also, if a file is deleted, it's not lost, as it can be viewed and restored by the group owner.

4.1.5. Event Systems:

We have two event systems:

- Observer event system
  We have observables that represent the source of data and observers that listen and consume the emitted data items by the observables.
- Callback event system
  Upon function completion, the callback function will be invoked with the resulted data.

4.2. User Manual

Distributed Mobile Cloud application support both dark and light mood:

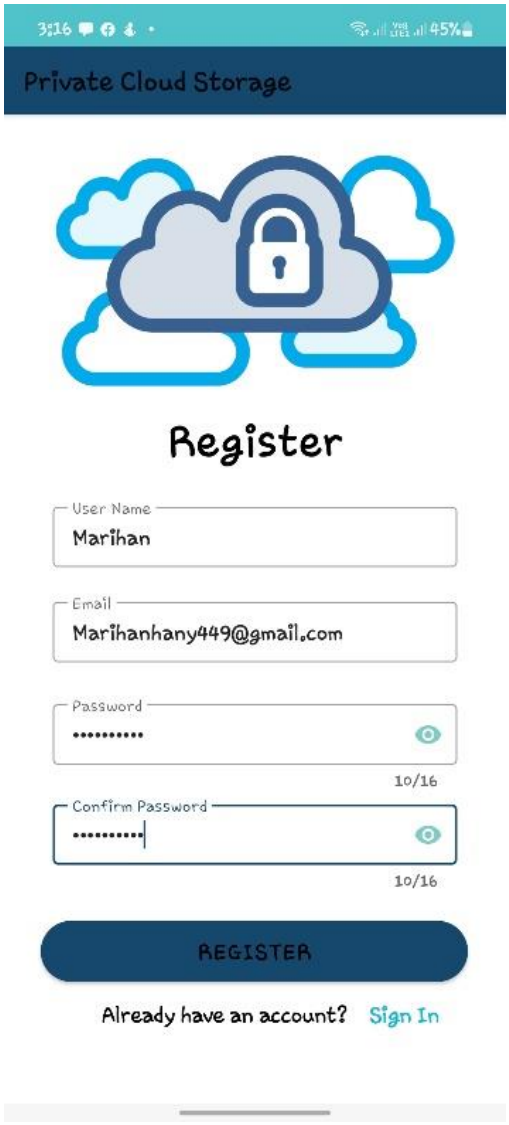


Figure 21. Application Dark Mood

Figure 22. Application Light Mood

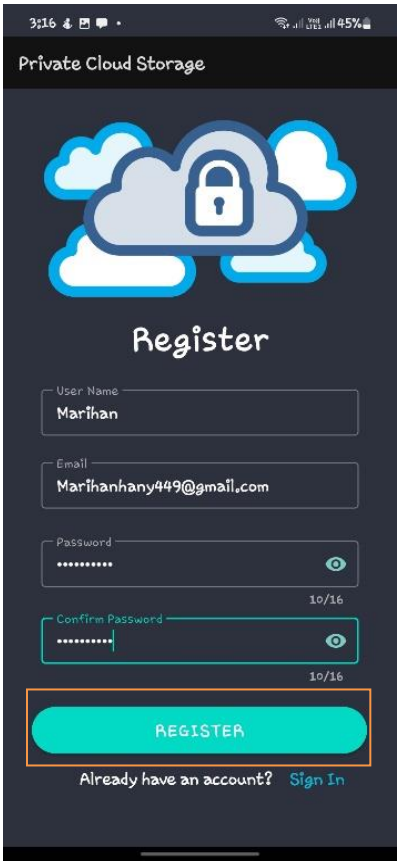


Figure 23. Registration

Registration Steps:

- Enter your name in “User Name” field.
- Enter your e-mail in “E-Mail” field.
- Enter a strong password its length should be > 6 characters containing at least 1 upper-case letter, 1 lower-case letter, 1 number and a character.
- Then re-enter the password to confirm.
- Then press “Register” button.

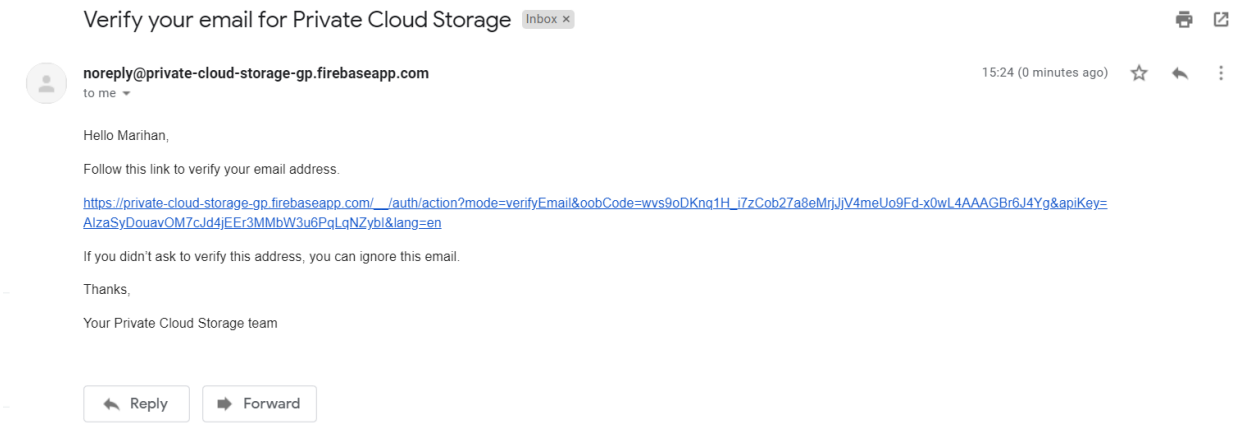


Figure 24. Email Verification

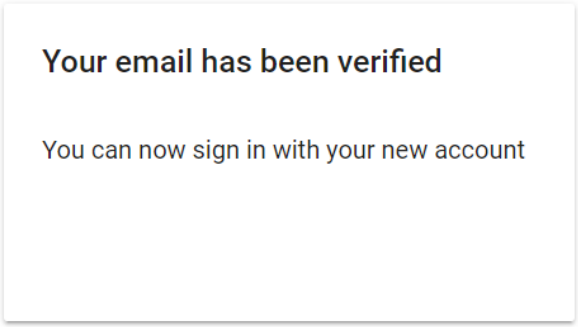


Figure 25. Email Confirmation

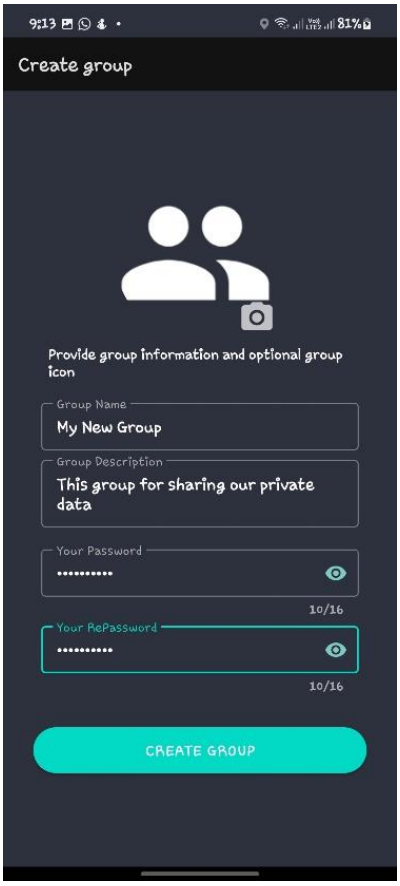
- When click the "Register" button, an e-mail verification is sent.
- Click on the link to verify your e-mail.
- Then, your e-mail is verified and you can go back to the application to sign in.

Figure 26. Sign In

1. Sign in Steps:
   - Enter your e-mail in "E-mail" field.
   - Enter your password in "password field.
   - Click "LOGIN" button to login.

2. You can also click "Forget Password" in case if you forgot your password.

3. If you aren't yet registered via the application, then you should register



Steps to Create Group:

- Enter group name.
- Enter group description.
- Enter group password.
- Re-enter group password to confirm.
- Click “Create Group” button.

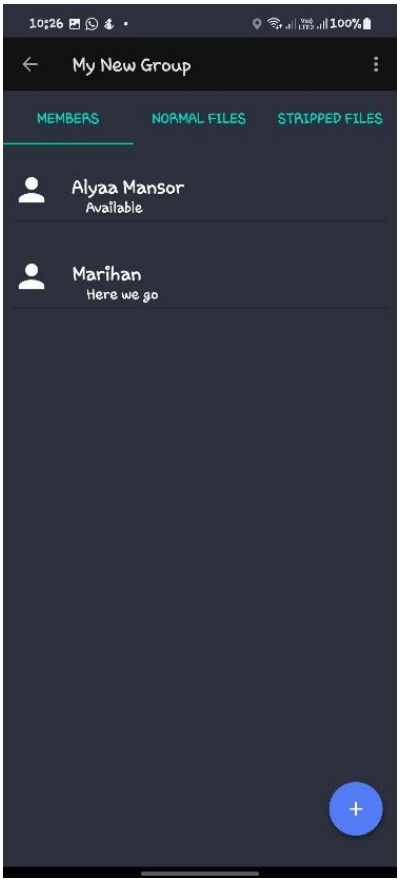
You can click on the eye symbol to show or hide the password.

Figure 27. Create Group

Here is where all the groups of you are listed, containing:

- The group name.
- The group description.
- The group picture.

Figure 28. Members' Groups



When the click in some group you find the group containing the following fields:

- Group Members
- Normal Files
- Stripped Files

Figure 29. Group Content

The group contains action button '+', when you click on it, the following can be happen:

- Share file.
- Show QR code.
- Create.

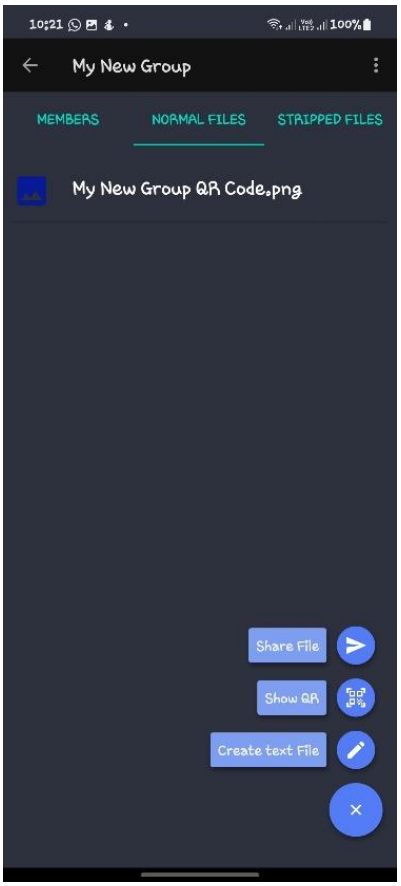


Figure 30. The Action Button

Invite your friends through the QR code of the group.

Figure 31. Invite others by QR code.

Create a text file:

- Choose share text file from action button.
- Name it in the box that will appear.
- Click ok or click cancel to cancel creating the text file.

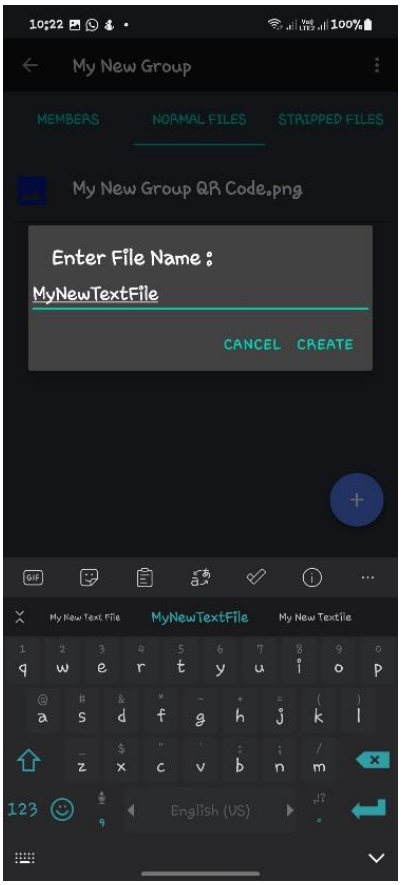
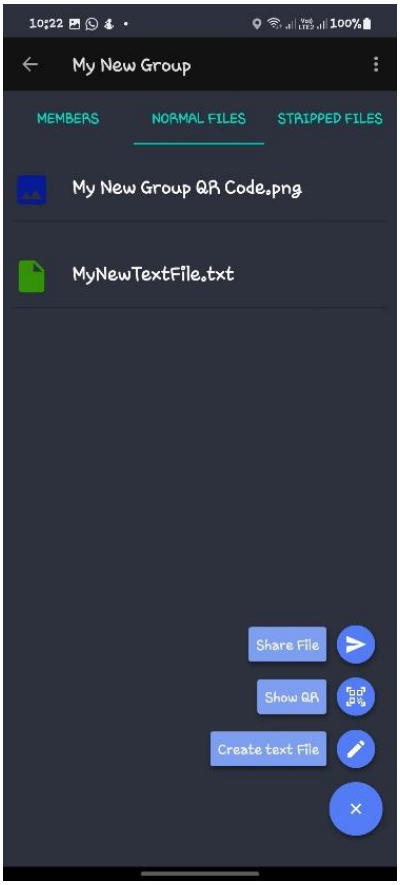


Figure 32. Create Text File



Here's the text file is shared.

Figure 34. After Sharing a Text File

To share a file:

- Select share file from action button.
- Choose the file you want to share from the file manager.
- Click share.

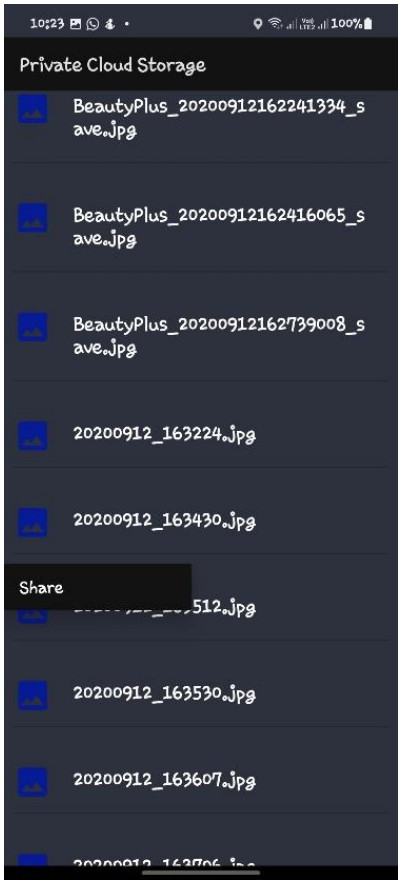


Figure 33. Share File

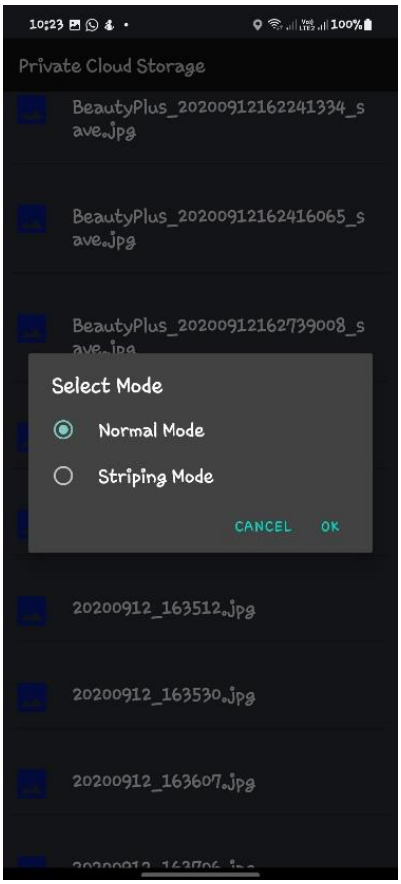


Figure 35. Select File Sharing Mode

- Select the mood of sharing (Normal Mode, Striping Mode).
- Click “OK” or click “Cancel” to cancel sharing the file.

Here’s the group content after sharing a normal file

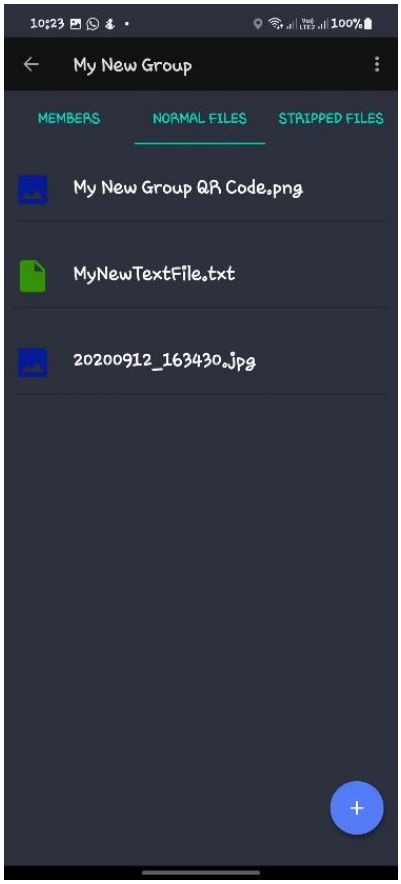


Figure 36. After Sharing a File

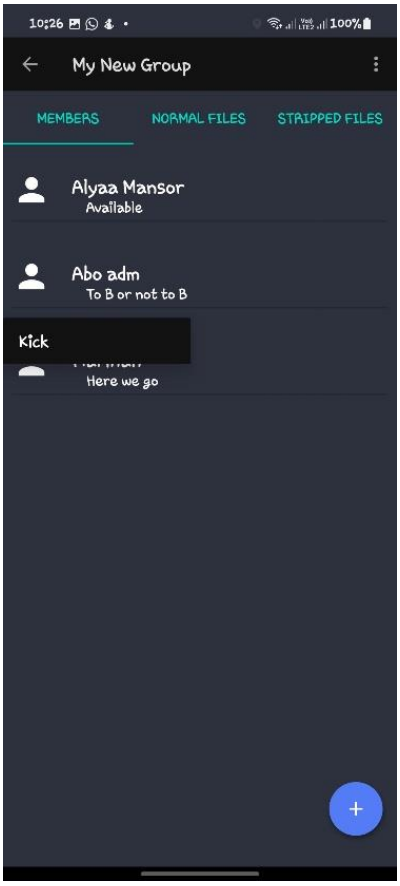


Figure 37. Kick Request

To kick member from a group:

- Long press on the user you want to kick from the group
- Click on “kick” option that appears.
- Kick request is sent to the rest of group members.
- They have two options (kick, don’t kick).

The name of the member requested to be kicked turn to red that the rest of group members can identify him.

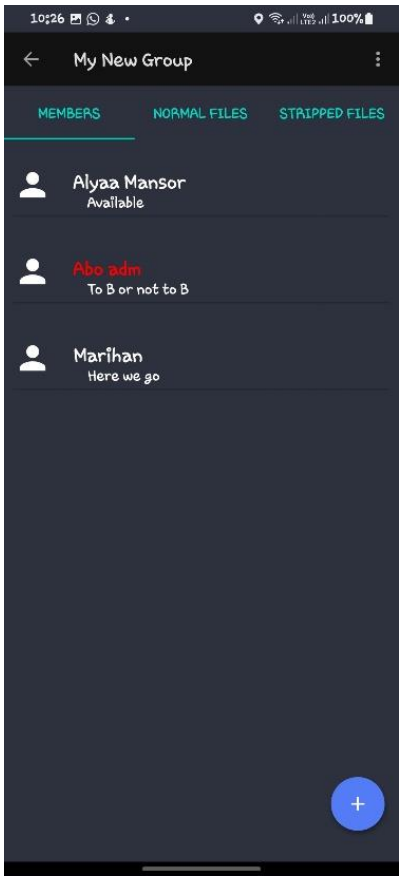


Figure 38. After Kick Request

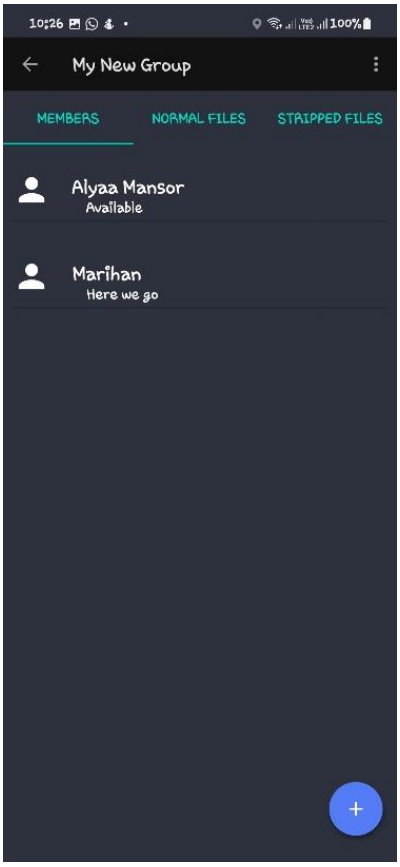


Figure 39. User Kicked

Kick happens when more than half of the group members vote to kick the user, otherwise the user isn't kicked.

You can see and update your profile's information.

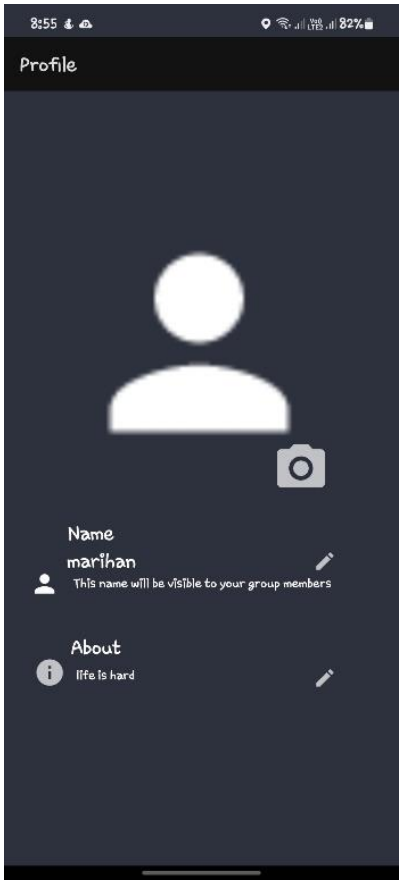
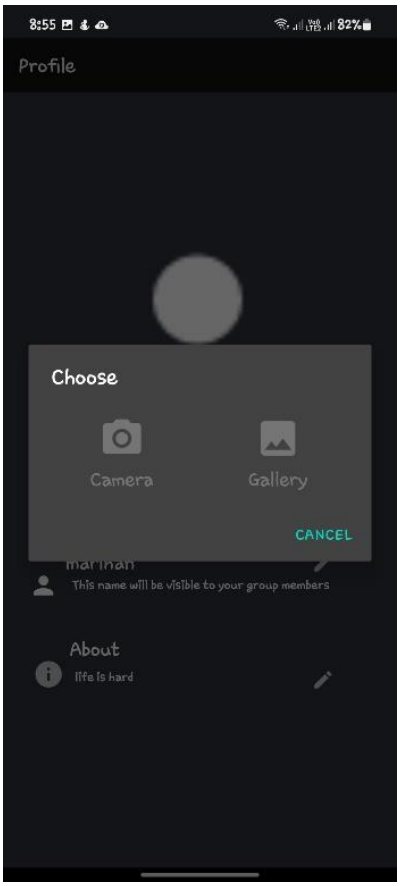


Figure 40. The User Profile



To show the profile picture just click on it.

To update the profile picture:

- Click on the camera symbol.
- Choose option (camera, Gallery).
- If you chose camera capture a picture.
- If you chose gallery, select a picture.

Figure 41. Update Picture

You can edit the name or the about info by click on edit symbol then enter data and press "OK" or press "cancel" to cancel editing.

Figure 42. User Information

5. Conclusion & Future Work

5.2. Conclusion

Keeping the privacy of people is a challenge, helping them share their data with no fear of spying is important.

Building our distributed mobile cloud application proved that the application is reliable for this purpose by keeping the shared data synchronized between users encrypted and stored via firebase platform.

Distributed mobile cloud helps users to recover their data in case of data loss easily. Also joining a group to share files with family and friends happens easily, safely and directly using “QR code scanning” between group members.

Users can have more and more privacy by using “Stripping Mode” while uploading the files, that no group member can see its content in case of one user at least is offline, so that this stripped data can be available and accessed by all the users when they are all online only.

5.3. Future Work

- Replace firebase backend with P2P network.
- The ability to edit files within the application.
- Add notification system.
- Improve the event system.
- Improve the UX/UI.
- Support multiple users from same device.
- Microservices Mobile Cloud Distribution.

6. References

[1] Android Documentation:
https://developer.android.com/

[2] Java Documentation:
https://www.javatpoint.com/

[3] Firebase Documentation:
https://firebase.google.com/

[4] Data Stripping by Academic Dictionaries and Encyclopedias:
https://en-academic.com/dic.nsf/enwiki/264639

[5] Design Patterns in Java by Steven John Metsker and William C. Wake, April 2006

[6]Cloud Storage Security by Aaron Wheeler and Michael Winburn, 2015.

[7] The Design of Rijndael: AES-The Advanced Encryption Standard by Joan Daemen ،Vincent Rijmen, 2013https://www.geeksforgeeks.org/file-handling-in-java/

[8]Google Android Firebase: Learning the Basics by Bill Stonehem9] .2016 ,]

[10] Modern Multithreading by Richard H. Carver, Kuo-Chung Tai,
2005.https://www.tabnine.com/code/java/methods/com.google.common.hash.Hashing/sha256

[11] Design of Hashing Algorithms. Josef Pieprzyk, Babak Sadeghiyan. 1993