



Department of Computer Engineering
Bilkent University

Senior Design Project

Project Short-Name: Nebula

Analysis Report

Group Members: *Arda Atacan Ersoy, Kübra Nur Güzel, Seyfullah Yamanoğlu, Başak Şevval Ekici, Serhat Bezmez*

Supervisor: *Eray Tüzün*

Jury Members: *Cevdet Aykanat, Abdullah Ercüment Çiçek*

Innovation Expert: *Duygu Gözde Tümer*

Table of Contents

1. Introduction	3
2. Current Systems	4
3. Proposed System	14
3.1. Overview	14
3.2. Functional Requirements	15
3.2.1. Server Requirements	15
3.2.2. Client Applications Requirements	16
3.2.2.1 - Mobile applications	16
3.2.2.2 - Web application	16
3.2.2.3 - Desktop application	16
3.3. Nonfunctional Requirements	16
3.3.1. Security	16
3.3.2. Usability	17
3.3.3. Cost	18
3.3.4. Performance	18
3.3.5. Extendibility	18
3.3.6. Marketability	18
3.3.7. Scalability	19
3.4. Pseudo Requirements	19
3.5. System Models	19
3.5.1 Scenarios	19
3.5.2 Use Case Model	25
3.5.3 Object and Class Models	26
3.5.4 Dynamic Models	31
3.5.4.1 Sequence Diagrams	31
3.5.4.2 Activity Diagrams	34
3.5.5 User Interface	36
4. References	39

1. Introduction

Cloud storage has come to rely almost exclusively on large storage providers as trusted third parties to transfer and store data. This system suffers from the inherent weaknesses of a trust-based model. Because client-side encryption is non-standard, the traditional cloud is vulnerable to a variety of security threats, including man-in-the-middle attacks, malware, and application flaws that expose private consumer and corporate data. The main infrastructures of Nebula is not very different regarding to the current systems that are being used to provide cloud storage to users, in terms of the storage, hardware and network components. However, with the unique architecture of the Nebula, it is aimed to provide a different system with enhanced security and privacy.

Nebula is a peer to peer decentralized blockchain based cloud storage system. What does these technical terms mean and why it is an important issue at the moment? Cloud storage becomes more popular everyday but also more people want to keep their data secure and private. With the not so pleasant past events from other centralized cloud storage products, such as data losses or hacking to private photographs of celebrities, the idea of Nebula arose. Nebula offers secure and distributed storage of personal data with the use of remote servers for requests and configurations as well as a remote data center infrastructure. Each data will be distributed in an encrypted fashion, with the use of blockchain to make the best possible security services up and running.

Nebula offers many advantages compared to data center-based cloud storage. Data security can be maintained using client-side encryption, while data integrity will be maintained via a proof of storage and retrievability that Hyperledger provides[1]. The impact of infrastructure failures and security breaches will be greatly reduced. An open market for data storage may drive down costs for various storage services by enabling more parties to compete using existing devices since with Nebula users will provide and use the storage themselves. Data on the Nebula network will be resistant to tampering, unauthorized access, and data failures.

Decentralized cloud storage network offers many advantages compared to datacenter-based cloud storage in terms of security, privacy, redundancy in data and cost reduction due to efficiency. Nebula will provide the data security with client-side encryption which is not a standard in the systems that are widely used now since they are datacenter-based models. Also the file loss can be avoided since extra copies will be transmitted in case of errors. With the help of blockchain storage, cloud computing cost can be reduced.

2. Current Systems

There are many options for cloud storage systems on the market. In this reports, the cloud systems will be analyzed in two sections: centralized cloud systems and decentralized cloud systems. Nebula aims to be a decentralized cloud system but it is a fairly new topic so there are not very popular systems to analyze at the moment. With the centralized cloud systems, very popular systems such as Google Drive, Apple iCloud and Dropbox will be discussed.

2.1 Comparing Two Systems

Before intensely analyzing the current systems, a brief overview of centralized and decentralized concepts is below in the given in TABLE 1 in order to understand effectiveness of inherited features of decentralization. Regardless of the specific services offered by companies, we have theoretically compared these two concepts from some point of views. TABLE 1 is a critical decision point for us to consider pros and cons of the decentralized systems.

	Centralized Cloud Systems	Decentralized Cloud Systems
Points of Failure	Single Point of Failure: If the master server machine goes down, clients are no longer able to process user requests since the machine that runs the core software to process requests is dead. Although the central systems are developed by dominating companies, theoretically single point of failure cannot be overlooked.	No single point: Client machines aren't relying on a single server (multiple nodes).
Maintenance	Easy to maintain as there is only a single point of failure.	Difficult to maintain since there are many participating nodes in the system.
Liquidity	Better market liquidity since it is a seated system in the current market.	Inadequate liquidity to compete with centralized due to it is a new technology.
Scalability	Scale out the system by vertical scaling — by adding more storage, I/O bandwidth, processing power (number of CPUs, number of cores) to the server machine. (Costly solution comparing to decentralized)[17]	Scale-out the system by adding more nodes.[17]
Fault Tolerance / Stability	Theoretically unstable. Single failure can occur on master server machine.	Very stable and has an immunity to a single failure.

Speed	Fast transactions	Slower. Considering the nodes are physically separated in space and encryption, separation and distribution of the data take some finite amount of time. Faster when the node number maximizes and geographical location of nodes are close.[2]
Evolution to new technologies	Based on single framework, which brokes diversity and evolve slowly. [16]	Once the basic infrastructure is in place, evolution is much easier in the structure.[16]
Trust Factor	Trusting a 3rd party with potentially sensitive data comes with all the risks of human vulnerabilities. Mistakes can be made and data can be lost, stolen, or even sold.[17]	The system is trustless (refer to Glossary): Trades are executed peer-to-peer and the exchange relies fully on its users. Encrypted and decentralized storage methods prevent other nodes from accessing data. Even the renter of hard drive space cannot access the information stored inside it.[17]

Ease of development	Fast development. Pick up a framework and apply it [17]	More complex design:Low level details like resource sharing (trade) and communications (transport). [17]
Ease of use	Simple UX and steps.	Complex user experience and confusing steps / transactions.
Service Fee	Not cost-effective for customers. For example, Amazon S3 has a fee \$23 per month 1TB storage.[2]	Less expensive for instant Siacoin charges around US \$2 for 1 TB per month. [2]

TABLE 1: Comparison Table of Decentralized and Centralized Cloud Systems

How many TB?

1

Storage Provider	Monthly Storage Cost	Download Bandwidth Cost	Private	Decentralized	Included Multi Region Redundancy
Sia	\$2	\$1	✓	✓	✓
Amazon S3	\$23	\$92	✗	✗	✗
Google Cloud	\$20	\$110	✗	✗	✗
Microsoft Azure	\$24	\$87	✗	✗	✗

Table 2: Comparison Table of Cloud Service Providers [2]

As it is seen from Table 2, 1 TB storage providing cost is shown in the picture. Decentralized cloud Sia has a much less cost comparing to the other major centralized cloud services. [2]

2.2 Centralized Cloud Systems

Centralized Cloud Systems are the most traditional cloud systems currently in the market. It has a single server architecture for multiple clients. The server has more computing resources than its clients, allowing all of the major processing done by one server computer. [3]

2.2.1 Apple iCloud



- iCloud is a mobile device friendly application.
- iCloud gives a 5GB free storage but for bigger storage usage, the service gets more expensive.
- Compatible with Mac and Windows PC
- It is used for file and document sharing and iCloud can also stores music, photos, videos, and documents.

2.2.2 Google Drive



Google Drive is one of the most used cloud storage services available at the moment. The advantage with Google Drive is that it works synchronized with other Google Services such as Google Documents and Gmail. User can modify documents on their Drive and also work collaboratively with other Google accounts. Google Drive can be used from the web or with the mobile applications. It does not have a desktop application.

- Google Drive is a mobile device friendly application.
- Drive gives a 15GB free storage but for bigger storage user needs to subscribe to a payment plan.
- Does have a desktop application.
- Allows storage for music, photos, videos, and documents.
- Google Drive also allows collaborative working on certain file types
- User can share their files directly with entering an email address

2.2.3 Dropbox



- Dropbox gives a 2GB free storage but paid subscriptions available that offer more capacity.
- Dropbox offers computer applications for Apple macOS, Windows, Linux computers.
- Dropbox has a desktop application.
- Dropbox mobile application is compatible with iOS, Android, and Windows Phone smartphones and tablets.[4]
- The Dropbox software enables users to drop any file into a designated folder.

2.3 Decentralized Cloud Systems

Decentralized file storage topic is popular even it is a new system to compete with centralized file storage companies. And lots of companies are investing their fundings to the Filecoin ICO and companies that are explained below which try to develop decentralized file storage system. [5] Current up-to-date decentralized cloud storage systems are all using blockchain at the moment. [14]

2.2.1 Inter-planetary File System (IPFS) with FileCoin



- It is a protocol to create a new way to server information on the web.
- It allows user to choose where to get content from and user can set privacy of peers user trust to receive his/her files.
- The files are stored in the nodes based on their content. Each node stores only content it is interested in.
- The content can be accessed offline.
- IPFS is an open source project.
- They use Proof of Spacetime and Proof of Replication techniques.
- They did their Initial Coin Offering (ICO) in September 2017.
- It is still in development stage.

2.2.2 SiaCoin



- It is a protocol to create a new way to server information on the web.
- They used Proof of Storage technique on their algorithm.
- SiaCoin has a working product.
- It provides an open source platform.
- Siacoin has a 10 min block time (same as bitcoin) producing 144 blocks a day. [4]
- The deals are secured by file contracts, Siacoin's form of smart contracts. [6]

2.2.3 Storj



- Their technology revolves around file sharing and separates parts of the files to users in the network. When the user requests for the file, Storj locates all the shards and piece them together. [3]
- Storj is a distributed cloud storage which means not all the processing of the transactions is done in the same place.[7]
- The files are encrypted before the separating process and the user who owns the file has his private key for validation.
- Storj claims that they have a working product, but it is not released yet and does not allow new user account registration.
- It provides an open source platform.

3. Proposed System

3.1. Overview

The Nebula platform is a decentralized data processing architecture designed for secure, scalable management of online data storage, file sharing, and user access for individual consumers. This technology seeks to displace single source Cloud Storage Providers (CSPs) [8] and Storage Partitions. The Nebula ecosystem aims to manage security for data-at-rest (storage), and data-in-use (sharing). The ecosystem is structured to provide scalable benefits and incentives for all participants to grow the security, integrity, financial competitiveness and performance of Nebula technology.

Nebula uses Hyperledger technology, which is a blockchain technology hosted by Linux Foundation[10], in order to decentralize Nebula's network and databases. There will be two immutable ledger distributed across the network: One is used for auditing user accounts. The other is used for the auditing files and peers relationship. Inherited blockchain features enhanced Nebula to being immune to attacks or single point failures since the peer nodes will continue to function of keeping blockchain ledger. Moreover, uploading the data on a centralized cloud, the data is distributed across the network. With the immutable ledger, cloud is shared across the all nodes and it is highly encrypted in such manner that is impossible the interrupt. In other words, only owner can access to the file. Thus blockchain technology is useful to decentralize and secure the data. Blockchain consensus mechanism and smart contracts make this interruption nearly impossible. Instead of based on blockchain technology, Nebula combines blockchain ledger with its P2P network, which contributes enhanced security, performance file transfer and decreased cost to the end user. This combination of distributed storage and blockchain provides a verification of the network without any third party.

The basic architecture of the Nebula system is illustrated in Figure 1. There are basically 3 main components of the system which are described as Backend Engine, Data Centers and Distributed File Storage network. These components are

interconnected each other through backend engine. Backend Engine serves as an API to the web, mobile and desktop clients. User logs and audits are kept in two different blockchain ledgers by using the 3rd party HyperLedger component such that each peer will be has the immutable blockchain ledger. Backend engine simply validates user registration and also their unique keys for their files. Moreover, data gathered from the peers are controlled by backend engine.

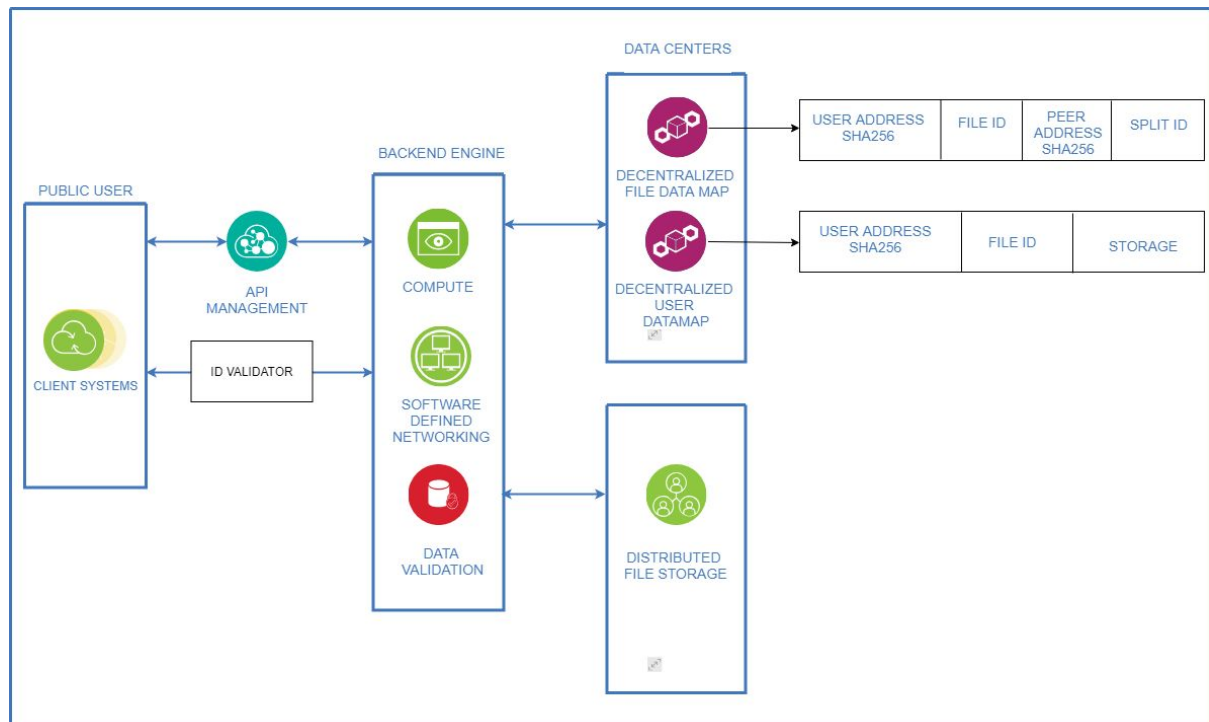


Figure 2: Nebula Drafted Architecture

The difference between blockchain based decentralized cloud:

Following table (TABLE 2) compares the key aspects of a decentralized peer to peer cloud network and a blockchain based decentralized cloud network. Overall, especially from the security perspective, blockchain usage in the network would increase the reliability of Nebula system.

	<i>Decentralized P2P Cloud Network</i>	<i>Blockchain based Decentralized Cloud Network</i>
<i>Need of a central server</i>	No	No
<i>Speed and efficiency</i>	<i>Nodes which hold user's file are known by the system so retrieving the file is fast</i>	<i>Nodes which hold user's file should be mined from the chain first to retrieve the file and this process spends some extra time</i>
<i>Security</i>	<i>This system is all about copying the information and this weakens the security.</i>	<i>Blockchain prevents copying of information</i>
<i>Reliability</i>	<i>Information could be lost in case of seeders quitting.</i>	<i>Because of distributed ledger information is never lost</i>

TABLE 2: Comparison Table of Blockchain Based and Regular Decentralized Cloud Systems

3.2. Functional Requirements

3.2.1. Server Requirements

- There should be unique private and public keys for each peer.
- Users audit information will be kept on server.
- Server has to know each active node and their properties such as bandwidth, storage amount, etc.

- Application should enable segmented and distributed files allowance for multi-threaded concurrent downloads.
- There will be an option for the users who want to use the cloud but don't want to become a peer (due to low storage support on their devices etc.). These users will pay such amount money for this service.

3.2.2. Client Applications Requirements

3.2.2.1 - Mobile applications

- Users can login with their username and passwords.
- Users will be able to upload and download files to/from their accounts.
- Users will be able to see their wallet information.
- Users will be returned a revenue for the allocated storage on their devices.
(For detailed information, see the glossary)
- Users will be able to share their storage from mobile device.

3.2.2.2 - Web application

- Users will be able to upload and download files to/from their accounts.
- Web application will have a responsive design to different resolutions of screens.

3.2.2.3 - Desktop application

- The system will store the user's data by splitting apart, encrypting and distributing it across the decentralized network.
- Users will be returned a revenue for the allocated storage on their devices.
- Users will be able to see their wallet information and let system make mining process.
- Users will be able share their storage from their PC.

3.3. Nonfunctional Requirements

3.3.1. Security

- Application should keep the data secure and private via a series of client-side processes before it enters the network. To do that, first the metadata required to remotely verify file integrity should be generated, then the data is split into shards, and each shard should be encrypted.
- End-to-End encryption and client-side encryption will be provided for each user from the network.
- The encrypted data should be then sent out into the network, and the integrity of the data should be checked at regular intervals. So unencrypted data should be never exposed, files in this system should be as secure as the user's key management.
- Program should manage this file storage in terms of blockchain technology and files should be separated into different users. So, no one device has an entire file and all the files are copied on numerous devices, so if a hard drives fails your files won't be lost.
- By the aim of decentralized storage, application should not be vulnerable to hacker attack since it is impossible for anyone without the private keys to decrypt the encoded files.
- With the help of blockchain inherited features (immutable ledger, consensus mechanisms), data confidentiality and security will be provided.

3.3.2. Usability

- Application needs to be user friendly and user interface should be understandable for the simplicity of the user commands.
- Applications , peculiarly Android version, must be efficient in terms of memory and CPU usage since mobile phones have limited resources on memory, battery and CPU. The optimal term "efficient" will be decided after some test on different mobile phones.

3.3.3. Cost

- Application should provide lots of hosts for decentralized file storage. Hosts should determine their own prices for renting out their hard drives (similar to the Airbnb model). This ensures that the prices will stay competitive as time goes on.
- Application should provide much less expensive services compared to major players. Application should offer from $\frac{1}{3}$ to $\frac{1}{5}$ cost of global centralized storage companies. (Per terabyte storage cost, Amazon: 23\$, Google Cloud: 20\$) [6]

3.3.4. Performance

- The speed of user's file storage will depend on the performance of local storage who services to our user. In order to compete with centralized system, we need more people to serve as storage disk. As we increase our competition between people who provide disk storage. We can increase our performance.
- Server should response less than 500 ms while doing the peer distribution.

3.3.5. Extendibility

- Application should be suitable for changes because when users demand new features from us, we should have such infrastructure that can provide it.
- Application should provide the payment method for storage and bandwidth as user use it. User should not be restricted to buy very huge storage files unless user needs it.

3.3.6. Marketability

- In terms of other non-functional requirements, decentralized file storage system has some advantages to centralized ones(See Table 1). This is very important to use this application for the users who concerns these non-functional requirements.

- Big database companies are willing to work with decentralized file storage services such as Couchbase, MongoDB, InfluxDb. [8] Thus, there is a demand in the market for such product.

3.3.7. Scalability

- Since user and file data tied on blockchain platform, scalability of the application is tied on the underlying blockchain platform which is currently determined as Hyperledger. With Hyperledger each node has their own responsibilities and that provides an independent scale for each node. [9] Nebula system should also perform in a similar way.
- Scalability of the application should be adjusted considering the cases when limited scalability harms the availability and reliability of the system such as potential overload of the network.

3.4. Pseudo Requirements

The systems will have a server, a web application, a desktop application for Windows and a mobile application for Android. Nebula will also use a third party effort called Hyperledger (For the details see Glossary section).

3.5. System Models

In this part, the report will be giving detailed information about possible user scenarios and how those scenarios will proceed. The most common possible scenarios.

3.5.1 Scenarios

Scenario 1

Use Case Name: Login as Bob

Actors: Bob

Entry Conditions:

- User Bob is on the login screen

Exit Conditions:

- User Bob is on the home page

Main Flow of Events:

1. User Bob enters his email as "bob@mail.com."
2. User Bob enters his password.
3. User Bob taps the "Login" button.
4. Nebula checks if the username and the password is correct
5. Nebula confirms that the credentials are correct.
6. Nebula navigates to the Home page of the user Bob.

Alternative Flow of Events:

- A. User enters wrong email and/or wrong password.
 - a. Nebula detects that the credentials are incorrect.
 - b. Nebula navigates to Login screen.
 - c. Nebula informs user about the wrong entries.
 - d. User enters the credentials correctly.
 - e. Nebula navigates the user to Home page.

Scenario 2

Use Case Name: Sign-up as Bob

Actors: Bob

Entry Conditions:

- User Bob is on the sign-up screen

Exit Conditions:

- User Bob is on the home page

Main Flow of Events:

1. User Bob enters his email as "bob@mail.com."
2. User Bob enters his password.
3. User Bob enters his password again for confirmation.
4. Nebula checks if the email address and the password is applicable.
5. Nebula confirms that the credentials are correct and creates the account.

6. Nebula navigates to the Home page of the user Bob.

Alternative Flow of Events:

A. User entries are inaccurate or missing.

a. Nebula returns to Login screen.

b. Nebula informs the user about missing and/or inaccurate entries.

c. User enter the missing and/or inaccurate informations again.

d. Nebula creates a new account for the user.

e. Nebula navigates to the Home page of the user.

Scenario 3

Use Case Name: Change Settings for Bob

Actors: Bob

Entry Conditions:

- User Bob is on the profile page

Exit Conditions:

- User Bob is on the profile page

Main Flow of Events:

1. User enters the settings section in the profile page.
2. Bob views his settings for his account.
3. Bob chooses the “Change Username” setting
 - a. Bob changes his username
4. Bob chooses the “Change Password” setting
 - a. Bob changes his password
5. Bob chooses the “Peer Options” setting
 - a. Bob selects “Become a Peer” and starts to contribute storage to cloud
6. Bob chooses the “Privacy” setting
 - a. Bob selects “Don’t share my information with third party companies” option
7. Bob chooses “Save Changes” option
 - a. Nebula checks if all required information is given
 - b. Nebula saves the new settings
 - c. Nebula navigates Bob back to the profile page.

Alternative Flow of Events:

- A. Bob does not change his username. (Skip Step 3)
- B. Bob does not change his password. (Skip Step 4)
- C. Bob does not change his peer preference. (Skip Step 5)
- D. Bob does not change his privacy preferences. (Skip Step 6)

Scenario 4

Use Case Name: Upload a File

Actors: Bob

Entry Conditions:

- User Bob is on the home page

Exit Conditions:

- User Bob is on the files page

Main Flow of Events:

1. User clicks the “+” icon on the home page
2. User drags and drops a file into the provided area
3. Nebula uploads the file and shows the progress with a bar
4. Once the upload completes, the user gets a notification.
5. Nebula redirect him back to homepage
5. Bob sees the new uploaded file on his homepage.

Alternative Flow of Events:

- A. Bob tries to upload a very large file
 - a. Nebula warns the user about the file size.
- B. Bob interrupts the upload
 - a. Nebula stops the uploading process

Scenario 5

Use Case Name: Download a File from the Web Application

Actors: Bob

Entry Conditions:

- User Bob is on the home page

Exit Conditions:

- User Bob is on the files page

Main Flow of Events:

1. User clicks the “Download” button under the file he wants to download
2. Nebula gives opens a pop-up and asks user to choose a destination to download the file
3. Bob selects “Desktop”
4. Nebula downloads the file to Bob’s desktop while showing him the progress.
5. Nebula gives a notification once the download completes
6. Bob can view the file on his Desktop

Alternative Flow of Events:

- A. Bob does not select a proper destination file to download
 - a. Nebula asks user to select a destination
- B. Bob interrupts the download
 - b. Nebula stops the downloading process

Scenario 6

Use Case Name: View a File

Actors: Bob

Entry Conditions:

- User Bob is on the home page

Exit Conditions:

- User Bob is on the files page

Main Flow of Events:

1. User selects one of his files
2. Nebula checks if the file can be previewed
 - a. The file is a .txt file so Nebula gives a preview
3. Bob reads his file
4. Bob closes the file
5. Nebula redirects him back to homepage.

Alternative Flow of Events:

- A. User selects a file that cannot be previewed
 - a. Nebula gives a brief information about to file

Scenario 7

Use Case Name: Becoming a Peer

Actors: Bob

Entry Conditions:

- User Bob is on the profile page

Exit Conditions:

- User Bob is on the home page

Main Flow of Events:

1. User enters the settings section in the profile page.
2. Bob views his settings for his account.
3. Bob selects the “Become a Peer” option.
4. Bob selects where from his device to contribute some storage to Nebula.
5. Bob enters the amount for the storage space he wants to provide.
6. Nebula allocates the space, communicating with the OS.

Alternative Flow of Events:

- A. Bob does not have a proper destination/enough storage to contribute
 - a. Nebula gives informs the user about the requirements to become a peer
- B. Bob enters an invalid storage space amount
 - a. Nebula gives a warning.
 - b. Bob enters valid inputs.
 - c. Nebula confirms and proceeds.

Scenario 8

Use Case Name: Getting a Revenue from storage usage

Actors: Bob

Entry Conditions:

- User Bob is on the settings page

Exit Conditions:

- User Bob is on the home page

Main Flow of Events:

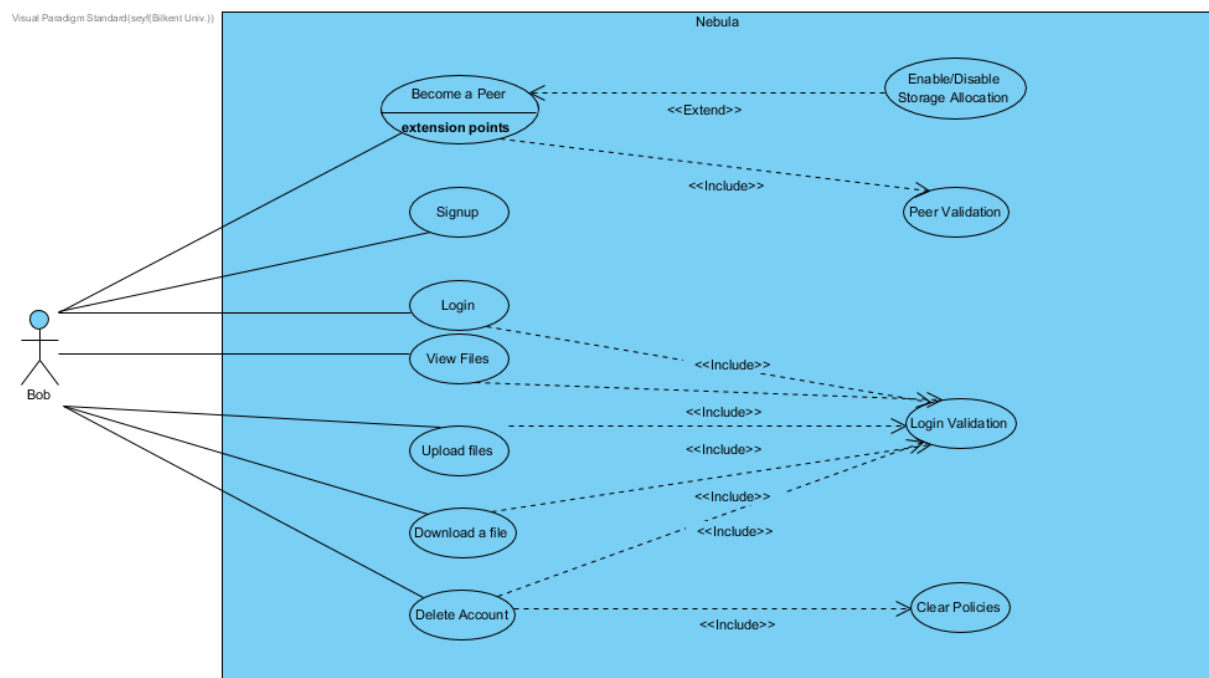
1. Bob is on the Become a Peer option
2. Bob enters the amount to contribute to the Nebula network.

3. Nebula calculates the revenue that he can get with respect to the amount Bob contributes.
4. Bob can now use that free rewarded space.

Alternative Flow of Events:

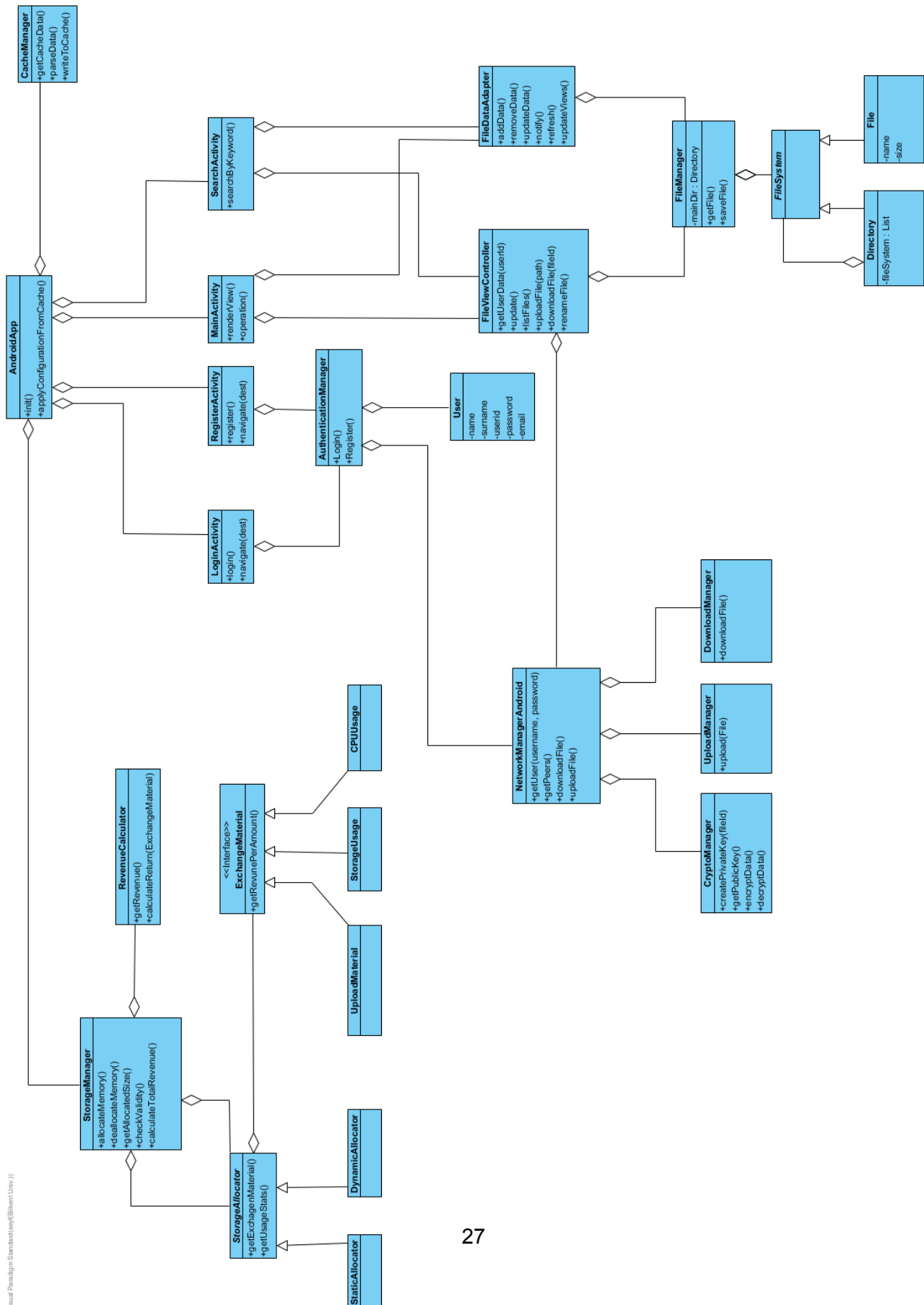
- A. Bob changes the amount he contributes to the Nebula network.
 - a. Nebula recalculates the revenue.

3.5.2 Use Case Model



3.5.3 Object and Class Models

Android Application Class Diagram:



Description:

The design of this application is close to the MVC architecture with some exceptions. Clarification of some of the classes are as follows:

Model:

FileManager: This class is responsible for local type file operations. Reading/writing from/to disk, etc.

StorageManager: This class is responsible for storage allocation for the network. Since our application enables users to allocate their space for application/network to use, this class handles those required operations including revenue calculations.

NetworkManagerAndroid: This class will do requests to Network API because p2p connection will be established when the application runs. We will handle connection type operations here. Since we offer end-to-end encryption, it also has CryptoManager which encrypts and decrypts the data when uploading or after download.

CacheManager: This class will manage the cache operations at the execution of the application, it will write/read to/from disk the user settings.

View:

ActivityClasses: Since native Android SDKs uses Activity classes as View components, they will be our main renderable components. They will also handle user interactions.

Note:

- Resource management is handled by Android itself, so any class for resource management is ignored

Controller:

AuthenticationManager: This class will manage authentication process with the use of web API.

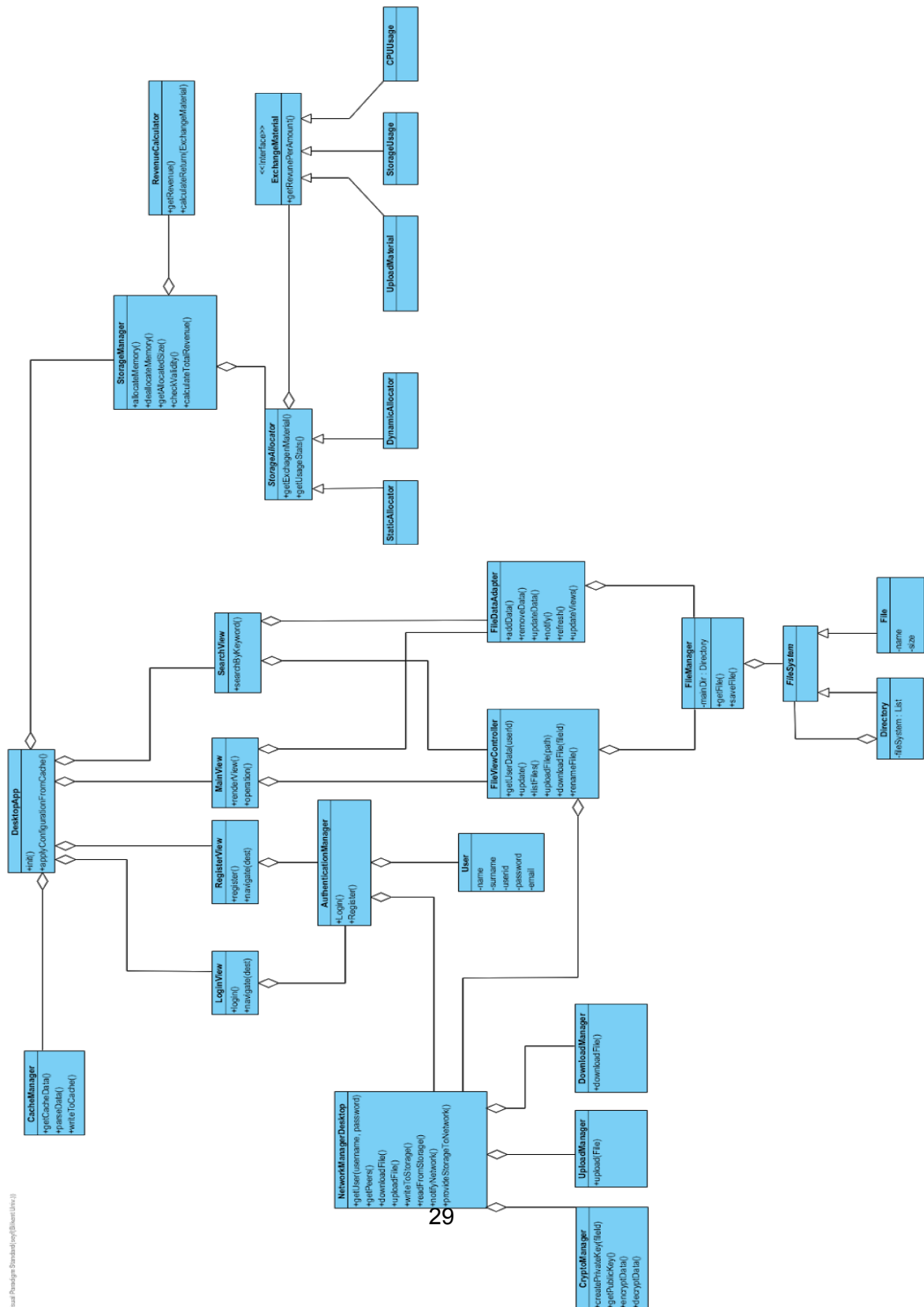
FileViewController: This class will communicate with File manager and will provide main and search view with list of files and abilities to do operation on files.

FileDataAdapter: This adapter will provide user interfaces with the list data, in Android those adapters are must for views that includes listing.

Note:

- Adapters are intentionally ignored in the design, because Android implicitly forces those classes to be existent.
- Listeners are intentionally ignored, instead, simply represented as `handleInput` methods.

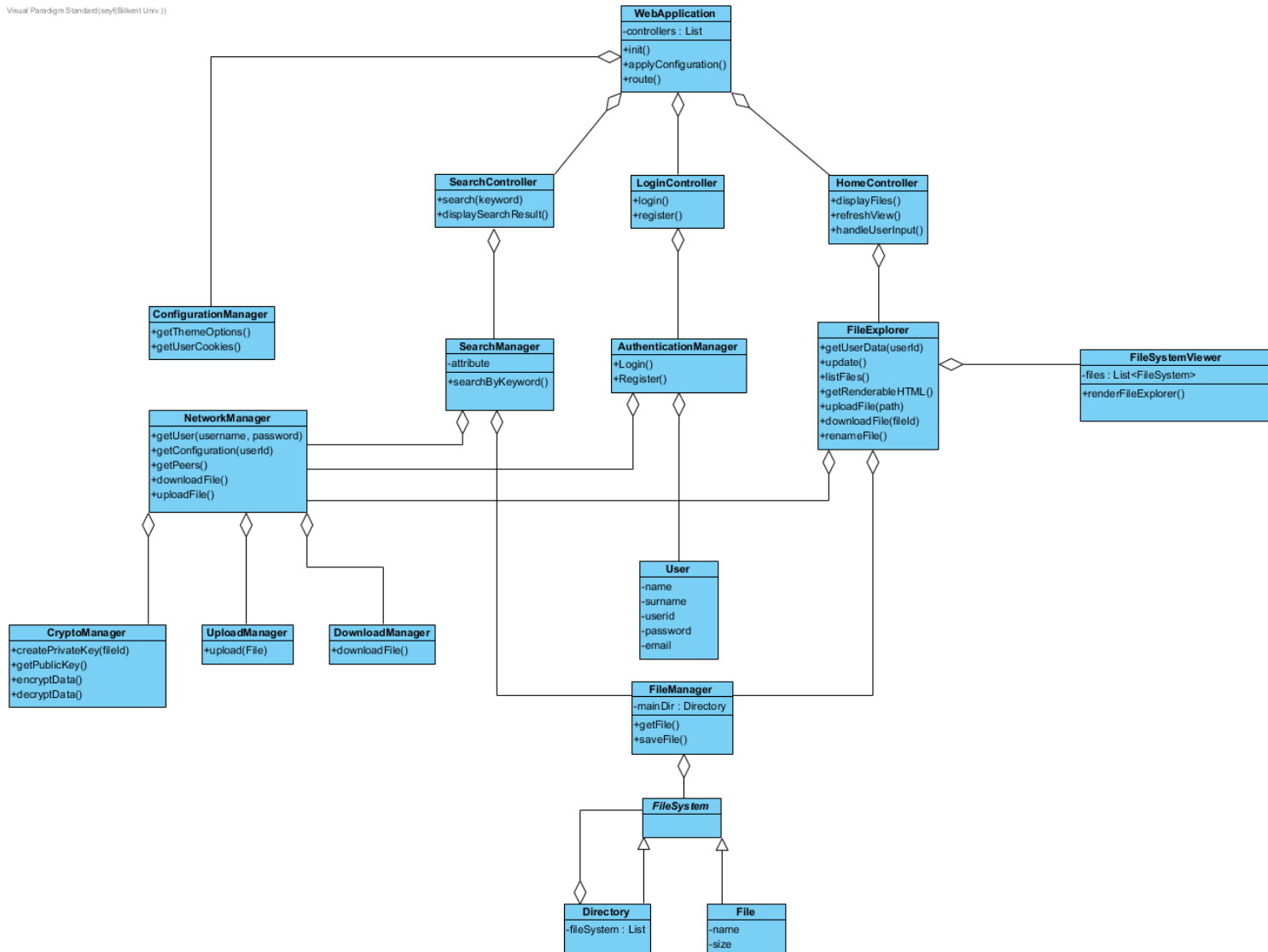
Desktop Application Class Diagram:



Description:

The design of this application is very close to the Android application. Because of the library and development environment, View classes have some little changes.

Web Application Class Diagram:



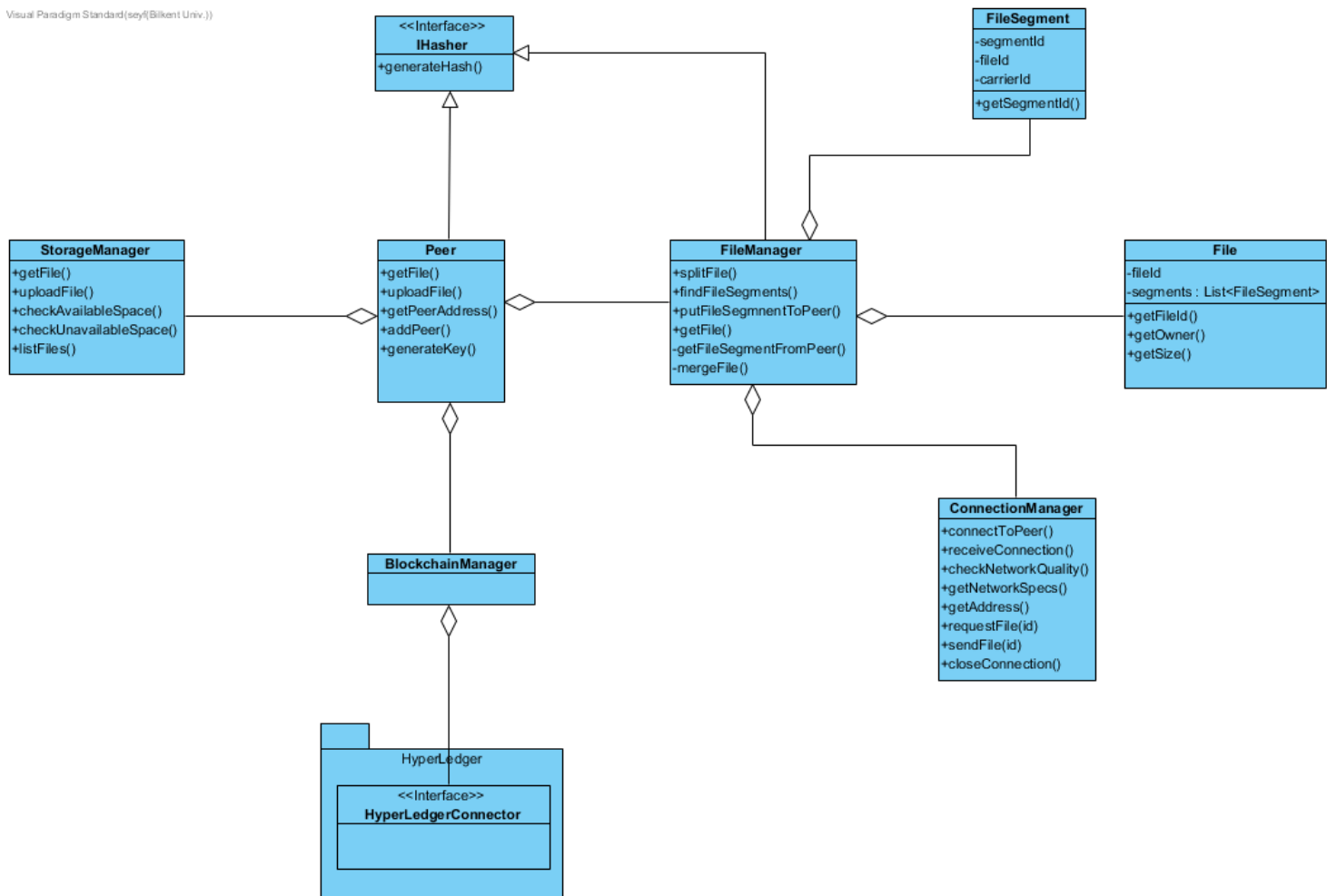
Description:

This application is designed with MVC architecture. Model classes are same with other applications. However, since it is not possible for Web application to manage local storage for providing distributed cloud storage, there is no StorageManager like components.

Like Cache manager in other applications, web application have ConfigurationManager, which will provide user settings via cookie caching. Controller and View classes are looking in inverse order but ASP.net and JSP libraries supports that kind of designs with their own configurations.

FileExplorer class will work like an inside application that is rendering file explorer and processes user inputs to manage files.

Network Class Diagram:



Description:

This system will work as services on web, desktop and Android applications. When Nebula is deployed to the client, this service will also be deployed and it will be open or will be opened at the time of application start.

This design could be merged with client for each client application but we decided to make it a different application that provides an API to the clients. In this way, we plan to have an opportunity for development process to be easier and manageable.

IHasher: Interface that enables classes for hashing for Blockchain Ledger.

Peer: Façade class that receives API requests and manages the network process. It can also generate hash for itself to write into Blockchain mechanism.

FileManager: In the distributed system, we need file segments created from files and a component to perform required operations. This class handles those, as well as, connection based operations with the Connection Manager class.

StorageManager: In this class, we can check the storage state, save/load a file, and list the existing files. It basically does the operations, which are related to shared memory segment with the Nebula network.

BlockchainManager: Since we will use 3rd party framework called HyperLedger, we need a class to have a connection with the framework. This class will be responsible for these operations. We did not specify any operation or attribute because we do not have enough knowledge about the HyperLedger.

3.5.4 Dynamic Models

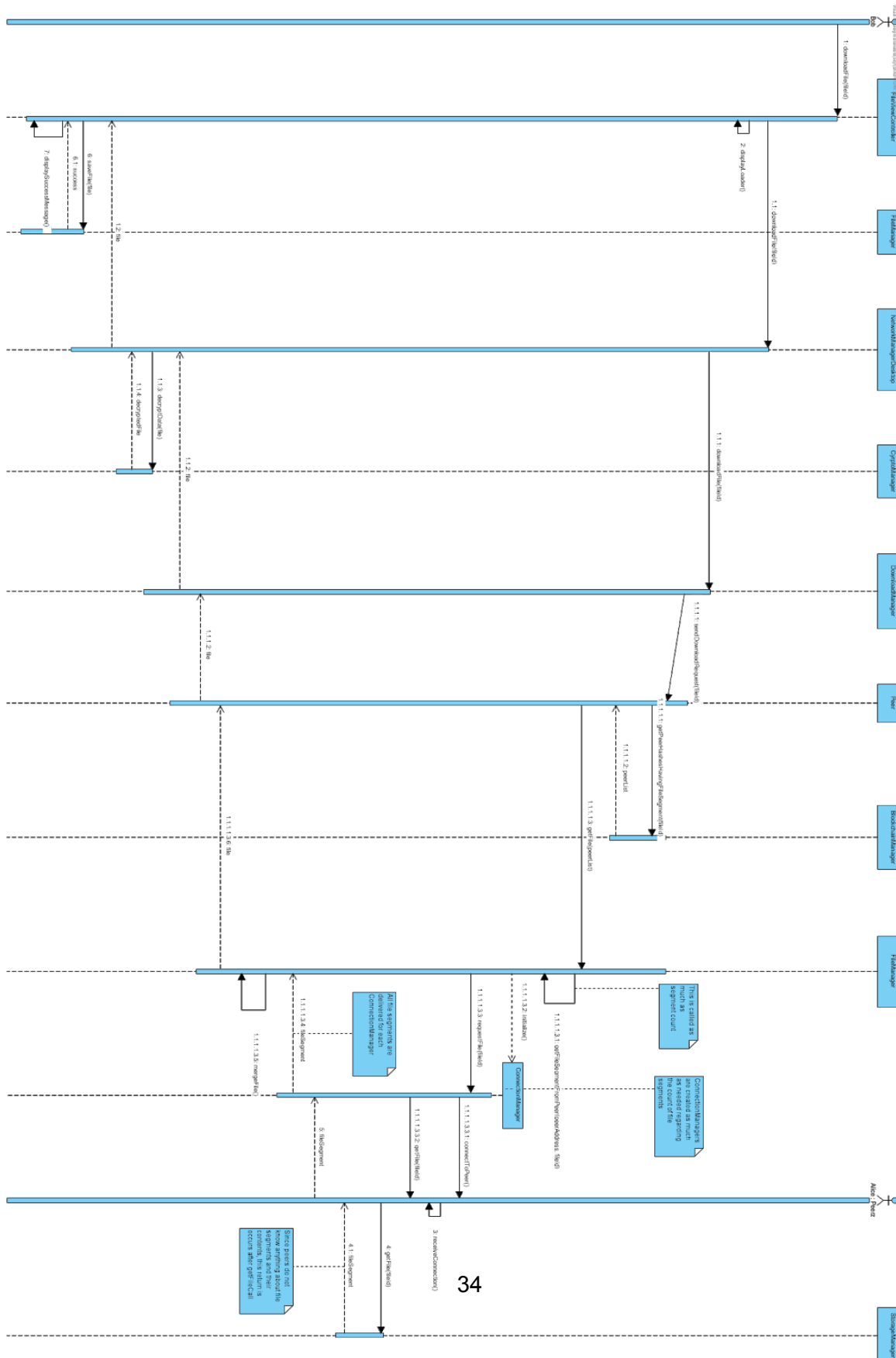
3.5.4.1 Sequence Diagrams

Scenario 1:

Bob is a user on the on the desktop application and wants to upload a new file to its storage area on the cloud and this process executes in a successful manner. Since the file will be stored at different peers actively sharing their storage areas on the Nebula, it is simulated that Alice is one of the peers that one of the file segments of Bob's new file stored.

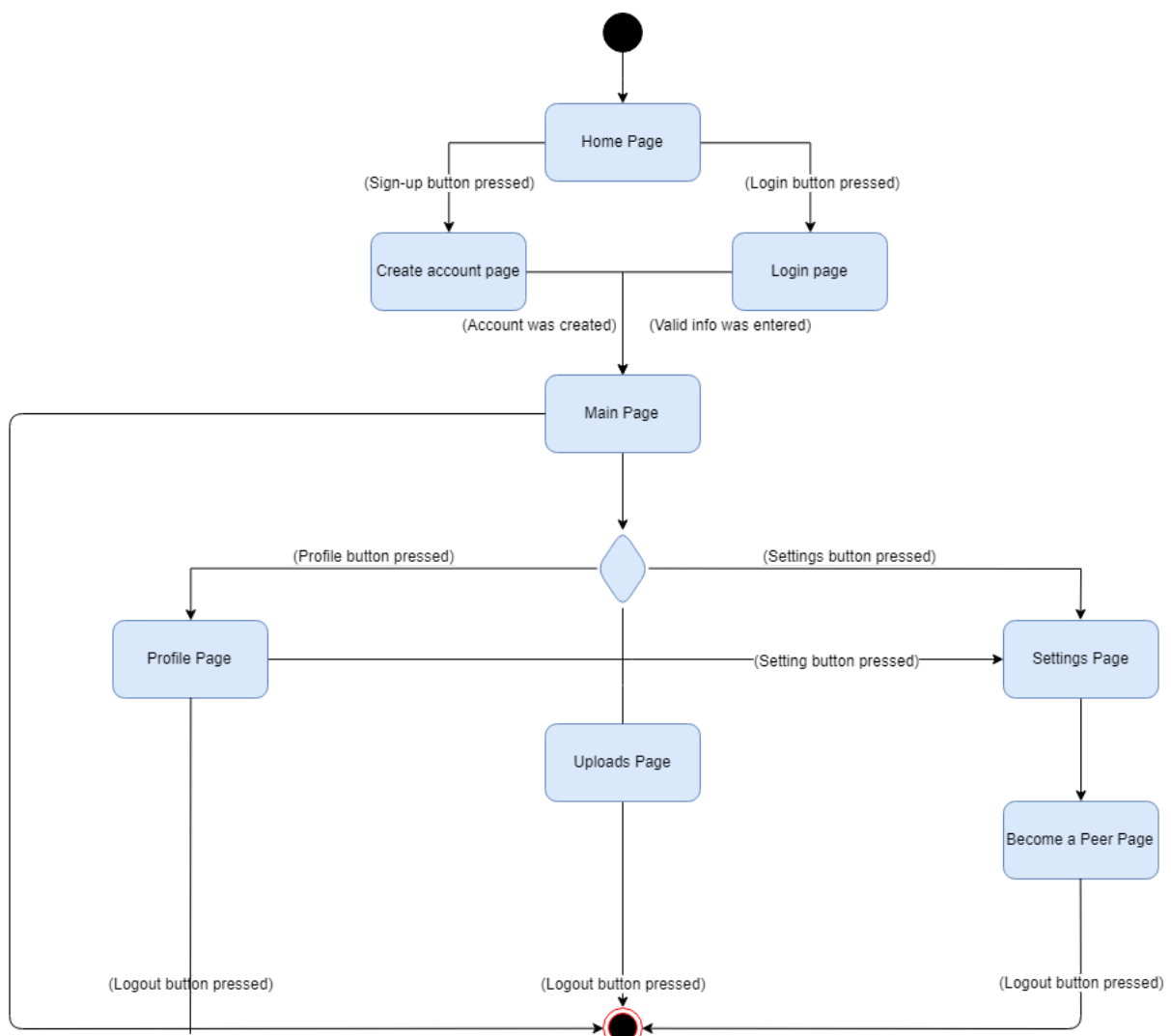
Scenario 2:

Bob is a user on the on the desktop application and wants to download a file from the application to its local storage. Alice is holding one of file segments of the Bob's file.



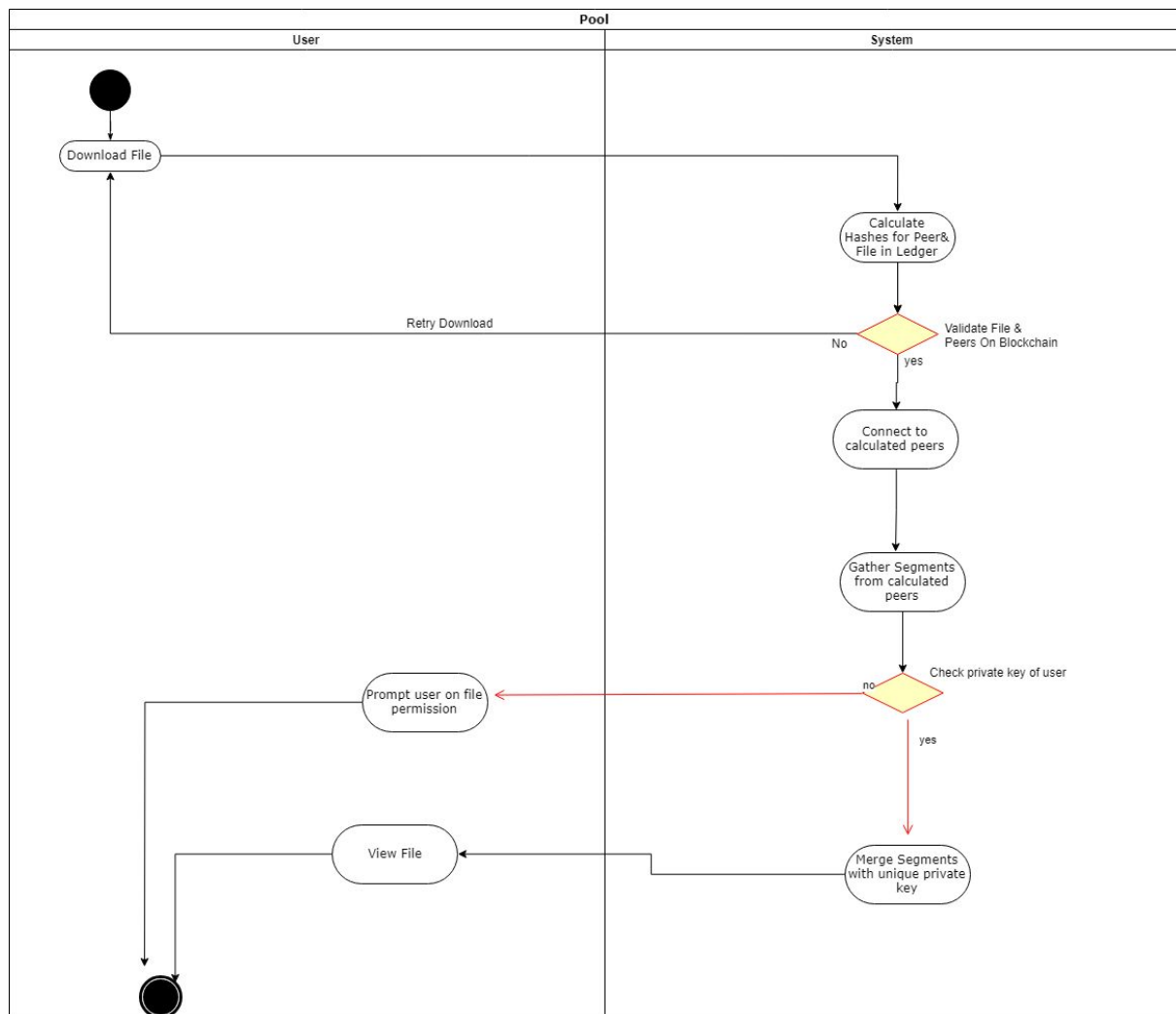
3.5.4.2 Activity Diagrams

Activity Diagram is used to show the flow of applications in Nebula. In order to see how a dynamic aspect of the system gets the user from one activity to another one. The below diagram (Main Flow Activity Diagram) can be used to visualize the activity flow of the system. First, the user enters the home page and chooses whether to sign-up or login, then with the main page can view the uploaded files, go to the profile page or change settings. Logging out from any of the pages terminates the activity flow.

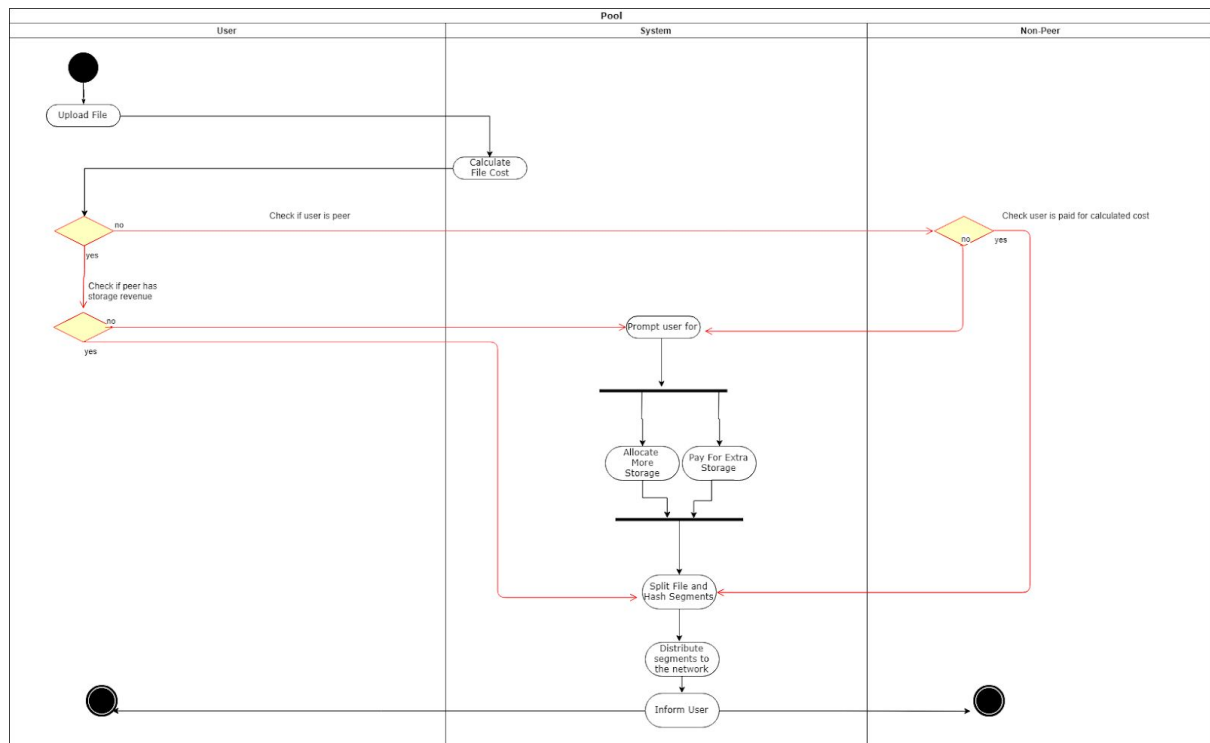


Main Flow Activity Diagram

Following activity diagram describes a detailed activity intent between system and user regarding to cases: download and upload file.



Download File Detailed Activity Diagram



Upload File Detailed Activity Diagram

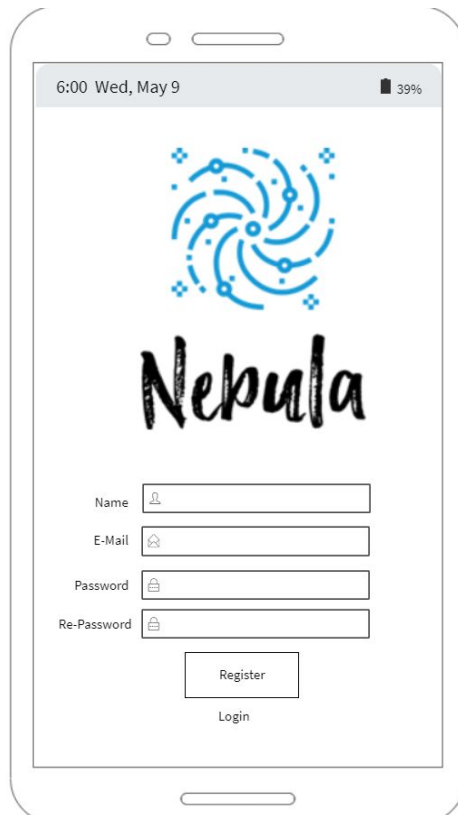
3.5.5 User Interface

Android Mockups:


Login:



Register:



6:00 Wed, May 9 39%



Nebula

Name

E-Mail

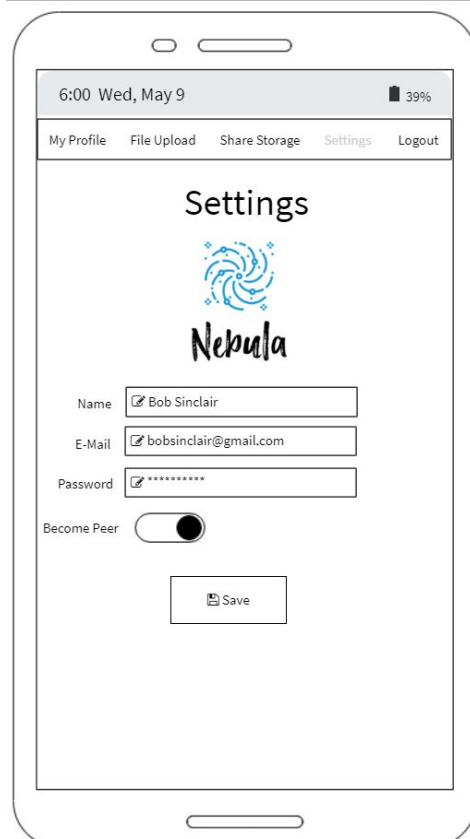
Password

Re-Password

Register

Login


Settings:



6:00 Wed, May 9 39%

My Profile File Upload Share Storage **Settings** Logout

Settings



Nebula

Name ☒ Bob Sinclair

E-Mail ☒ bobsinclair@gmail.com

Password ☒ *****

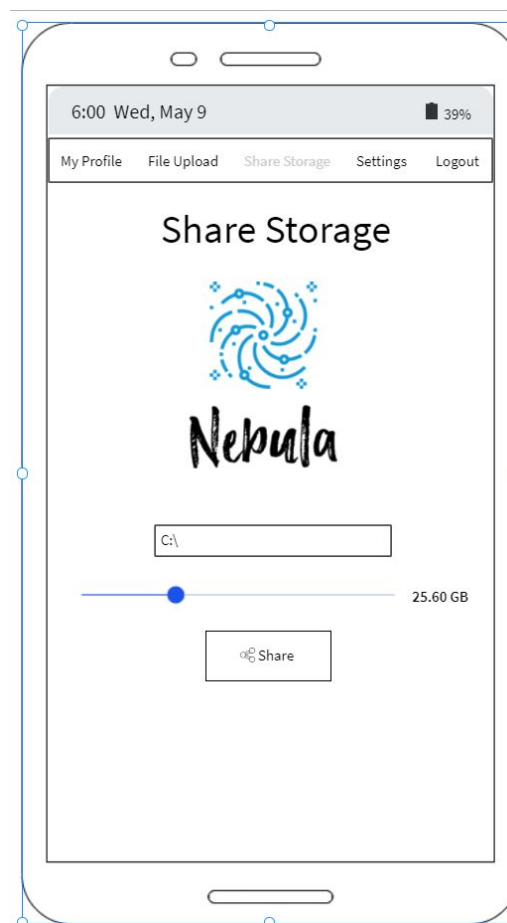
Become Peer ☒

Save

Upload File:



Share Storage:



4. Glossary

Hyperledger: Hyperledger is mainly an open source blockchains and tools collection. Main collection is backed by Linux Foundation but also various companies (including IBM) have their product in hyperledger. The reason hyperledger is used in the project Nebula is instead of putting too much effort on building blockchain infrastructure again, hyperledger collection can be re-usable for the boost innovation speed. Since the terms related to blockchain such as smart contracts and consensus mechanisms etc. are difficult to implement, we will use hyperledger components.

For more architectural information please see the reference 11 and 12.

Revenue: Nebula offers a revenue in return for the reserved storage on the devices. It will be a storage privilege for the user, which will be defined as a constant factor of reserved storage. Since Nebula will have non-peer customers, who is using the Nebula network for cloud storage but not allocated a storage for the network, it encourages customers to contribute to the network by returning storage revenue. The system will use the storage, cpu and bandwidth of the peers for the sake of Nebula network. We will reward the peers in the return of their system usage i.e free storage space. This given free space and its calculation mechanism is called Revenue.

Proof of storage / Proof of replication: Proof of storage is a scheme used by Filecoin, to prove to the client that the data is still stored. This can be done multiple times. In addition to that, Filecoin implements proof of retrievability, which proves to the client that the data can be retrieved. This is slightly different from the proof of storage, as it could have been stored correctly, but retrieving the data is not possible, as some chunks cannot be accessed. [13]

Proof of space-time: Proof of space-time is the scheme that combines time and space. It proves to the client that the data is stored at the time of the challenge. As this proof gives the time-lapse of storage it proves to the client that the data has been stored during all this period. [13]

Trustless System: Blockchains don't actually eliminate trust. What they do is minimize the amount of trust required from any single actor in the system. They do this by distributing trust among different actors in the system via an economic game that incentivizes actors to cooperate with the rules defined by the protocol. [15]

5. References

- [1] H. G. Do and W. K. Ng, "Blockchain-Based System for Secure Data Storage with Private Keyword Search," 2017 IEEE World Congress on Services (SERVICES), Honolulu, HI, 2017, pp. 90-93.
- [2] <https://www.investinblockchain.com/what-is-siacoin/>
- [3] <https://www.forbes.com/sites/forbestechcouncil/2017/12/05/todays-centralized-cloud-and-the-emerging-decentralized-edge/#4de8d2376b3c>
- [4] [https://en.wikipedia.org/wiki/Dropbox_\(service\)](https://en.wikipedia.org/wiki/Dropbox_(service)) Accessed: 31.10.2018
- [5] What is Decentralized Storage? (IPFS, FileCoin, Sia, Storj & Swarm)
<https://medium.com/bitfwd/what-is-decentralised-storage-ipfs-filecoin-sia-storj-swarm-5509e476995f>, Accessed: 31.10.2018
- [6] Decentralized storage wars. Storj v Sia v Filecoin v Maidsafe
<https://decentralize.today/decentralized-storage-wars-storj-v-sia-v-filecoin-v-maidsafe-27dc3d37434f>, Accessed: 31.10.2018
- [7] Battle of decentralized storages: SiaCoin (SC) vs Storj (STORJ) vs Filecoin (FIL)
<https://captainaltcoin.com/filecoin-vs-siacoin-vs-storj/>, Accessed: 31.10.2018
- [8] Decentralized Cloud Storage — Storj. (2018). Available: <https://storj.io/>. [Accessed: 31- Oct- 2018].
- [9] <https://umu.diva-portal.org/smash/get/diva2:1111497/FULLTEXT01.pdf>
- [10] The Meaning of Decentralization – Medium. (2017) Available: <https://medium.com/@VitalikButerin/the-meaning-of-decentralization-a0c92b76a274> [Accessed: 31- Oct- 2018].
- [11] "Hyperledger Architecture, Volume 1", 2018. https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf.

Accessed 10 Oct 2018.

[12] "About – Hyperledger". Hyperledger, 2018, <https://www.hyperledger.org/about>. Accessed 3 Nov 2018.

[13]

https://courses.cs.ut.ee/MTAT.07.022/2018_spring/uploads/Main/bruno-report-s17-18.pdf

[14] 4 of the Best Decentralized Cloud Storage Solutions to use in 2018
<https://windowsreport.com/decentralized-cloud-storage/>

[15] "What do we mean by "blockchains are trustless"?" 03.02.2018
<https://medium.com/@preethikasireddy/eli5-what-do-we-mean-by-blockchains-are-trustless-aa420635d5f6>

[16] <https://www.softwareadvice.com/resources/it-org-structure-centralize-vs-decentralize/>

[17]

<https://www.quora.com/How-does-centralized-and-decentralized-computing-differ>