Department of Computer Engineering

Bilkent University

Senior Design Project

Project Short-Name: Nebula

Low Level Design 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

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 nodes 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.

This report features an overview of the low-level architecture and design of Nebula’s system. Also, the trade-offs of our design and engineering standards will be explained. Documentation guidelines for the ease of understanding the rest of the report will be provided. After that, information about the packages and interfaces of Nebula’s systems will be presented. Finally, the class diagrams and a detailed look into each of Nebula’s software components will conclude the report.

1.1 Object Design Trade-Offs

1.1.1 Functionality vs Usability

Nebula offers great functionalities but they will offer no value if the user cannot interact with them. So our user interface will be very easy to understand and use. There will be simple instructions for operations like *“upload file”* or *“download selected file”*. We will not compromise the usability for increasing the functionality of the application.

1.1.2 Security vs Cost

Nebula stores all data in the blocks of ledger rather than within the application itself. The security of the user files is highly valuable and must be kept confidential at all costs. Hyperledger is a trust based blockchain system and so the files of the customer will always be in safe and secured by the system itself. However this security level requires some time cost, for example if the user wants to retrieve some or all of his/her files, Nebula needs to check all the ledger and this would cause some performance problems in terms of time cost.

1.1.3 Space vs Time

Number of ledgers in the blockchain may decrease the response time since in order to find every single file chunk in a correct order, all ledgers will be visited. Space required for that file does not change but the amount of chunk it is separated into, even though it will have a positive effect for security, will be negative for the time component of our system.

1.1.4 Compatibility vs Extensibility

In terms of extensibility, Nebula has to evolve with updates in the future. Also Nebula is aiming to be compatible with mobile in the future plans. After releasing the desktop application, we plan to have a web browser version and also mobile applications compatible with both IOS and Android systems.

1.1.5 Quality vs Time

We will always test our code in every stage to verify our code quality however maintaining the quality of our code would be consume a lot of time and our application also has a developing time limit. Therefore we have to balance time cost and quality in the most efficient way.

1.2 Interface Documentation Guidelines

In this report, all the class names are named in the standard 'ClassName' format, where all of these names are singular. The variable and method names follow a similar rule as in 'variableName' and 'methodName()'. In the class description hierarchy, the class name comes first, seconded by the attributes of the class, and finally concluded with the methods. The detailed outline looks similar to the one presented below:

Class Name
Description of Class
Attributes
Type of attribute: Name of attribute
Methods
MethodName(parametersOfMethod) : description of method

1.3 Engineering Standards

For the descriptions of the class interfaces, diagrams, scenarios, use cases, subsystem compositions and hardware depictions, this report follows the UML guidelines[1]. UML is a commonly used way to generate these diagrams, easy to use and since it is the method taught at Bilkent University, we chose to utilize it in the following pages. For the citations, the report follows IEEE's standards [2]. Again, this is a commonly used method and the one preferred in Bilkent University.

1.4 Definitions, acronyms, and abbreviations

TCP: Transmission Control Protocol

P2P: Peer to Peer

API: Application Programming Interface

HTTP: Hypertext Transfer Protocol

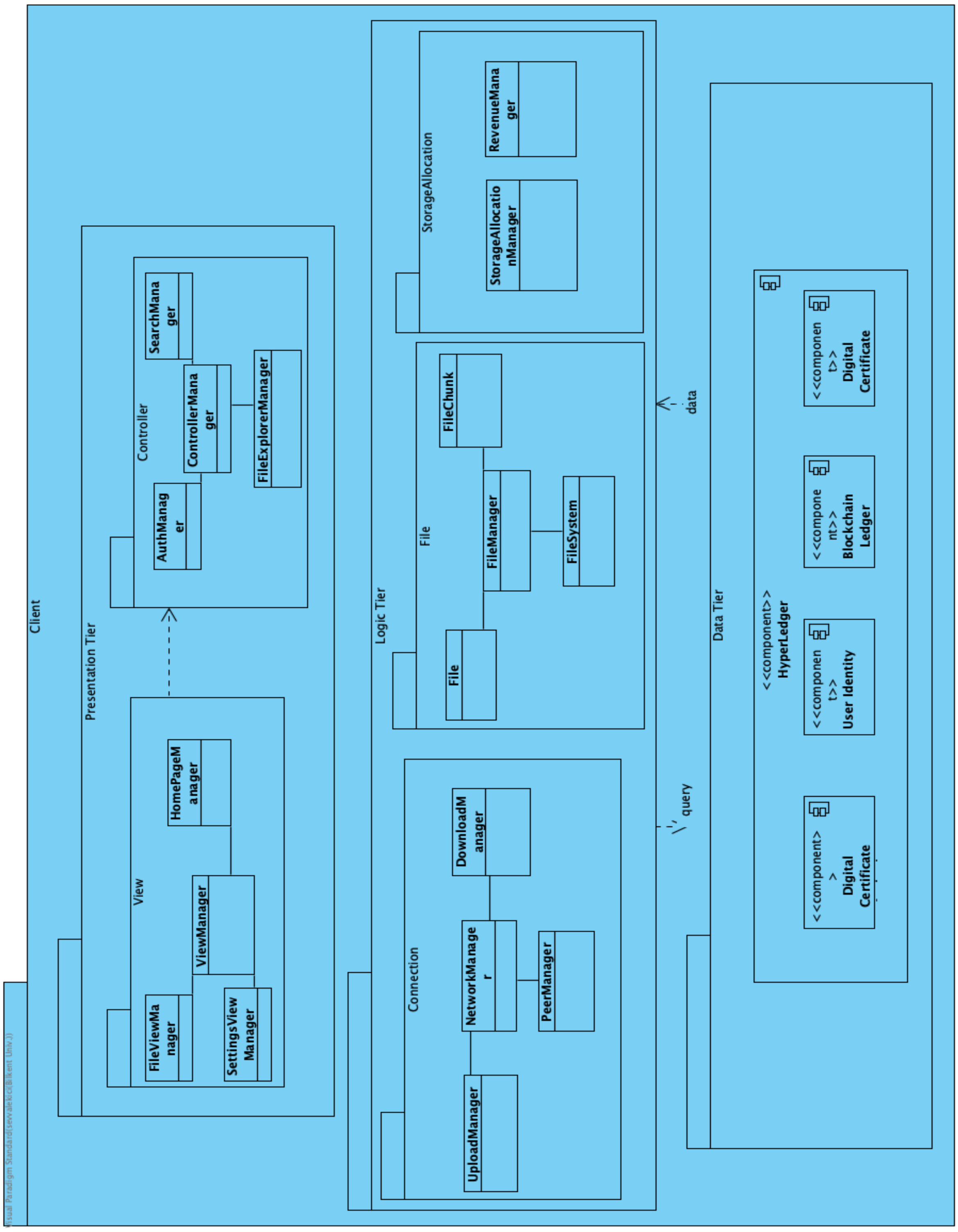
GUI: Graphical User Interface is a program interface that takes advantage of the computer's graphics to make the program easier to use.

Client: User End of the Application. For Nebula it is the web application.

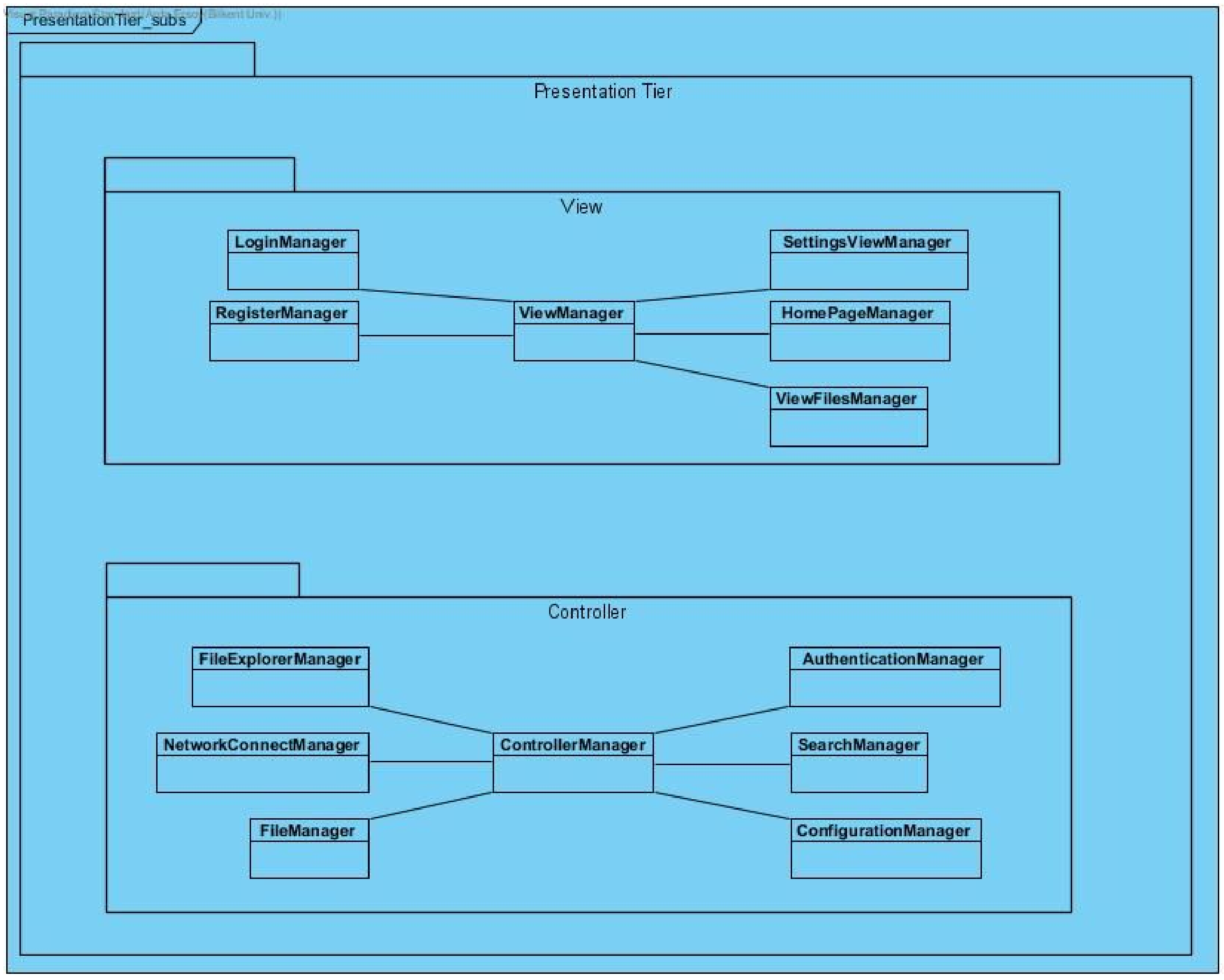
Server: The part of the application which responds to Client's requests. Responsible for data management and API interactions. In Nebula distributed network pretends server role.

2. Packages

2.1 Client



2.1.1 Presentation Tier



Presentation Tier is only responsible for the UI and the interaction between the application and the user.

View Subsystem takes care of visual interaction between user and the system. Individual explanation of components of the module are listed below:

View Manager: It controls the task distribution between other components of the view subsystem.

Home Page View Manager: The View Manager initially directing user to the home page. In this page download option is featured alongside with upload option. User will decide which one to do and Home Page View Manager will redirect the user to one of those.

File View Manager: User can view the files he/she uploaded to Nebula Cloud and download a prefered file if desired.

Settings View Manager: User can view the settings of his/her account and can change these options if desired.

Controller Manager: It controls and combines the task distribution between other controllers in this subsystem.

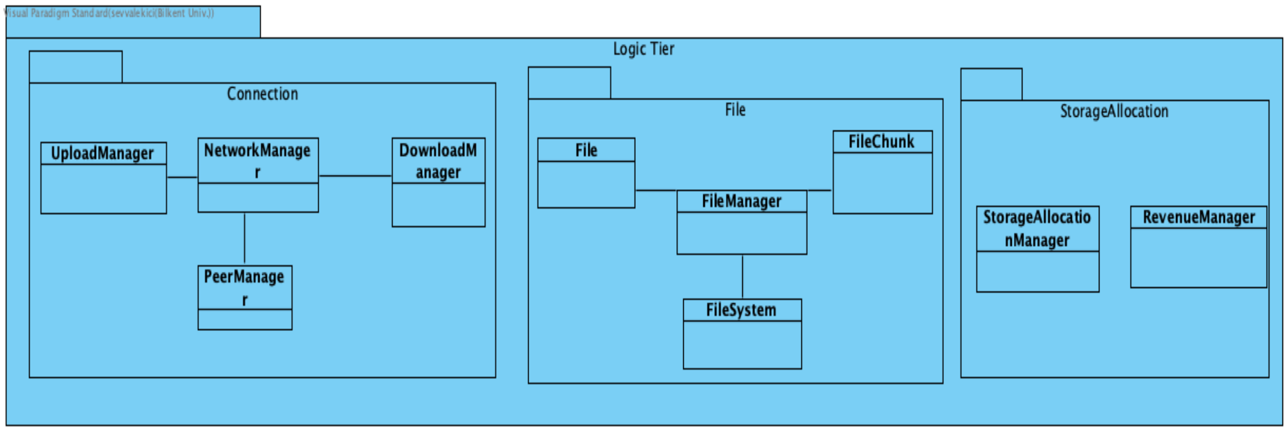
Authentication Manager: This manager controls the login and registry of the user interface. It is a control label between the client UI and the data tier which makes the data validation and sanitization.

File Explorer Manager: File Explorer Manager handles the actions of the user towards their files. These actions are downloading, uploading, deleting or viewing the files.

Account Manager: This manager is used to handle users actions for accounts such as signing up a user, checking sign up informations, logging in, changing password, deleting account and changing the option of being a peer.

Search Manager: It's a sub component of the file explorer manager. It searches the file in the system directory that client application uses.

2.1.2 Logic Tier



Logic tier subsystem consists of mission critical operations. In this subsystem, application connection operations, file separation and distribution operations, and storage allocation in the located machine is handled. With respect to the operation types, it can also query to the blockchain database and retrieve the necessary information from data tier.

Download Manager: Download Manager handles the download operation from the Nebula system to the users own storage.

Upload Manager: Upload Manager handles uploading a file to the storage system of Nebula from the local storage unit of user.

Network Manager: This is the central class of Connection Package. Network Manager is responsible for the functions of taking input elements and identifies the event that occurs.

Peer Manager: Peer manager is the connection control mechanism for the client application. It finds the peers from blockchain database and establishes a connection with those peers.

File Manager: This is the central class for the File Package. It is responsible for the communication between other classes in the package.

File System: This is the corresponding file manager agent for Nebula application. Nebula will traverse the file directory in the located operating system.

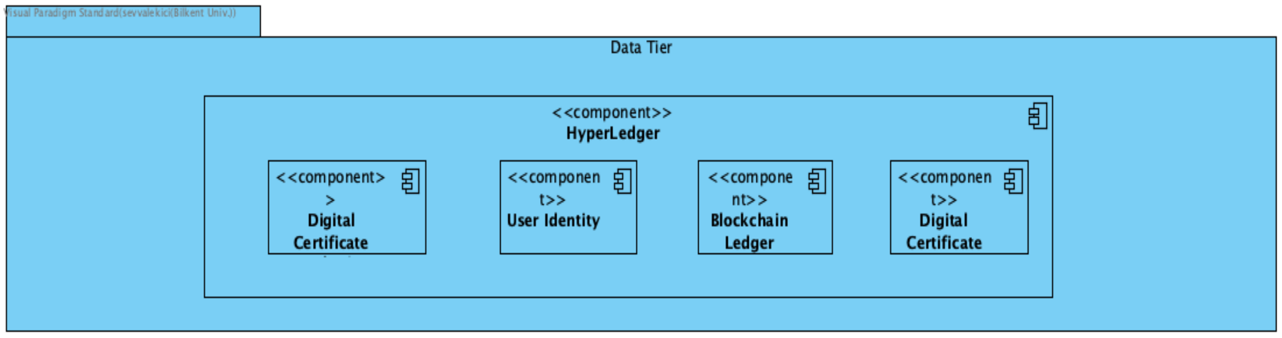
File Chunk: This is a Data-Transfer-Object(DTO) for the file chunks. The related operations regarding to file chunks will be handled here.

File: This is a Data-Transfer-Object(DTO) for the whole file. The related operations regarding to file will be handled here.

Storage Allocation Manager: This manager handles the allocated space by the user to the system. It calculates the usage of the bandwidth and the storage and decides how much storage space should be provided.

Revenue Manager: Revenue manager calculates the cost of the user to the system. Also it calculates the revenue space for a user who allocated space by the user.

2.1.3 Data Tier


Visual Paradigm Standard(sevvalekici(Bilkent Univ.))
Data Tier
<<component>>
HyperLedger
<<component>> Digital Certificate
<<component>> User Identity
<<component>> Blockchain Ledger
<<component>> Digital Certificate

These are the inherited components of the Hyperledger Fabric. These components will be used for our blockchain infrastructure. We will configure these components in order to fit Nebula network.

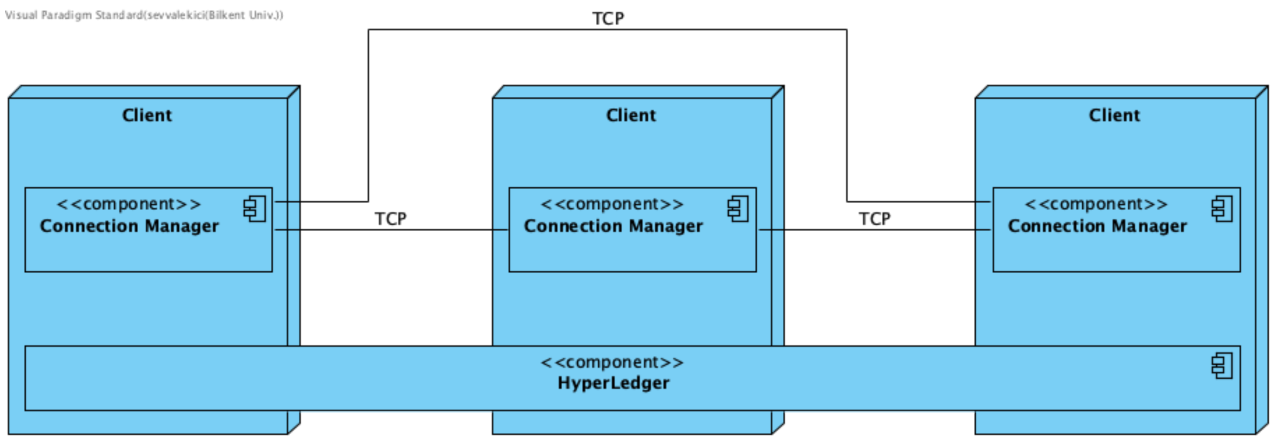
Blockchain Ledger: Blockchain ledger is responsible for the file and user audits in a distributed network.

User Identity: User identity corresponds the different actors in the blockchain network including peers, applications, administrators etc. These identities determines the permissions over the resources and the information access**.**

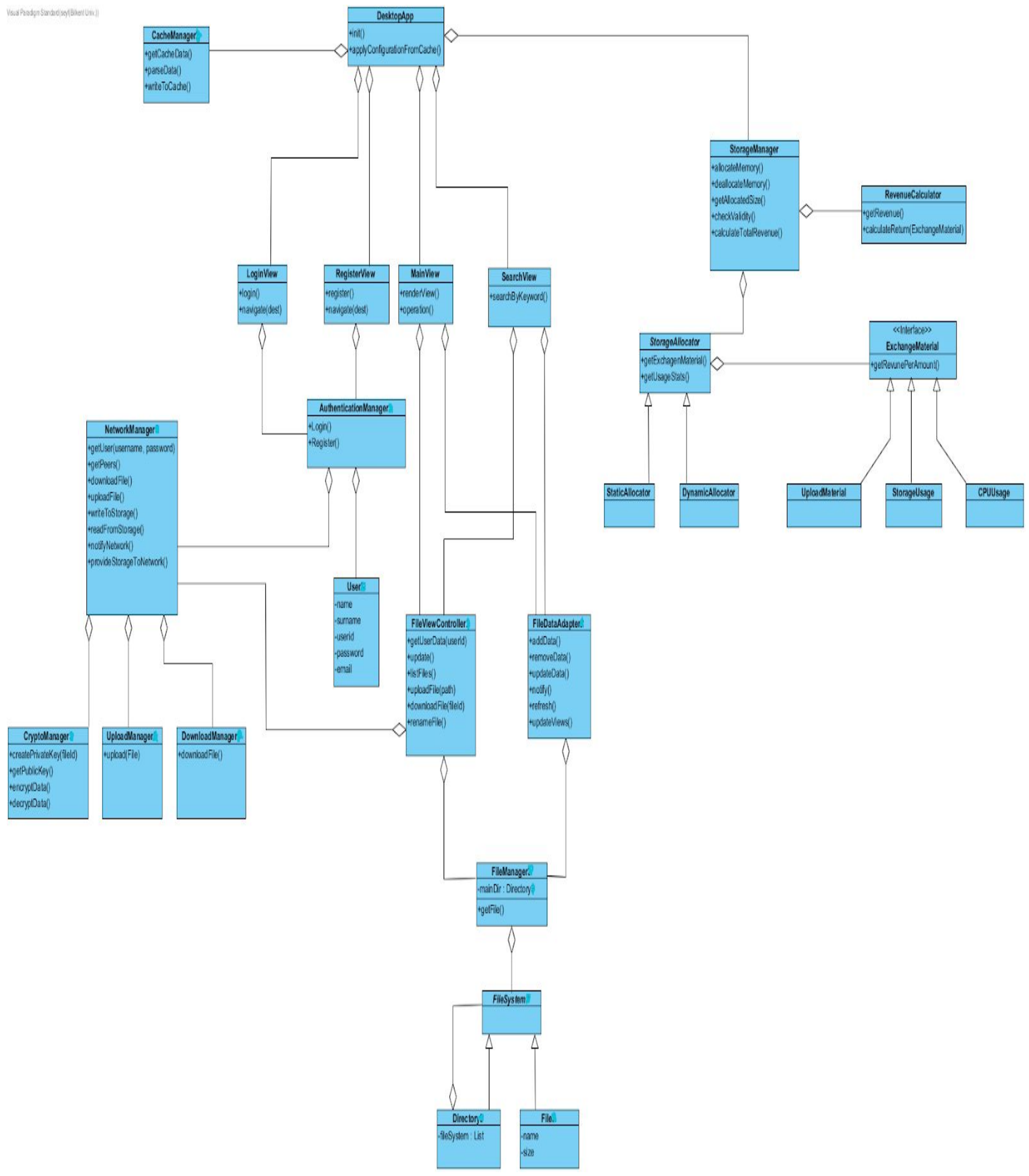
Digital Certificate: This is a digital certificate which hold a set of attributes relating to the holder of the certificate. [19]

Digital Certificate Authority: The certificate authority is a validation components which permits users to take undesired roles or do a job which is not allowed by a certain type of user identity.

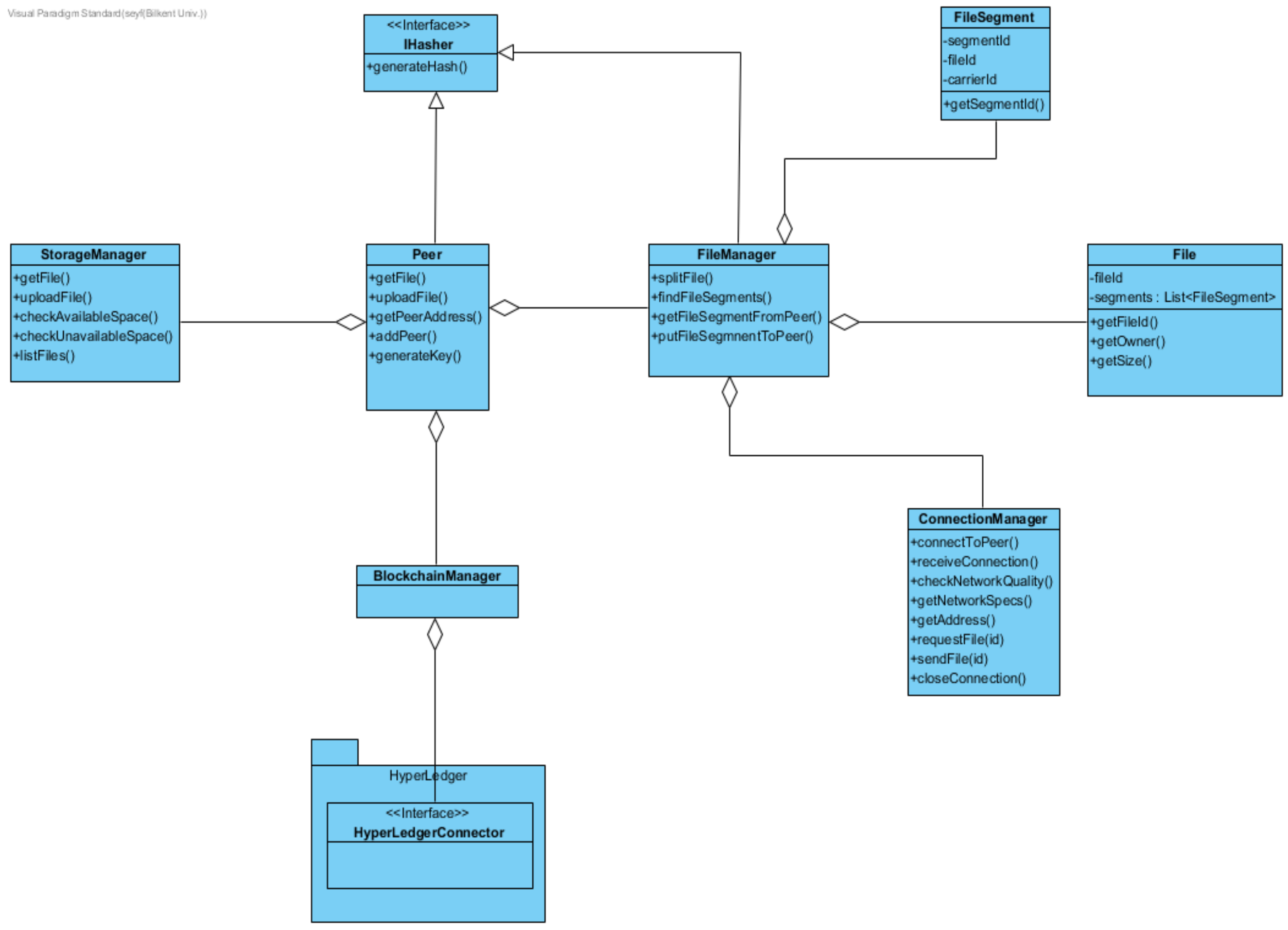
2.2 Hardware Software Mapping - Data Tier


Visual Paradigm Standard(sevvalekici(Bilkent Univ.))
TCP
Client
Client
Client
<<component>> Connection Manager
<<component>> Connection Manager
<<component>> Connection Manager
TCP
TCP
<<component>> HyperLedger

3. Class Interfaces



Nebula Desktop Client Class Diagram



Nebula Network Class Diagram

3.1. Client

3.1.1. Presentation Tier

3.1.1.1. View

View Manager
It controls the task distribution between other components of the view subsystem.
Attributes
private views : List<View> private history: Stack<View>
Methods
This class does not have any specific methods.

Login View
It checks the username and the password and leads the user into cloud application.
Attributes
private String userInput private String passwordInput
Methods
public void login() : This method checks whether the entered username is correct or not. public void navigate(dest) : If the login is succeeds, redirects the user to the home page.

Signup
It takes inputs from the user to sign up an account in the system.
Attributes
private String username private String email private String password private String reEnteredPassword
Methods
public int createAccount(String username, String email, String password): This method gets username, email and password. It creates an account with given parameters for the user.

Home Page View Manager
The View Manager initially directing user to the home page. In this page download option is featured alongside with upload option. User will decide which one to do and Home Page View Manager will redirect the user to one of those.
Attributes
private int userID
Methods
public void uploadFile(parametersOfMethod) : method that takes your file and redirects it to public void myFiles() : directs the user File View Manager page

File View Manager
User can view the files he/she uploaded to Nebula Cloud and download a prefered file if desired.
Attributes
private files : List<File>
Methods
public void viewFile() : user selects from the uploaded file list to view public void downloadFile(file) : downloads specifically chosen file

Settings View Manager
User can view the settings of his/her account and can change these options if desired.
Attributes
private String settingsPath private Map <key: String, value:String> userSettings
Methods
public String getSetting() public void setSetting() public List getSettingList()

3.1.1.2. Controller

Controller Manager
It controls and combines the task distribution between other controllers in this subsystem. This class will work as registry.
Attributes
private List<Object> controllers
Methods
public void setController() public Object getController()

Authentication Manager
This manager controls the login and registry of the user interface. It is a control label between the client UI and the data tier which makes the data validation and sanitization.
Attributes
private Path userSettings private Session session
Methods
public boolean checkIfUserActive() public boolean isUserValid()

File Explorer Manager
File Explorer Manager handles the actions of the user towards their files. These actions are downloading, uploading, deleting or viewing the files.
Attributes
URL url
Methods
public List<Object> getFileList() public void uploadFile(File file) public void downloadFile(File file)

Account Manager
This manager is used to handle users actions for accounts such as signing up a user, checking sign up informations, logging in, changing password, deleting account and changing the option of being a peer.
Attributes
User user
Methods
public void login() public void logout() public void changePassword()

Search Manager
It’s a sub component of the file explorer manager. It searches the file in the system directory that client application uses.
Attributes
FileManager fileManager
Methods
public List<File> search()

3.1.2. Logic Tier

3.1.2.1. Connection

Download Manager
Download Manager handles the download operation from the Nebula system to the users own storage.
Attributes
Path localStorage Path tempStorage List<Segments> fileSegment
Methods
public boolean download(File) : Downloads the selected file from Nebula to local drive. public boolean isFileSegmentsCompleted() public boolean isFileValid()

Upload Manager
Upload Manager handles uploading a file to the storage system of Nebula from the local storage unit of user.
Attributes
File: file
Methods
public boolean upload(File) : Uploads the selected file from local drive to Nebula.

Network Manager
This is the central class of Connection Package. Network Manager is responsible for the functions of taking input elements and identifies the event that occurs.
Attributes
Map networkCredentials
Methods
public String getUser(username, password) public Peer getPeers() public boolean downloadFile() public boolean uploadFile() public void writeToStorage() public void notifyNetwork() public void provideStorageToNetwork()

Peer Manager
Peer manager is the connection control mechanism for the client application. It finds the peers from blockchain database and establishes a connection with those peers.
Attributes
List<Peer> fileHolders List<Peer> fileUploadHolders
Methods
public List<Peer> getFileHolders(file: File) public List<Peer> getAvailablePeers() public List<Peer> findAlternativePeer(FileChunk chunk)

3.1.2.2. File

File Manager
This is the central class for the File Package. It is responsible for the communication between other classes in the package.
Attributes
Path localStorage Path tempStorage
Methods
public void splitFile() : this method splits the byte array of file into chunks. public void findFileSegment() : this method checks the peers and finds the desired file’s chunks. public void getFileSegmentFromPeer() : this method gets the related file segments from the peers public void putFileSegmentToPeer() : this method is for distribution of the chunked file segments. It puts a file segment to a peer.

File System
This is the corresponding file manager agent for Nebula application. Nebula will traverse the file directory in the located operating system.
Attributes
Path root
Methods
public void writeToDisk() public byte[] readFromDisk(Path path)

File Chunk
This is a Data-Transfer-Object(DTO) for the file chunks. The related operations regarding to file chunks will be handled here.
Attributes
String fileId String peerId String offset byte[] data
Methods
public void addHeader(String fileId, int sequenceNumber) public Map getHeaderInfo() public void removeHeader()

File
This is a Data-Transfer-Object(DTO) for the whole file. The related operations regarding to file will be handled here.
Attributes
private int FileId private int size private chunks : List<FileChunk>
Methods
private int getFileId() : This method returns File id number. private int getOwner() : This method returns owner’s id number. private int getSize(): This method returns the size of the file.

Connection Manager
Connection manager is the connection control mechanism for the files. It finds the peers from blockchain database and establishes a connection with those peers.
Attributes
This class does not have any attributes.
Methods

connectToPeer():
receiveConnection():
checkNetworkQuality():
getNetworkSpecs():
getAddress():
requestFile(id):
sendFile(id):
closeConnection():

3.1.2.3. Storage Allocation

StorageManager
This manager handles the allocated space by the user to the system. It calculates the usage of the bandwidth and the storage and decides how much storage space should be provided.
Attributes
This class does not have any attributes.
Methods
public void allocateMemory() public void deallocateMemory() public double getAllocatedSize() public boolean checkValidity() public double calculateTotalRevenue()

StorageAllocator
This manager handles the allocated space by the user to the system. It calculates the usage of the bandwidth and the storage and decides how much storage space should be provided.
Attributes
This class does not have any attributes.
Methods
public getExchangeMaterial()

RevenueCalculator
Revenue manager calculates the cost of the user to the system. Also it calculates the revenue space for a user who allocated space by the user.
Attributes
This class does not have any attributes.
Methods
public getRevenue() public calculateReturn(ExchangeMaterial)

3.1.3. Data Tier

3.1.3.1. Hyperledger

Hyperledger has its own class structure so we are directly integrating that into our project Nebula.

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 3 and 4.

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.

Consensus: The process of keeping the ledger transactions synchronized across the network — to ensure that ledgers update only when transactions are approved by the appropriate participants, and that when ledgers do update, they update with the same transactions in the same order. [7]

5. References

[1] IBM, "UML - Basics," June 2003. [Online]. Available: http://www.ibm.com/developerworks/rational/library/769.html

[2] IEEE, "IEEE Citation Reference," September 2009. [Online]. Available: https://ieee-dataport.org/sites/default/files/analysis/27/IEEE%20Citation%20Guidelines.pdf [Accessed 12- Feb- 2019].

[3] “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.

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

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

[6] “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

[7] “Introduction -- Hyperledger-Fabric”,2018. https://hyperledger-fabric.readthedocs.io/en/release-1.3/blockchain.html. [Accesssed: 19.12.2018].