CHAPTER 1 INTRODUCTION

This project is a stepping stone towards the building of a decentralized internet. This will not only improve the user experience by providing a fast service, but also improve the quality of product and help to curb piracy.

This project is an immense learning experience for us as it will help us to understand the working of the decentralized internet with an in-depth knowledge of the decentralized architecture. Whatever we learnt during our time in college, this project is something that inculcates all the attributes of almost every aspect of Information Technology Engineering.

1.1 NEED

The current video streaming system has some vulnerability which can be improved so as to provide a seamless way to the users to interact with the website and enjoy fast and secure browsing experience. Let us look into some of the most common vulnerabilities.

1.1.1 Lack of Robustness

Servers can be easily taken down using attacks such as Distributed Denial of Service (DDoS) attack, which is the most commonly used method. Such attacks exploit the fact that legitimate users can be denied access to the server/service by flooding the server with multiple requests.

1.1.2 Security

Since most of the data is centralized, it is more vulnerable to security threats. If the server is compromised, the entire user data and information may be compromised.

1.1.3 Concurrent Content Delivery

It is often observed that servers go down due to the fact that multiple users try to access a given data stream. Upgradation of server is an expensive solution to the above stated problem.

1.2 BASIC CONCEPT

The components of the decentralized web include Peer to Peer (P2P) file sharing, Distributed Hash Tables (DHT), blockchain, Self-certifying File Systems, Consensus Protocols and Smart Contracts.

1.2.1 Peer to Peer File Sharing

Applications of P2P file sharing such as BitTorrent leverage its users' resources to distribute all types of digital files to its consumer without the need of a central governing model. Client server architecture based content sharing services often incur high electricity cost to maintain high speeds of content delivery, to maintain the temperature of the servers among many other factors [1]. Since P2P architecture is self-maintaining, resilient and only need limited infrastructure and control, it is vastly superior, faster, more secure and robust than the existing client server architecture [2, 3].

1.2.2 Distributed Hash Table

Distributed Hash Tables are used as a lookup service in distributed and decentralized services [2]. They are used to map identifiers from a common pool of peers or nodes in an overlay network [4]. A DHT is an extension of a simple hash table that saves data in the form of key-value pairs on Node IDs. The Node Id is generated using the nodes' IP address or geographic information. The key is generated using a custom hash function using the data item as a parameter [2]. Most of the existing DHT assume that its peers are spread over the ID space uniformly [4].

1.2.3 Blockchain

Blockchain is a distributed, transparent, immutable ledger having a consensus protocol at the root of it. It is a growing database of records that have been executed among peers who have taken part in the transaction [5, 6]. These ledgers are visible and using a P2P approach, the peers or nodes in the Blockchain network can edit the distributed ledger [5, 7, 8, 9]. This makes tampering with the blocks comprised within the Blockchain extremely

challenging given the cryptographic data structure used in Blockchain and no necessity for secrets.

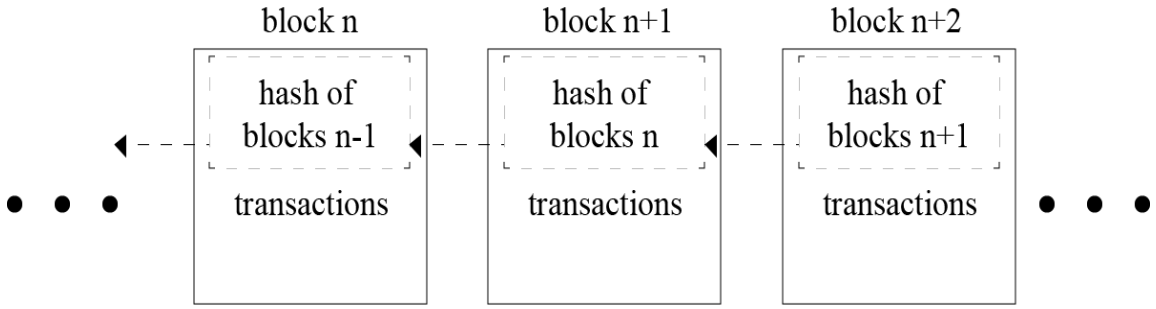

Fig. 1.1 Blockchain

1.2.4 Self-certifying File System

SFS is a secure file system spread over the internet. It provides one namespace for all the files in the world. SFS is inherently secure, and hence its users can share sensitive files without worrying about a third party tampering or reading the file [10]. IPFS uses the public key cryptography used in SFS [11].

1.2.5 Consensus Protocols

Consensus Protocols are the backbone of any blockchain application. Such protocols are used to provide authenticity, non-repudiation and integrity to the blockchain network, by utilizing a decentralized peer-to-peer network for verification of transactions before adding a block to the public ledger [5, 8]. The bitcoin blockchain uses the concept of Proof of Work (PoW) to help decide validate the transactions occurring and also helps in avoiding the forking problem in blockchain [5, 8]. Some other types of Consensus Protocols are Delegated Proof of Stake (DPoS), Proof of Activity (PoA) - an amalgamation of PoW and PoS [5].

1.2.6 Smart Contracts

A smart contract is “a digital contract that is written in source code and executed by computers, which integrates the tamper-proof mechanism of blockchain” [12, 13]. Smart contracts have transformed the blockchain scenario from a financial transaction protocol to an all-purpose utility. They are pieces of software, not contracts in the legal sense, that extend blockchain’s utility from maintaining a ledger of financial transactions to automatically

implementing conditions of multi-party agreements. Smart contracts are executed by a computer network that uses consensus protocols to agree upon the series of actions resulting from the contracts content [14]. The high-level programming languages used for writing smart contracts are mainly Solidity, Serpent and Low-level Lisp-like Language (LLL) [13].

1.2.7 Git

Technological growth has happened at a very high rate in the recent decades, especially in the field of computers. Computers have evolved from huge ineffective mainframe computers to today's portable highly effective laptops, mobile phones and desktops. Software being an integral part of the computer system, large number of project files are created. To keep track of all the changes in the files, a version control system is used. One of the most popular version control system is Git. One of the advantages for using Git is being open source in nature, and data can be extracted easily through the change logs maintained by it [12, 15].

Fig. 1.2 Forking and merging of git branches

1.3 APPLICATIONS

- Stream video and audio uninterruptedly.
- Provides a platform for independent creators as well as corporate companies alike to upload, distribute and share their videos.
- It can help to spread educational content to places with low internet bandwidth.
- The content which is uploaded on our platform is immutable.

CHAPTER 2 LITERATURE SURVEY

Before developing any application, product or service, an extensive survey of related technologies and field of work is necessary. The following section contains the work of people in the related field.

2.1 Related Work Done

Some of the works of distinguished people in the direction of our project scope are:

- **Satoshi Nakamoto, Bitcoin**

  Satoshi Nakamoto is the name used by the unknown person or people who developed bitcoin, and created and deployed bitcoin's original reference implementation. As part of the implementation, they also devised the first blackchain database. In the process, they were the first to solve the double spending problem for digital currency using a peer-to-peer network.

- **V. Buterin, Ethreum**

  Buterin is a co-founder and inventor of Ethereum, described as a "decentralized mining network and software development platform rolled into one" that facilitates the creation of new cryptocurrencies and programs that share a single Blockchain (a cryptographic transaction ledger). It supports a modified version of Nakamoto consensus via transaction-based state transitions.

- **Juan Benet,IPFS**

  Juan Benet is the inventor of the InterPlanetary File System (IPFS), a new protocol to make the web faster, safer, and more open, and Filecoin, a cryptocurrency incentivized storage network. The IPFS Project has grown into a large open source movement to re-decentralize the web, safeguard our data, and improve our applications.

Various sources of information for the completion of our work are:

2.1.1 Journal Paper

1. Donhee Han, Hongjin Kim, Juwook Jang, “Blockchain based Smart Door Lock System”, IEEE, Information and Communication, pp. 1165, 2017.
2. Robert W. Lucky, "The Lure of Decentralisation", IEEE Spectrum, pp. 23, 2017.
3. Y. Sompolinsky and A. Zohar, “Secure high-rate transaction processing in Bitcoin,”, Springer, Financial Cryptography, pp. 507-527, 2015.
4. Konstantinos Christidis, and, Michael Devetsikiotis, “Blockchains and Smart Contracts for the Internet of Things”, IEEE Access, Special Section on the Plethora of Research in Internet of Things, 2016.

2.1.2 Conference Paper

1. Malik Muhammad Imran Pattal, Li Yuan, Zeng Jianqiu, “Web 3.0: A real personal Web”, IEEE, Third International Conference on Next Generation Mobile Applications, Services and Technologies, pp. 125128, 2009
2. Lakshmi Siva Sankar, Sindhu M, M. Sethumadhavan, “Survey of Consensus Protocols on Blockchain Applications”, IEEE, International Conference on Advanced Computing and Communication Systems, 2017.
3. Ruchika Malhotra, NakulPritam, KanishkNagpal, "Defect Collection and Reporting System for Git based Open Source Software", IEEE, International Conference on Data Mining and Intelligent Computing (ICDMIC) 2014
4. Jiin-Chiou Cheng, Narn-YihLee ,Chien Chi , and Yi-Hua Chen, “Blockchain and Smart Contract for Digital Certificate”, IEEE, Proceedings of IEEE International Conference on Applied System Innovation, 2018.
5. FabiusKlemm, SarunasGirdzijauskas, Jean-Yves Le Boudec, Karl Aberer, “On Routing in Distributed Hash Tables”, IEEE, 7th

International Conference on Peer-to-Peer Computing, pp. 113-120, 2007.

6. Deepak K. Tosh, Sachin Shetty, Xueping Liang, “Consensus Protocols for Blockchain-based Data Provenance: Challenges and Opportunities”, IEEE, 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), pages 469–474, 2017.
7. Sachchidanand Singh, Nirmala Singh, “Blockchain: Future of Financial and Cyber Security”, IEEE, 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), pp. 463–467, 2016.
8. DejanVujičić, DijanaJagodić, SinišaRanđić , “Blockchain Technology, Bitcoin, and Ethereum: A Brief Overview”, IEEE, 17th International Symposium INFOTEH-JAHORINA, 2018.

2.1.3 Study Papers

1. Satoshi Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System”, White Paper, 2008.
2. V. Buterin, “Ethereum white paper: a next generation smart contract & decentralized application platform,” Ethereum White Paper, 2013.
3. Smith, Jerry. “Distributed Computing with Aglets”. White Paper, 1999.
4. J. Blackburn, K. Christensen, ”A Simulation Study of a New Green BitTorrent,” IEEE, Proc. Green Communications Workshop in conjunction with IEEE ICC’09, 2009.
5. Olaf Landsiedel, Stefan Gotz, Klaus Wehrle, “Towards Scalable Mobility in Distributed Hash Tables” IEEE, Peer-to-Peer Computing, 2006.
6. David Mazieres, “Self Certifying File System”, Doctor of Philosophy, Massachusetts Institute of Technology, Massachusetts, USA, 2000.
7. Juan Benet, “IPFS - Content Addressed, Versioned, P2P File System (Draft 3)”, White Paper, 2014.

8. Konstantinos Christidis, and, Michael Devetsikiotis, “Blockchains and Smart Contracts for the Internet of Things”, IEEE Access, Special Section on the Plethora of Research in Internet of Things, 2016.

2.2. EXISTING TECHNOLOGIES

The blockchain is a relatively new approach in the field of information technology. The complexity of the technology poses many challenges and foremost amongst these are management and monitoring of blockchain based decentralized applications. Next section takes a deep dive in two such technologies - Ethereum and IPFS.

2.2.1. Ethereum

Ethereum, proposed by a cryptocurrency researcher and programmer Vitalik Buterin, is a public, open-sourced, blockchain-based distributed computing platform having smart contract functionality [16, 17]. Ethereum represents a blockchain with a built-in Turing complete programming language called Solidity. It facilitates an abstract layer allowing anyone to create their own rules for ownership, formats of transactions, and state transaction functions. This is achieved by inculcating smart contracts, a set of cryptographic rules that are processed only if all necessary conditions are satisfied [4, 18].

2.2.2. InterPlanetary File System

The InterPlanetary File System (IPFS) is a distributed file system which incorporates ideas from existing technologies like BitTorrent, Git, SFS, and Kademlia and models them into a complete system [11]. IPFS, a peer-to-peer distributed file system, aims to replace HTTP and build a better web for us all [19]. The torrent protocol facilitates relocation of data between nodes comprising the infrastructure and the Kademlia DHT is used for the

management of metadata [5]. IPFS can be assumed as a single BitTorrent collection which exchanges data within one Git repository. IPFS facilitates a high-put content-addressed block storage model, with content addressed hyperlinks. IPFS includes a distributed hashable, reward-driven block exchange, and a self-certifying namespace. IPFS doesn't have more than one point of failure, and it is not mandatory for the peers to trust one another [11, 13]. IPFS borrows the concept of Merkle Directed Acyclic Graphs (DAGs) from the Git Version Control System.

The Merkle DAG object model helps capture changes to the IPFS tree, or even a permanent web, in a distributed-friendly way [11]. Figure 2.1 depicts the creation of a new object: the client sends its object to any node on its nearest site [20].

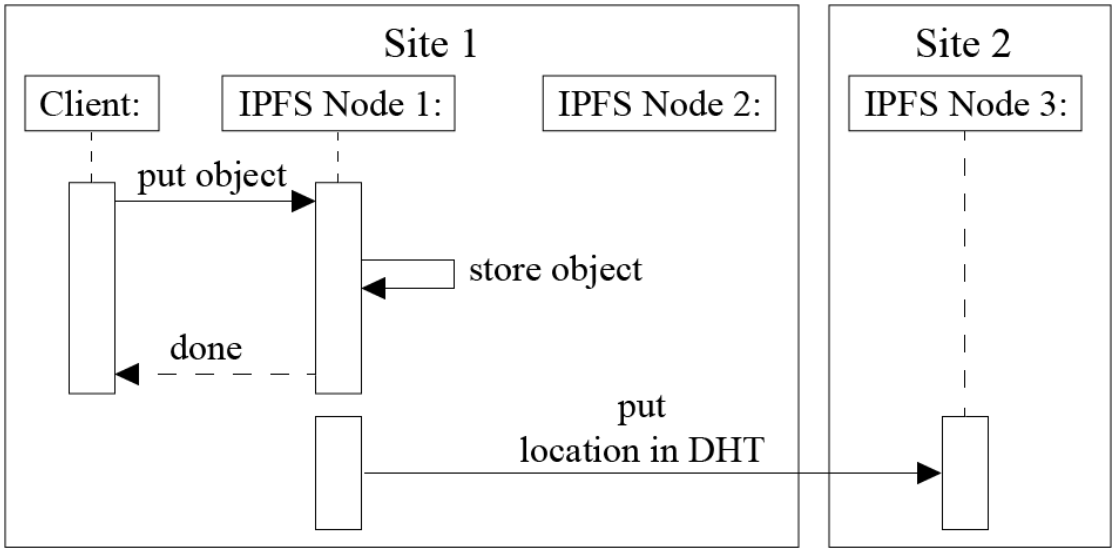

Fig. 2.1 Writing an IPFS Block

This node stores the object locally and put the location of the object in the DHT. Because the DHT does not provide locality, the node storing this metadata can be located in any node composing the Fog infrastructure. In our example, Node 4 belonging to Site 2 stores the location of the object that has been created on Node 1. Figure 2.2 illustrates what happens when the client reads an object stored on its local site [20].

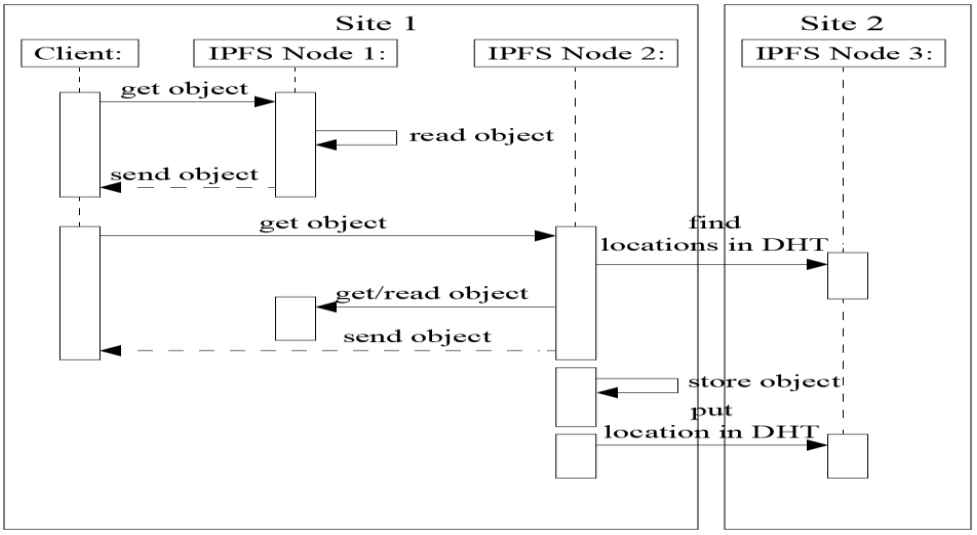

Fig. 2.2 Reading to a block to IPFS

.

Each time an IPFS node receives a request for a particular object, it first, checks whether this object is available on the node. In this case, the node sends the object directly to the client. Otherwise, the IPFS node should rely on the DHT protocol to locate the requested object. That is, it should compute the hash based on the object id, contact the node in charge of the metadata, retrieve the object from the node(s) storing it (using the BitTorrent protocol), make a local copy while sending the data to the client, and lastly update the DHT in order to inform that there is a new replica of the object available on that node [11, 20]. Figure 2.3 describes what happens when an object is requested from another site (because the client moves from a site to another one or because the object is accessed by a remote client) [20].

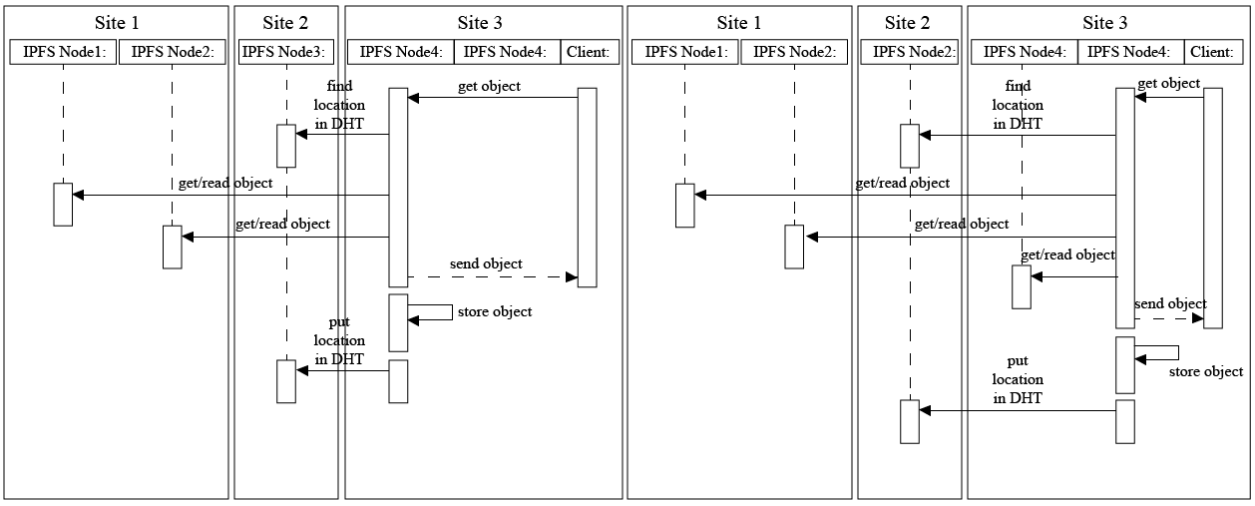

Fig 2.3 Read an object stored remotely

In any case, the client contacts a node on the site it belongs to. Because the node does not store the object, the protocol is similar to the previous one involving the extra communication with the DHT [11, 20]. In the network model of decentralized services, applications distribute their workload over multiple hosts [21]. The administration of the network is also very complex. A problem of these approaches arises when a node fails to provide the desired service. In this case, the network has to look for the service in another node. If it still cannot find the node with the desired service, it will keep on looking in different nodes. If the requested service is not found, there has to be some kind of timeout agreement in order to prevent the user having to wait endlessly [21]. The problem with system implemented with peers is that no one is responsible. If someone uploads a family picture, and ten years later, he wants the photo that is stored in the decentralized network, all those nodes may have been gone. The data might have been erased unknowingly years before [22].

CHAPTER 3 PROJECT STATEMENT

The first and foremost goal of our proposed system will be to deliver media content requested by a user in a fast and timely manner, limiting the packet losses using a decentralized app (dApp). Future scope will include developing an Android application and if possible iOS clients for the service. On the Android and iOS clients, the users should be able to download the content.

3.1. What is to be developed?

We have to develop a progressive web application for blockchain based video and audio streaming services, inculcating the concept of peer- to-peer communication and decentralized framework.

Project objectives:

- To develop video and audio streaming service with minimum failure points.
- To understand how the blockchain works.

3.2. Technologies used

i. **React**

React is a JavaScript library for building user interfaces. It is maintained by Facebook and a community of individual developers and companies.

ii. **HTML**

Hypertext Markup Language is the standard markup language for creating web pages and web applications. With CSS and Javascript, it forms a cornerstone technologies for the World Wide Web.

iii. **CSS**

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language like HTML.

iv. **Solidity**

Solidity is an object oriented programming language for writing smart contracts. It is used for implementing smart contracts on various blockchain platforms.

CHAPTER 4 SOFTWARE AND HARDWARE REQUIREMENTS

Any software being developed needs a minimum system requirement. The settings are based on the configurations on which the program can run without any delay in providing efficient services. The optimum requirements for efficiently running the software on the system are as follows.

4.1 Hardware and Software Specifications

4.1.1 Hardware

- **Processor**: Intel Core 2 Duo or later.
- **Memory**: 2 GB RAM.
- **Graphics**: Video card must be 256 MB or more and should be a DirectX 9compatible.
- **Storage**: 2 GB available space

4.1.2 Software

- **Operating System**: Windows 7/8/10, MacOS, any Linux based OS
- **Browsers** : Mozilla Firefox and Google Chrome with Metamask plugin, Opera Browser

CHAPTER 5 DESIGN

Software design is the process by which an agent creates a specification of a software artifact, intended to accomplish goals, using a set of primitive components and subject to constraints. Software design refers to all the activities involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems.

5.1 Data Flow Diagrams

A Data Flow Diagram is a way of representing a flow of a data of a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself.

It has three levels: Level 0, Level 1 and Level 2.

5.1.1 Level 0:

A context diagram is a top level (also known as "Level 0") data flow diagram. It only contains one process node ("Process 0") that generalizes the function of the entire system in relationship to external entities.

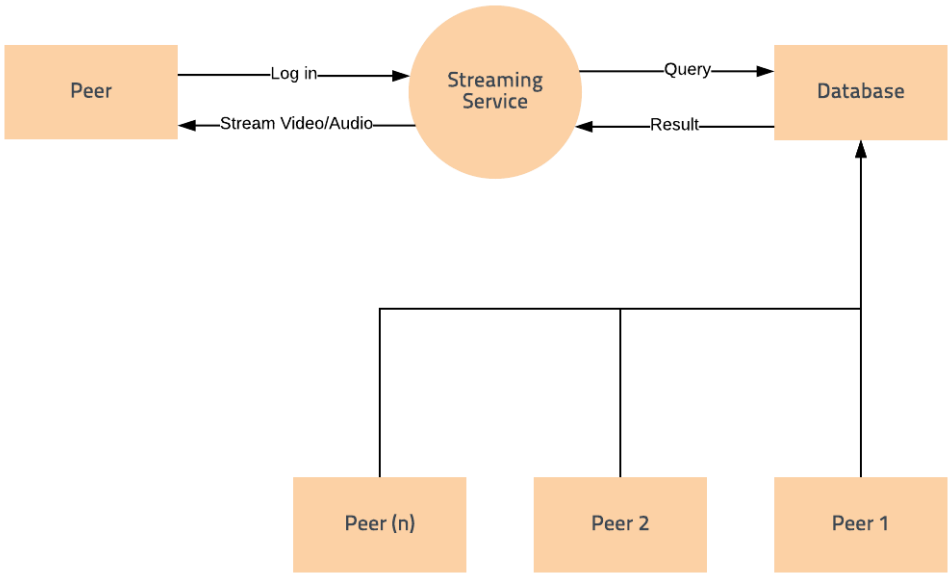

Fig 5.1 Data Flow Diagram Level 0

5.1.2 Level 1:

The **Level 1 DFD** shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole.

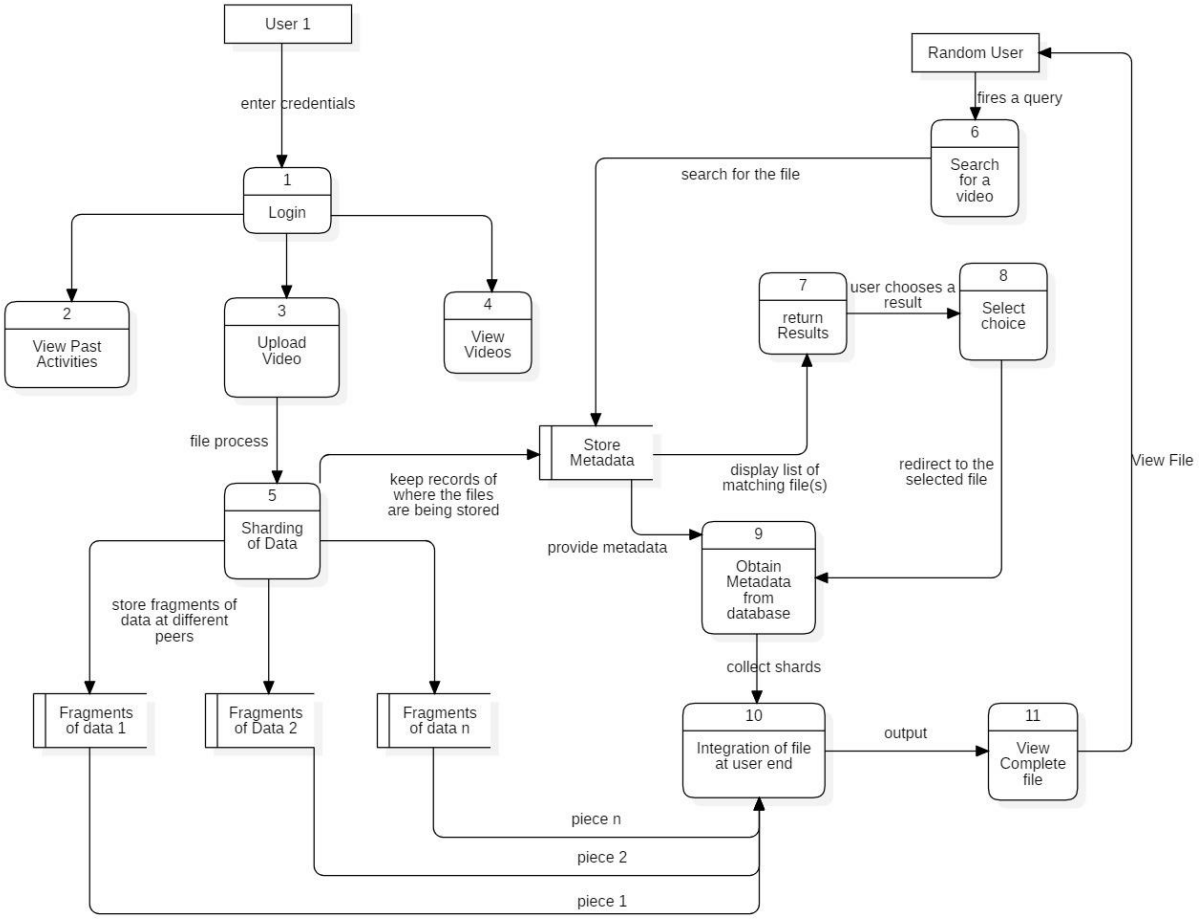

Fig 5.2 Data Flow Diagram Level 1

5.1.3. Level 2:

A **level 2 data flow diagram** (**DFD**) offers a more detailed look at the processes that make up an information system than a **level** 1 **DFD** does. It can be used to plan or record the specific makeup of a system. You can then input the particulars of your own system.

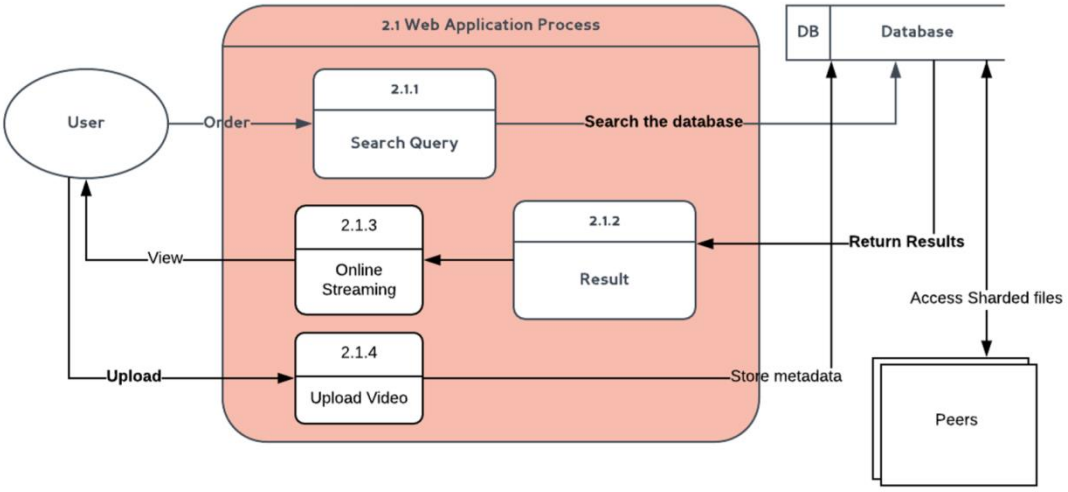

Fig 5.3: Data Flow Diagram Level 2

5.2 UML DIAGRAMS

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

5.2.1 Use Case Diagram

Use case diagrams are usually referred to as behavior **diagram used** to describe a set of actions (**use** cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors).

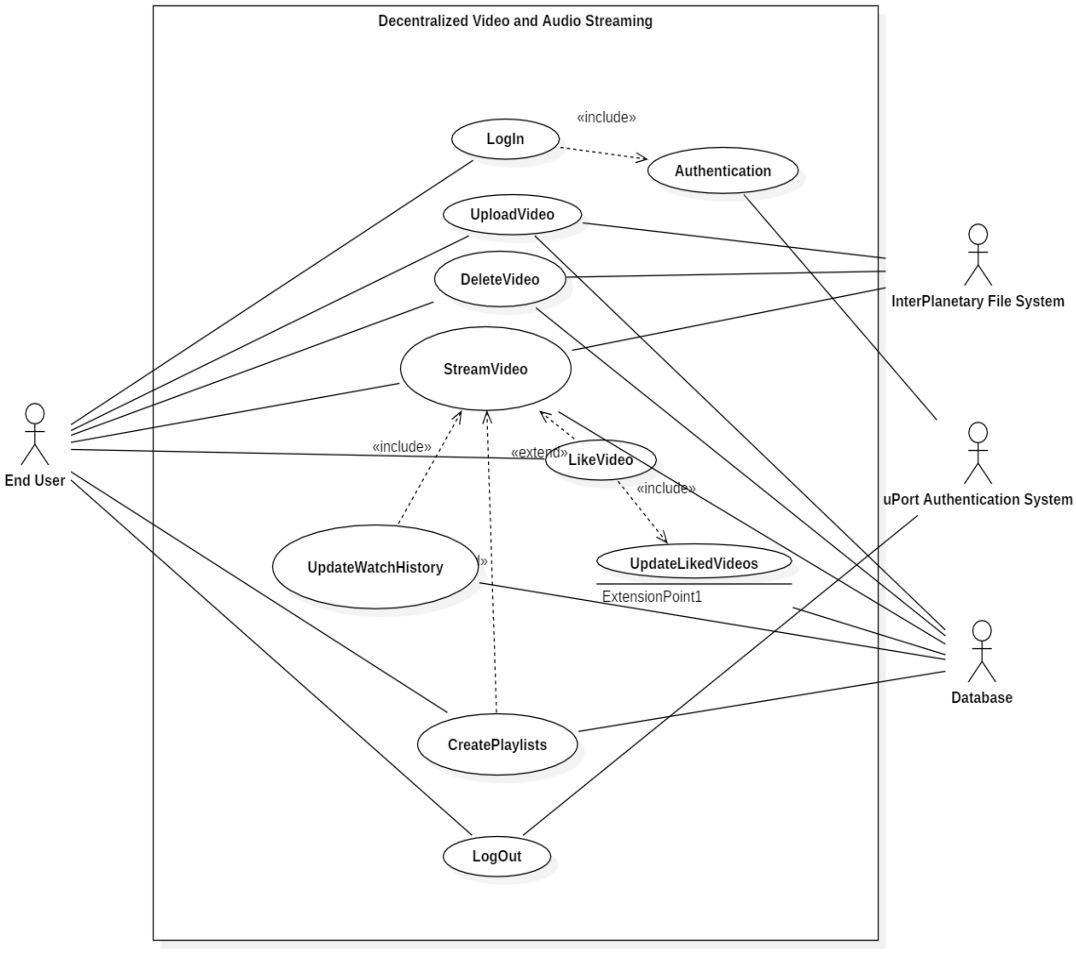

Fig. 5.4 Use Case Diagram

5.2.2 Class Diagram

In software engineering, a **class diagram** in the Unified Modeling Language (UML) is a type of static structure **diagram** that describes the structure of a system by showing the system's **classes**, their attributes, operations (or methods), and the relationships among objects.

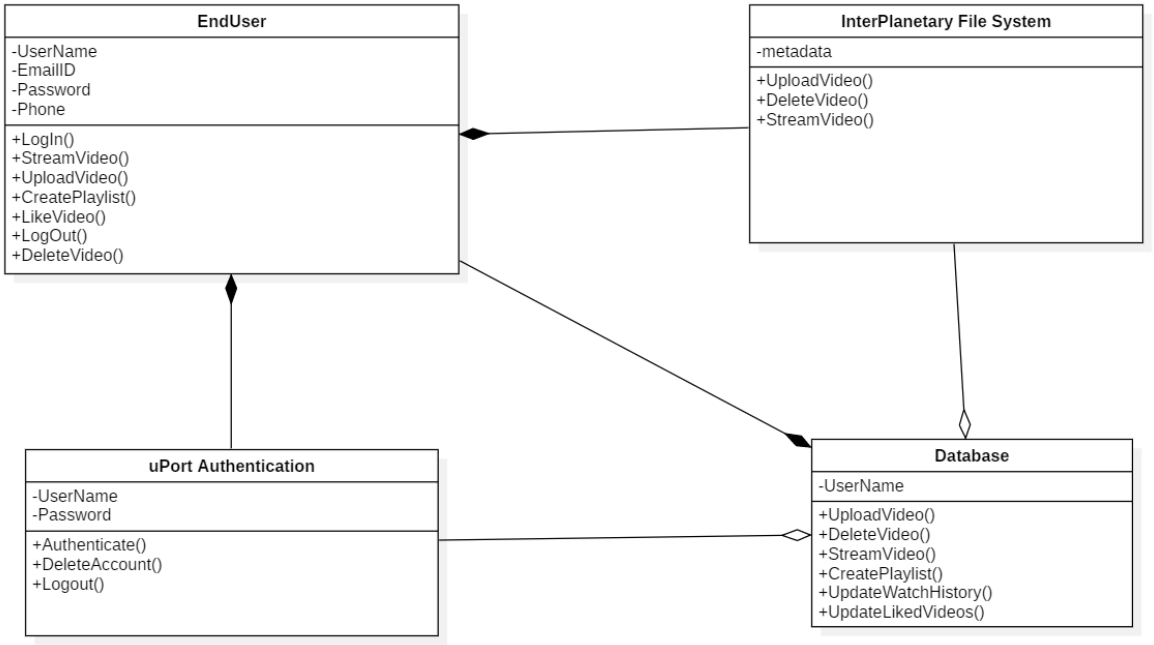

Fig. 5.5 Class Diagram

5.2.3 Sequence Diagram

Sequence diagrams are sometimes called event **diagrams** or event scenarios. A **sequence diagram** shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur.

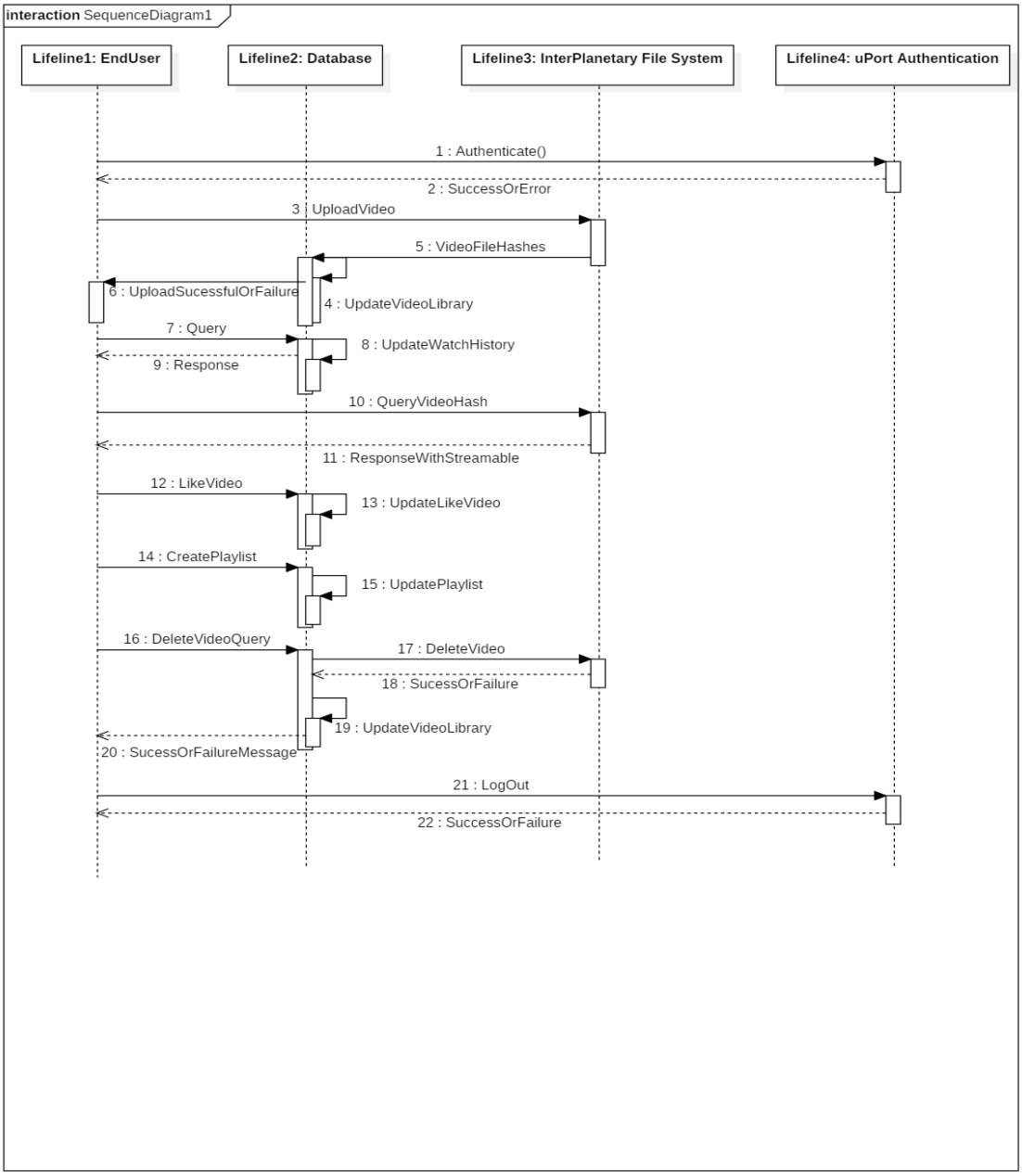

Fig 5.6 Sequence Diagram

5.2.4 State chart Diagram

It's a behavioral diagram and it represents the behavior using finite state transitions. State diagrams are also referred to as State machines and State-chart

Client Diagram:

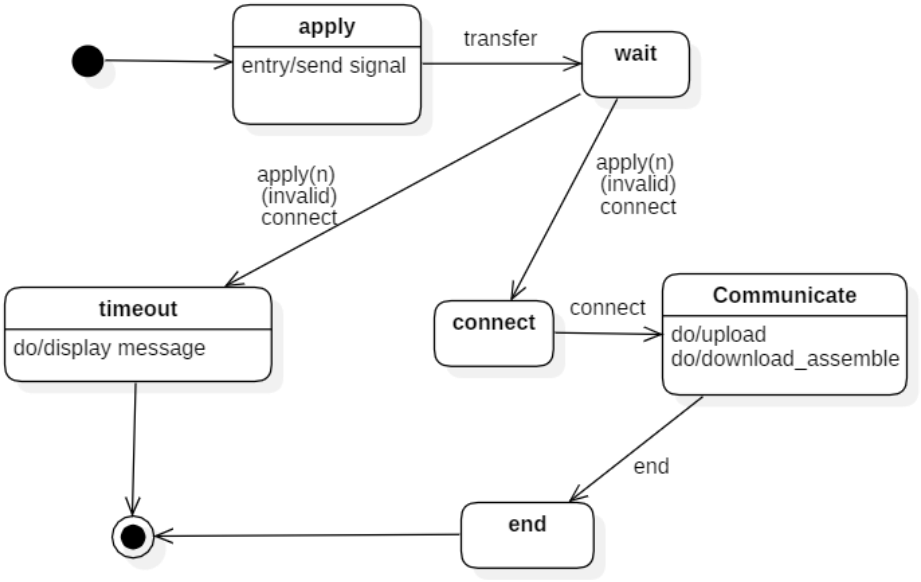

Fig 5.7 State Chart Client Diagram

Server Diagram:

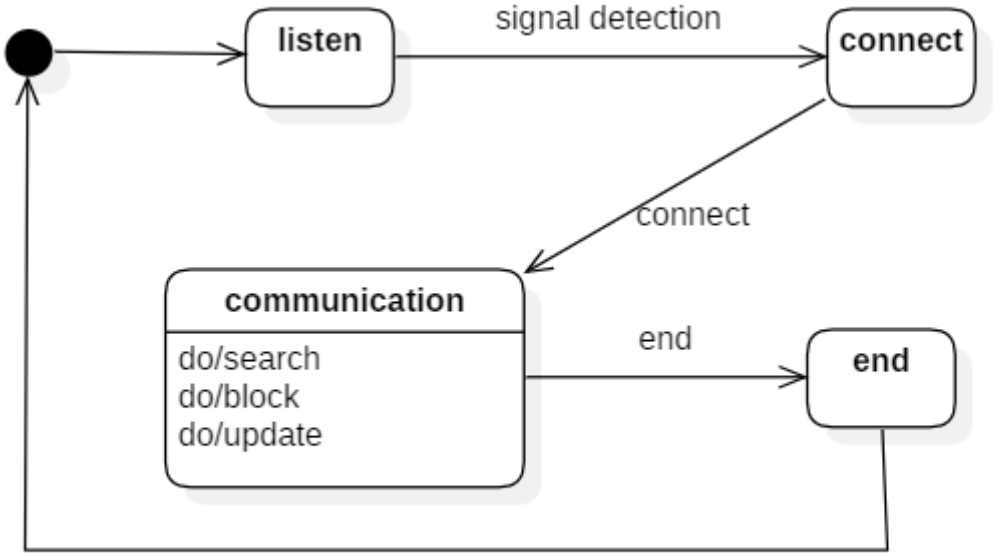

Fig 5.8 State Chart Server Diagram

5.2.5 Activity Diagram

Activity diagram is another important **diagram** in UML to describe the dynamic aspects of the system. **Activity diagram** is basically a flowchart to represent the flow from one **activity** to another **activity**. The **activity** can be described as an operation of the system. The control flow is drawn from one operation to another.

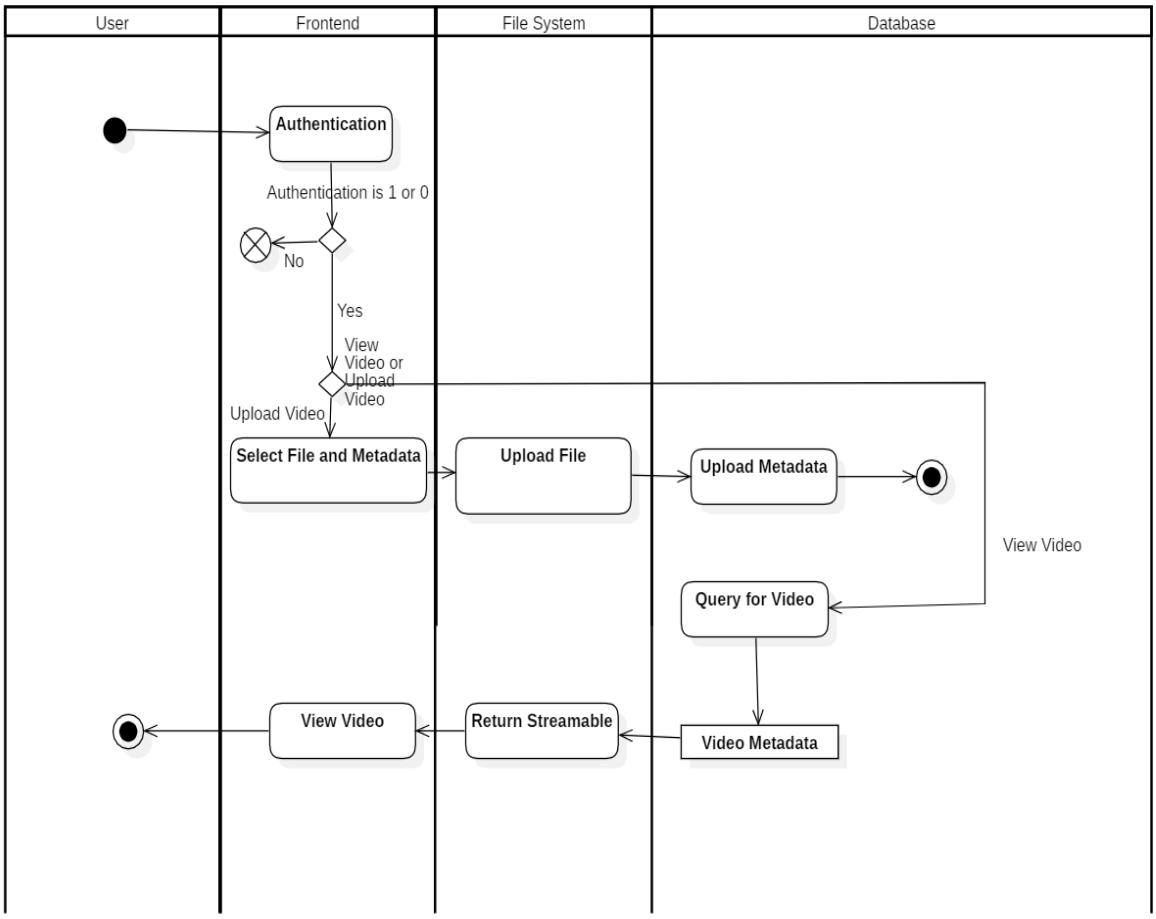

Fig 5.9 Activity Diagram

5.2.6 Collaboration Diagram

A **collaboration diagram**, also called a communication **diagram** or interaction **diagram**, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). The relationships between the objects are shown as lines connecting the rectangles.

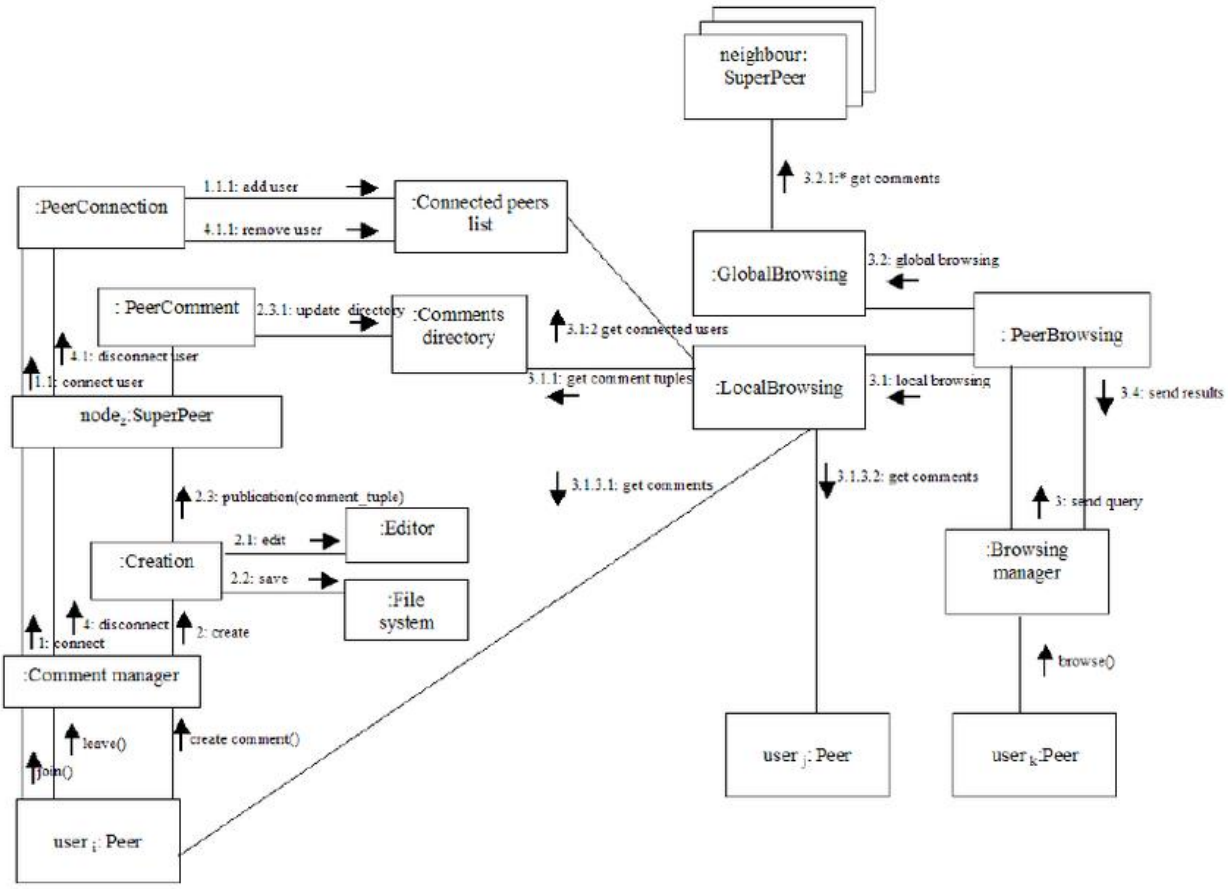

Fig 5.10 Collaboration Diagram

5.3 Planning and Scheduling

PERT Chart: A PERT is a project management tool used to schedule, organize, and coordinate tasks within a project. PERT stands for Program Evaluation Review Technique, a methodology developed by the US Navy in the 1950s to manage the Polaris submarine missile program.

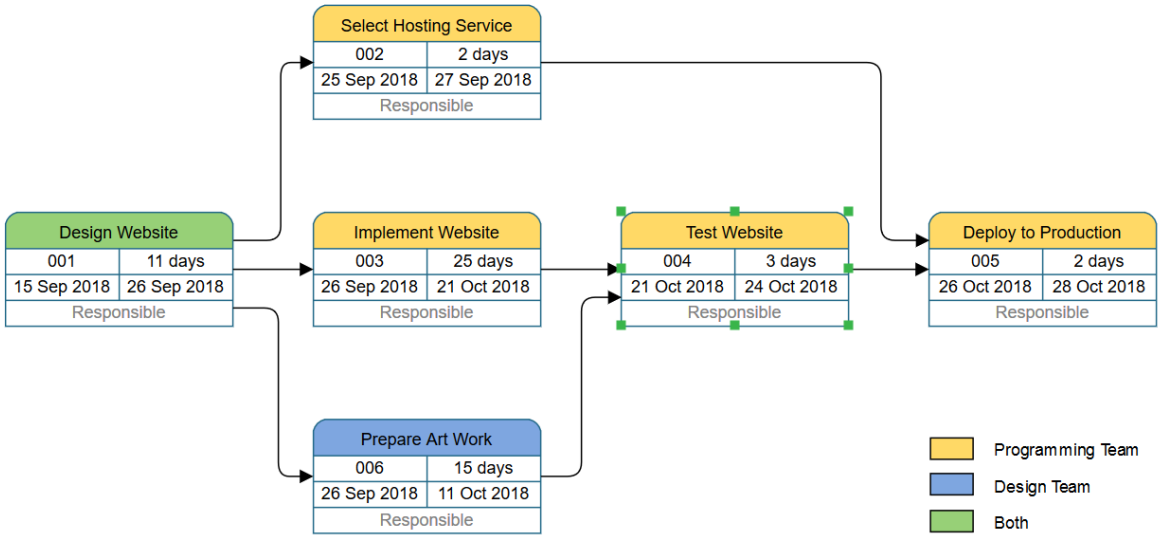

Fig 5.11 PERT (A)

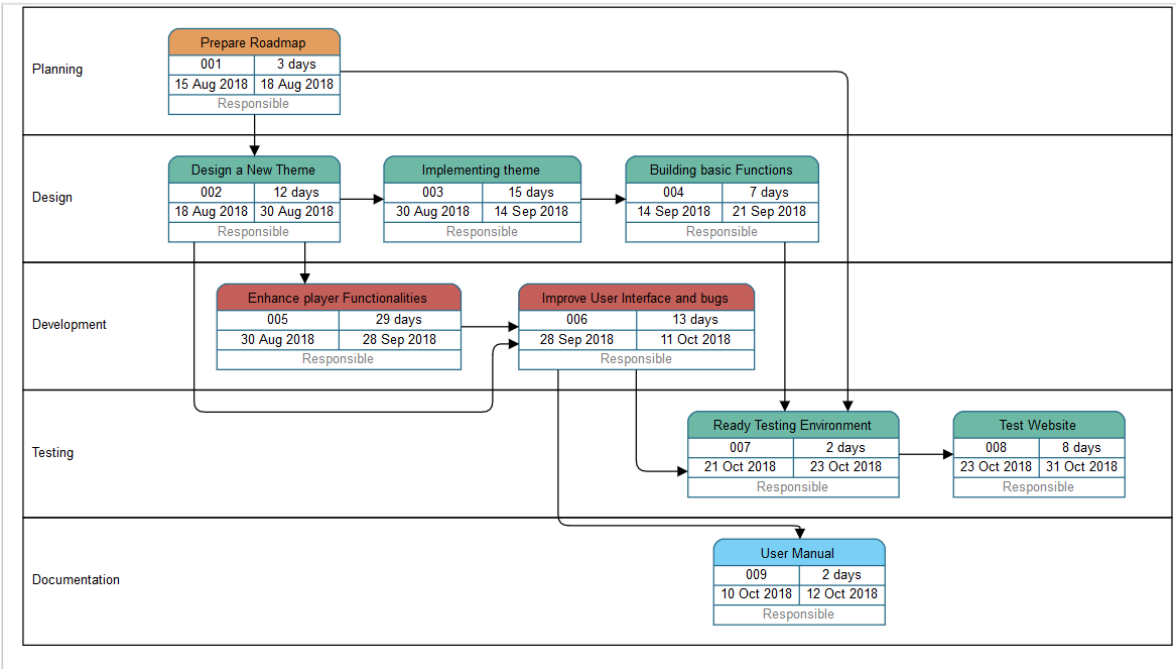

Fig 5.12 PERT (B)

CHAPTER 6 CONCLUSION

With blockchain, we can imagine a world in which contracts are embedded in digital code and stored in transparent, shared databases, where they are protected from deletion, tampering, and revision. In this world every agreement, every process, every task, and every payment would have a digital record and signature that could be identified, validated, stored, and shared. Intermediaries like lawyers, brokers, and bankers might no longer be necessary. Individuals, organizations, machines, and algorithms would freely transact and interact with one another with little friction. This is the immense potential of blockchain.

Thus, we have understood what the new technology is, and how it works. We understood its principles and mechanism of working. It is safe to say that blockchain is the technology for tomorrow. It can effectively help in decentralizing the net and providing better security and faster transfer of data across the new internet.

REFERENCES

[1] Malik Muhammad Imran Pattal, Li Yuan, Zeng Jianqiu, ―Web 3.0: A real personal Web‖, IEEE, Third International Conference on Next Generation Mobile Applications, Services and Technologies, pp. 125128, 2009.

[2] Keshab Nath, Sourish Dhar, Subhash Basishtha, ―Web 1.0 to Web 3.0 - Evolution of the Web and its Various Challenges‖, IEEE, International Conference on Reliability, Optimization and Information Technology pp. 86 – 89, 2014.

[3] J. Blackburn, K. Christensen, ‖A Simulation Study of a New Green BitTorrent,‖ IEEE, Proc. Green Communications Workshop in conjunction with IEEE ICC'09, 2009.

[4] Olaf Landsiedel, Stefan Gotz, Klaus Wehrle, ―Towards Scalable Mobility in Distributed Hash Tables‖ IEEE, Peer-to-Peer Computing, 2006.

[5] Ling Zhong, Xiofan Wang, Maria Kihl, ―Topological Model and Analysis of the P2P BitTorrent Protocol‖, IEEE, Proceedings of the 8th World Congress on Intelligent Control and Automation, pp. 754758, 2011.

[6] Fabius Klemm, Sarunas Girdzijauskas, Jean-Yves Le Boudec, Karl Aberer, ―On Routing in Distributed Hash Tables‖, IEEE, 7th International Conference on Peer-to-Peer Computing, pp. 113-120, 2007.

[7] Lakshmi Siva Sankar, Sindhu M, M. Sethumadhavan, ―Survey of Consensus Protocols on Blockchain Applications‖, IEEE, International Conference on Advanced Computing and Communication Systems, 2017.

[8] Donhee Han, Hongjin Kim, Juwook Jang, ―Blockchain based Smart Door Lock System‖, IEEE, Information and Communication, pp. 1165, 2017.

[9] Jatinder Singh, John David Michels, ―Blockchain as a Service (Baas): Providers and Trust‖, IEEE, European Symposium on Security and Privacy Workshops, pp. 1165, 2018. [10] Deepak K. Tosh, Sachin Shetty, Xueping Liang, ―Consensus Protocols for Blockchain-based Data Provenance: Challenges and Opportunities‖, IEEE, 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), pages 469–474, 2017.

[11] Sachchidanand Singh, Nirmala Singh, ―Blockchain: Future of Financial and Cyber Security‖, IEEE, 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), pp. 463–467, 2016.

[12] Dejan Vujičić, Dijana Jagodić, Siniša Ranđić , ―Blockchain Technology, Bitcoin, and Ethereum: A Brief Overview‖, IEEE, 17th International Symposium INFOTEH-JAHORINA, 2018.

[13] Konstantinos Christidis, and, Michael Devetsikiotis, ―Blockchains and Smart Contracts for the Internet of Things‖, IEEE Access, Special Section on the Plethora of Research in Internet of Things, 2016.

[14] Sreehari P , M Nandakishore, Goutham Krishna, ―SMART WILL: Converting the Legal Testament into a Smart Contract‖, IEEE, International Conference on Networks & Advances in Computational Technologies, pp. 203-207, 2017.

[15] Satoshi Nakamoto, ―Bitcoin: A Peer-to-Peer Electronic Cash System‖, White Paper, 2008.

[16] David Mazieres, ―Self Certifying File System‖, Doctor of Philosophy, Massachusetts Institute of Technology, Massachusetts, USA, 2000.

[17] Juan Benet, ―IPFS - Content Addressed, Versioned, P2P File System (Draft 3)‖, White Paper, 2014.

[18] Jiin-Chiou Cheng, Narn-Yih Lee , Chien Chi , and Yi-Hua Chen, ―Blockchain and Smart Contract for Digital Certificate‖, IEEE, Proceedings of IEEE International Conference on Applied System Innovation, 2018.

[19] R. G. Shirey, K. M. Hopkinson, K. E. Stewart, ―Analysis of implementations to secure git for use as an encrypted distributed version control system", IEEE, 48th Hawaii International Conference on System Sciences, pp. 5310–5319, 2015.

[20] Ruchika Malhotra, Nakul Pritam, Kanishk Nagpal, "Defect Collection and Reporting System for Git based Open Source Software", IEEE, International Conference on Data Mining and Intelligent Computing (ICDMIC) 2014.

[21] HaeJun Lee, Bon-Keun Seo, Euiseong Seo, "A Git Source Repository Analysis Tool Based on a Novel Branch-oriented Approach", IEEE, International Conference on Information Science and Applications (ICISA), 2013.

[22] Nida Khan , Abdelkader Lahmadi , Jerome Francois and Radu State, ―Towards a Management Plane for Smart Contracts: Ethereum Case Study‖, IEEE, IFIP Network Operations and Management Symposium, 2018.