

**PROJECT REPORT ON**

**ONLINE VIDEO/AUDIO STREAMING SERVICE BASED ON**

**DECENTRALIZED ARCHITECTURE**

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE IN THE  
PARTIAL FULFILLMENT FOR THE AWARD OF THE DEGREE

**OF**

**BACHELOR OF ENGINEERING**

**IN**

**INFORMATION TECHNOLOGY**

**BY**

<b>ISHAN JOSHI</b>	<b>– B150388568</b>
<b>KISHLAYA KUNJ</b>	<b>– B150388586</b>
<b>NEERAJ LAGWANKAR</b>	<b>– B150388597</b>

**UNDER THE GUIDANCE OF**

**PROF. SHAMLA MANTRI**



## **DEPARTMENT OF INFORMATION TECHNOLOGY**

**MIT- COLLEGE OF ENGINEERING**

**Pune, Maharashtra, India**

**2018-19**

## **CERTIFICATE**

This is to certify that the project report entitled

### **ONLINE VIDEO/AUDIO STREAMING SERVICE BASED ON DECENTRALIZED ARCHITECTURE**

#### **Submitted by**

Ishan Joshi – B150388568

Kishlaya Kunj – B150388586

Neeraj Lagwankar – B150388597

is a bonafide work carried out by them under the supervision of Prof. Shamla Mantri and it is approved for the partial fulfillment of the requirement of Savitribai Phule Pune University for the award of the Degree of Bachelor of Engineering  
(Information Technology)

This project report has not been earlier submitted to any other Institute or University for the award of any degree or diploma.

Prof. Shamla Mantri  
Internal Guide  
Department of Information Technology

Dr. Krishna Warhade  
Head of Department  
Department of Information Technology

Name  
External Guide  
Date:

Dr. Krishna Warhade  
Head of Department  
MIT College of Engineering,Pune

Place: MITCOE, Pune  
Date: 17/12/2018

## **ACKNOWLEDGEMENT**

We take this opportunity to thank our project guide Prof. Shamlam Mantri and Head of the Department Dr. Krishna Warhade for their valuable guidance and for providing all the necessary facilities, which were indispensable in the completion of this project report. We are also thankful to all the staff members of the Department of Information Technology of MIT College of Engineering, Pune for their valuable time, support, comments, suggestions and persuasion. We would also like to thank the institute for providing the required facilities, internet access and important books.

**Ishan Joshi  
Kishlaya Kunj  
Neeraj Lagwankar**

## LIST OF FIGURES

<b>S.No</b>	<b>Fig Num</b>	<b>Name of the figure</b>	<b>Page Number</b>
1	2.1	Blockchain	5
2	2.2	Forking and merging of git branches	6
3	2.3	Reading an IPFS block	8
4	2.4	Writing to an IPFS block	8
5	2.5	Read an object stored remotely	9
6	4.1	Data Flow Diagram Level 0	17
7	4.2	Data Flow Diagram Level 1	18
8	4.3	Data Flow Diagram Level 2	19
9	4.4	Use Case Diagram	20
10	4.5	Class Diagram	21
11	4.6	Sequence Diagram	22
12	4.7	State Chart Client Diagram	23
13	4.8	State Chart Server Diagram	23
14	4.9	Activity Diagram	24
15	4.10	Collaboration Diagram	25
16	4.11	PERT chart(A)	26
17	4.12	PERT chart (B)	26
18	5.1	Uploading to IPFS	27
19	5.2	uPort Authentication	28
20	5.3	Obtain user data in Solidity	28
21	5.4	React component that renders the video	29
22	5.5	React component that renders a dashboard	30
23	5.6	Redux Dispatch Function	31
24	6.1	Welcome Page	32
25	6.2	Upload Page	32
26	6.3	User Profile Page	33
27	6.4	Video Streaming Page	33

## III

## NOMENCLATURE

1	DoS	Denial of Service
2	DDoS	Distributed Denial of Service
3	DAG	Directed Acyclic Graph
4	DApp	Decentralized App
5	P2P	Peer to peer
6	DHT	Distributed Hash Table
7	IP	Internet Protocol
8	IPFS	InterPlanetary File System
9	SFS	Self-Certifying File System
10	PoW	Proof of Work
11	DPoS	Delegated Proof of Stake
12	PoA	Proof of Activity
13	PoS	Proof of Stake
14	LLL	Lisp – like – Language
15	HTTP	Hyper Text Transfer Protocol
16	CSS	Cascading Style Sheet
17	NPM	Node Package Manager
18	iOS	Apple Operating System
19	QR	Quick Response
20	JSX	JavaScript Extended
21	URL	Uniform Resource Loader
22	DFD	Data Flow Diagram
23	UML	Unified Modeling Language
24	PERT	Program Evaluation Review Technique

## IV

## CONTENTS

<b>CERTIFICATE</b>	<b>I</b>
<b>ACKNOWLEDGEMENT</b>	<b>II</b>
<b>LIST OF FIGURES</b>	<b>III</b>
<b>NOMENCLATURE</b>	<b>IV</b>

<b>CHAPTER</b>	<b>TITLE</b>	<b>PAGE NO.</b>
<b>1.</b>	<b>INTRODUCTION</b>	<b>01</b>
1.1.	BACKGROUND	01
1.2.	RELEVANCE	01
1.3.	PROJECT UNDERTAKEN	02
1.4.	ORGANIZATION OF PROJECT REPORT	03
<b>2.</b>	<b>BACKGROUND</b>	<b>04</b>
2.1.	LITERATURE SURVEY	04
2.2.	EXISTING TECHNOLOGIES	07
2.3.	TOOLS FOR DEVELOPMENT	10
2.4.	RELATED WORK	11
<b>3.</b>	<b>SPECIFICATION</b>	<b>12</b>
3.1.	PROBLEM STATEMENT	12
3.2.	REQUIREMENT SPECIFICATION	12
3.3.	HARDWARE AND SOFTWARE SPECIFICATIONS	16
<b>4.</b>	<b>DESIGN</b>	<b>17</b>
4.1.	DATA FLOW DIAGRAMS	17
4.1.1.	LEVEL 0	17
4.1.2.	LEVEL 1	18
4.1.3.	LEVEL 2	19
4.2.	UML DIAGRAMS	20
4.2.1.	USE CASE DIAGRAM	20
4.2.2.	CLASS DIAGRAM	21
4.2.3.	SEQUENCE DIAGRAM	22
4.2.4.	STATECHART DIAGRAM	23
4.2.5.	ACTIVITY DIAGRAM	24
4.2.6.	COLLABORATION DIAGRAM	25



4.2.7.	PERT CHART	26
<b>5.</b>	<b>IMPLEMENTATION</b>	<b>27</b>
<b>6.</b>	<b>RESULTS AND EVALUATION</b>	<b>32</b>
6.1.	WELCOME PAGE	32
6.2.	UPLOAD PAGE	32
6.3.	USER PROFILE PAGE	33
6.4.	VIDEO UPLOAD PAGE	33
<b>7.</b>	<b>CONCLUSIONS AND FUTURE WORK</b>	<b>34</b>
7.1.	CONCLUSION	34
7.2.	FUTURE WORK	34
	<b>REFERENCES</b>	<b>35</b>

## **CHAPTER 1**

## **INTRODUCTION**

### **1.1. BACKGROUND**

The current video streaming systems have some shortcomings which can be improved to provide a seamless way to the users to interact with the website and enjoy fast and secure browsing experience. Some of the shortcomings are:

#### **A. 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.

#### **B. 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.

#### **C. 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. RELEVANCE**

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 have learnt during our time in college, this project is something that inculcates all the attributes of almost every aspect of Information Technology Engineering.

**Aim of this project is:**

- To understand the concept of decentralized network.
- To understand the concept behind block chain.
- To implement smart contracts using Solidity.

### **1.3. PROJECT UNDERTAKEN**

In our project, we aim to eliminate or reduce the problems explained in the above section. This will greatly improve the user experience and the concept of decentralized internet will be implemented successfully. The solution to the above mentioned problems are:

- Cyber-attacks like DoS/DDoS would be redundant against Web 3.0 due its structure. Since, the data is never stored at one location/machine, the attack can be made only on a single system, which can be detected and necessary actions can be taken, without affecting the network.
- The decentralized nature of data also makes the network much more secure. This is due to the fact that current exploits would be rendered useless, due to the group of technologies constituting Web 3.0. Thus, manipulating data or illegal access of data would be impossible, making the system secure, and free of plagiarized content.
- In our proposed system, the number of users will only strengthen the network, instead of degrading it. Video and audio streaming service using decentralized architecture which is based on technologies such as Peer to Peer networks, Merkle DAGs, and, Blockchain, which would reduce the cost of service since there would be no need of renting, upgrading or maintaining servers. It would provide media content faster to end user. The inherent security of the network would also render conventional exploits useless, adding to the robustness of the system. Thus, providing a fast, secure and robust decentralized app (DApp) for users.

#### 1.4. ORGANIZATION OF PROJECT REPORT

This report covers all the necessary details regarding the project. The chapters have been arranged systematically so that users can understand the framework step by step.

- The **Background** chapter deals with the back-end of the system, providing an insight to what happens behind-the-scenes.
- The **Specification** chapter deals with the hardware and software specifications required to run the application smoothly.
- The **Design** chapter deals with the entire architectural design of the application.
- The **Implementation** chapter deals with our steps to put the project into execution.
- The **Results and Evaluation** chapter deals with the tests carried out by us and their corresponding results, and further evaluating the performance of the application developed.
- The **Conclusions and Future Works** chapter wraps up the project in a nutshell.

## **CHAPTER 2 BACKGROUND**

The fruition of any project is always achieved by extensive research and information provided by those who paved way for us through their insight into various technologies.

### **2.1. LITERATURE SURVEY**

A decentralized web will be able to solve the problems faced by any service based on client server architecture. In the last decade, advancement in web technology has led to the concept of decentralized network, thus allowing the rise of peer-to-peer communication. Peer-to-Peer communication circumvents this problem by relaying traffic through peers instead of a dedicated server.

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.

#### **A. 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 needs limited infrastructure and control, it is vastly superior, faster, more secure and robust than the existing client server architecture [2, 3].

#### **B. 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].

### C. 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.

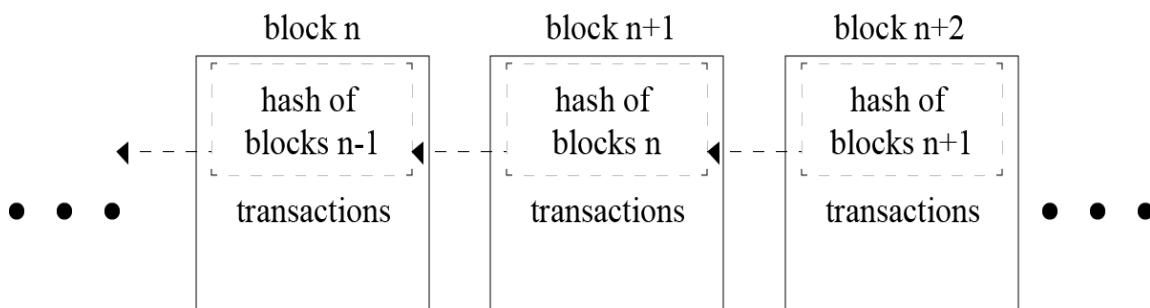


Fig. 2.1 Blockchain

### D. 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].

### E. 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].

## F. 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].

## G. 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 2.2 Forking and merging of Git branches

## 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.

### A. 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].

### B. 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 [13, 11]. 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 3 depicts the creation of a new object: the client sends its object to any node on its nearest site [20].

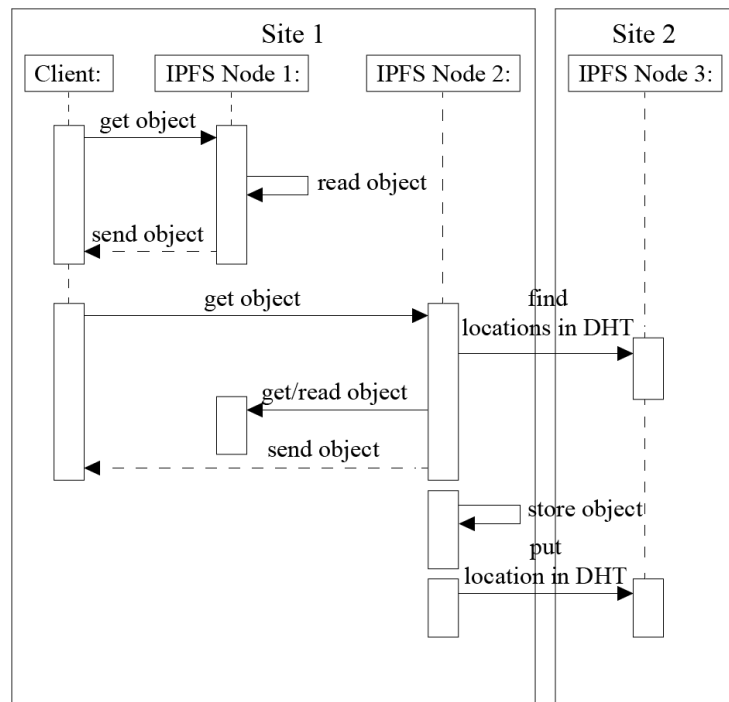


Fig. 2.3 Reading an IPFS Block

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 [20]. Figure 4 illustrates what happens when the client reads an object stored on its local site [20].

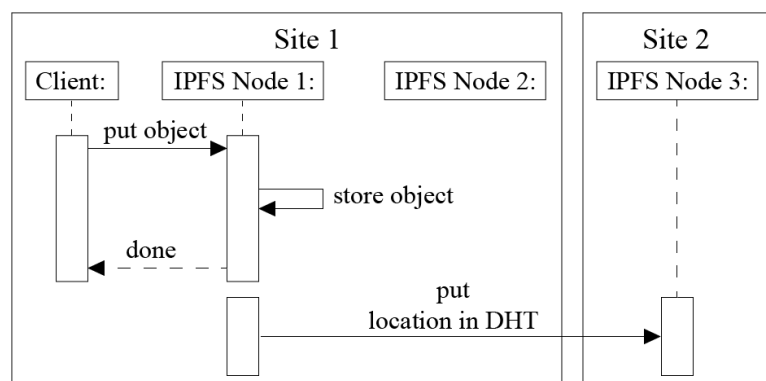


Fig. 2.4 Writing a block to IPFS

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 5 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].

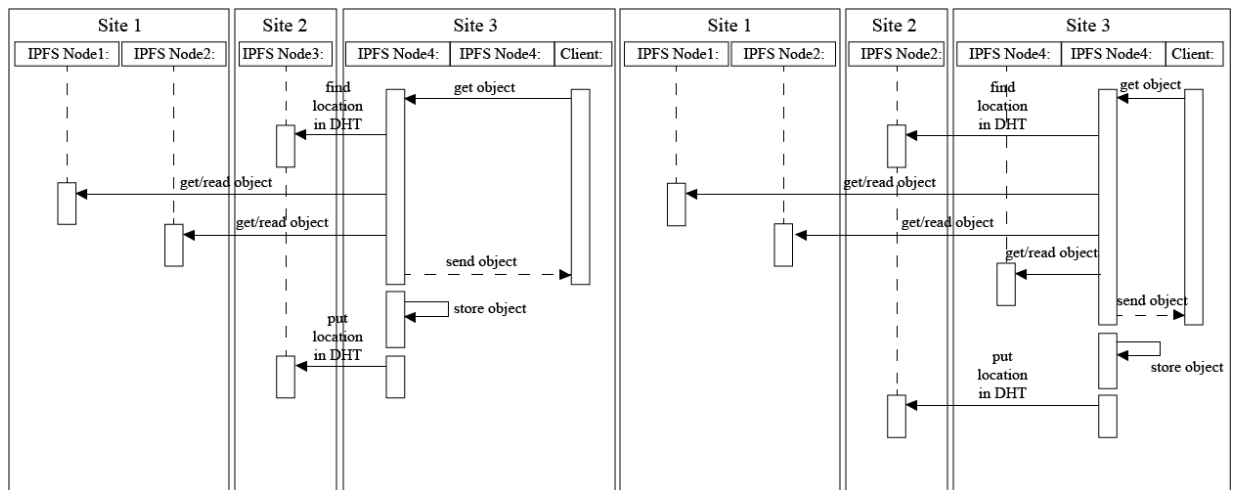


Fig 2.5 Read an object stored remotely

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].

## 2.3 TOOLS FOR DEVELOPMENT

Some of the tools used in the development of the project are:

- **HTML, CSS, JavaScript:** The basic web development languages have been used to design the front-end of the application. The web page also acts as a link between the user and the back end, fulfilling the requests made by the user.
- **React:** React is a JavaScript library for building user interfaces. It is maintained by Facebook and a community of individual developers and companies. It makes the web page responsive without needing to reload the entire content again.
- **Blockchain:** Blockchain serves as the backbone of our application. It supports the video streaming service that we are implementing through our project.
- **Solidity:** This is an Ethereum based programming language used for the development of smart contracts.
- **IPFS:** InterPlanetary File System is a protocol and network designed to create a content-addressable, peer-to-peer method of storing and sharing hypermedia in a distributed file system.
- **NPM:** NPM is a package manager for the JavaScript programming language.

## 2.4. RELATED WORKS

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

### 1. 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 blockchain database. In the process, they were the first to solve the double spending problem for digital currency using a peer-to-peer network.

## **2. V. Buterin, Ethereum**

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.

## **3. 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.

## **CHAPTER 3 SPECIFICATION**

### **3.1. PROBLEM STATEMENT**

To develop an application program for blockchain based audio and video streaming services, inculcating the concept of peer- to-peer communication and decentralized architecture.

#### **Project objectives:**

- To develop video and audio streaming service with minimum failure points.
- To understand how the blockchain works.
- To implement an algorithm to achieve minimum time and space complexity.

### **3.2. REQUIREMENT SPECIFICATION**

#### **3.2.1 Purpose**

The purpose of the document is to give a detailed description of Online Video and Audio Streaming Service Based on Decentralized Architecture. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. Users will be able to stream videos and audios via our network which are provided by and for the users.

#### **3.2.2 Intended Audience and Reading Suggestions**

The intended audiences of stakeholders for this specification of the Online Video and Audio Streaming Service Based on Decentralized Architecture include:

- User who is interested in listening music or watching videos
- Content creators who want to showcase their talent and upload their creations

#### **3.2.3 Product Scope**

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). The user should be able to sign up/sign in to the network using a secure authentication system. The user should also have the ability

to like or dislike a video/audio. The user should be able to upload his/her own content to the DApp system. The more the users are engaged in the system, stronger the network becomes. 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.

### **3.2.1 Overall Description**

#### **3.2.1.1 Product Perspective**

Our streaming service deals with streaming of videos and audios based on decentralized architecture. Users who are connected to the network will contribute to the system while watching video or listening to audios by simultaneously uploading some part of the media they are consuming.

#### **3.2.1.2 Classes and Characteristics**

The **User** has following objectives and characteristics:

- The user will have to enter his/her name, mobile number and an avatar to have a unique identity.
- The user will enter these credentials while signing up for the first time.
- The next time, the user will have to scan a QR code to log in.
- The user will also be able to update and delete his/her profile at any time.

The **Video** has following characteristics:

- The video will have a title and a description.
- It will also have a genre which will further tell the user, the type of media he/she is consuming.
- After the following criteria are met, the user then can upload the videos to watch.

The **Playlist** characteristics:

- The user will be able to create a playlist of media he/she likes.
- The user can then edit the playlist by adding or deleting media.
- The user can even create multiple playlists and delete the ones which are not needed.

### **3.2.5 Data and Content Constraints**

#### **3.2.5.1 Database**

- BigchainDB

#### **3.2.5.2 Hardware Constraints**

- None

#### **3.2.5.3 Technologies Used**

- JavaScript
- JSX
- HTML
- CSS

### **3.2.6 User Documentation**

- Type the system URL in the browser.
- The user should scan the QR code to sign in.
- The user can view the details in the profile section.
- The information filled by the user can be submitted by pressing on the submit button.

### **3.2.7 Assumptions and Dependencies**

#### **3.2.7.1 Assumptions**

- The users are assumed to have working knowledge of how the browsers work.

- The users are also assumed to have preliminary knowledge of filling forms and submitting information on the internet.

#### **3.2.7.2 Client Hardware:**

A compatible operating system with sufficient storage space.

#### **3.2.7.3 Networks:**

- **Internet**, which is the global network used for communication.
- **Browser**, which is the software tool that runs on the user's computer to communicate via internet.

### **3.2.8 Other Nonfunctional Requirements**

#### **3.2.8.1 Performance Requirements**

- There should be good internet connection to use system seamlessly.
- The unauthorized user is not permitted to use system.
- The system shouldn't lag on low performance machines.

#### **3.2.8.2 Safety Requirements**

- An unauthorized user is not permitted to use the system.
- Being decentralized it is not possible to hack the system, as there is no centralized server.

### **3.2.9 Interoperability**

The decentralized streaming service shall interoperate with the following browsers:

- Mozilla Firefox
- Google Chrome

### **3.2.10 Maintainability**

The system shall permit some authorized users to maintain the repository.



### **3.2.11 Envisioned Future Enhancements**

- There will be a recommendation algorithm based on user's likes to recommend which videos to watch next.
- There will also be an Android or iOS app based on React-Native which will enable users to access the website from their phones.

## **3.3. HARDWARE AND SOFTWARE REQUIREMENTS**

### **3.3.1 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

### **3.3.2 Software**

- **Operating System:** Windows 7/8/10, MacOS, any Linux based OS.
- **Browsers:** Mozilla Firefox, Google Chrome.

## CHAPTER 4

## DESIGN

This chapter deals with the application design. Let us have a look at them.

### 4.1 Data Flow Diagrams:

#### 4.1.1 Level 0:

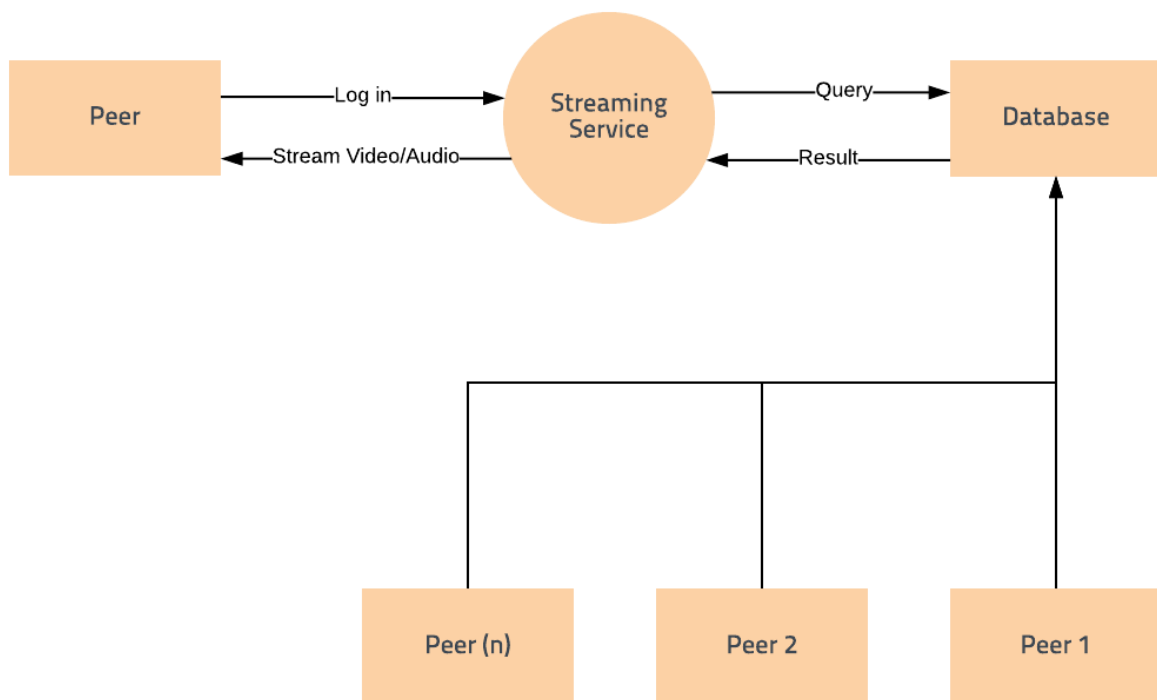


Fig 4.1 Data Flow Diagram Level 0

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.

#### 4.1.2 Level 1:

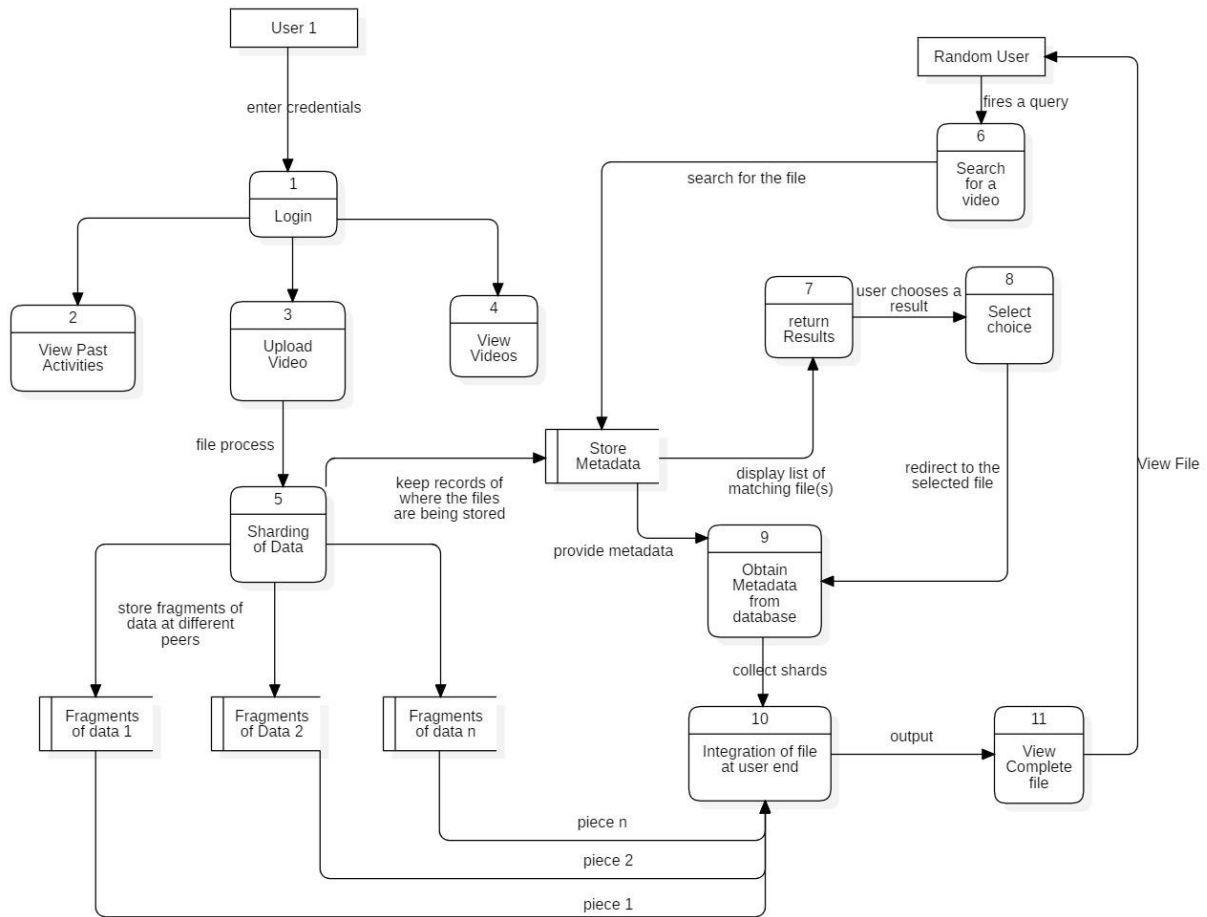


Fig 4.2 Data Flow Diagram Level 1

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.

#### 4.1.3. Level 2:

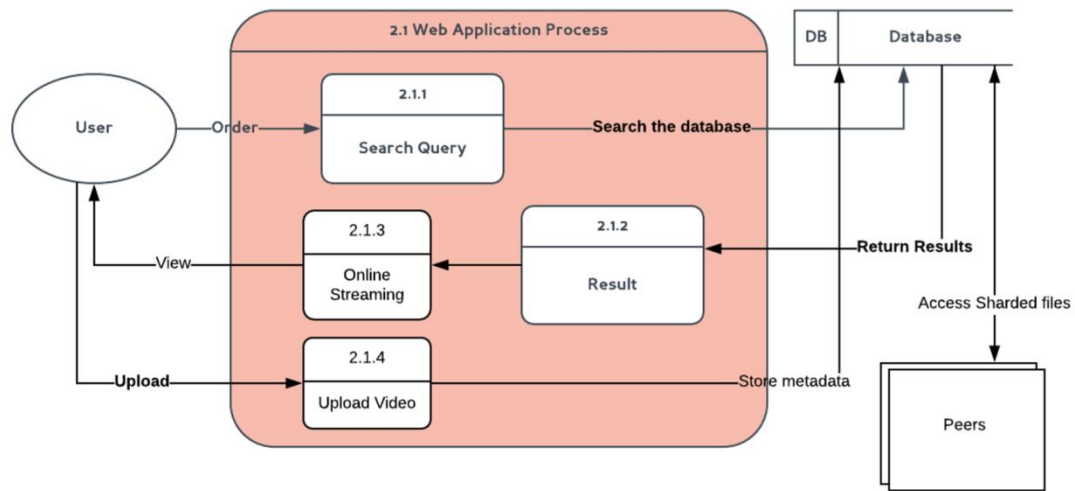


Fig 4.3 Data Flow Diagram Level 2

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.

## 4.2 UML DIAGRAMS

### 4.2.1 Use Case Diagram

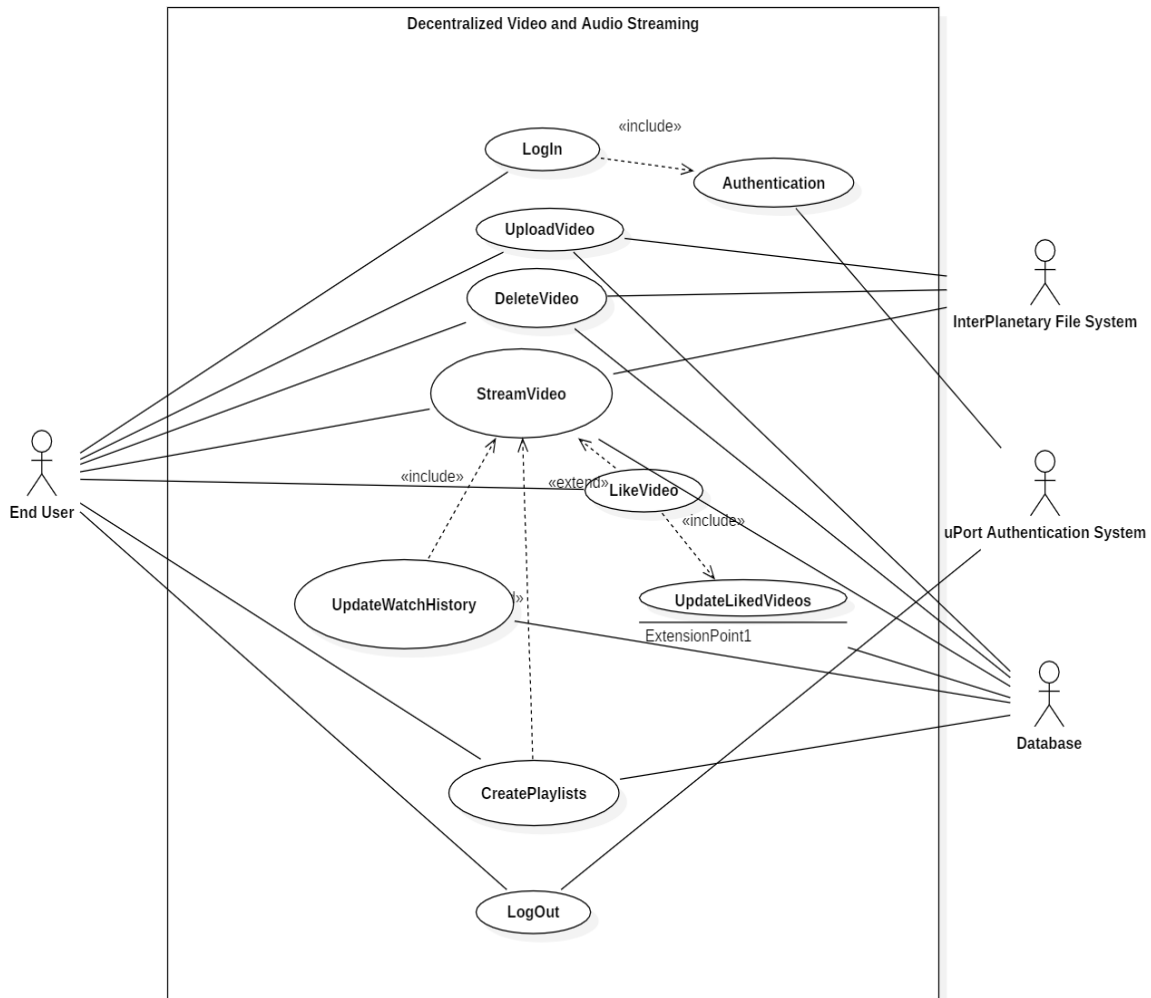


Fig. 4.4 Use Case Diagram

**Use case diagrams** are usually referred to as behavior **diagrams 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).

### 4.2.2 Class Diagram

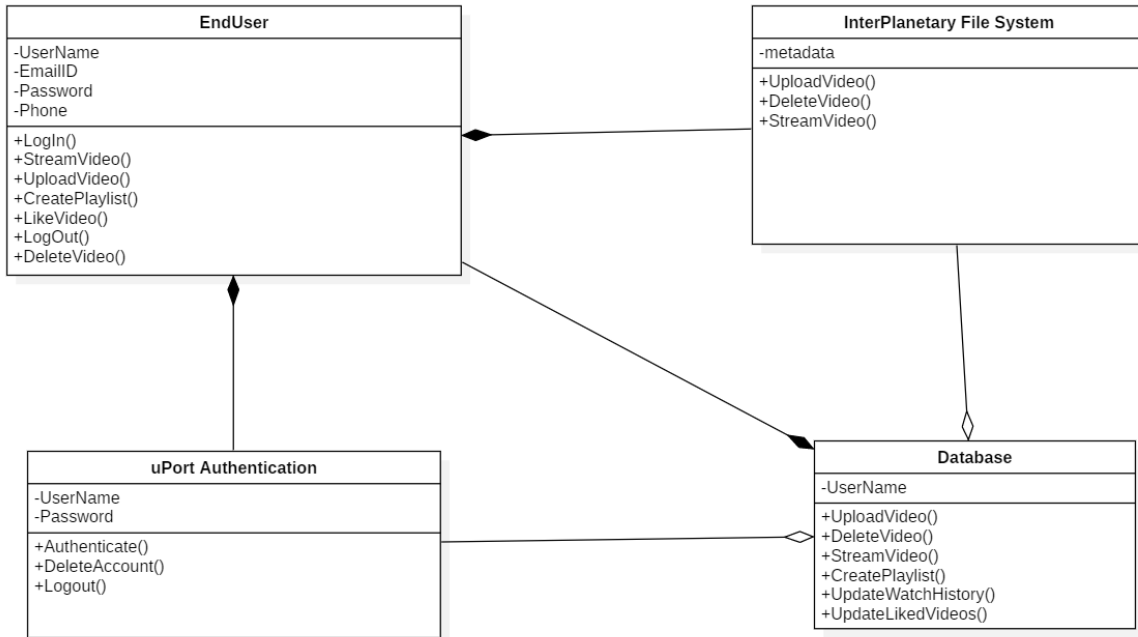


Fig. 4.5 Class Diagram

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.

### 4.2.3 Sequence Diagram

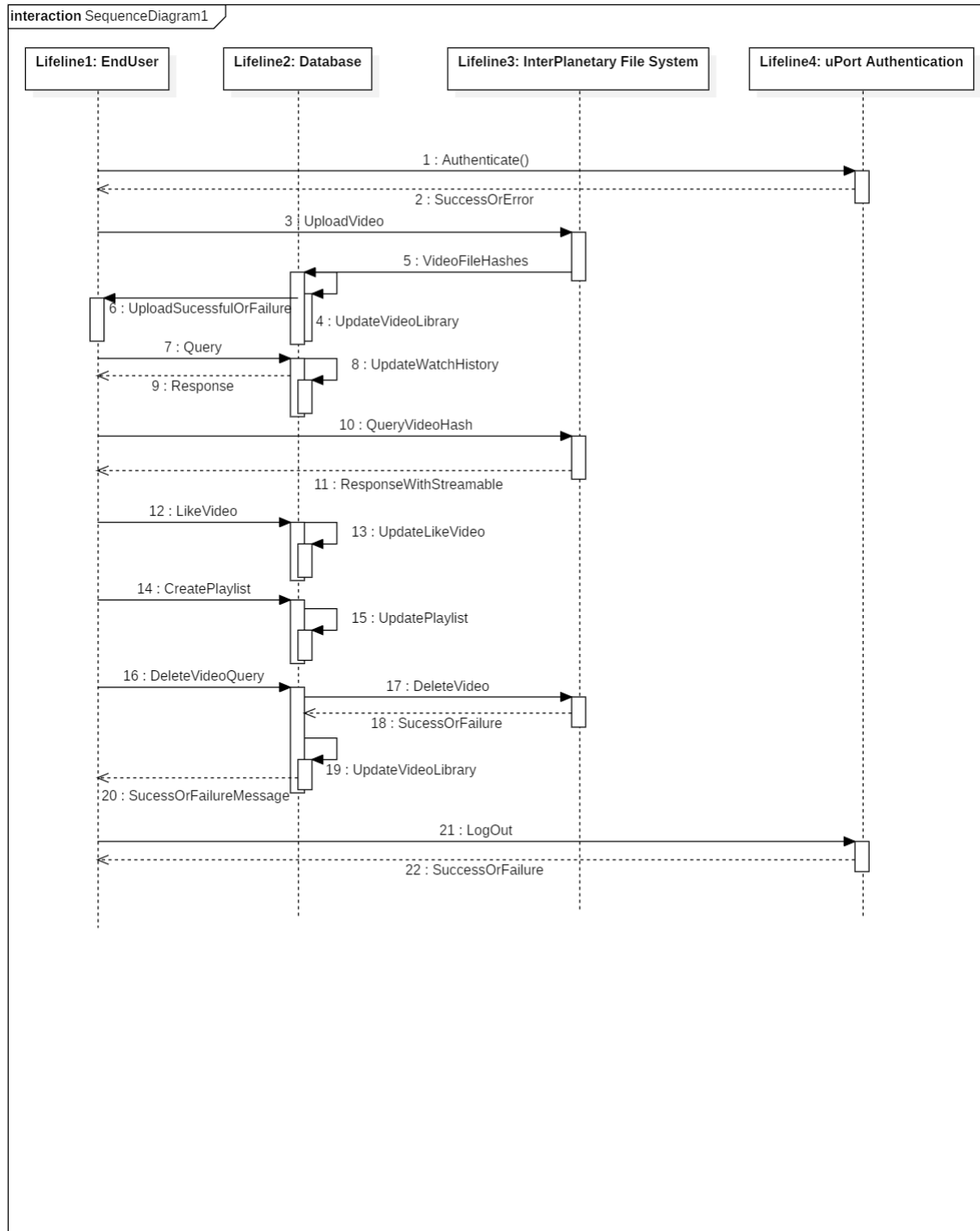


Fig 4.6 Sequence Diagram

**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.

#### 4.2.4 State chart Diagram

##### Client Diagram:

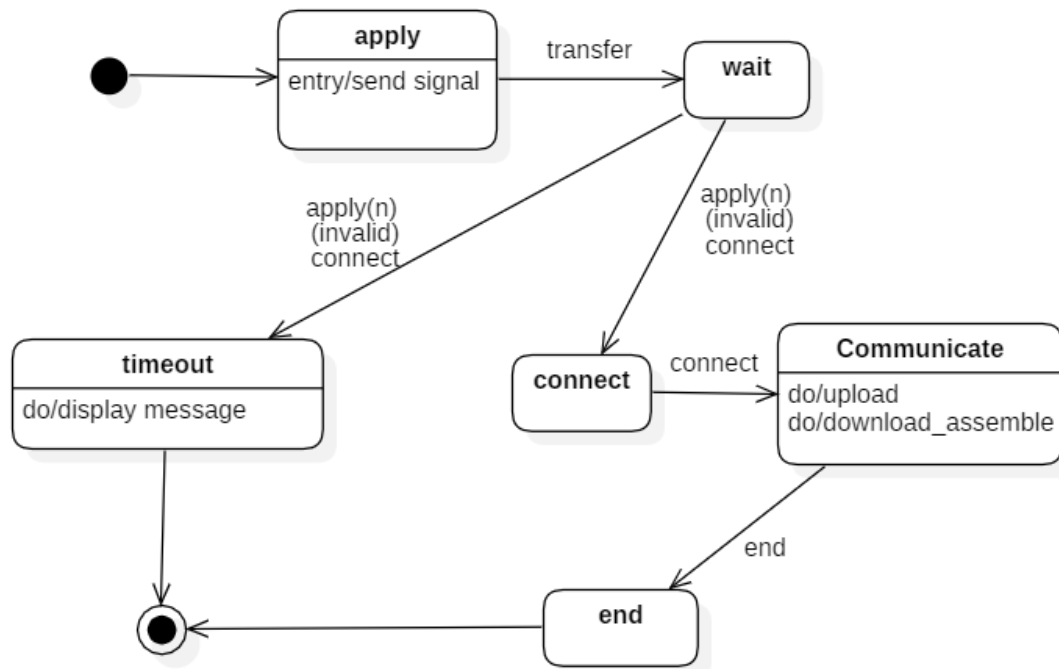


Fig 4.7 State Chart Client Diagram

##### Server Diagram:

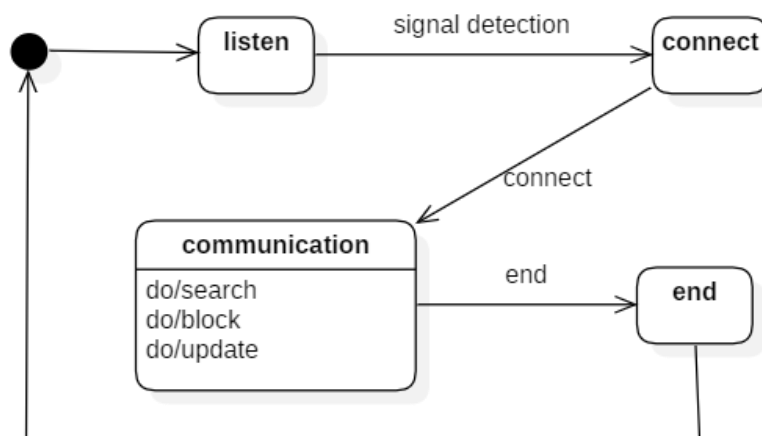


Fig 4.8 State Chart Server Diagram



### 4.2.5 Activity Diagram

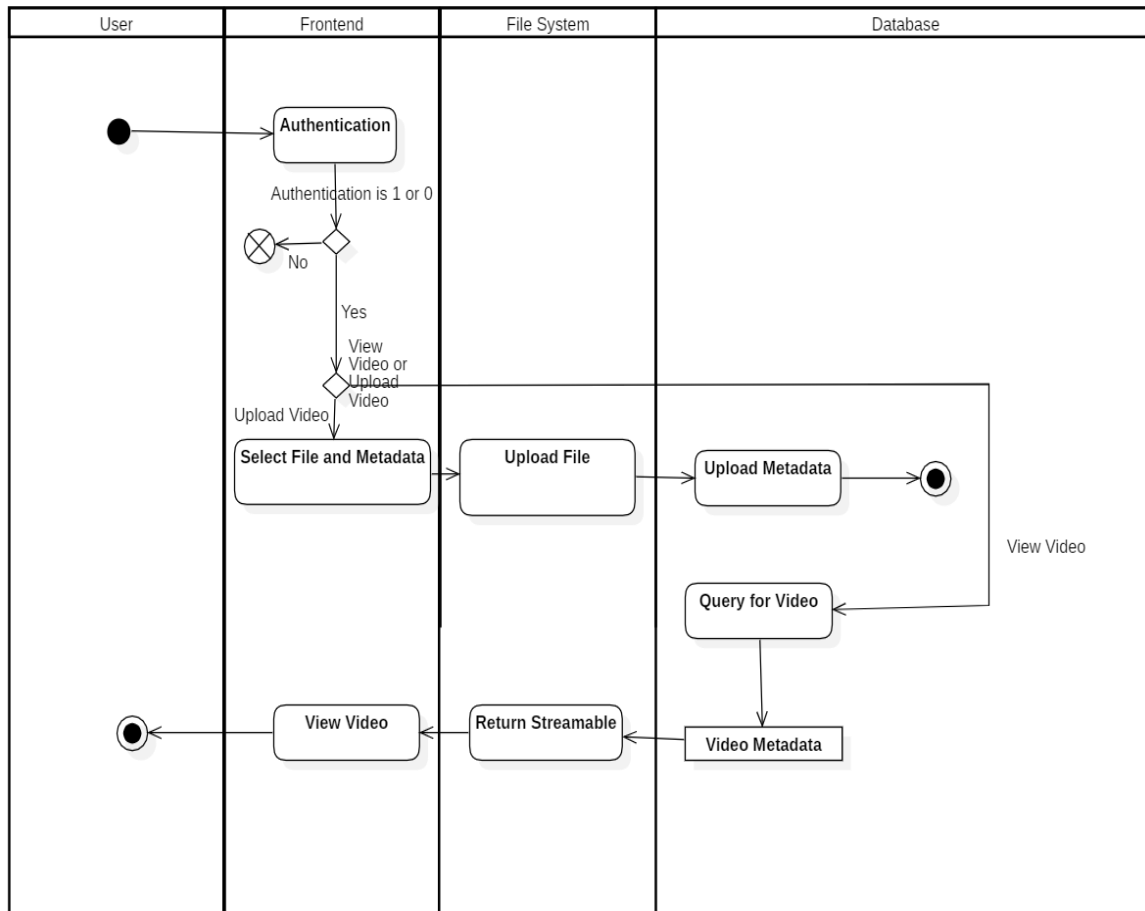


Fig 4.9 Activity Diagram

**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.

#### 4.2.6 Collaboration Diagram

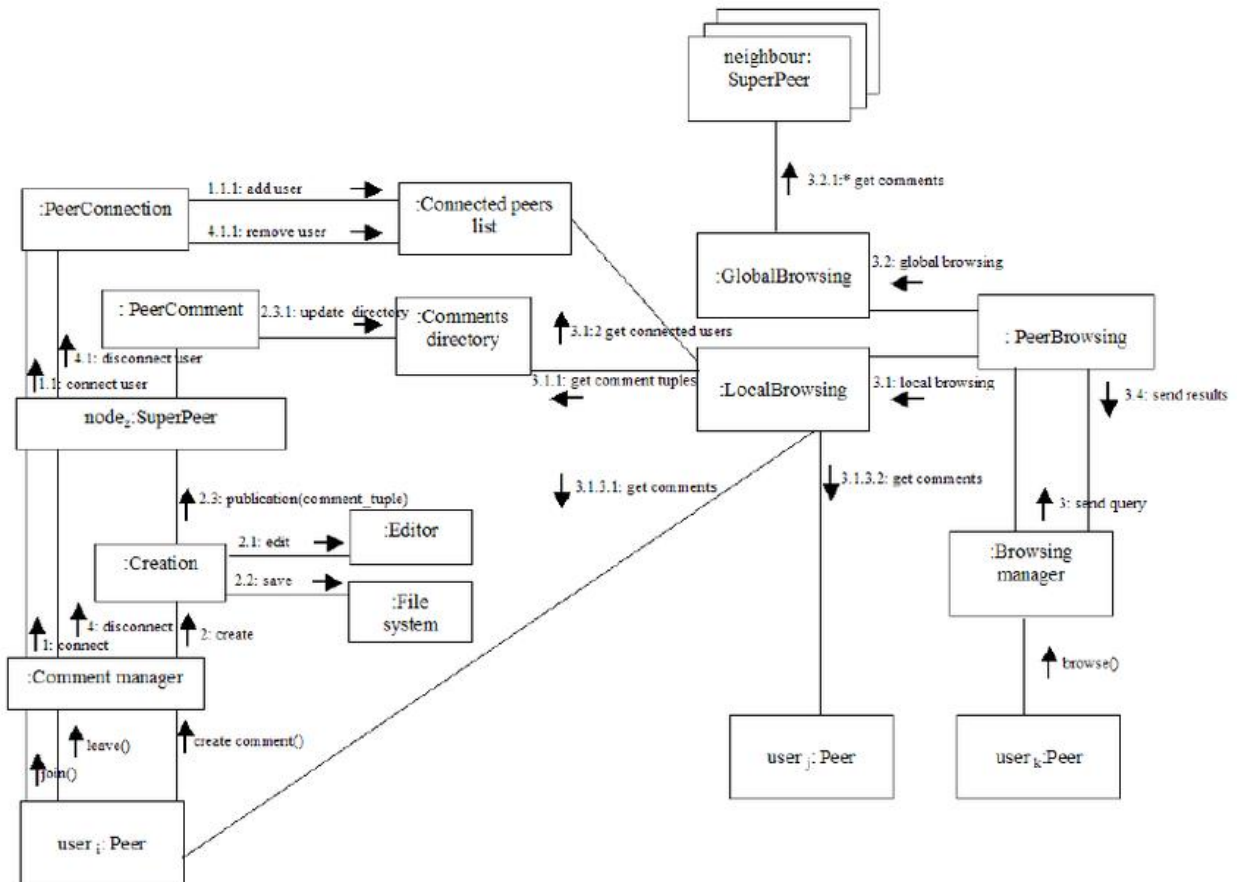


Fig 4.10 Collaboration Diagram

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.

#### 4.2.7 PERT chart

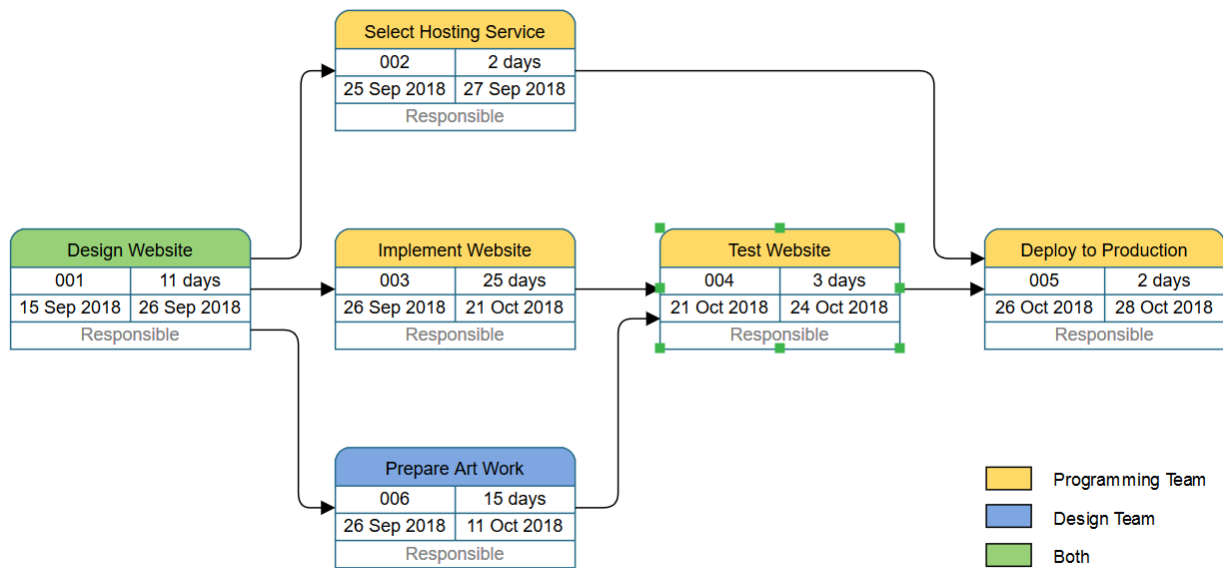


Fig 4.11 PERT (A)

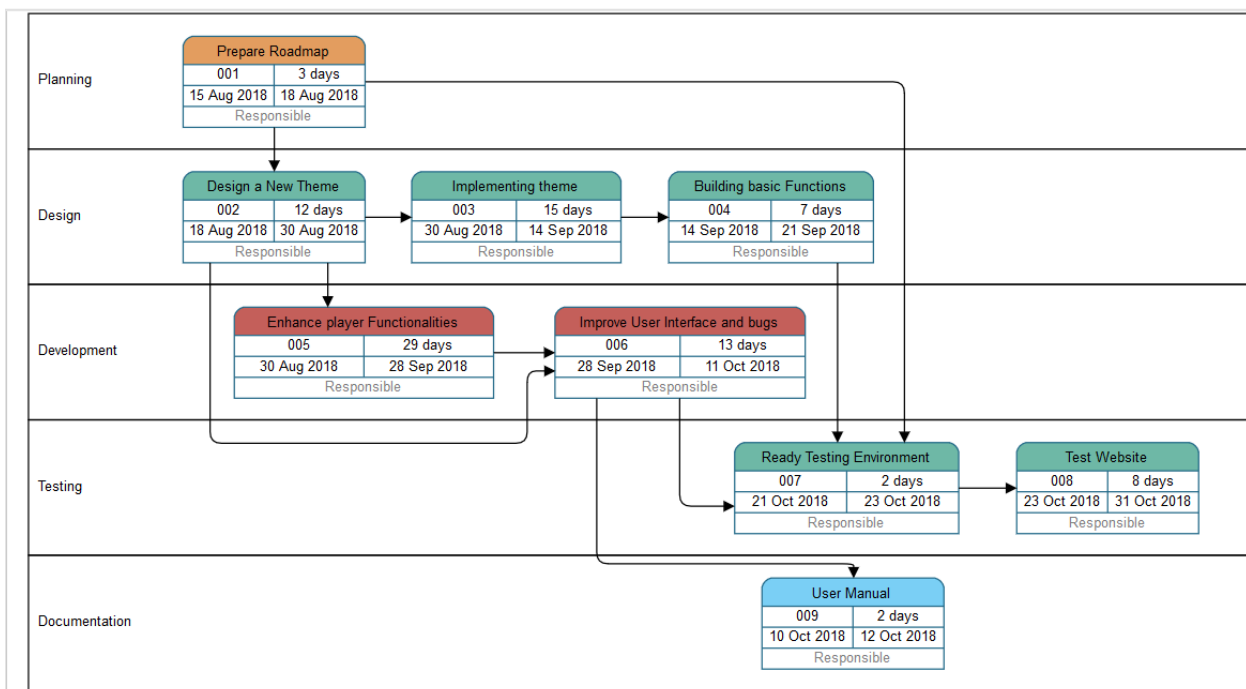
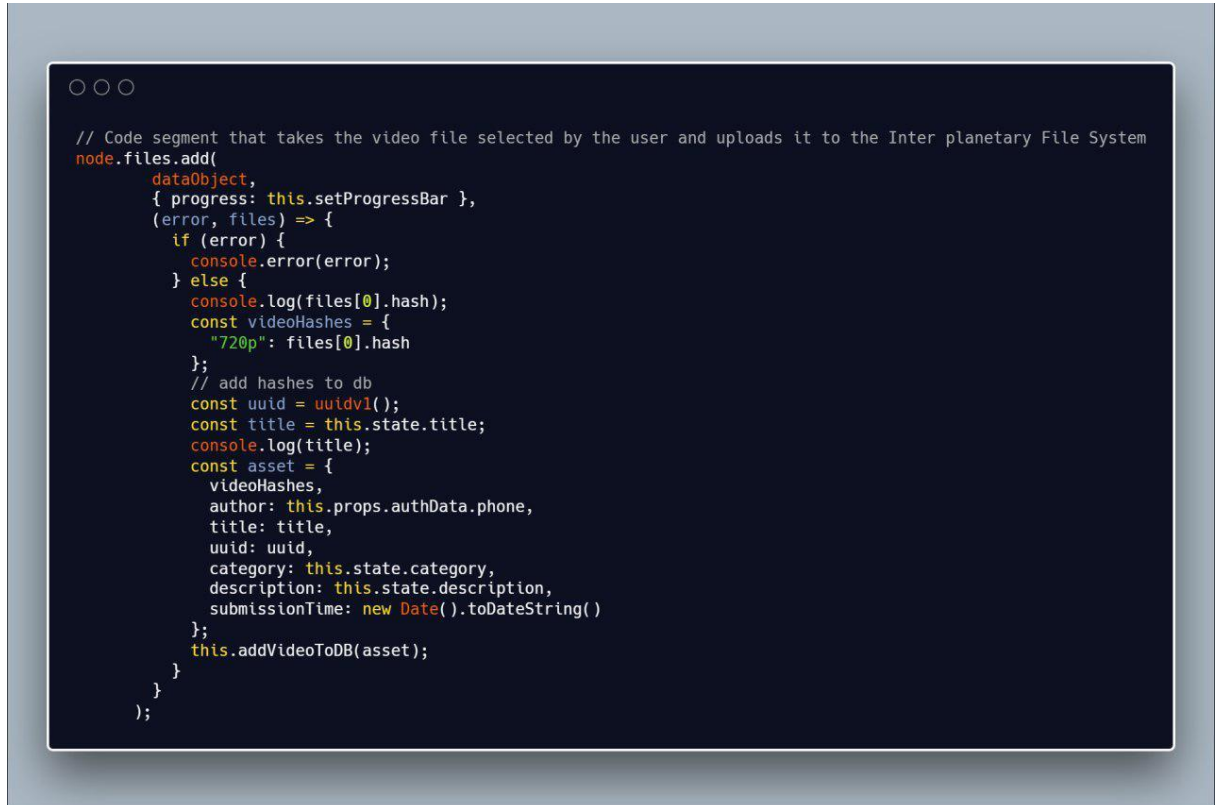


Fig 4.12 PERT (B)

A **PERT chart** 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 U.S. Navy in the 1950s to manage the Polaris submarine missile program.

## CHAPTER 5 IMPLEMENTATION

The following section covers the code implementations of the project. It will give a brief overview of the task it is trying to accomplish.



```
// Code segment that takes the video file selected by the user and uploads it to the Inter planetary File System
node.files.add(
  dataObject,
  { progress: this.setProgressbar },
  (error, files) => {
    if (error) {
      console.error(error);
    } else {
      console.log(files[0].hash);
      const videoHashes = {
        "720p": files[0].hash
      };
      // add hashes to db
      const uuid = uuidv1();
      const title = this.state.title;
      console.log(title);
      const asset = {
        videoHashes,
        author: this.props.authData.phone,
        title: title,
        uuid: uuid,
        category: this.state.category,
        description: this.state.description,
        submissionTime: new Date().toString()
      };
      this.addVideoToDB(asset);
    }
  }
);
```

Fig 5.1 Uploading to IPFS

```

// UPort developer credentials that help us interact with the uPort authentication system

import { Connect, SimpleSigner } from "uport-connect";
import { SigningKey } from "../keys/uportkeys.js";

export let uport = new Connect("INK", {
  clientId: "2ozFbmN51xFHTdd97gCLRkMDNqCwey67xMA",
  network: "rinkeby",
  signer: SimpleSigner(SigningKey)
});
export const web3 = uport.getWeb3();

```

Fig 5.2 uPort Authentication

```

// A function written in solidity that helps get user details from the blockchain

function getUser(string _email)
public
view
returns(
    string username,
    string email,
    address addr
)
{
    // Some validations (also called Guard Functions) are required before creating the user in the store.
    // 1. Check if the user already exists.
    require(checkUserExistence(_email), "User does not already exists"); // Using require function. Please refer
solidity documentation for more information.
    // Other guard functions go here.

    username = userMapping[_email].username;
    email = _email;
    addr = userMapping[_email].ethereumAddress;
}

```

Fig 5.3 Obtain User data in Solidity



```
// react component that renders the video

import React, { Component } from "react";

class INKVideo extends Component {
  state = {};

  render() {
    return (
      <div className="uploaded-videos">
        <video controls={true} width="50%" height="50%">
          <source src={this.props.src} />
        </video>
      </div>
    );
  }
}

export default INKVideo;
```

Fig 5.4 React component that renders the video

```
// React component that renders a Dashboard

import React, { Component } from "react";
import "../Dashboard.css";

class Dashboard extends Component {
  constructor(props, { authData }) {
    super(props);
    authData = this.props;
  }

  render() {
    return (
      <div className="container-fluid" style={{ paddingTop: "5%" }}>
        <div className="card text-black border">
          <div className="card-header">Dashboard</div>
          <div className="card-body">
            <p className="card-text">
              Welcome <strong>{this.props.authData.name}</strong>
            </p>
          </div>
        </div>
      </div>
    );
  }
}

export default Dashboard;
```

Fig 5.5 React component that renders a dashboard

```
// Redux function that gets activated once user enters his/her credentials. It also helps sign in the user
export function loginUser() {
  return function(dispatch) {
    // UPort and its web3 instance are defined in ../../../../util/wrappers.
    // Request uPort persona of account passed via QR
    uport
      .requestCredentials({
        requested: ["name", "avatar", "phone", "country"]
      })
      .then(credentials => {
        dispatch(userLoggedIn(credentials));
        getUser(credentials);
        localStorage.setItem(
          "uPortUserCredentials",
          JSON.stringify(credentials)
        );
        var currentLocation = browserHistory.getCurrentLocation();
        if ("redirect" in currentLocation.query) {
          return browserHistory.push(
            decodeURIComponent(currentLocation.query.redirect)
          );
        }
        return browserHistory.push("/dashboard");
      });
  });
}
```

Fig 5.6 Redux Dispatch Function



## CHAPTER 6 RESULTS AND EVALUATION

The following section contains the snapshots of the web application.

### 1) The Welcome Page

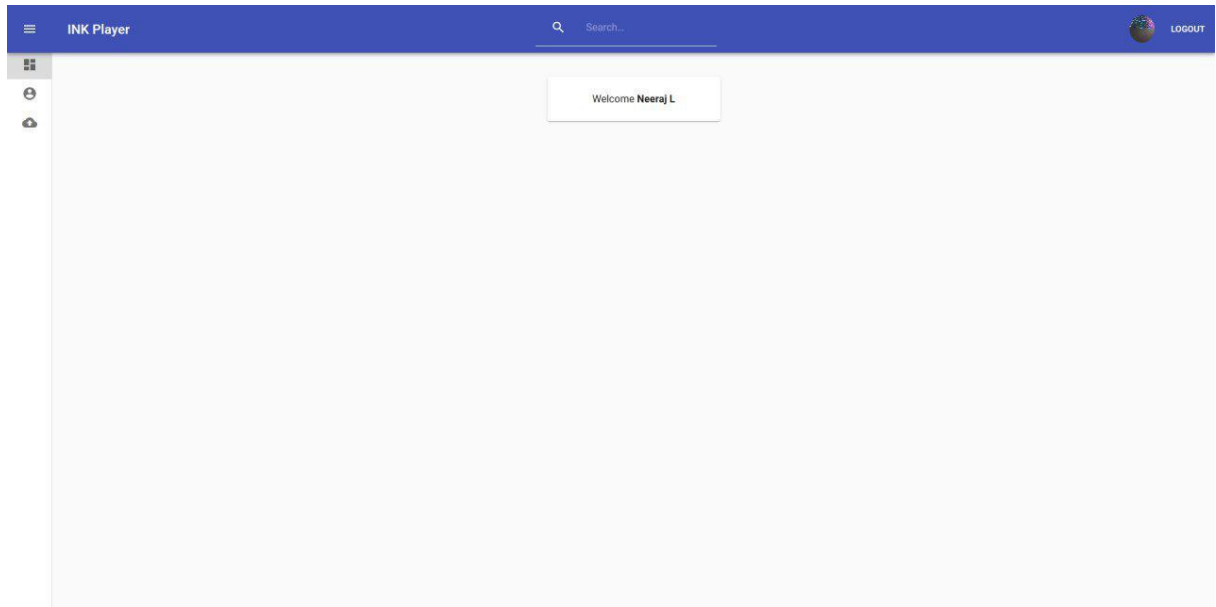


Fig 6.1 Welcome Page

### 2) The Upload Page

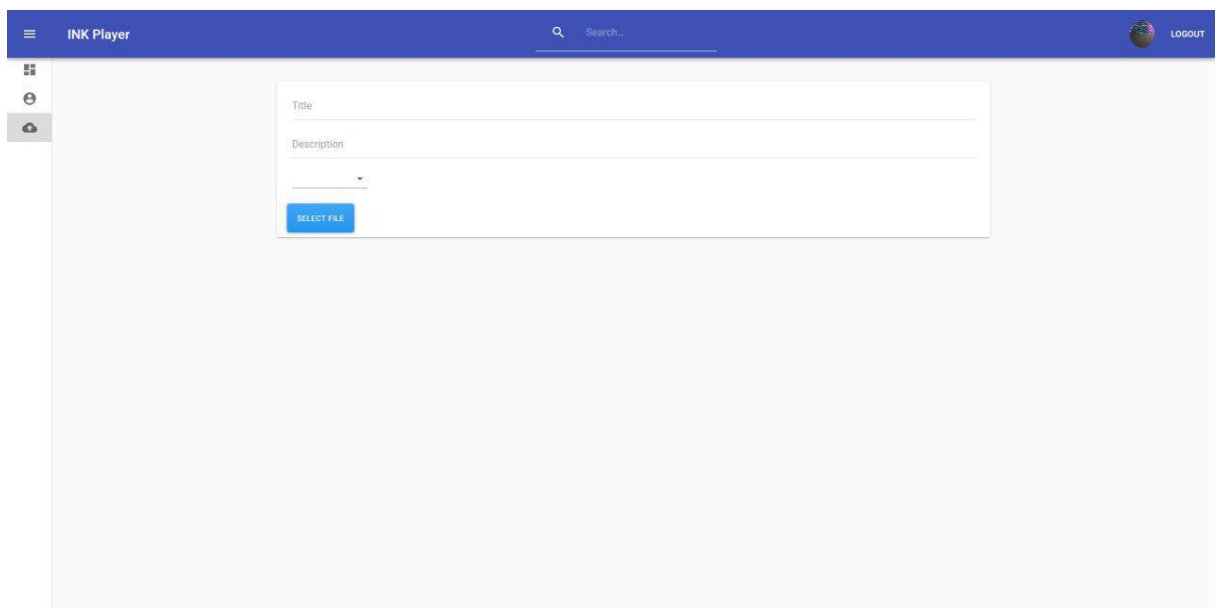


Fig 6.2 Upload Page

### 3) The User Profile Page

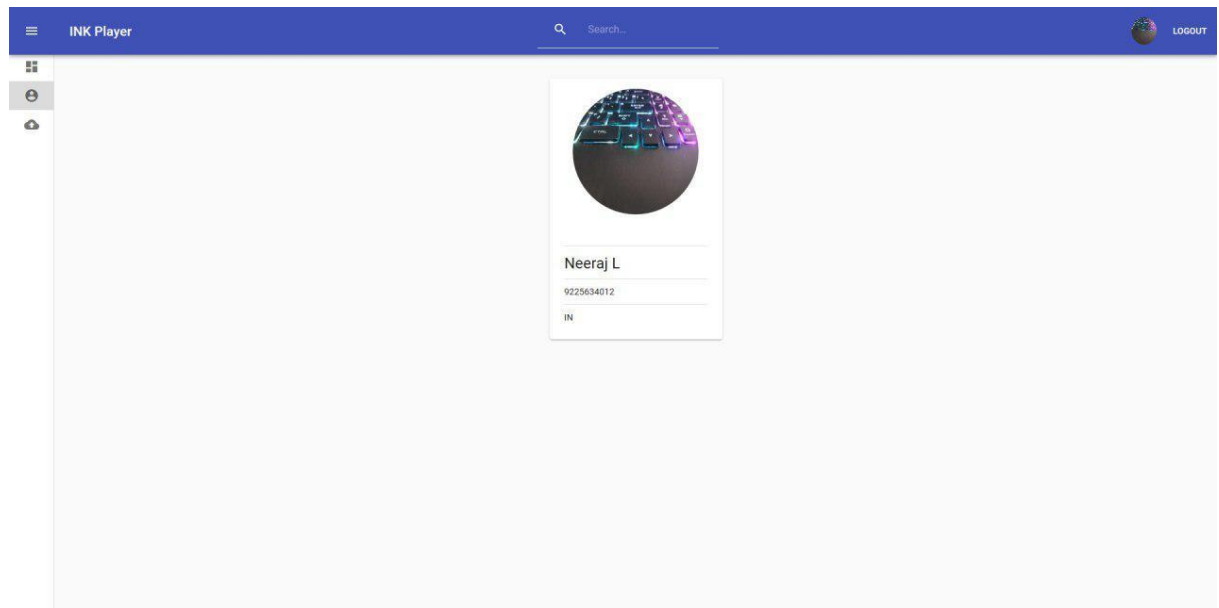


Fig 6.3 User Profile Page

### 4) The Uploaded page

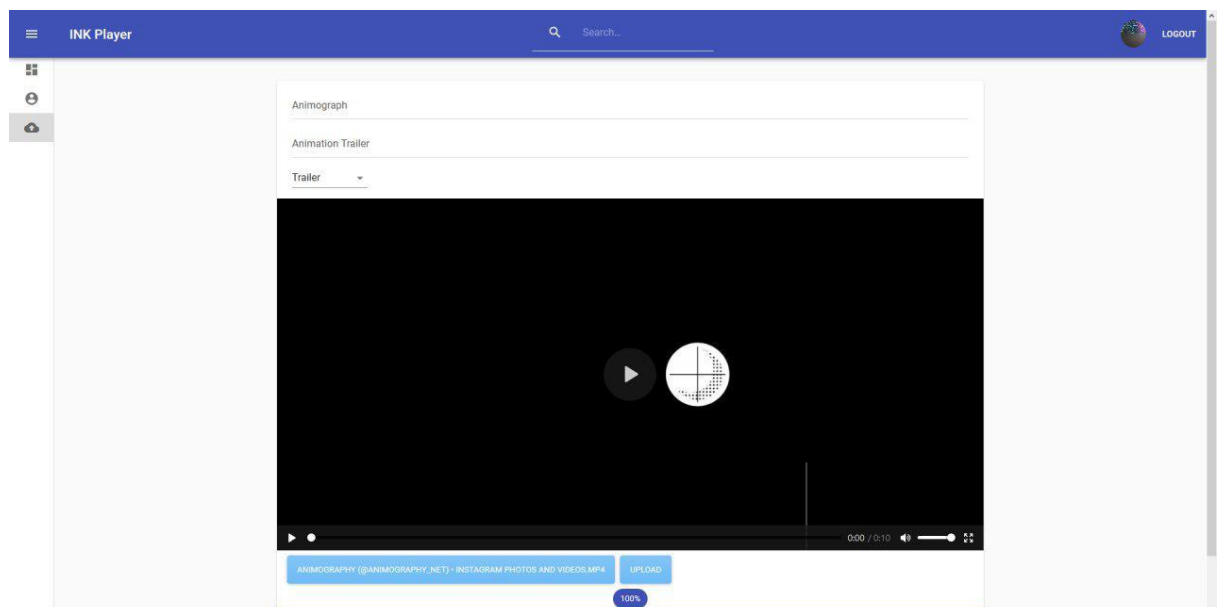


Fig 6.4 Uploaded Page

## **CHAPTER 7 CONCLUSION AND FUTURE WORK**

### **7.1 Conclusion**

The number of people using internet is increasing exponentially on a daily basis. Several upgrades have been made to the infrastructure supporting the current web, such as optic fiber cables, better servers etc. The client server architecture currently in use puts tremendous load on the servers to cater the ever increasing growth of the users. P2P architecture solves this by distributing the load among multiple peers who are currently using the service, eliminating the need of the centralized servers and providing faster, safer and private access to the content. IPFS is one such technology, which uses multiple peers in its network spread all across the world to transfer files without the need of a server. The restriction in data access through a decentralized application can be bypassed on the Ethereum blockchain and this fact can be utilized in the existing services. Applications of blockchain, once a farfetched goal, are finally making inroads in the mainstream. The idea of distributed consensus is proving to be a prime substitute to maintain security. The introduction of smart contract has radically transformed traditional contracts. People are opting to use smart contract instead of a traditional contract as it is safe, tamper-proof and has a wide variety of functionality.

### **7.2 Future Work**

The future scope in this applications are as follows:

- To embed unique key in the file so as to prevent piracy.
- To make the application open source
- To enhance the speed of data access

## REFERENCES

- [1] J. Blackburn, K. Christensen, A Simulation Study of a New Green BitTorrent, IEEE, Proc. Green Communications Workshop in conjunction with IEEE ICC'09, 2009.
- [2] Olaf Landsiedel, Stefan Gotz, Klaus Wehrle, - Towards Scalable Mobility in Distributed Hash Tables IEEE, Peer-to-Peer Computing, 2006.
- [3] 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.
- [4] 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.
- [5] Lakshmi Siva Sankar, Sindhu M, M. Sethumadhavan, - Survey of Consensus Protocols on Blockchain Applications, IEEE, International Conference on Advanced Computing and Communication Systems, 2017.
- [6] Donhee Han, Hongjin Kim, Juwook Jang, - Blockchain based Smart Door Lock System, IEEE, Information and Communication, pp. 1165, 2017.
- [7] Jatinder Singh, John David Michels, - Blockchain as a Service (Baas): Providers and Trust, IEEE, European Symposium on Security and Privacy Workshops, pp. 1165, 2018.
- [8] 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.
- [9] 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.
- [10] David Mazieres, - Self Certifying File System, Doctor of Philosophy, Massachusetts Institute of Technology, Massachusetts, USA, 2000.
- [11] Juan Benet, - IPFS - Content Addressed, Versioned, P2P File System (Draft 3), White Paper, 2014.

- [12] Jiin-Chiou Cheng, Narn-Yih Lee , Chien Chi , and Yi-Hua Chen, - Blockchain and Smart Contract for Digital Certificatell, IEEE, Proceedings of IEEE International Conference on Applied System Innovation, 2018.
- [13] 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.
- [14] 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.
- [15] Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", White Paper, 2008.
- [16] Dejan Vujičić, Dijana Jagodić, Siniša Randić, "Blockchain Technology, Bitcoin, and Ethereum: A Brief Overview", IEEE, 17th International Symposium INFOTEH-JAHORINA, 2018.
- [17] 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.
- [18] V. Buterin, "Ethereum white paper: a next generation smart contract & decentralized application platform," Ethereum White Paper, 2013.
- [19] Yongle Chen, Hui Li , Kejiao Li and Jiyang Zhang, "An improved P2P File System Scheme based on IPFS and Blockchain", IEEE International Conference on Big Data, pp. 2652- 2657, 2017.
- [20] Bastien Confais, Adrian Libre,Benoît Parrein, "An Object Store Service for a Fog/Edge Computing Infrastructure based on IPFS and Scale-out NAS", IEEE, 1st International Conference on Fog and Edge Computing (ICFEC), 2017.
- [21] Ludwig, Simone. "Comparison of Centralized and Decentralized Service Discovery in a Grid Environment." Fifteenth IASTED International Conference on Parallel and Distributed Computing and Systems, vol. 1, pp. 12- 17, 2003.
- [22] Robert W. Lucky, "The Lure of Decentraisation", IEEE Spectrum, pp. 23, 2017.