PROJECT REPORT

Digital Signatures using Smart Contracts

20ECTE401 - LIVE IN LAB 2

Submitted by

ADHAVAN M	**-**	**412420104006**
DHANUSH G	**-**	**412420104030**
SURAJ R	**-**	**412420104131**

In Partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

SRI SAIRAM INSTITUTE OF TECHNOLOGY (AUTONOMOUS),

SAI LEO NAGAR, CHENNAI-44

ANNA UNIVERSITY: CHENNAI 600 025

ANNA UNIVERSITY: CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this project report **"Digital Signatures using Smart Contracts"** is the Bonafide work of **"DHANUSH G 412420104030, SURAJ R 412420104131, ADHAVAN M 412420104006"** who carried out the 20ECTE401 - LIVE IN LAB 2 Project Work under my supervision.

SIGNATURE	**SIGNATURE**	**SIGNATURE**
Guide	Lab in Charge	HOD
Dr. D NAVEEN RAJU, Asst. Professor, Department of Computer Science and Engineering Sri Sairam Institute of Technology, Chennai-44.	Dr. D RAJALAKSHMI Asst. Professor, Department of Computer Science and Engineering Sri Sairam Institute of Technology, Chennai-44.	Dr. B. SREEDEVI Head Of Department, Department of Computer Science and Engineering Sri Sairam Institute of Technology, Chennai-44.

Submitted for project Viva – Voce Examination held on _ _ _ _ _ _ _ _ _ _ _ _

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Table of Contents

LIST OF FIGURES

ACKNOWLEDGEMENT

A successful man is one who can lay a firm foundation with the bricks others have thrown at him **-David Brinkley.**

Such a successful personality is our beloved founder Chairman, **Thiru MJF. Ln. LEO MUTHU**. At first, we express our sincere gratitude to our beloved chairman through prayers, who in the form of a guiding star has spread his wings of external support with immortal blessings.

We express our gratitude to our **CEO Mr. J. SAI PRAKASH LEO MUTHU** and our Trustee **Mrs. J. SHARMILA RAJA** for their constant encouragement for completing this project.

We express our sincere thanks to our beloved principal, **Dr. K. PALANIKUMAR** for having given us spontaneous and whole hearted encouragement for completing this project.

We are indebted to our **HEAD OF THE DEPARTMENT** **Dr. B. SREEDEVI** for her support during the entire course of this project work.

Our sincere thanks to our project guide **Dr. D NAVEEN RAJU** for his kind support in bringing out this project.

We express our gratitude and sincere thanks to our coordinator **Dr. D RAJALAKSHMI** for her valuable suggestions and constant encouragement for successful completion of this project. We thank all the teaching and non-teaching staff members of the Department of Computer Science and Engineering and all others who contributed directly or indirectly for the successful completion of the project.

ABSTRACT

A Blockchain is a decentralized entity hosted by many people across the internet, and each transaction in the blockchain is immutable which ensures data integrity and eliminates the need for trust in the system as no single user can modify or hack the records. Smart Contracts are programs that run on the blockchain when conditions are met, and these smart contracts can be used for many purposes. One such purpose is to use it to verify the integrity and origin of digital documents and verify ownership of them. Smart Contracts offer a safe and tamper proof way for dealing with digital documents safely and verify ownership of the same. The Smart Contracts are programmed in a language known as Solidity and is hosted in the Polygon Blockchain which is a layer 2 scaling chain which runs on top of the Ethereum Network, which makes it faster and scalable with the same security of the Ethereum network and widespread compatibility with various tools which are already available. The smart contracts mint a Non-Fungible Token or NFT which is based on the ERC 721 Token Standard. The ERC 721 standard provides a way of identifying unique entities in the blockchain. These tokens can then be sold, bought, transferred, auctioned to other people in the same network.

CHAPTER I
INTRODUCTION

With the world increasingly going digital, the usage of real paper documents is getting lesser and lesser. Even though this paradigm shift is good for the environment, and for accountability, the originality and integrity of these documents are questioned heavily as digital documents can be easily forged and tampered with. This also means that physical ink signatures are going out of fashion and a better replacement for the same is needed. And that is where the use of smart contracts on the blockchain is implemented. Smart Contracts are pieces of code that runs on the underlying blockchain network when certain conditions are met and these records are permanent and there is no way of editing or altering these records which is essential for all digital documents to verify their integrity and origin. The Blockchain used in this project is the Polygon Blockchain which is a Layer 2 Scaling Sidechain built on top of the Ethereum Chain. Polygon offers low GAS fees and is multiple times faster and scalable than the Ethereum Main Net.

CHAPTER II
LITERATURE SURVEY

1. **PROJECT TITLE: Using blockchain and semantic web technologies for the implementation of smart contracts between individuals and health insurance organizations.**

   Authors: Efthymios Chondrogiannis, Vassiliki Andronikou, Efstathios Karanastasis, Antonis Litke, Theodora Varvarigou

   Abstract: Blockchains and smart contracts are gaining momentum as enabling technologies for a wide set of applications where data distribution and sharing among decentralized infrastructures is required. In this work, we present a distributed application developed using blockchain technologies that allows individuals and health insurance organizations to come into agreement during the implementation of the healthcare insurance policies in each contract. For this purpose, health standards and semantic web technologies were used for the formal expression of both the insured individual's data and contract terms. Accordingly, a fine-grained data access policy was applied for evaluating contract terms on the basis of relevant data captured in healthcare settings. A prototype was implemented involving the development of several different smart contracts for the Ethereum platform as well as the necessary visual environment for accessing them. The developed system validates various features related to blockchain and smart contract features that are briefly discussed in this work, part of which can be mitigated or resolved through the use of a private permissioned blockchain. The application of well-established techniques for potential malfunctions of external services could also boost the security of the system and prevent it from potential attacks.

2. **PROJECT TITLE: A survey on blockchain technology and its security**

   Authors: Huaqun Guo, Xingjie Yu

   Abstract: Blockchain is a technology that has desirable features of decentralization, autonomy, integrity, immutability, verification, fault-tolerance, anonymity, auditability, and transparency. In this paper, we first carry out a deeper survey about blockchain technology, especially its history, consensus algorithms' quantitative comparisons, details of cryptography in terms of public key cryptography, Zero-Knowledge Proofs, and hash functions used in the blockchain, and the comprehensive list of blockchain applications. Further, the security of blockchain itself is a focus in this paper. In particular, we assess the blockchain security from risk analysis to derive

comprehensive blockchain security risk categories, analyze the real attacks and bugs against blockchain, and summarize the recently developed security measures on blockchain. Finally, the challenges and research trends are presented to achieve more scalable and securer blockchain systems for the massive deployments.

3. **PROJECT TITLE: Blockchain technology applications for Industry 4.0: A literature-based review**

Authors: Mohd Javaid, Abid Haleem, Ravi Pratap Singh, Shahbaz Khan, Rajiv Suman

Abstract: Industry 4.0 involves innovations with upcoming digital technologies, and blockchain is one of them. Blockchain can be incorporated to improve security, privacy, and data transparency both for small and large enterprises. Industry 4.0 is a synthesis of the new production methods that allow manufacturers to achieve their target more rapidly. Research has been conducted on various Industry 4.0 technologies like Artificial Intelligence (AI), Internet of Things (IoT), Big data, and Blockchain, and how they could create significant interruptions in recent years. These technologies provide various possibilities in the world of manufacturing and supply chain. Blockchain is a technology that has gained much recognition and can enhance the manufacturing and supply chain environment. Various fields now have fascinating insights into the advantages of blockchain. Several research articles on "Blockchain" and "Industry 4.0" from Google Scholar, Scopus, and other relevant sources are identified and reviewed for this study. This paper discusses the major potential of Blockchain in Industry 4.0. Various drivers, enablers, and associated capabilities of Blockchain technology for Industry 4.0 are discussed for insights. Different Industry 4.0 spheres/sub-domains for Blockchain technology realization are also discussed. Finally, we have identified and studied fourteen significant applications of Blockchain in Industry 4.0. It is a range of new developments and hope for immense opportunities that are changing Industry 4.0. This technology would work to achieve amplified outcomes and work individually to enhance the process.

4. **PROJECT TITLE: Contract law 2.0: 'Smart' contracts as the beginning of the end of classic contract law**

Author: Alexander Savelyev

Abstract: The paper analyzes legal issues associated with the application of existing contract law provisions to so-called Smart contracts, defined in the paper as 'agreements existing in the form of software code implemented on the Blockchain platform, which ensures the autonomy and self-executive nature of Smart contract terms based on a predetermined set of factors. The paper consists of several sections. In the second section, the paper outlines the peculiarities of Blockchain technology, as currently implemented in Bitcoin cryptocurrency, which forms the core of Smart contracts. In the third section, the main characteristic features of Smart contracts are described. Finally, the paper outlines key tensions between classic contract law and Smart contracts. The concluding section sets the core question for analysis of the perspectives of implementation of this technology by governments: 'How to align the powers of the government with Blockchain if there is no central authority but only distributed technologies. The author suggests two solutions, neither of which is optimal: (1) providing the state authorities with the status of a Superuser with extra powers; and (2) relying on traditional remedies and enforcement practices, by pursuing specific individuals – parties to a Smart contract – in offline mode.

CHAPTER III
REQUIREMENT SPECIFICATION

3.1. Introduction:

A software requirements specification is a document that captures a complete description about how the system is expected to perform. It is usually signed off at the end of requirements engineering phase. A Requirement Specification is a collection of the set of all requirements that are to be imposed on the design and verification of the product. The specification also contains other related information necessary for the design, verification, and maintenance of the product.

3.2 Hardware and Software Requirements

3.2.1 Hardware Requirements

- Memory: Minimum 2GB
- Screen Resolution: 1280x1024 or larger
- Application Window Size: 1024x680 or larger

3.2.2 Software Requirements:

- A Web Browser:
  - Google Chrome 54 or later
  - Mozilla Firefox ESR 45 or version 48 or later
- Meta Mask Crypto Wallet Browser Extension
- A Crypto Wallet Address

TECHNOLOGIES USED

3.3.1 BLOCKCHAIN

A blockchain is a distributed database that is shared among the nodes of a computer network. As a database, a blockchain stores information electronically in digital format. Blockchains are best known for their crucial role in cryptocurrency systems, such as Bitcoin or Ethereum for maintaining a secure and decentralized record of transactions. The innovation with a blockchain is that it guarantees the fidelity and security of a record of data and generates trust without the need for a trusted third party.

A blockchain collects information together in groups, known as blocks, that hold sets of information. Blocks have certain storage capacities and, when filled, are closed and linked to the previously filled block, forming a chain of data known as the blockchain. All new information that follows that freshly added block is compiled into a newly formed block that will then also be added to the chain once filled.

The goal of blockchain is to allow digital information to be recorded and distributed, but not edited. In this way, a blockchain is the foundation for immutable ledgers, or records of transactions that cannot be altered, deleted, or destroyed. This is why blockchains are also known as a distributed ledger technology (DLT).

Security:

Blockchain technology achieves decentralized security and trust in several ways. To begin with, new blocks are always stored linearly and chronologically. That is, they are always added to the "end" of the blockchain. After a block has been added to the end of the blockchain, it is extremely difficult to go back and alter the contents of the block unless a majority of the network has reached a consensus to do so. That's because each block contains its own hash, along with the hash of the block before it, as well as the previously mentioned time stamp. Hash codes are created by a mathematical function that turns digital information into a string of numbers and letters. If that information is edited in any way, then the hash code changes as well.

3.3.2 Smart Contracts

A "smart contract" is simply a program that runs on the Ethereum blockchain. It's a collection of code (its functions) and data (its state) that resides at a specific address on the blockchain.

Smart contracts are a type of the parent chain account. This means they have a balance and they can send transactions over the network. However, they're not controlled by a user, instead they are deployed to the network and run as programmed. User accounts can then interact with a smart contract by submitting transactions that execute a function defined on the smart contract. Smart contracts can define rules, like a regular contract, and automatically enforce them via the code. Smart contracts cannot be deleted by default, and interactions with them are irreversible**.**

Uses Case for Smart Contracts

- Accuracy, Speed, and Efficiency
    - The contract is immediately executed when a condition is met.
    - Because smart contracts are digital and automated, there is no paperwork to deal with.
    - No time was spent correcting errors that can occur when filling out documentation by hand.

- Trust and Transparency
    - There's no need to worry about information being tampered with for personal gain because there's no third party engaged and
    - Encrypted transaction logs are exchanged among participants.

- Security
    - Because blockchain transaction records are encrypted, they are extremely difficult to hack.
    - Furthermore, because each entry on a distributed ledger is linked to the entries before and after it, hackers would have to change the entire chain to change a single record.

- Savings
    - Smart contracts eliminate the need for intermediaries to conduct transactions, as well as the time delays and fees that come with them.

3.3.3 ERC 721 Tokens – Non-Fungible Token Standard

A Non-Fungible Token (NFT) is used to identify something or someone in a unique way. This type of Token is perfect to be used on platforms that offer collectible items, access keys, lottery tickets, numbered seats for concerts and sports matches, etc. This special type of Token has amazing possibilities so it deserves a proper Standard, the ERC-721 came to solve that.

The ERC-721 (Ethereum Request for Comments 721), proposed by William Entriken, Dieter Shirley, Jacob Evans, Nastassia Sachs in January 2018, is a Non-Fungible Token Standard that implements an API for tokens within Smart Contracts. It provides functionalities like to transfer tokens from one account to another, to get the current token balance of an account, to get the owner of a specific token and also the total supply of the token available on the network. Besides these it also has some other functionalities like to approve that an amount of token from an account can be moved by a third-party account.

3.3.4 The Polygon Blockchain Network

Polygon is a “layer two” or “sidechain” scaling solution that runs alongside the Ethereum blockchain — allowing for speedy transactions and low fees. MATIC is the network’s native cryptocurrency, which is used for fees, staking, and more. The Polygon Blockchain provide faster transactions and lower costs for users. It acts as a speedy parallel blockchain running alongside the main Ethereum blockchain. To use it, you can “bridge” some of your crypto over to Polygon, and then interact with a wide range of popular crypto apps that were once exclusive to the main Ethereum blockchain.

3.3.5 MetaMask Wallet

MetaMask is a popular cryptocurrency wallet known for its ease of use, availability on both desktops and mobile devices, the ability to buy, send, and receive cryptocurrency from within the wallet, and collect non-fungible tokens (NFTs) across two blockchains. MetaMask wallet is a versatile product in the marketplace, supporting multiple protocols across blockchains. In addition to supporting over 450,000 coins on the Ethereum ERC-20 protocol, users can also store non-fungible tokens (NFTs) in the wallet and connect them with marketplaces like OpenSea. MetaMask also supports multiple blockchains, including the BNB Chain (formerly known as the Binance Smart Chain), Polygon, Avalanche, and several other test blockchains.

3.3.6 Solidity

Solidity is an object-oriented, high-level language for implementing smart contracts. Smart contracts are programs which govern the behavior of accounts within the Ethereum state. Solidity is a curly-bracket language designed to target the Ethereum Virtual Machine (EVM). It is influenced by C++, Python and JavaScript. You can find more details about which languages Solidity has been inspired by in the language influences section. Solidity is statically typed, supports inheritance, libraries and complex user-defined types among other features. With Solidity you can create contracts for uses such as voting, crowdfunding, blind auctions, and multi-signature wallets.

3.3.7 Node.js and Next.js

Node.js

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser, which was designed to build scalable network applications. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts. Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications

Next.js

Next.js is an open-source web development framework built on top of Node.js enabling React-based web applications functionalities such as server-side rendering and generating static websites. React documentation mentions Next.js among "Recommended Toolchains" advising it to developers as a solution when "Building a server-rendered website with Node.js". Where traditional React apps can only render their content in the client-side browser, Next.js extends this functionality to include applications rendered on the server-side.

CHAPTER IV
SYSTEM DESIGN AND ARCHITECTURE

4.1 SYSTEM DESIGN

In our implementation we used a Next.JS Frontend Web Application to interact with the solidity smart contract that is deployed on the Polygon Blockchain. Firstly, the user connects their wallet to the application, then the user uploads a document to IPFS, which is the inter-planetary file system for signing. IPFS is a peer-to-peer decentralized file system. Once the document is uploaded, the user receives a success message for document upload and then is asked to sign with their wallet. When the user clicks on "Sign with Wallet" a MetaMask Prompt is shown which shows the Transaction Fees (GAS FEE) for the particular Transaction. And if the user has sufficient balance in their account the user is asked to confirm the transaction. On confirmation which takes a certain time based on the network traffic the NFT is added to the user's wallet. This document is now signed by the user which stays on the blockchain and can be only accessed with the private key of the wallet that is used for minting. Thus, the document is signed and can be viewed by anyone in the network.

4.2 SYSTEM ARCHITECTURE

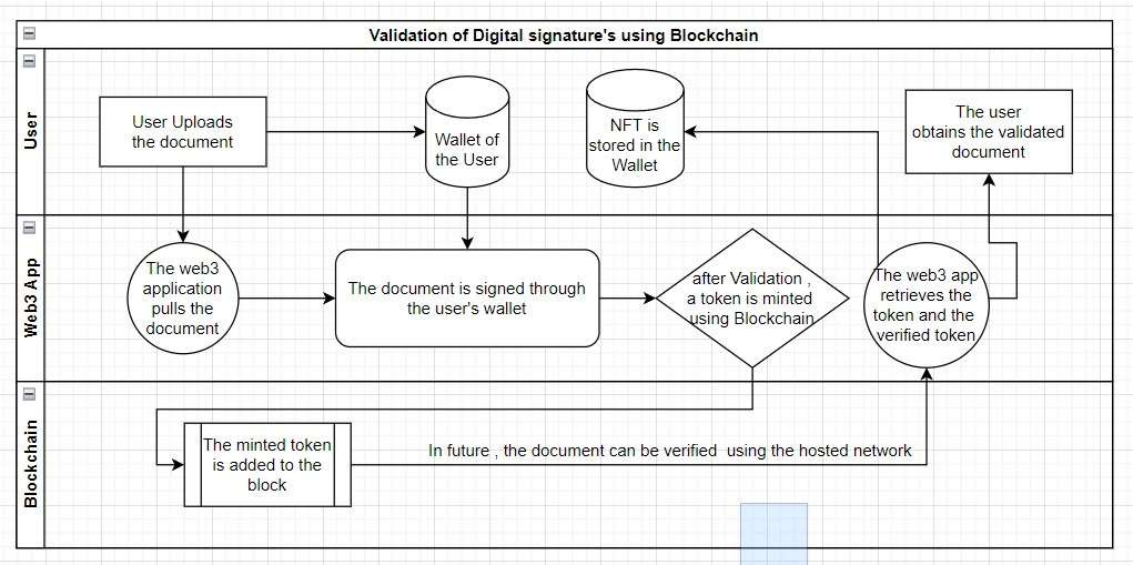

Fig 4.1

CHAPTER V
SYSTEM TESTING

In order to make the Blockchain App to work with the Polygon / Ethereum Main Net, severe system testing has to be done before deploying to the public.
To perform tests there are several tools and applications available for the Web3 world.

Polygon Mumbai Test Net
https://docs.polygon.technology/docs/develop/network-details/network/
The Mumbai Polygon Test Net is a Hosted Test Net that has its primary Node in Mumbai and all across the world. This Test Net Helps in Testing the Blockchain Applications that is built for Ethereum and Polygon Networks to be tested to solve real world use cases.

Polygon Faucet:
https://faucet.polygon.technology
To facilitate Transactions even on the Test Net, we need the Polygon Matic ERC20 Token, which is usually bought through a Crypto Exchange in the real world since they have real value. But in the Test Net these tokens don't have any value, so they are given out for free using the Polygon Faucet. If you enter your Wallet ID – The Faucet will add some minimal number of Tokens to Provide for Gas Fees and other fees.

Block Explorer Polygon Mumbai Test Net
https://mumbai.polygonscan.com
A Block Explorer is an online tool that enables you to search for real-time and historical information about a blockchain, including data related to blocks, transactions, addresses and more. A block explorer is used to check the status of a transaction, for buying or selling crypto. It is also used to acquire information associated with a blockchain address, including the transaction history, the total value of the assets held at the address, the total amount of crypto received at the address, and the total amount of crypto sent from the address, among other data points.

Opensea Test Net
https://testnets.opensea.io
Opensea is an NFT marketplace which facilitates buying selling and listing of NFTs from the user's wallet. Since Opensea is a real marketplace, it uses real Ethereum and Polygon to facilitate transactions on the main net. But for the purposes of testing, there is an option to use the polygon testing Mumbai Net which works same as the Main Net but is for testing purposes.

CHAPTER VI
SOURCE CODE

File 1: /pages/index.js

```
import Head from 'next/head'
import Link from 'next/link'
import Image from 'next/image'
import { Navbar } from '../components/Navbar'
import styles from '../styles/Home.module.css'

export default function Home() {
  return (
    <div>
      <Head>
        <title>🖊 Polysign - Sign Documents Digitally</title>
      </Head>
      <Navbar />
      <main className='container'>
        <section>
          <div className='landing-wrapper'>
            <div className="landing-image">z`
              <Image src='/assets/landing-background.avif' alt='Landing Page
Background' width="100%" height="100%" layout='fill'></Image>
            </div>
            <h1 className='titletext'>Digitally Sign Documents on the <br />
Blockchain with 🖊 Polysign</h1>
            <div className='button-group'>
              <Link href='/start'><button className='btn-filled link'>Get
Started</button></Link>
              <Link href='#know-more'><button className='btn-outline'>Know
More</button></Link>
            </div>
          </div>
        </section>
        <section id='know-more'>
          <div className='know-more-wrapper'>
              <h1>How it Works?</h1>
          </div>
        </section>
        <div className="sizedbox"></div>
      </main>
    </div>
  )
}
```

File 2: /pages/start.js

```
/* eslint-disable @next/next/no-img-element */
import React, { useState } from 'react'
import * as IPFS from 'ipfs-core'
import { Navbar } from '../components/Navbar'
import Link from 'next/link'
import { contractABI, contractAddress } from '../contract'
import { useWeb3 } from "@3rdweb/hooks"
import Web3 from 'web3'

const web3 = new Web3(Web3.givenProvider)
const Start = () => {

  const [document, setDocument] = useState(null)
  const [tokenId, setTokenId] = useState(null);
  const [response, setResponse] = useState(null);
  const [file, setFile] = useState(false);
  const [Url, setUrl] = useState(false);
  const { address } = useWeb3();
  const [title, setTitle] = useState('');
  const [description, setDescription] = useState('');


  const upload = async (title, desc, file) => {
    console.log('Uploading Document to IPFS');
    return new Promise(async (resolve, reject) => {
      const ipfs = await IPFS.create({ repo: 'ipfs-' + Math.random() });
      try {
        // const ipfs = await IPFS.create()
        const filesAdded = await ipfs.add(file);
        console.log('IPFS upload Succesful');
        resolve(filesAdded);
      } catch (ex) {
        console.error(ex)
        reject(ex);
      }
    })
  }

  const worker = async (ipfspath) => {
    console.log('Started Worker For Minting');

    const CONTRACT = await new web3.eth.Contract(contractABI,
contractAddress);
    const metadata = { name: title, description, image: ipfspath, address,
mintedAt: new Date().getTime() }
```

```
    console.log(address);
    const response = await CONTRACT.methods.mintTo(address,
JSON.stringify(metadata)).send({ from: address });
    // setTokenId(response.events.Transfer.returnValues.tokenId);
    console.log('response',response, response.receipt, response.id);
    setResponse(response);
  }


  const onSubmit = async (e) => {
    e.preventDefault();
    console.log('Submitted Form');
    const file = e.target.document.files[0]
    if (!file) {
      console.log('File Doesn\'t Exist, Upload and Continue!')
      return;
    }
    var reader = new FileReader();
    var url = reader.readAsDataURL(file)
    setUrl(url);
    const { title, description } = e.target
    setTitle(title.value);
    setDescription(description.value);
    const id = await upload(title, description, file);
    console.info('onChange', id);
    const path = `https://ipfs.io/ipfs/${id.path}`
    id && setDocument(path);
  }

  return (
    <div>
      <div><Navbar /></div>
      <main className="container">
        <section>
          <div className="center-all">
          <h1>Sign Your Documents in Three Simple Steps</h1>
          { !document && <div className='flow-wrap'>
          <div className="round">1</div>
          <h2> Upload your Document</h2>
          <br />
          {
            <form onSubmit={onSubmit}>
            <input type="text" name="title" id="title" placeholder='Title' />
            <input type="text" name="description" id="description"
placeholder='Description' />
            <input type="file" name="document" id="document"  />
            { file && <p>{file.name }</p> }
            <br />
```

```
            <button type="submit">Submit</button>
           </form>
          }

          </div> }
          <br /><br />
          {
            document && !response && <div className='flow-wrap'>
              <div className="round">2</div>
              <h2>Document Uploaded to IPFS</h2>
              { Url && <img src={Url} width="300px" height="300px"
className="preview-img" alt="Document" /> }
              { document &&  <img src={document} alt='ipfs link resource'
width='600px' /> }
              { document && <button onClick={() => {worker(document)}}>Sign
with Wallet</button>}
            </div>
          }
          <br />
          {
            response && <div className='flow-wrap'>
                <div className="round">3</div>
                { response && <h2>Token Mint Successful</h2> }
                <br />
                {document &&  <img src={document} className="preview-img"
alt='ipfs link resource' /> }
                {
                  document && <Link href="/nfts"><button>View All Signed
Documents</button></Link>

                }
              </div>
          }
          </div>
        </section>
      </main>
    </div>
  )
}

export default Start
```

File 3: /pages/nfts.js

```
import React, { useState, useEffect } from 'react'
import { Navbar } from '../components/Navbar';
import { ThirdwebSDK } from "@thirdweb-dev/sdk";
import { contractABI, contractAddress } from '../contract'

import Web3 from 'web3'

const web3 = new Web3(Web3.givenProvider)


const NFT = () => {

   useEffect(()=>{
       getNFTS()

   },[])

    const [NFTS, setNFTS] = useState([]);

    const getNFTS = async () => {
        //setNFTS([]);
        console.log("try")
        const CONTRACT = await new web3.eth.Contract(contractABI,
contractAddress);
        const response = CONTRACT.methods
        const totalSupply = await response.totalSupply().call();
        for (let i = 0; i < totalSupply; i++) {
            const token = await response.tokenURI(i).call();
            try {
                const metadata = JSON.parse(token);
                setNFTS(prev => [...prev, metadata]);
            } catch {
                console.log('Not a valid metadata');
            }
        }

    }



    return (
        <div>
            <Navbar />
            <main className='nft-container'>
                { NFTS.length == 0 && <h1>Loading Signed Documents...</h1> }
                {NFTS.map((nft, index) => {
                    return (
```

```
                        <div key={index} className="nft-card">
                            <h2>{nft.name}</h2>
                            <img src={nft.image} alt="nft-image" />
                            <p className="description">{nft.description}</p>
                            {
                                nft.address && <p>ownedBy:
{nft.address?.substring(0,5) + "...." +
nft.address?.substring(nft.address.length-5, nft.address.length -1)}</p>
                            }
                        </div>
                    )
                })}
            </main>
        </div>

    )
}

export default NFT
```

CHAPTER VII
SCREENSHOTS

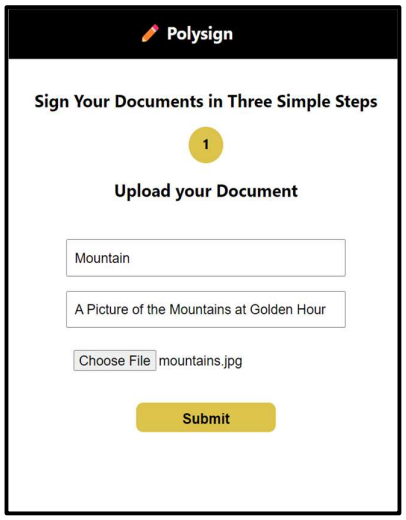

Fig 7.1

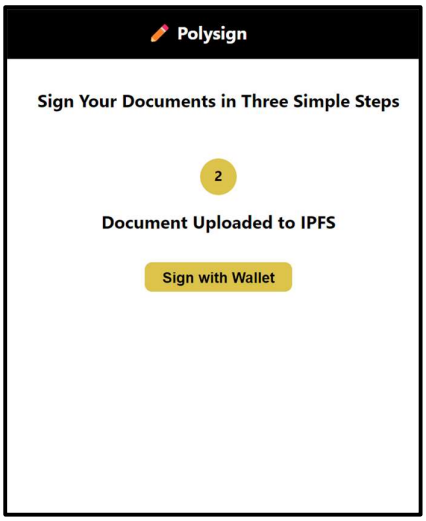

Fig 7.2

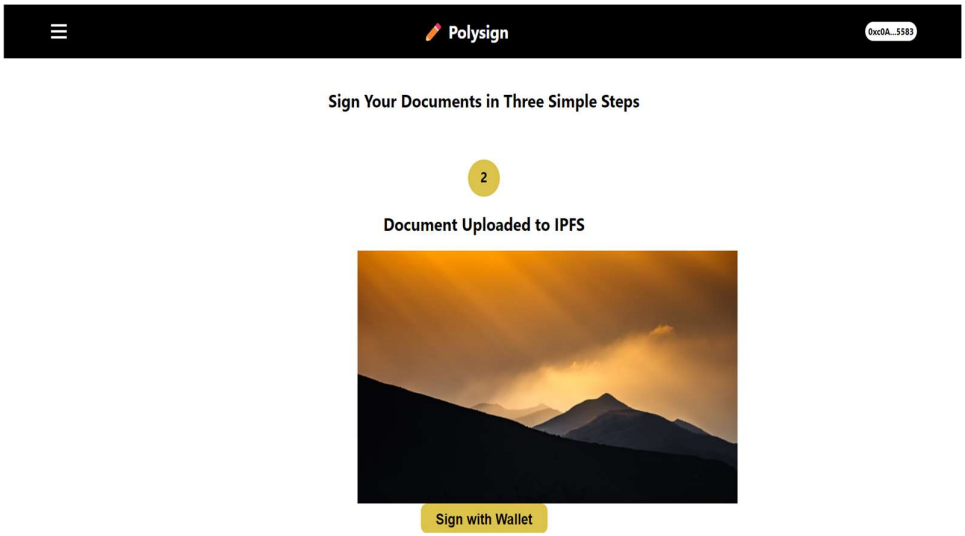

Fig 7.3


Polysign
0xc0A...5583
Mountains
Picture of mountains at golden hour
ownedBy: 0xc0A....f558

Fig 7.4

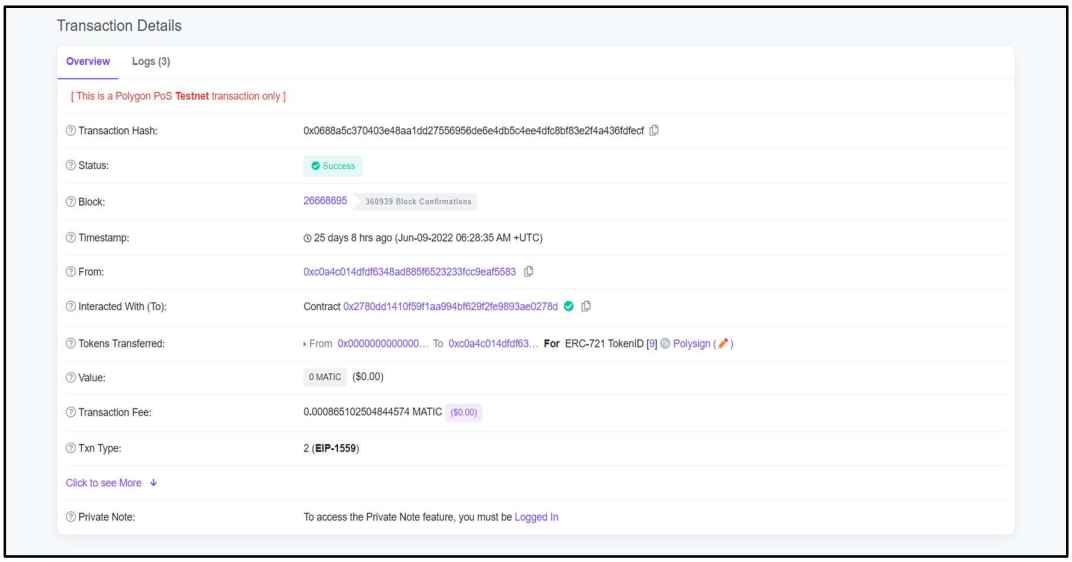
Transaction Details
Overview
Logs (3)
[This is a Polygon PoS Testnet transaction only]
Transaction Hash:
0x0688a5c370403e48aa1dd27556956de6e4db5c4ee4dfc8bf83e2f4a436fdfecf
Status:
Success
Block:
26668695
360939 Block Confirmations
Timestamp:
25 days 8 hrs ago (Jun-09-2022 06:28:35 AM +UTC)
From:
0xc0a4c014dfdf6348ad885f6523233fcc9eaf5583
Interacted With (To):
Contract 0x2780dd1410f59f1aa994bf629f2fe9893ae0278d
Tokens Transferred:
From 0x0000000000000... To 0xc0a4c014dfdf63... For ERC-721 TokenID [9] Polysign
Value:
0 MATIC ($0.00)
Transaction Fee:
0.000865102504844574 MATIC ($0.00)
Txn Type:
2 (EIP-1559)
Click to see More
Private Note:
To access the Private Note feature, you must be Logged In

Fig 7.5

CHAPTER VIII
FUTURE ENHANCEMENT

1. Make the minting of tokens faster by speeding up IPFS upload or using an alternative to IPFS.
2. Add support for more Metadata to be minted along with the document.
3. Enable the signing of documents other than Images and add the ability to view them instantly.
4. Add functionality for multiple parties to sign on a single document like a sale deed or rent agreement.

CHAPTER IX
CONCLUSION

The current implementation simply demonstrates the basic capabilities of the WEB 3 world, and this project is aimed to use the WEB 3 Framework fully to better suit the technology for real world applications and improve the ease of use for lots of people. With that in mind this project is going to be developed further and further by refining each component and step involved and make it efficient and easy to use for all people. Roughly 30 Years ago the Internet was launched and no one anticipated that it would be an essential part of our lives 30 years later. Today the Web 3 technologies provides the capability for enabling a decentralized world with no one central authority but involving all the stakeholders as authority and I'm sure that in the near future Web 3 technologies will play a vital role in the future world similar to how the Internet does now.

CHAPTER X
REFERENCES

1. https://docs.polygon.technology/docs/develop/getting-started/
2. https://portal.thirdweb.com/
3. https://nextjs.org/docs
4. https://docs.soliditylang.org/en/v0.8.15/
5. https://portal.thirdweb.com/pre-built-contracts/nft-collection
6. https://medium.com/coinmonks/get-started-with-building-ethereum-dapps-and-smart-contracts-d86b9f7bd1c
7. https://yourstory.com/the-decrypting-story/exclusive-polygon-sandeep-nailwal-investors-funding-sequoia-tiger-softbank/amp
8. https://docs.polygon.technology/docs/develop/alchemy/
9. https://www.youtube.com/watch?v=mhYsUeE7Cy4
10. https://docs.polygonscan.com/v/mumbai-polygonscan/

Full Project Hosted at https://github.com/dnashh/polysign