

# **Social Media Web 3.0 Decentralized app**

**A report submitted for the course named Project - III (CS400)**

*Submitted by*

Name: Amit Kumar

Roll No: 19010111

Semester: 7th Semester

**Supervised by**

**Dr. Kishorjit Nongmeikapam**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
INDIAN INSTITUTE OF INFORMATION TECHNOLOGY MANIPUR  
MANTRIPUKHRI, IMPHAL-795002, INDIA  
September 2022**

## **Declaration**

I hereby declare that this submission is our own work and that to the Best of our knowledge and beliefs. It contains no material previously Published or written by neither any person nor material which to a Substantial extent has been accepted for the award of any other degree Or diploma of the university or other institute of higher learning, except Where due acknowledgement has been made in the text.

Signature of Student

Name: Amit Kumar

Roll No: 19010111

Signature of Supervisor

Dr. Kishorjit Nongmeikapam

# Certification

The work embodied in the present report entitled  
“**Social Media Web 3.0 Decentralized app**” has been carried  
out in the Department of Computer Science & Engineering  
The work reported herein is original and does not  
form part of any other report or dissertation on the basis of which a degree  
or award was  
conferred on an earlier occasion or to any other student.  
I understand the Institute’s policy on plagiarism and declare that the report  
and publications  
are my own work, except where specifically acknowledged and has not  
been copied from  
other sources or been previously submitted for award or assessment.

Amit Kumar  
19010111  
Department of Computer Science & Engineering  
IIIT Senapati, Manipur

Signature of Examiner

# Abstract

In this report , we are going to present detailed implementation of Social Media Application Using Blockchain.Our project, Block chain based “**Social Media Web 3.0 Decentralized app** ” using Next.js,Solidity,Sanity,Pinata.Blockchain technology is much more than a system for securely transferring cryptocurrencies.

The social media is something almost everyone using either on a daily basis or at least once in a while.it is Web 3.0 app where user can post text, video, image. User can like, share, comment to the post.

NFT based profile image. NFTs (**non-fungible tokens**) are unique cryptographic tokens that exist on a block chain and cannot be replicated.

For many, social media appears to have a range of benefits. **It provides a way for many of us to connect with others.** We can support other people and feel supported by them. It may even be a useful way for those with social anxiety and those who have a hard time with face-to-face interactions to connect with others.

# Acknowledgement

Firstly, I would like to thank Dr. Kishorjit sir for guiding us through Each and every step of the process with knowledge and support. His Thoughts have been a constant source of inspiration for us. I would also Like to acknowledge the contribution of all faculty members of the Department for their kind assistance, suggestions and cooperation Throughout the development of the project.

Finally, we would like to thank our classmates for the encouragement And help during the project.

Signature of Student

Name: Amit Kumar

Roll No: 19010111

# Table of Contents

Chapter 1 .....	8
Introduction .....	8
1.1 Problem statement and Objective .....	8
1.2 Project Features .....	9
1.3 Project Scope .....	10
1.4 Gantt chart.....	10
1.5 Summary .....	11
Chapter 2: .....	12
Existing System Study .....	12
2.1 Introduction.....	12
2.2 Software and Applications .....	12
2.3 Project SetUp .....	13
2.4 Summary .....	14
Chapter 3: .....	15
System Design.....	15
3.1 Introduction.....	15
3.2 Life Cycle.....	15
3.2.1 Prototype Model .....	15
3.3 Data-Flow Diagram.....	18
3.4 Justification of Chosen Lifecycle .....	19
3.5 Methodology .....	20
3.6 Summary .....	20
Chapter 4: .....	21
Implementation .....	21
4.1 Introduction.....	21
4.2 Steps of implementaion .....	21
4.3 Web3 dapp .....	22
How does the frontend of Web 3.0 dApp communicate with the Blockchain network? .....	23
4.4 Working of Web3 dapp .....	25
4.5 Working of Metamask .....	27
4.6 Front-end implementaion detail.....	27

4.7 Smart Contract implementaion detail .....	29
Chapter 5: .....	35
Result analysis and Testing .....	35
5.1 Deploy and host app on Vercel .....	37
Chapter 6: .....	43
Conclusion .....	43
6.1 How Will Web3.0 Change Future of Social Media? .....	44
5.2 Bibliography .....	46

# Chapter 1

## Introduction

The project “Social Media Application using Blockchain” using Next.js, Solidity, sanity, Pinata. Blockchain technology is much more than a system for securely transferring cryptocurrencies. Social media is important because it allows you to share your knowledge and gain knowledge.

Also it reach, nurture, and engage with your target audience — no matter their location. When a business can use social media to connect with its audience, it can use social media to generate brand awareness, leads, sales, and revenue.

### 1.1 Problem statement and Objective

The social media is something almost everyone using either on a daily basis or at least once in a while. it is Web 3.0 app where user can post text, video, image. user can like, share, comment to the post.

NFT based profile image. NFTs (**non-fungible tokens**) are unique cryptographic tokens that exist on a blockchain and cannot be replicated.

#### 1. Free Speech & Censorship Resistance

A decentralised social network allows users more control. There is no central authority (person, server or company) that dictates on users and block their freedom.



## 2. Data Privacy & Security

We rely on public-key cryptography for account security, rather than relying on a single organisation to protect user data. We don't track users.

## 3. Economic Neutrality

We want to liberate users from invasive advertising and the risk to privacy.

## 4. Digital Ownership

It stores a user's knowledge on-chain, so that the user becomes the owner of the content he creates, which means greater control and greater rewards

## 5. Community Management

It provide a better management of any gated community and proved a secure place to share member-only content on social network.

# 1.2 Project Features

list the core features of the project which I am planning to build

- Users can create post, like them, tag people, and share the post.
- Users can share text, images, gif and video on the post.
- Users can mint their nft profile and set it as their avatar.
- Users can become the member of token gated communities
- Crypto Communities can create their own gated communities for members and followers
- Community member need to have token or community nft to join a particular community as member.

## 1.3 Project Scope

Outside of finance, blockchain can be used in applications including **healthcare, insurance, voting, welfare benefits**. NFTs can represent real-world items like artwork and real estate. "Tokenizing" these real-world tangible assets makes buying, selling, and trading them more efficient while reducing the probability of fraud. The project "Social Media Application using Blockchain" using Next.js, Solidity, Sanity, Pinata.

## 1.4 Gantt chart

	July	August	September	October	November
<i>Problem identification</i>					
<i>Planning</i>					
<i>learning</i>					
<i>Coding</i>					
<i>Testing</i>					
<i>Report</i>					

## 1.5 Summary

Blockchain technology is much more than a system for securely transferring cryptocurrencies. Social media is important because it allows you to share your knowledge and gain knowledge.

NFT based profile image. NFTs (**non-fungible tokens**) are unique cryptographic tokens that exist on a blockchain and cannot be replicated.

Social media plays an important role in every student's life. It is often easier and more convenient to access information, provide information and communicate via social media. Tutors and students can be connected to each other and can make good use of these platforms for the benefit of their learning and teaching.

## Chapter 2:

### Existing System Study

#### 2.1 Introduction

The project “Social Media Application using Blockchain” using Next.js, Solidity, sanity.io, pinata. So in this chapter we talk about software and application, project setup.

#### 2.2 Software and Applications

1. **Visual Studio Code** : Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.
2. **Next.js** : Next.js is an open-source web development framework created by Vercel enabling React-based web applications with server-side rendering and generating static websites.
3. **Solidity**: Solidity is an object-oriented programming language for implementing smart contracts on various blockchain platforms, most notably, Ethereum. It was developed by Christian Reitwiessner, Alex Beregszaszi, and several former Ethereum core contributors. Programs in Solidity run on Ethereum Virtual Machine.

4. **Sanity.io** :Sanity is the platform for structured content that lets your team work together in real-time to build engaging digital experiences across channels. It is backend database for storing post.
5. **Pinata**: Pinata is an NFT media management service that allows users to host, manage and share files of any kind on the blockchain of their choice. As an IPFS pinning service, we focus on giving both technical and non-technical creators a fast, easy, and reliable way to share content without limits.
6. **Polygon's Mumbai Testnet** : **Mumbai** Testnet is used by web3 developers building applications on the Polygon sidechain.

## 2.3 Project SetUp

**Visual Studio Code** : in Visual Studio Code firstly we create new folder name SocialMediaBlockchain then I will install dependencies for my project .Then I will create folder and file for coding.

Then for Front-end setup by using command "npx create-react-app project".  
Then create account in sanity.io for start it use command" sanity start".  
Backend setup by using command " npx hardhat ".

Blockchain – Polygon\_Mumbai\_Testnet  
Writing Smart Contract - Solidity  
UI library – Next.js & Web3UIKit  
Solidity development environment - Hardhat  
File Storage – Web3 Storage

## 2.4 Summary

Blockchain technology is much more than a system for securely transferring cryptocurrencies. we will first write our own Solidity-based Smart Contract, Deploy that Smart Contract on Mumbai Testnet is used by web3 developers building applications on the Polygon sidechain .

A decentralised social network allows users more control. There is no central authority (person, server or company) that dictates on users and block their freedom.

## **Chapter 3:**

# **System Design**

### **3.1 Introduction**

This chapter focuses on the systematic approaches to be adopted to guide help in the starting to the completion of the project, it will mainly focus on a development life cycle model and its advantages and disadvantages. The chapter aims to conclude with why the methodology was chosen for the duration of the project and the reasons behind choosing it. If the correct methodology is chosen and followed correctly, it should help to ensure the project stays on schedule.

### **3.2 Life Cycle**

#### 3.2.1 Prototype Model

In Prototype Model, a throw-away prototype is built with potentially few features included to closely understand the requirements. The prototype is not the complete system because many of the features are not built in the prototype, it is simply a prototype of what the final system will look like so that the client/user can get close feel of the system before the final system is even built.

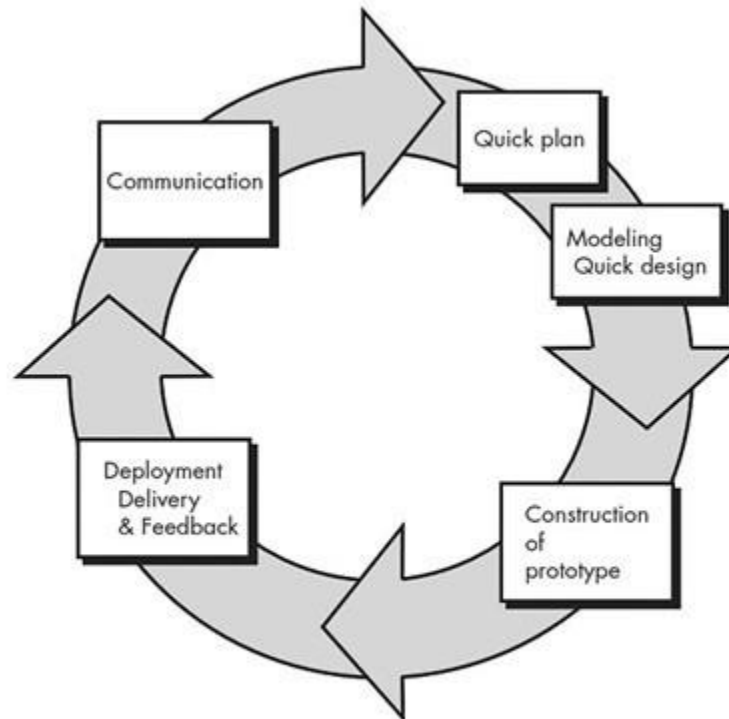


Figure: Prototype Model

#### **Advantages of Prototype Model to the project:**

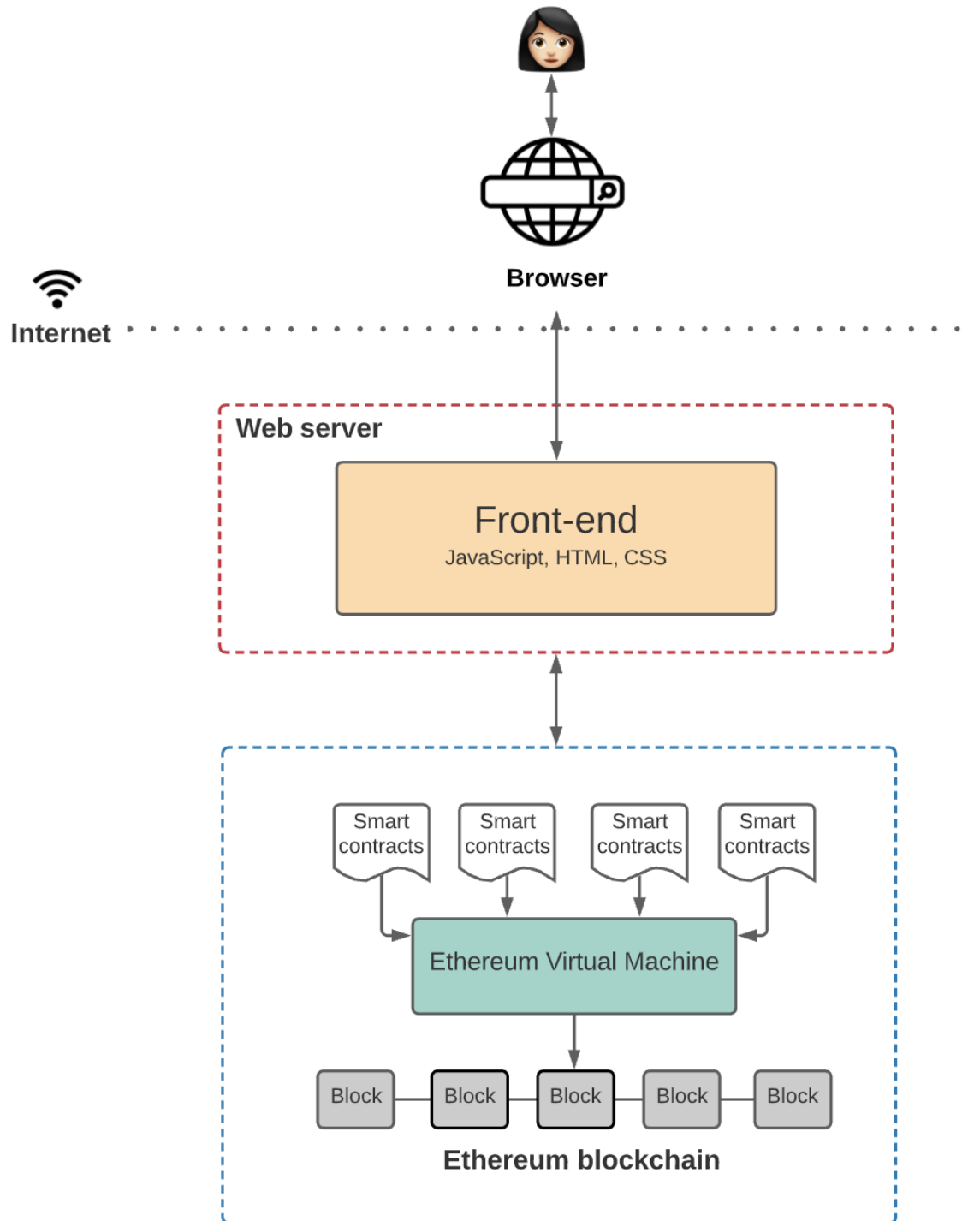
- Errors can be detected much earlier in the lifecycle since therefore requirements can be changed with the feedback of the client, who are the students and myself in this case, if the error is time consuming to fix, as the project will have to be delivered on time.
- Feedback can be gathered early on in the development lifecycle through prototypes, this means that the feedback received from supervisor and participants students can be applied earlier than later in the lifecycle, saving time and helping to complete the project on schedule. This also means that the final system will satisfy the problem identified, as feedbacks from both supervisors and students will be regularly inputted in very prototype developed.



**Disadvantages of Prototype Model to the project:**

- Prototype have to be regularly developed which means, this could lead to continuous implantation and repairing, which could be time consuming.
- This methodology could increase problems, such as getting attached to prototype built and using that prototype design to the final system rather than changing it according to the feedback received. This means the final system cannot be according to the requirements and the requirements are not met.

### 3.3 Data-Flow Diagram



### **3.4 Justification of Chosen Lifecycle**

The chosen lifecycle methodology to be followed by the system was Prototype Model. This methodology was chosen because it allows to work on different aspects of the system requirements separately, this allows to get feedbacks on different prototypes very quickly and allows changes to be made, so that the final system matches with the requirements specified. Furthermore, it also helps to see that the project schedule, Gantt chart, is accurate and achievable. The chosen lifecycle has different variations which can be used, chosen is that Evolutionary Prototyping variation is very effective for myself. It is first time making a system which requires different components to be researched and implemented because usually this was done in a group which meant shared workload and made it easier as each member were allocated certain aspects of the system. However, as this time it is an individual work, Evolutionary Prototyping lifecycle helps to create a prototype and keep adding new prototypes on top of the initial prototype and tested along with the components of the first prototype, i.e., Database, will have to bear in mind that the first prototype must be robust. This is better than the Waterfall Model as the waterfall model needs have all the requirements stated, along with the design and by the time it is coding stage, it could be late to make changes to the requirements, if a requirement proved to be hard to do or time-consuming, this could lead to delay in project schedule, prolong the delivery of the system and add additional constraints to other aspects of the project, i.e., Write-ups.

### 3.5 Methodology

Front-end setup by using command "npx create-react-app project". Then create account in sanity.io for start it use command" sanity start".

Back-end setup by using command "npx hardhat ".

Blockchain – Polygon\_Mumbai\_Testnet

Writing Smart Contract - Solidity

UI library – Next.js & Web3UIkit

Solidity development environment - Hardhat

File Storage – Web3 Storage

### 3.6 Summary

The chapter aims to conclude with why the methodology was chosen for the duration of the project and the reasons behind choosing it. If the correct methodology is chosen and followed correctly, it should help to ensure the project stays on schedule.

## Chapter 4:

# Implementation

### 4.1 Introduction

The project “Social Media Application using Blockchain” using Next.js, Solidity, sanity.io, pinata. So in this chapter we talk about coding and testing part.

### 4.2 Steps of implementaion

NFT images have a unique hexagon shape.

- Building a Web 3.0 Application with **Next.js**.
- Creating, auditing and deploying our smart contract using **Solidity**.
- Styling our app with **Tailwind CSS**.
- Adding **Web 3.0** authentication using **MetaMask**.
- Storing the tweets in **Sanity.io**.
- **Mint** our own **NFT profile picture** and store it on **Pinata**.
- Deploy and host the app on **Vercel**.

## 4.3 Web3 dapp

With decentralized applications or dApps, there is also a frontend client and a backend server. But unlike Web 2.0 applications, Web 3.0 eliminates the need for a centralized database and a centralized web server; instead, the Blockchain is used. The most used Blockchain platform for implementing dApps is Ethereum. Other Blockchain platforms that are also available for dApp implementation are Hyperledger Sawtooth, Hyperledger Fabric, EOS, etc.

**Frontend:** Like web 2.0 applications, the frontend of the dApp is written using HTML, CSS, and JavaScript programming languages.

**Web3.js:** Web3.js is a Javascript library that enables your frontend to interact with the Ethereum network. To connect with the Blockchain, you need to connect to a node. It can be done by running a node on your computer. And if you don't want to run a node, you can connect with a third party who is running the nodes, e.g., Infura. The nodes that you connect with to interact with the Blockchain network are often called "providers."

**Smart contract:** [Smart contract](#) is a self-executing program that runs on the Ethereum blockchain and defines the logic of a dApp. Smart contracts are written Solidity language. Because the smart contract code is stored on the Ethereum Blockchain, anyone on the network can inspect the code.

Additionally, smart contracts can also connect to any API by integrating Oracle. This helps the smart contracts to extract external data like weather information,

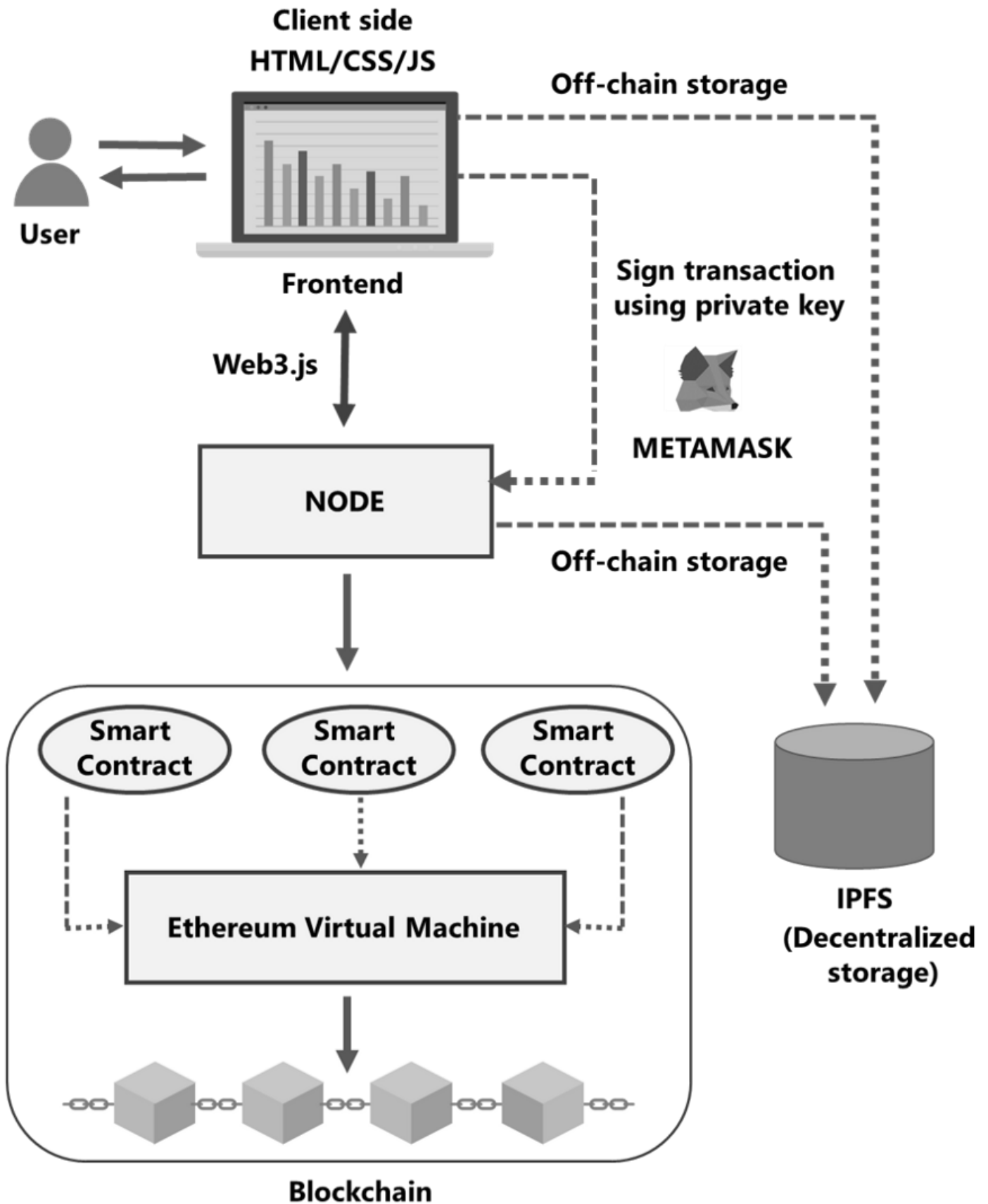
market data, etc. One can also make payments through a bank on dApp because of Oracle.

**Ethereum Virtual Machine or EVM:** EVM executes the logic defined in the smart contracts and processes the updates that happen on the globally accessible Blockchain. The EVM doesn't understand Solidity language, which is used to write smart contracts. Instead, you have to compile the code written in Solidity language into bytecode, which the EVM can then execute.

<b>How does the frontend of Web 3.0 dApp communicate with the Blockchain network?</b>
---

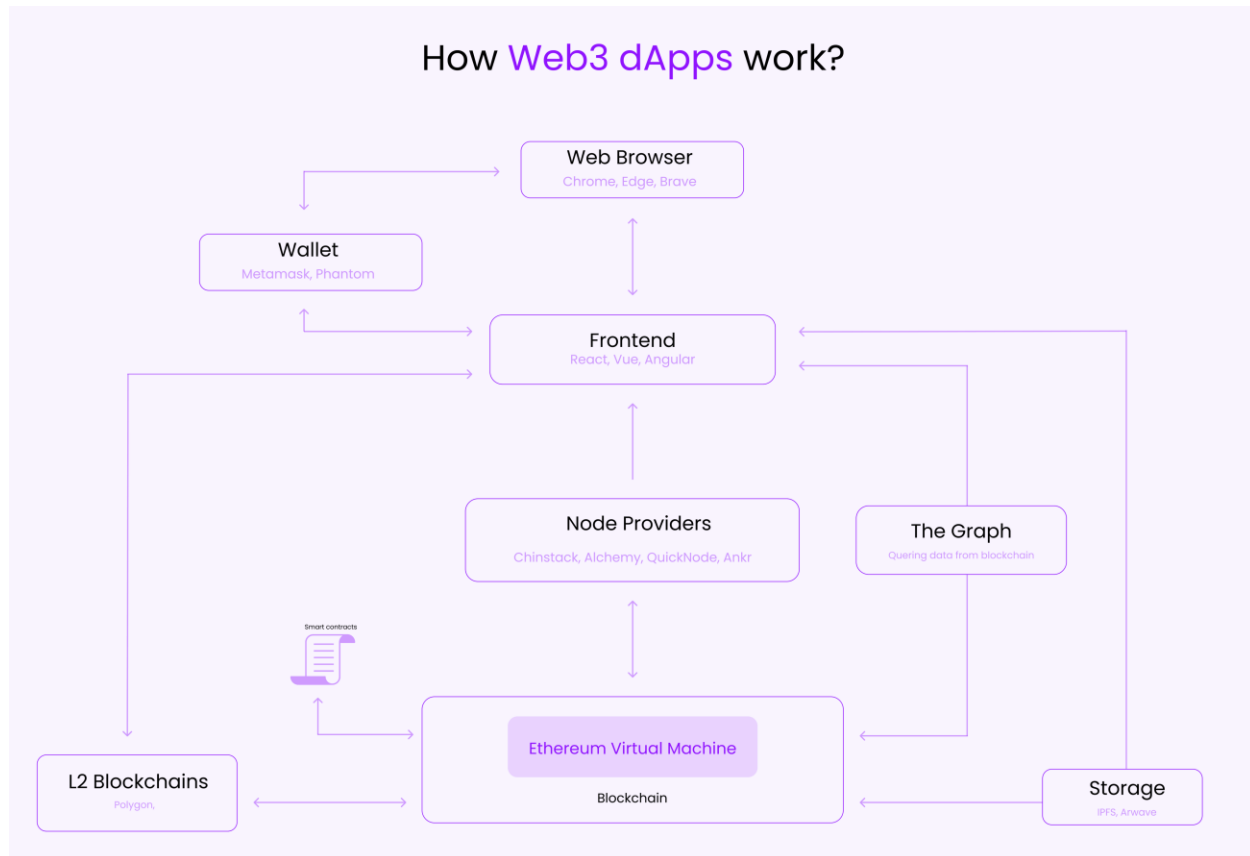
Because of the decentralized network, every node in the Blockchain network is required to keep a copy of the full Blockchain, including the code associated with smart contracts. Once you connect to the Blockchain through the node, you can submit the transaction to the Blockchain only after signing it using your private key. For instance, imagine you have a dApp that lets users view or post videos to the Blockchain. You might have a button on frontend that allows anyone to view the videos. Viewing videos on the Blockchain does not require a user to sign a transaction. However, when a user wants to post a new video onto the Blockchain, the dApp asks the user to sign the transaction using their private key; only then transaction, i.e., the new video will be posted. Otherwise, the nodes wouldn't accept the transaction.

For “signing” the transactions, the third-party Metamask is required, which stores the private keys of a user in the browser.





## 4.4 Working of Web3 dapp



The architecture of [Web 3.0](#) applications (or “DApps”) are completely different from Web 2.0 applications.

Take Medium, for example, a simple blogging site that lets users publish their own content and interact with content from others.

As a web 2.0 application, it may sound simple, but there’s a lot that goes into Medium’s architecture to make it all possible:

**First**, there must be a place to store essential data, such as users, posts, tags, comments, likes, and so on. This requires a constantly updated database.

**Second**, backend code (written in a language like Node.js, Java, or Python) must define Medium’s business logic. For example, what happens when a new user signs up, publishes a new blog, or comments on someone else’s blog?

**Third**, frontend code (typically written in JavaScript, HTML, and CSS) must define Medium’s UI logic.

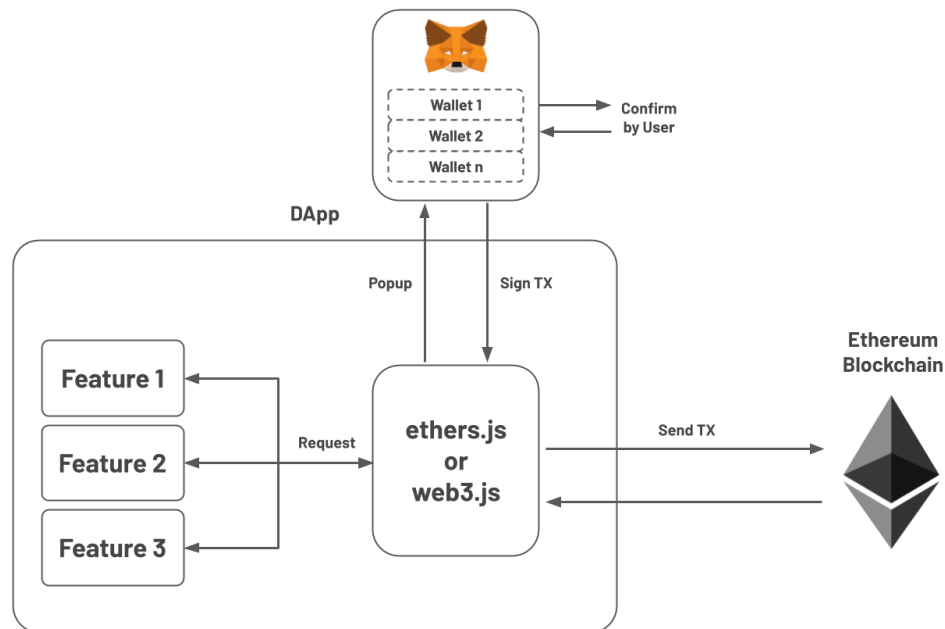
Web 3.0 DeFi (abbreviation for Decentralized Finance) Dapps (or decentralized applications) are the popular technologies that are secured using Blockchain. A DeFi dApp solution helps users perform seamless financial transactions, such as lending, borrowing, mortgaging, loaning, saving, and more. In addition, these platforms take advantage of crypto tenders (cryptocurrencies and stable coins) and collaterals such as NFTs that are an integral part of Web 3.0.

Some Web 3.0 applications have already established a strong market presence, including Sushiswap, PancakeSwap, and MakerDAO. Like these applications, investors can build secure and robust applications under the guidance of a Web 3.0 DeFi dApp development company.

These adopt the concept of decentralization, thereby making the user data susceptible to forgery. The transactions are clearly visible without disclosing the identity of the users. The information is shared with several devices; even the refrigerator working on IoT holds the necessary information.

1. Web 3.0 doesn't use centralized servers, so no server or server-based protocols are implemented during the process.
2. Blockchains are an integral part of DeFi dApp solution Most DeFi dApps operate on Ethereum and other networks such as Hyperledger Sawtooth, HyperLedger Fabric, EOS, and Polygon.
3. The front end of decentralized applications is HTML, CSS, and JavaScript, which are a crucial part of existing Web 2.0 applications.
4. Web 3.0 DeFi dApp development is incomplete without smart contracts. Their code is written in Solidity, and the smart contracts execute when certain predefined conditions are fulfilled. These have a complete financial history and have become a preferred choice of decentralized finance companies globally.
5. Smart Contracts run on Ethereum Virtual Machine and, according to the Blockchain information, are updated. Since they are Ethereum-based, smart contracts can run on various blockchains.

## 4.5 Working of Metamask



## 4.6 Front-end implementaion detail

1.Client folder : inside client we work on frond-end part.

A decentralized application (dapp) where users can upload posts to the **network** as well as mint their own profile image as **NFT**.

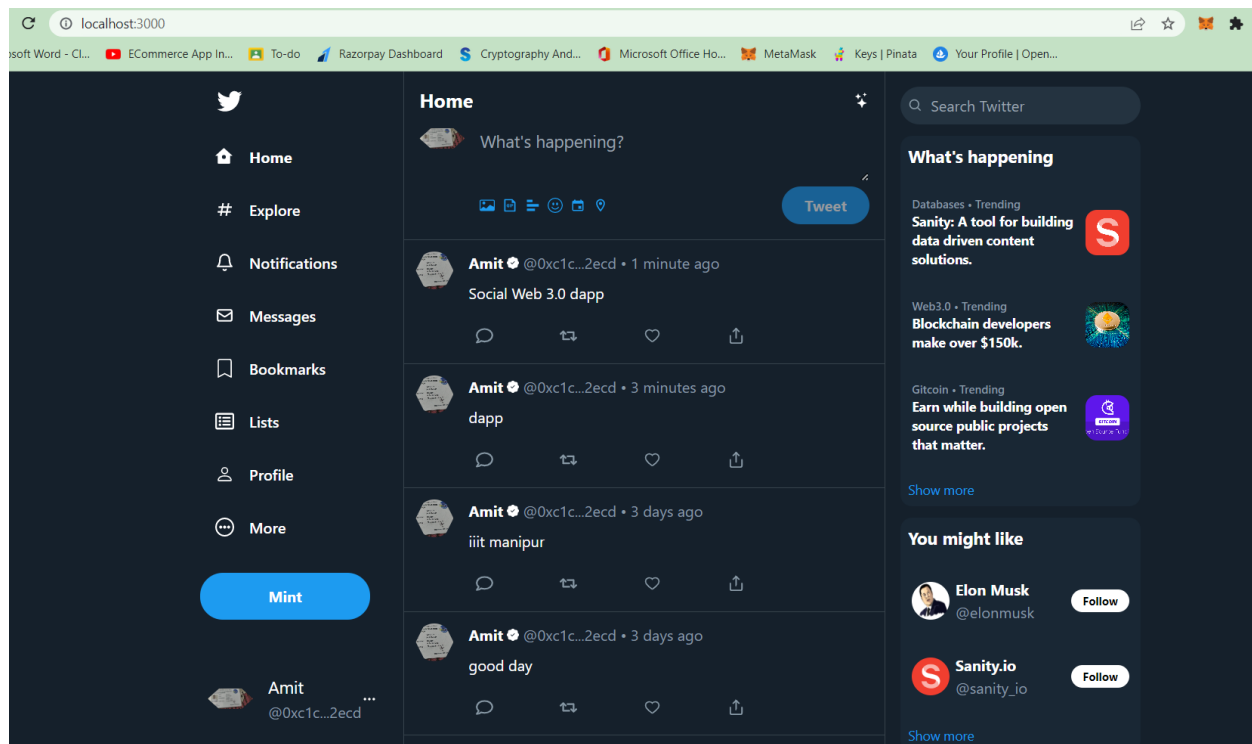
- Building a Web 3.0 Application with **Next.js**.
- Open package.json and add the following scripts:
  - "scripts": {
  - "dev": "next dev",
  - "build": "next build",
  - "start": "next start",
  - "lint": "next lint"
  - }
- These scripts refer to the different stages of developing an application:
  - dev - Runs next dev to start Next.js in development mode
  - build - Runs next build to build the application for production usage

- start - Runs next start to start a Next.js production server
- lint - Runs next lint to set up Next.js' built-in ESLint configuration
- Create two directories pages and public at the root of your application:
- pages - Associated with a route based on their file name. For example pages/about.js is mapped to /about
- public - Stores static assets such as images, fonts, etc. Files inside public directory can then be referenced by your code starting from the base URL (/).
- Next.js is built around the concept of pages. A page is a React Component exported from a .js, .jsx, .ts, or .tsx file in the pages directory. You can even add dynamic route parameters with the filename.
- Inside the pages directory add the index.js file to get started. This is the page that is rendered when the user visits the root of your application.
- Populate pages/index.js with the following contents:
- ```
function HomePage() {
```
- ```
  return <div>Welcome to Next.js!</div>
```
- ```
}
```
- 
- ```
export default HomePage
```
- After the set up is complete:
- After the installation is complete:
- Run npm run dev or yarn dev or pnpm dev to start the development server on http://localhost:3000
- Visit http://localhost:3000 to view your application
- Edit pages/index.js and see the updated result in your browser
- Next.js based on react.
- Styling our app with **Tailwind CSS**.
- **Advantages of Tailwind CSS:**
- Highly Customizable.
- Enables building complex responsive layout.
- Responsive and development is easy.
- Components creation is easy.

- **Disadvantages of Tailwind CSS:**
- There are missing headers, and navigation components.
- It takes time to learn how to implement inbuilt classes.

After completing it.Type command to see result yarn build.

Then yarn run dev. See result in localhost:3000

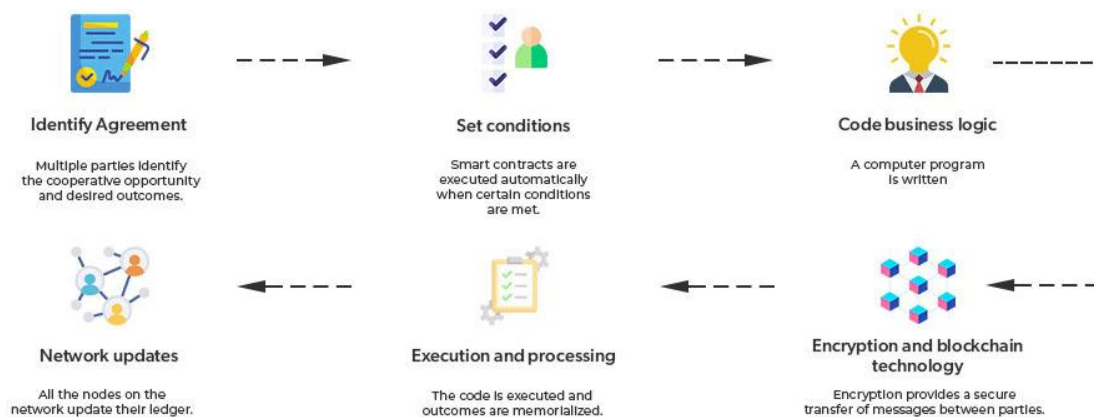


## 4.7 Smart Contract implementaion detail

- Smart contracts are self-executing lines of code with the terms of an agreement between buyer and seller automatically verified and executed via a computer network.
- Nick Szabo, an American computer scientist who invented a virtual currency called "Bit Gold" in 1998, defined smart contracts as computerized transaction protocols that execute terms of a contract.
- Smart contracts deployed to blockchains render transactions traceable, transparent, and irreversible.

- We're going to create a simple smart contract that implements a token that can be transferred. Token contracts are most frequently used to exchange or store value. We won't go in depth into the Solidity code of the contract on this tutorial, but there's some logic we implemented that you should know:
- There is a fixed total supply of tokens that can't be changed.
- The entire supply is assigned to the address that deploys the contract.
- Anyone can receive tokens.
- Anyone with at least one token can transfer tokens.
- The token is non-divisible. You can transfer 1, 2, 3 or 37 tokens but not 2.5.

## How does a Smart Contract Work?



in Smart contract folder : here we work on

- Creating, auditing and deploying our smart contract using **Solidity**.
- Adding **Web 3.0** authentication using **MetaMask**.

```

• /**
•  * Checks if there is an active wallet connection
•  */
• const checkIfWalletIsConnected = async () => {
•   if (!window.ethereum) return setAppStatus('noMetaMask')
•   try {
•     const addressArray = await window.ethereum.request({

```

```

    method: 'eth_accounts',
  })
  if (addressArray.length > 0) {
    setAppStatus('connected')
    setCurrentAccount(addressArray[0])

    createUserAccount(addressArray[0])
  } else {
    router.push('/')
    setAppStatus('notConnected')
  }
} catch (err) {
  router.push('/')
  setAppStatus('error')
}
}

/**
 * Initiates MetaMask wallet connection
 */
const connectToWallet = async () => {
  if (!window.ethereum) return setAppStatus('noMetaMask')
  try {
    setAppStatus('loading')

    const addressArray = await window.ethereum.request({
      method: 'eth_requestAccounts',
    })

    if (addressArray.length > 0) {
      setCurrentAccount(addressArray[0])
      createUserAccount(addressArray[0])
    } else {
      router.push('/')
      setAppStatus('notConnected')
    }
  } catch (err) {
    setAppStatus('error')
  }
}

const createUserAccount = async (userAddress = currentAccount) => {
  if (!window.ethereum) return setAppStatus('noMetaMask')
  try {
    const userDoc = {

```

```

•     _type: 'users',
•     _id: userAddress,
•     name: 'Amit',
•     isProfileImageNft: false,
•     profileImage:
•         'https://about.twitter.com/content/dam/about-twitter/en/brand-
•         toolkit/brand-download-img-1.jpg.twimg.1920.jpg',
•     walletAddress: userAddress,
•     }
•
•     await client.createIfNotExists(userDoc)
•
•     setAppStatus('connected')
• } catch (error) {
•     router.push('/')
•     setAppStatus('error')
• }
• }

```

- Back-end setup by using command “npx hardhat compile”.
- Writing automated tests when building smart contracts is of crucial importance, as your user's money is what's at stake.
- To test our contract, we are going to use Hardhat Network, a local Ethereum network designed for development. It comes built-in with Hardhat, and it's used as the default network. You don't need to setup anything to use it.

```

• require("@nomiclabs/hardhat-waffle");
• require("dotenv").config();
•
• const privateKey = process.env.DEPLOYER_SIGNER_PRIVATE_KEY;
•
• module.exports = {
•     solidity: "0.8.17",
•     networks: {
•         goerli: {
•             url: 'https://eth-
•             goerli.g.alchemy.com/v2/qgaQewusGJERrhvFq1zmhrc2rZIwZqsT',

```

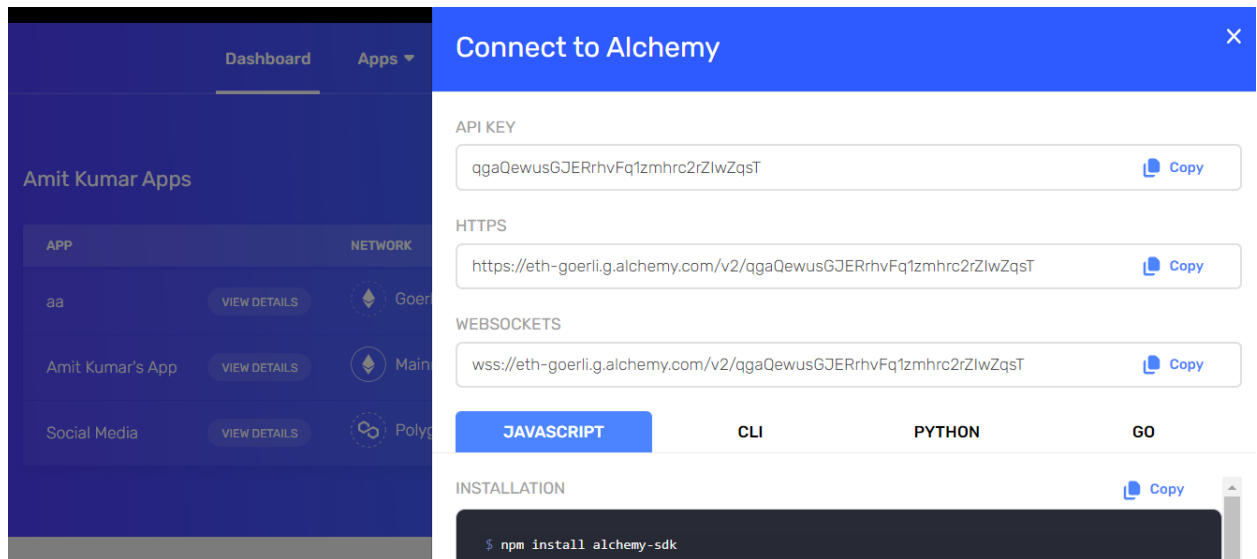


```

•   accounts:[
•   'abcc7c603c02299d02d0be46fa5bd57beb982bd966db8927841e0d6f9fce1c50'
•   ,
•   ],
•   },
•   },
•   };

```

- Here add goerli network , url of alchemy and metamask account.



3. Set up Sanity which is backend Database.

```

{
  "root": true,
  "project": {
    "name": "social-app"
  },
  "api": {
    "projectId": "cfd5x2km",
    "dataset": "production"
  },
  "plugins": [
    "@sanity/base",
    "@sanity/default-layout",
    "@sanity/default-login",
    "@sanity/desk-tool"
  ],
  "env": {
    "development": {
      "plugins": [

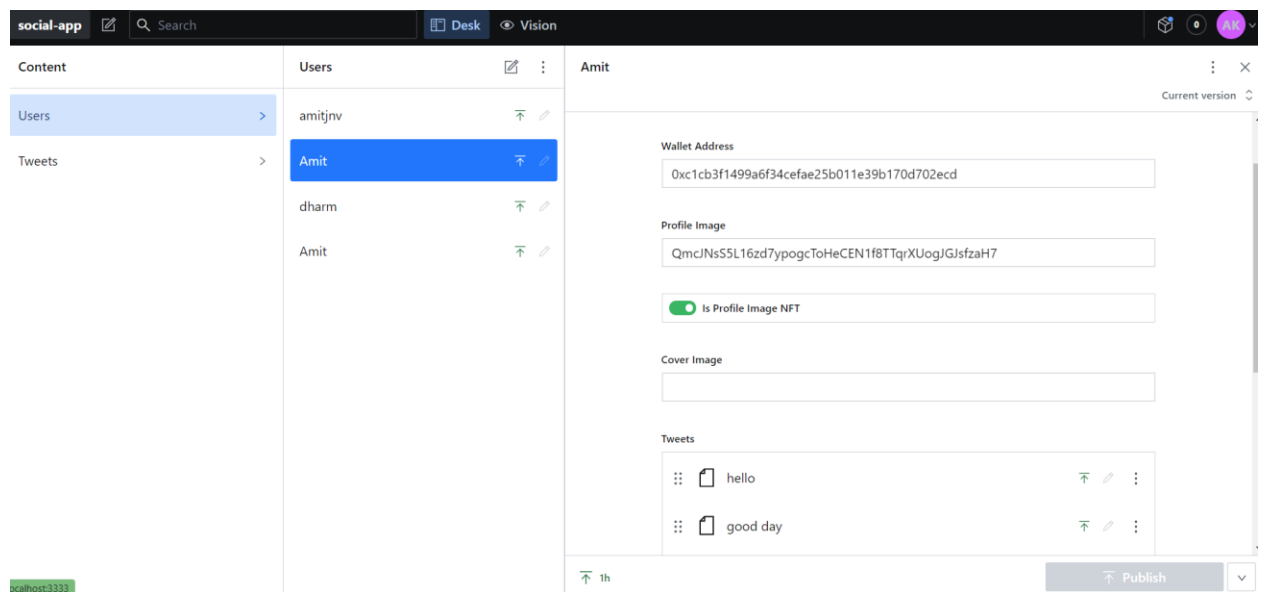
```

```

    "@sanity/vision"
  ]
},
"parts": [
  {
    "name": "part:@sanity/base/schema",
    "path": "./schemas/schema"
  }
]
}

```

- Storing the tweets in **Sanity.io**.
- After completing it. go to "cd studio" and "sanity start" command .
- See result in localhost:3333
- In content we have Users and Tweets .

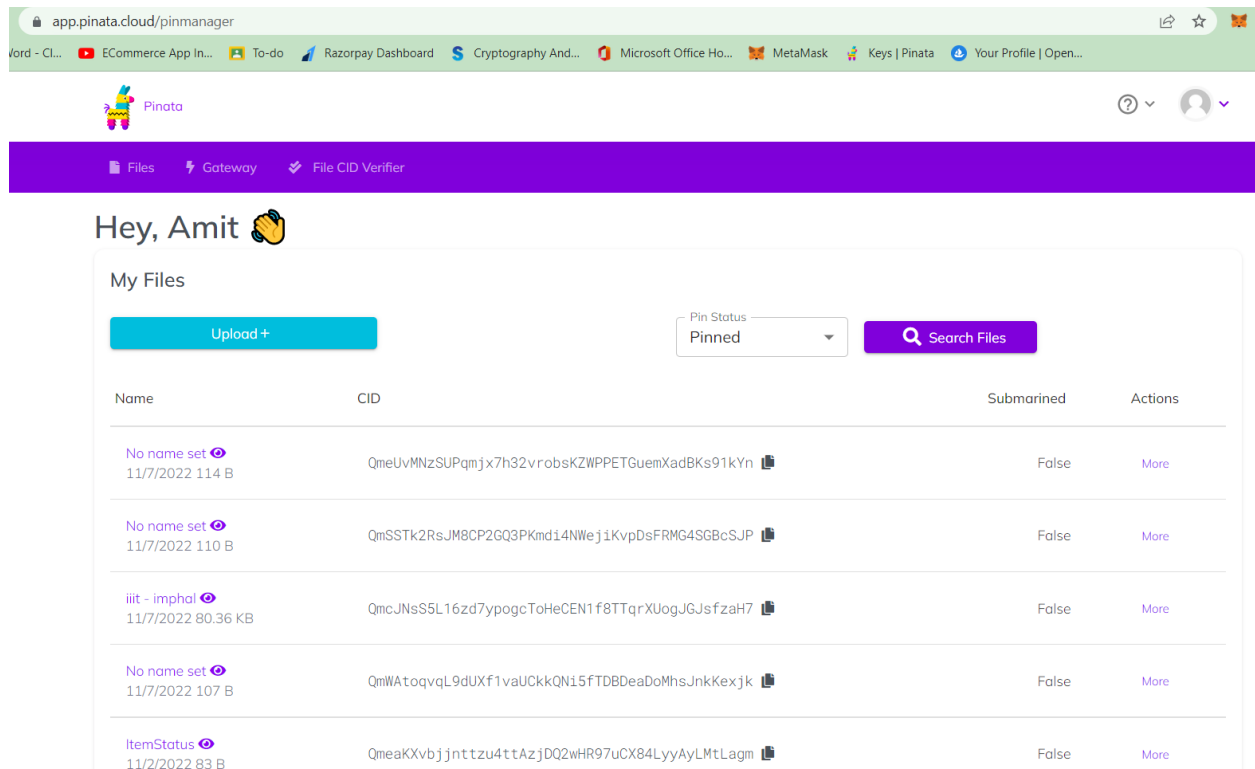


## Chapter 5:

# Result analysis and Testing

**Mint** our own **NFT profile picture** and store it on **Pinata**.

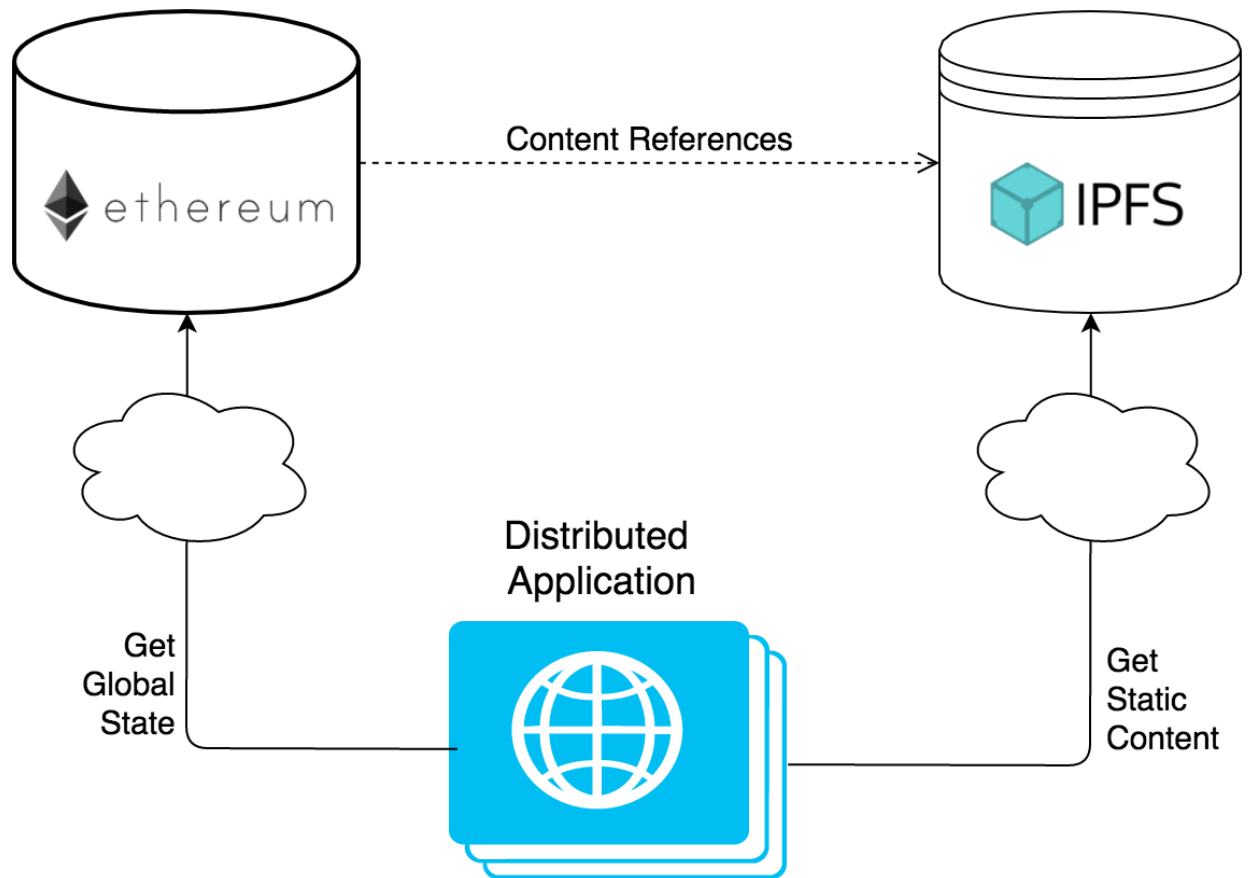
Storage on Blockchain: The smart contracts are stored on the Ethereum blockchain. But storing every dApp data on the Blockchain gets really expensive. The user needs to pay every time they add new data to the Blockchain. Asking your users to pay extra fees for every new transaction on your dApp is not a good experience for the user. This issue can be solved by using a decentralized off-chain storage solution, like the Interplanetary File system abbreviated as IPFS. Instead of storing data in a centralized database, the IPFS system distributes and stores the data in a peer-to-peer network.



The screenshot shows the Pinata web interface at [app.pinata.cloud/pinmanager](https://app.pinata.cloud/pinmanager). The interface includes a navigation bar with links for Files, Gateway, and File CID Verifier. Below the navigation bar, there is a greeting "Hey, Amit" and a section titled "My Files". This section contains an "Upload +" button, a "Pin Status" dropdown menu set to "Pinned", and a "Search Files" button. A table lists the files stored on IPFS, with columns for Name, CID, Submerged, and Actions.

Name	CID	Submerged	Actions
No name set 11/7/2022 114 B	QmeUvMNzSUPqmjx7h32vrobsKZWPETGuemXadBKs91kYn	False	More
No name set 11/7/2022 110 B	QmSSTk2RsJM8CP2GQ3PKmd14NWej1KvpDsFRMG4SGBcSJP	False	More
iiit - imphal 11/7/2022 80.36 KB	QmcJNsS5L16zd7ypogcToHeCEN1f8TTqrXUogJGJsfaH7	False	More
No name set 11/7/2022 107 B	QmWAtovqL9dUXf1vaUCkkQN15FTDBDeaDoMhsJnkKexjk	False	More
ItemStatus 11/2/2022 83 B	QmeaKXvbjjnttzu4ttAzjDQ2wHR97uCX84LyyAyLmtLagm	False	More

Here IPFS storage for storing NFT profile.



To link to content on IPFS by

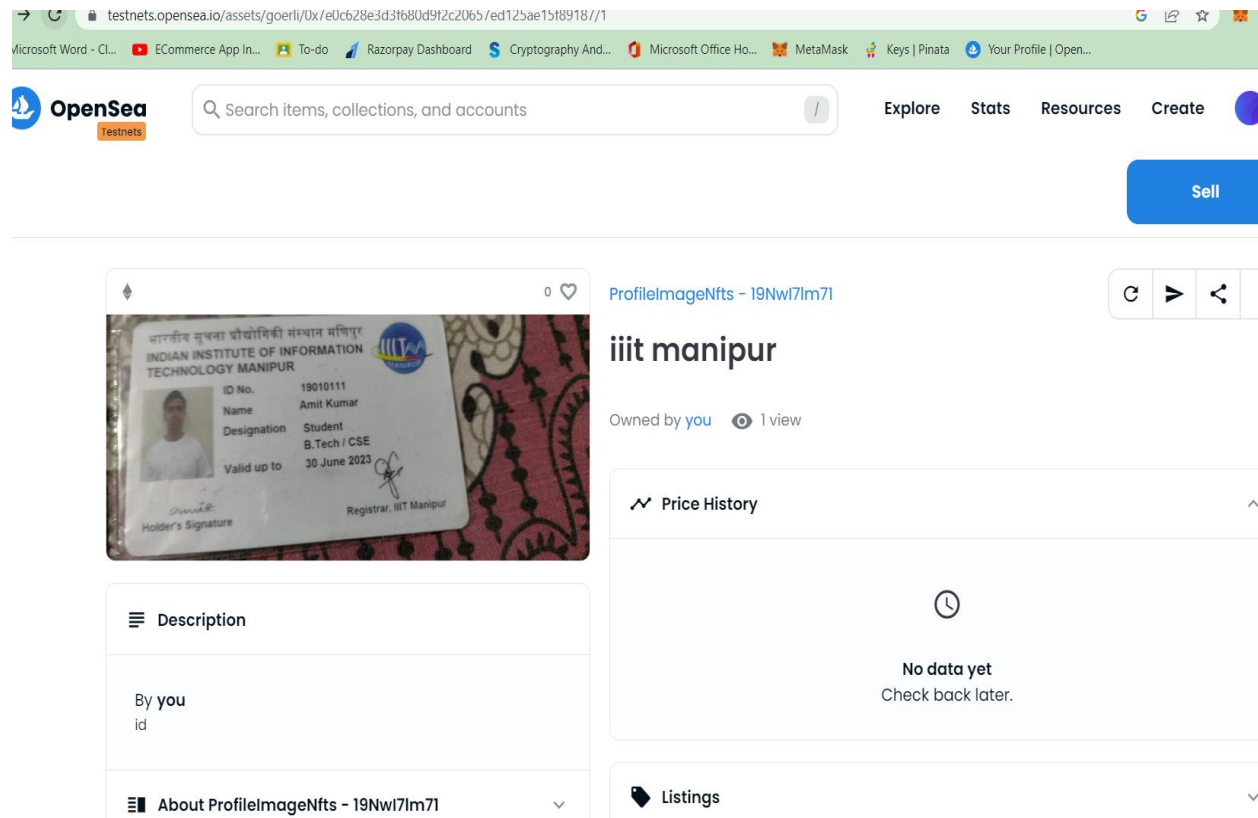
"https://ipfs.io/ipfs/<CID>"

For connecting Sanity Database and Pinata with Frond-end in client folder  
".env".

```

NEXT_PUBLIC_SANITY_PROJECT_ID=cfd5x2km
NEXT_PUBLIC_SANITY_TOKEN=skvpTYMmT1oRzAfVqk1poWACm6NtRIDcFdbU8xZUFV4WnU6V7jCOLXHG
yBXogFF0vS0t9mSfERHp7GIPLD4jP9TwCPd4ugQuGox7GCJsm474T1BLmYeKISJY1RyMtUy5D27sru3fX
m6NzuFsKNQmae3eEPj1d1NQFCurJ1x4CmssUWJVoE3T
NEXT_PUBLIC_PINATA_API_KEY=8fdec48249e364d3ce43
NEXT_PUBLIC_PINATA_API_SECRET=ad8a51e78cc388e3f1958320a28e2bcfed4536cc7990c6ffffa
2b9b43fd34bfd
  
```

OpenSea is the world's first and largest web3 marketplace for NFTs and crypto collectibles. Browse, create, buy, sell, and auction NFTs using OpenSea today.



## 5.1 Deploy and host app on Vercel

To download and install Vercel CLI, run the following command:  
`npm i -g vercel.`

### Next.js Build API :

next build generates an optimized version of your application for production. This standard output includes:

- HTML files for pages using `getStaticProps` or [Automatic Static Optimization](#)
- CSS files for global styles or for individually scoped styles
- JavaScript for pre-rendering dynamic content from the Next.js server
- JavaScript for interactivity on the client-side through React.

This output is generated inside the .next folder:

- .next/static/chunks/pages – Each JavaScript file inside this folder relates to the route with the same name. For example, .next/static/chunks/pages/about.js would be the JavaScript file loaded when viewing the /about route in your application
- .next/static/media – Statically imported images from next/image are hashed and copied here
- .next/static/css – Global CSS files for all pages in your application
- .next/server/pages – The HTML and JavaScript entry points prerendered from the server. The .nft.json files are created when [Output File Tracing](#) is enabled and contain all the file paths that depend on a given page.
- .next/server/chunks – Shared JavaScript chunks used in multiple places throughout your application
- .next/cache – Output for the build cache and cached images, responses, and pages from the Next.js server. Using a cache helps decrease build times and improve performance of loading images

All JavaScript code inside .next has been **compiled** and browser bundles have been **minified** to help achieve the best performance.

### Hardhat to connect to a specific network:

We can use the --network parameter when running any task, like this:

```
npx hardhat run scripts/deploy.js --network goerli
```

With our current configuration, running it without the --network parameter would cause the code to run against an embedded instance of Hardhat Network. In this scenario, the deployment actually gets lost when Hardhat finishes running, but it's still useful to test that our deployment code works:

```
$ npx hardhat run scripts/deploy.js
```

Deploying contracts with the account:

0xf39Fd6e51aad88F6F4ce6aB8827279cFfB92266

Account balance: 10000000000000000000000

Token address: 0x5FbDB2315678afecb367f032d93F642f64180aa3

```
{
  "name": "hardhat-project",
  "dependencies": {
    "@openzeppelin/contracts": "^4.6.0",
    "dotenv": "^16.0.3"
  },
  "devDependencies": {
    "@nomiclabs/hardhat-waffle": "^2.0.3",
    "ethereum-waffle": "^3.4.4",
    "hardhat": "^2.12.2"
  }
}
```

## Managed Next.js with Vercel:

Vercel is the fastest way to deploy your Next.js application with zero configuration.

When deploying to Vercel, the platform automatically detects Next.js, runs next build, and optimizes the build output for you, including:

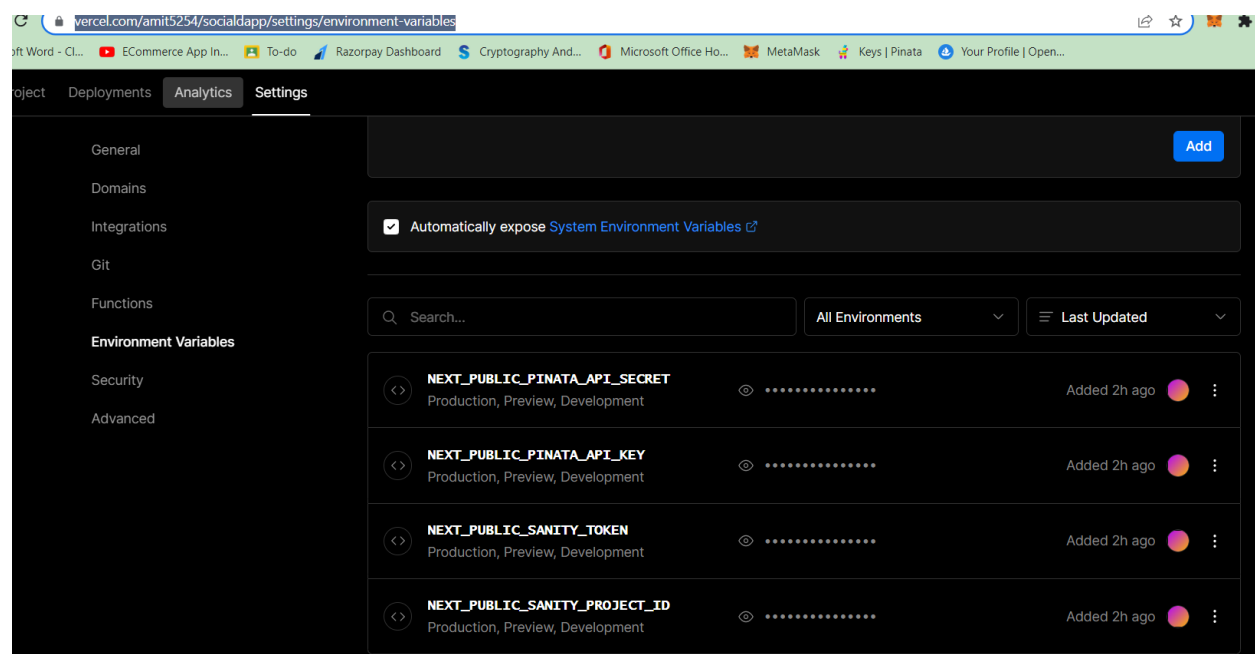
- Persisting cached assets across deployments if unchanged
- Immutable deployments with a unique URL for every commit
- Pages are automatically statically optimized, if possible
- Assets (JavaScript, CSS, images, fonts) are compressed and served from a Global Edge Network
- API Routes are automatically optimized as isolated Serverless Functions that can scale infinitely
- Middleware are automatically optimized as Edge Functions that have zero cold starts and boot instantly

In addition, Vercel provides features like:

- Automatic performance monitoring with Next.js Analytics
- Automatic HTTPS and SSL certificates
- Automatic CI/CD (through GitHub, GitLab, Bitbucket, etc.)
- Support for Environment Variables
- Support for Custom Domains
- Support for Image Optimization with next/image
- Instant global deployments via git push.

## Adding of environment variables from .env file in vercel:

```
NEXT_PUBLIC_SANITY_PROJECT_ID=cfd5x2km
NEXT_PUBLIC_SANITY_TOKEN=skvpTYMmT1oRzAfVqk1poWACm6NtRIDcFdbU8xZUFV4WnU6V7jCOLXHGYBXogFF0vS0t9mSfERHp7GIPLD4jP9TwCPd4ugQuGox7GCJsm474T1BLmYeKISJY1RyMtUy5D27sru3fXm6NzuFsKNQmae3eEPj1d1NQFCurJ1x4CmssUWJVoE3T
NEXT_PUBLIC_PINATA_API_KEY=8fdec48249e364d3ce43
NEXT_PUBLIC_PINATA_API_SECRET=ad8a51e78cc388e3f1958320a28e2bcfed4536cc7990c6ffffa2b9b43fd34bfd
```



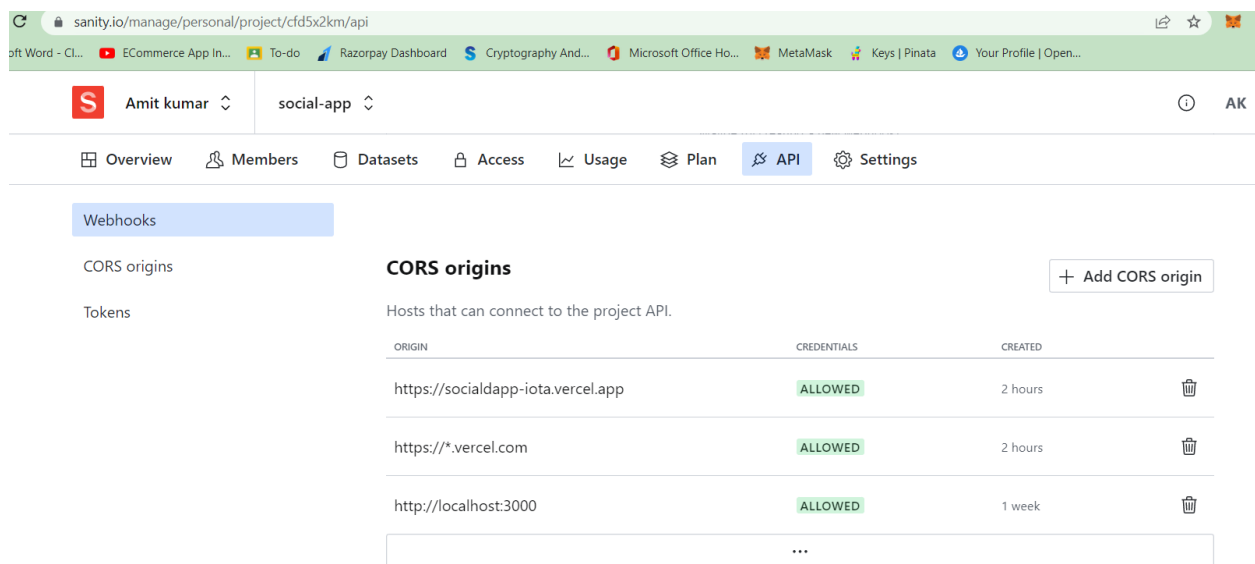
Apart from development and production environments, there is a 3rd option available: test. In the same way you can set defaults for development or production environments, you can do the same with a .env.test file for the testing environment (though this one is not as common as the previous two). Next.js will not load environment variables from .env.development or .env.production in the testing environment.

This one is useful when running tests with tools like jest or cypress where you need to set specific environment vars only for testing purposes. Test default values will be loaded if NODE\_ENV is set to test, though you usually don't need to do this manually as testing tools will address it for you.



There is a small difference between test environment, and both development and production that you need to bear in mind: `.env.local` won't be loaded, as you expect tests to produce the same results for everyone. This way every test execution will use the same env defaults across different executions by ignoring your `.env.local` (which is intended to override the default set).

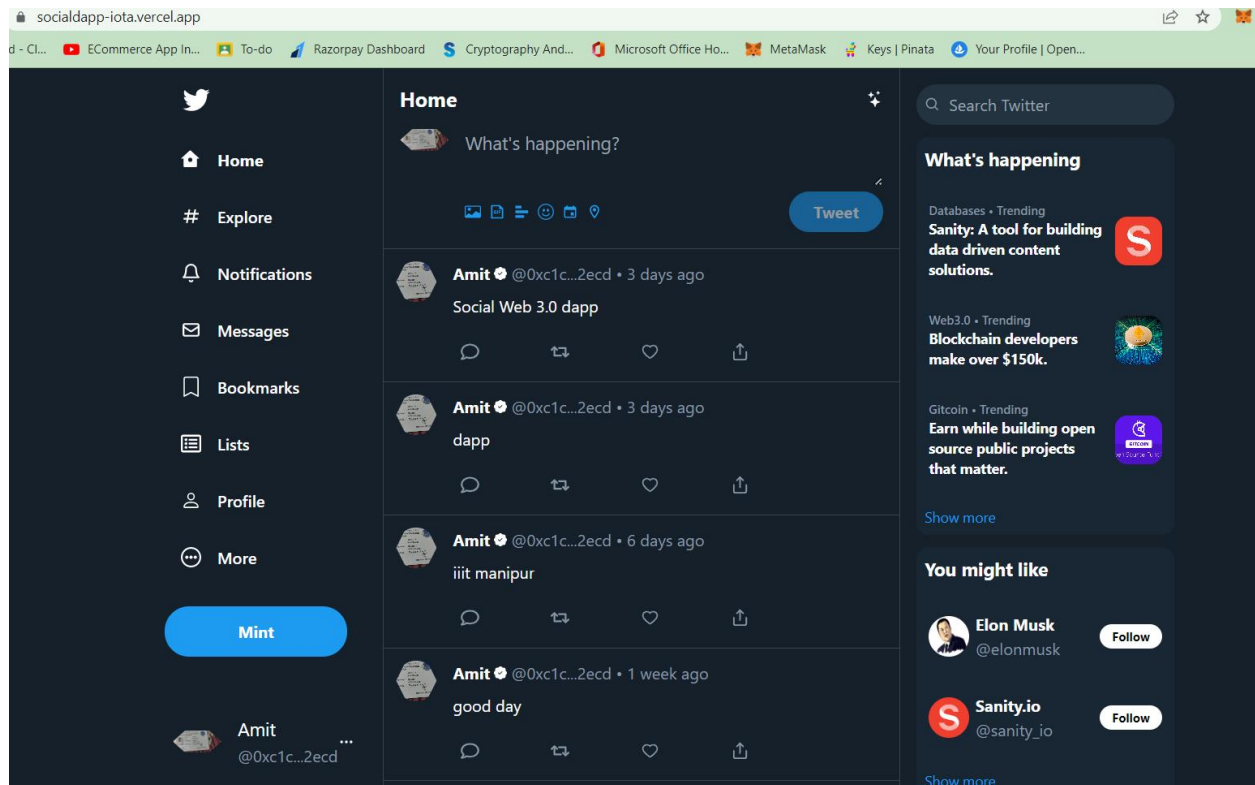
## Then allowed Hosts that connect to Project API:



The screenshot shows the Sanity.io management interface for a project named 'social-app'. The 'API' tab is selected, and the 'CORS origins' section is active. A sidebar on the left lists 'Webhooks', 'CORS origins', and 'Tokens'. The main content area displays a table of allowed CORS origins with columns for 'ORIGIN', 'CREDENTIALS', and 'CREATED'. Three origins are listed: 'https://socialdapp-iota.vercel.app', 'https://\*.vercel.com', and 'http://localhost:3000', all marked as 'ALLOWED'. A '+ Add CORS origin' button is in the top right. A table with 3 columns (ORIGIN, CREDENTIALS, CREATED) and 3 rows of CORS origins is shown. The origins are https://socialdapp-iota.vercel.app, https://\*.vercel.com, and http://localhost:3000. All are marked as ALLOWED. The created times are 2 hours, 2 hours, and 1 week respectively. A trash icon is next to each row. A '+ Add CORS origin' button is in the top right.

ORIGIN	CREDENTIALS	CREATED
https://socialdapp-iota.vercel.app	ALLOWED	2 hours
https://*.vercel.com	ALLOWED	2 hours
http://localhost:3000	ALLOWED	1 week

After successfully deployed and Host:  
See result:



## **Chapter 6:**

### **Conclusion**

Social media allows individuals to keep in touch with friends and extended family. Some people will use various social media applications to network and find career opportunities, connect with people across the globe with like-minded interests, and share their own thoughts, feelings, and insights online.

- Users can create post, like them, tag people, and share the post.
- Users can share text, images, gif and video on the post.
- Users can mint their nft profile and set it as their avatar.
- Users can become the member of token gated communities
- Crypto Communities can create their own gated communities for members and followers
- Community member need to have token or community nft to join a particular community as member.

Blockchain Social Media are decentralized networking platforms built using blockchain protocols/platforms that allow the development of applications and smart contracts. To name a few, Ethereum, Steem, Stellar are some of the blockchain protocols that support the development of social media dApps. Being decentralized, Blockchain social media networks are not under any central proprietary authority holding all the data. Rather, the data gets stored in a homogenous and decentralized manner across servers of each node of the network. Blockchain-based social media platforms support social networking, content sharing, and even blogging, but being decentralized, they enable end-to-end encryptions for every interaction.

## 6.1 How Will Web3.0 Change Future of Social Media?

Web3 will redefine how we socialize. Designed to be community-driven, ad-free, and self-monetized, Web3 is built on block chain technology where, instead of having large corporations owning all of the content and acting as gatekeepers, it serves as a collective of decentralized networks.

- **Expressing our individuality**

Features in Web3 allow users to customize their avatar's skin, hair and clothes depending on their mood, and you can buy, sell and wear outfits so creative and fantastical they would not be possible to achieve in real life.

- **Re-building sense of community**

We will bond over worlds, spaces, and activities in the metaverse. Whether we are attending Burning Man in VR, or we are part of an NFT collection and its associated Discord community, Web3 allows us to reconnect.

- **Giving power back to creators.**

### ***Advantages of dApps***

**(i) Open source:** The dApp code is open source, which means the code is transparent, and anyone can look at the code. This allows a lot more trust in the application.

**(ii) Censorship Resistant:** dApps are censorship-resistant. It means they don't have to answer government, corporations, or specific people. In the case of web 2.0 applications, anything or any service which may violate codified laws and is against the interests of powerful political groups or government, etc., is at risk. This is why Facebook or YouTube removes content that could be considered objectionable or is not conducive to their growth. But this is not possible with dApps. If the smart contract is coded to do something and is triggered, nobody

can stop it. Censorship resistance ensures that no one on the network can block users from submitting transactions, deploying dApps, or reading data from the Blockchain unless the code is written to do this.

**(iii) No downtime:** Another essential benefit of dApps is that there is no downtime, or the app will never go offline. There have been times when Facebook or YouTube are down, maybe because of a single point of failure like the servers have a bug, or there is an outage. Since dApps run on Blockchain, they will actually never go down because they are run by hundreds and thousands of computers all around the world. It would be infeasible to turn them all off.

### **Disadvantages of Dapp**

- (i) It is challenging to modify smart contracts once deployed.
- (ii) Also, it becomes difficult to fix any issues in dApps because fixes require every node in the network to update all the copies in the network.

## 5.2 Bibliography

<https://dashboard.alchemy.com/apps/t5fyma69koo1t5cy>

<https://www.sanity.io/manage/personal/project/cfd5x2km/api>

<https://vercel.com/amit5254/socialdapp/settings/environment-variables>

<https://socialdapp-iota.vercel.app/>

<https://gateway.pinata.cloud/ipfs/QmcJNsS5L16zd7ypogcToHeCEN1f8TTqrXUogJGJsfaH7>

<https://app.pinata.cloud/pinmanager>

<https://docs.ipfs.tech/how-to/address-ipfs-on-web/>

<https://goerli.etherscan.io/tx/0x08ff679ef30453f85f913188444364eb15a5061a0c1ca4b7d1e334f5e876d868>