INSTITUTE OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
BUNDELKHAND UNIVERSITY, JHANSI

A Major Report

on

"PIXELVAULT – A NFT MARKETPLACE"

Submitted to the Department of Computer Science & Engineering for the fulfilment of Four Years Degree Bachelor of Technology in Computer Science and Engineering

Session 2023-2024

15/05/24

GUIDED BY
Er. Manoj Verma
Assistant Professor
Dept. of Computer Science
& Engineering
IET BU, Jhansi

COORDINATOR
Er. Priyanka Pande
Head of Department
Dept. of Computer Science
& Engineering
IET BU, Jhansi

PROJECT IN CHARGE
Dr. Sadik Khan
Assistant Professor
Dept. of Computer Science
& Engineering
IET BU, Jhansi

SUBMITTED BY:
1. Abhay Singh – 201381030001
2. Abhishek Singh - 201381030004 (Team Leader)
3. Aditya Gautam - 201381030007
4. Aditya Raj Pathak - 201381030008
5. Amarendra Singh Yadav - 201381030013
6. Tanu - 191381030060

INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BUNDELKHAND UNIVERSITY, JHANSI

तार : विश्वविद्यालय
Gram : UNIVERSITY

टेलीफोन : कार्यालय : 2320496
मुख्यालय : निवास : 2321214
फैक्स : 0510 : 2321467

बुन्देलखण्ड विश्वविद्यालय, झाँसी
BUNDELKHAND UNIVERSITY, JHANSI
झाँसी (उ.प्र.) 284128

CERTIFICATE

To whomsoever it may concern

This is to certify that **Abhay Singh (201381030001), Abhishek Singh (201381030004), Aditya Gautam (201381030007), Aditya Raj Pathak (201381030008), Amarendra Singh Yadav (201381030013) and Tanu (191381030060)** are bonafide students of Computer Science & Engineering Final year in the academic year of 2023-24 have shown their competence in completing Project Report on **"PIXELVAULT – A marketplace for NFT"** for the fulfilment of the requirements for the award of the degree of Bachelor of Technology, Bundelkhand University, Jhansi (U.P.). Their work is highly appreciable.

Under the Guidance of

Er. Manoj Verma
Assistant Professor
Dept. of Computer Science
& Engineering
IET BU, Jhansi

ACKNOWLEDGEMENT

It gives us immense pleasure to present this report. We, the students of Computer Science and Engineering, were given, as a part of our curriculum, to make a Project on a topic of our choice. Our group choose **"PIXELVAULT – A marketplace for NFT"** as our project. We take this privilege to express our gracious thanks and regards to **Er. Manoj Verma** our guide and the project in-charge **Dr. Sadik Khan** for providing his continuous assistance and coordination.

We express our sincere and profound sense of gratitude to our learned and respected **Head of the Department Er. Priyanka Pande, Dr. Lalit Kumar Gupta, Er. B.P. Gupta, Er. Vijay Kr. Verma, Er. B.B. Niranjan, Dr. Sadik Khan, Er. Keshav Tiwari, Er. Akhilesh Kumar and Er. Manoj Verma** without whom realizing the project was a tough job.

Lastly, I would like to thank the Almighty, my parents and my friends for their constant encouragement and invaluable support from the conceptualization and culmination of the project, without which this task would not be possible.

Abhay Singh	201381030001
Abhishek Singh	201381030004
Aditya Gautam	201381030007
Aditya Raj Pathak	201381030008
Amarendra Singh Yadav	201381030013
Tanu	191381030060

DECLARATION

We, **Abhay Singh, Abhishek Singh, Aditya Gautam, Aditya Raj Pathak, Amarendra Singh Yadav, Tanu** hereby declare that the work which is being presented in the file, **"PIXELVAULT – A marketplace for NFT"**, is in partial fulfilment of the requirement of the major project that has been given by us, as a scheduled program of our B.Tech education. To the best of our knowledge and belief, we also declare that we worked as per the university norms. This is an authentic record of our own, carried out **under the guidance of Er. Manoj Verma.**

Student Name/Signatures	Roll Number
Abhay Singh Abhay	**201381030001**
Abhishek Singh Abhishek Singh.	**201381030004**
Aditya Gautam Aditya Gautam	**201381030007**
Aditya Raj Pathak Aditya Raj Pathak	**201381030008**
Amarendra Singh Yadav Amarendra	**201381030013**
Tanu Tanu	**191381030060**

TABLE OF CONTENTS

LIST OF FIGURES

SYNOPSIS

INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BUNDELKHAND UNIVERSITY, JHANSI

A Project Synopsis

on

"PIXELVAULT – A NFT MARKETPLACE"

Submitted to the Department of Computer Science & Engineering,

Institute of Engineering and Technology

Bundelkhand University, Jhansi, for Fulfilment of the degree of B.Tech

(VIIth SEM 2023-2024)

GUIDED BY

Er. Manoj Verma

Assistant Professor

Dept. of Computer Science & Engineering

IET BU, Jhansi

COORDINATOR

Er. Priyanka Pande

Head of Department

Dept. of Computer Science & Engineering

IET BU, Jhansi

PROJECT IN CHARGE

Dr. Sadik Khan

Assistant Professor

Dept. of Computer Science & Engineering

IET BU, Jhansi

SUBMITTED BY:

1. Abhay Singh – 201381030001
2. Abhishek Singh - 201381030004 (Team Leader)
3. Aditya Gautam - 201381030007
4. Aditya Raj Pathak - 201381030008
5. Amarendra Singh Yadav - 201381030013
6. Tanu - 191381030060

INTRODUCTION

Problem Analysis:

In a rapidly evolving digital landscape, where creativity mingles with technology, we present an exploration into the creation and operation of an NFT Marketplace. This endeavor delves into the heart of blockchain innovation, where ownership and authenticity are redefined through the ingenious concept of Non-Fungible Tokens (NFTs).

The objective of this project is to design, develop, and deploy an NFT Marketplace named **PIXELVAULT** - a digital platform that revolutionizes the way we perceive, trade, and value digital assets. NFTs, being unique tokens secured by blockchain, grant us the ability to own and trade distinct digital items, from artwork and music to virtual real estate, backed by an immutable ledger of ownership.

Our venture into the NFT Marketplace domain involves intricate technical considerations. We dive into the mechanics of blockchain technology, analyzing its consensus protocols, cryptographic security, and decentralized architecture that ensure the authenticity and provenance of each NFT transaction. This project aims not just to understand the technology but to harness it for creating a seamless and secure platform.

Objective:

The primary objective of our project is to conceptualize, develop, and launch a cutting-edge NFT Marketplace that redefines the digital landscape. Through this venture, we aim to achieve the following goals:

- **Technical Innovation:** Design and implement an NFT Marketplace that leverages blockchain technology to ensure the security, immutability, and authenticity of NFT transactions. By employing advanced cryptographic techniques and decentralized architecture, our platform will establish a new standard for secure digital asset ownership and trading.
- **User-Centric Experience:** Craft an intuitive and user-friendly interface that caters to both creators and collectors. Our platform will provide a seamless and engaging experience, enabling users to seamlessly mint, list, and trade NFTs while fostering a sense of community within the digital art and collectibles space.
- **Ensuring Authenticity:** Establish a transparent and tamper-proof ledger through blockchain technology, assuring buyers of the authenticity and provenance of the NFTs they acquire. By implementing smart contracts, we will automate the enforcement of ownership rights and streamline the process of NFT transfer.
- **Educational Outreach:** Provide educational resources to users, creators, and collectors about NFT technology, its benefits, and potential risks. By fostering awareness and understanding, we aim to contribute to the responsible growth and adoption of NFTs.
- **Research and Analysis:** Conduct an in-depth analysis of the evolving NFT landscape, including market trends, legal considerations, and technological advancements. This research will enable us to adapt and innovate within a dynamic and rapidly changing environment.

Through the achievement of these objectives, our project seeks to contribute to the ongoing transformation of digital ownership, creative expression, and value exchange. We aim to leave a lasting

impact by creating a platform that not only embraces innovation but also empowers individuals in the realm of digital art, collectibles, and creative entrepreneurship.

Proposed System: Making NFT Marketplaces More Reliable

NFT marketplaces, where digital stuff like art and collectibles are bought and sold, have a big challenge. There's no clear way to prove if things are real or not, which makes people doubt if what they're buying is genuine. Sometimes, fake items and arguments over who owns what pop up because of this. Also, using these marketplaces can be tricky, especially for regular folks.

Our plan is to use a special kind of technology called blockchain to fix this. With blockchain, we can be sure things are real and where they came from. We want to create a marketplace that's easy to use and helps everyone feel safe when they buy or sell digital things. This way, people can trust NFTs more and use them without any confusion. Our goal is to make NFT marketplaces simple, secure, and welcoming for everyone.

FEASIBILITY STUDY

A feasibility study is an assessment of the practicality of a proposed plan or project. A feasibility study analyzes the viability of a project to determine whether the project or venture is likely to succeed. The study is also designed to identify potential issues and problems that could arise while pursuing the project.

As part of the feasibility study, project managers must determine whether they have enough of the right people, financial resources, and technology. The study must also determine the return on investment, whether this is measured as a financial gain or a benefit to society, as in the case of a nonprofit project.

DIMENSIONS OF FEASIBILITY

The study considers the feasibility of four aspects of a project:

- **Technical:** A list of the hardware and software needed, and the skilled labor required to make them work.
- **Financial:** An estimate of the cost of the overall project and its expected return.
- **Market:** An analysis of the market for the product or service, the industry, competition, consumer demand, sales forecasts, and growth projections
- **Organizational:** An outline of the business structure and the management team that will be needed.

TECHNICAL FEASIBILITY

A technical feasibility study consists in determining if your organization has the technical resources and expertise to meet the project requirements. A technical study focuses on assessing whether your organization has the necessary capabilities that are needed to execute a project, such as the production capacity, facility needs, raw materials, supply chain and other inputs. In addition to these production inputs, you should also consider other factors such as regulatory compliance requirements or standards for your products or services.

Web Application

It is a measure of the practicality of a specific technical solution and the availability of technical resources and expertise. The project is a complete web-based application. The main technologies and tools that are associated with the “PIXELVAULT” are

- HTML5
- CSS
- Javascript
- Tailwind
- Nextjs
- Nodejs
- Expressjs

ThirdWeb

The technical feasibility of integrating a Non-Fungible Token (NFT) marketplace within the ThirdWeb platform holds significant promise and potential. Leveraging blockchain technology, the platform can establish a decentralized and secure environment for NFT transactions, ensuring transparency and authenticity. The immutability and traceability features of blockchain contribute to the prevention of fraudulent activities and provide a reliable record of ownership for digital assets.

Smart contracts, a key component of blockchain technology, can be employed to automate various aspects of the NFT marketplace, such as ownership transfers and royalty distributions, reducing the need for intermediaries and enhancing efficiency. Additionally, the platform can explore interoperability standards, allowing users to seamlessly trade and interact with NFTs across different blockchain networks.

The feasibility of the NFT marketplace on ThirdWeb also hinges on addressing scalability challenges associated with blockchain networks, ensuring smooth and cost-effective transactions even during periods of high demand. Integration with emerging technologies, such as decentralized storage solutions, can enhance the platform's ability to handle the increasing volume of digital assets and associated metadata.

Moreover, considerations for user experience, such as intuitive interfaces and wallet integrations, are essential for encouraging widespread adoption. Security measures, including robust encryption and authentication protocols, must be prioritized to protect users' digital assets and personal information.

MongoDB

MongoDB, the most popular NoSQL database, is an open-source document-oriented database. The term ‘NoSQL’ means ‘non-relational’. It means that MongoDB isn’t based on the table-like relational database structure but provides an altogether different mechanism for storage and retrieval of data. This format of storage is called BSON (similar to JSON format).

SQL databases store data in tabular format. This data is stored in a predefined data model which is not very much flexible for today’s real-world highly growing applications. Modern applications are more networked, social and interactive than ever. Applications are storing more and more data and are accessing it at higher rates.

Relational Database Management System(RDBMS) is not the correct choice when it comes to handling big data by the virtue of their design since they are not horizontally scalable. If the database runs on a single server, then it will reach a scaling limit. NoSQL databases are more scalable and provide superior performance. MongoDB is such a NoSQL database that scales by adding more and more servers and increases productivity with its flexible document model.

FINANCIAL FEASIBILITY

The economic feasibility of developing an NFT marketplace web app involves a comprehensive analysis of costs, revenue potential, and overall financial viability. Here are key considerations:

Costs:

- **Development Costs:** This includes expenses related to hiring developers, designers, and other technical personnel, as well as costs associated with software tools and frameworks needed for the development of the web app.

- **Blockchain Integration Costs:** Incorporating blockchain technology for NFT transactions involves costs related to smart contract development, integration with blockchain networks, and testing to ensure secure and efficient transactions.
- **Security Measures:** Implementing robust security measures to protect user data, digital assets, and the platform itself may require investments in encryption technologies, secure protocols, and ongoing security audits.
- **Operational Costs:** These encompass day-to-day operational expenses such as server hosting, cloud services, and maintenance.
- **Marketing and Promotion:** To attract users and creators to the NFT marketplace, budgeting for marketing campaigns, community building, and promotional activities is essential.

Revenue Potential:

- **Transaction Fees:** The primary revenue source for an NFT marketplace is transaction fees charged on each sale or transfer of NFTs. Establishing competitive and transparent fee structures is crucial for user adoption.
- **Premium Features:** Offering premium features, such as enhanced visibility for NFT listings, analytics tools, or exclusive promotional opportunities, can generate additional revenue streams.
- **Partnerships and Collaborations:** Forming strategic partnerships with artists, brands, or other platforms can lead to revenue-sharing agreements, sponsorship deals, or joint ventures.
- **Membership Models:** Introducing subscription-based models for advanced features, premium access, or reduced transaction fees for members can create recurring revenue.
- Advertising and Sponsorship: Allowing relevant advertisements or sponsorships within the platform can contribute to revenue, provided it aligns with the platform's user experience and values.

MARKET FEASIBILITY

Market feasibility for an NFT marketplace web app involves analyzing the demand, competition, and overall dynamics of the market to assess the viability and potential success of the platform. Here are key considerations for evaluating the market feasibility:

Market feasibility for an NFT marketplace web app involves analyzing the demand, competition, and overall dynamics of the market to assess the viability and potential success of the platform. Here are key considerations for evaluating the market feasibility:

- **Market Demand:** User Interest-Assess the current level of interest in NFTs and digital assets. Analyze trends in online communities, social media, and art-related platforms to gauge user enthusiasm. Creator Engagement-Evaluate the willingness of artists, musicians, and other content creators to participate in the NFT marketplace. Understand their preferences and requirements for listing and selling NFTs.
- **Target Audience:** Identify the User Base-Define the target audience for the NFT marketplace. Consider whether the focus is on art, music, gaming, collectibles, or a combination of these. Understand the demographics and preferences of the intended users.

- **Competitive Analysis:** Analyze the strengths and weaknesses of existing NFT marketplaces. Identify gaps in the market and areas where the proposed platform can offer unique features or improvements. Understand the strategies and success factors of leading NFT marketplaces. Assess user reviews and feedback to identify pain points that can be addressed.
- **Regulatory Environment:** Investigate the legal landscape surrounding NFTs and blockchain technology in relevant jurisdictions. Ensure compliance with regulations related to digital assets, copyright, and consumer protection.
- **Technology Trends:** Blockchain Adoption-Evaluate the level of adoption and acceptance of blockchain technology. Assess whether potential users are familiar with and comfortable using blockchain for NFT transactions. Stay informed about emerging technologies that could impact the NFT space, such as advancements in blockchain scalability, interoperability, and layer 2 solutions.
- **Monetization Opportunities:** Analyze the fee structures of existing NFT marketplaces and determine a competitive yet sustainable pricing strategy. Consider transaction fees, listing fees, and any other potential revenue streams. Explore opportunities for partnerships with influencers, brands, or organizations that could enhance the visibility and reach of the NFT marketplace.
- **Community Building:** Engagement Strategies-Develop strategies for community building within the NFT marketplace. Consider features like forums, social interactions, and events that can foster a sense of belonging among users. Implement mechanisms for collecting and incorporating user feedback to continuously improve the platform based on user preferences.
- **Scalability:** Estimate the potential user growth and scalability requirements. Ensure that the infrastructure and technology stack can handle increased user activity without compromising performance.

ORGANIZATIONAL FEASIBILITY

Organizational feasibility for an NFT market web app involves assessing the capability of the organization to successfully plan, develop, launch, and sustain the platform. It considers factors related to the team, structure, and management strategies. Here are key considerations for evaluating organizational feasibility:

- **Expertise and Skills:** Assess the technical expertise within the organization, particularly in blockchain technology, smart contract development, and web application development. Ensure the team possesses the necessary skills to implement and maintain the required functionalities of the NFT marketplace.Evaluate the team's understanding of the NFT market, blockchain trends, and the broader digital asset landscape. Knowledge of the unique challenges and opportunities in the NFT space is crucial for making informed decisions.
- **Team Composition:** Form cross-functional teams that bring together skills in software development, blockchain technology, user experience (UX) design, legal, and marketing. This ensures a holistic approach to building and operating the NFT marketplace. Consider establishing an advisory board comprising individuals with experience in the NFT space, legal experts, and industry influencers. Their guidance can provide valuable insights and enhance the credibility of the project.
- Leadership and Management: Ensure the presence of strong leadership capable of steering the organization through the complexities of the NFT market. Effective decision-making, strategic vision, and adaptability are key leadership qualities. Implement robust project management

practices to oversee the development lifecycle. Adopt agile methodologies to facilitate iterative development, adapt to changing requirements, and ensure timely deliveries.

- **Financial Resources:** Develop a comprehensive budget that covers development costs, marketing, operations, and potential contingencies. Explore funding options such as investment, grants, or partnerships to secure the necessary financial resources. Establish effective financial management practices to monitor expenses, track revenue, and ensure the financial sustainability of the NFT marketplace.
- **Legal and Compliance:** Employ legal experts or consultants with knowledge of blockchain, NFTs, and relevant regulations. Ensure compliance with intellectual property laws, data protection regulations, and other legal considerations. Conduct thorough audits of smart contracts to identify and address any vulnerabilities. This ensures the security and reliability of the NFT marketplace.
- **Risk Management:** Identify potential risks related to market dynamics, technology, and regulatory changes. Develop risk mitigation strategies to address challenges proactively. Create contingency plans for unexpected events that could impact the development or operation of the NFT marketplace.
- **Partnerships and Collaborations:** Explore partnerships with blockchain technology providers, payment processors, and other industry stakeholders to enhance the functionality and reach of the NFT marketplace. Develop strategies for engaging with the NFT community, artists, and potential users. Building a supportive community can contribute to the success and sustainability of the platform.
- **Scalability:** Ensure the scalability of the platform by implementing a robust and scalable infrastructure. Be prepared to handle increased user activity and transaction volumes as the platform grows. Choose a technology stack that allows for scalability and easy integration with emerging technologies in the blockchain space.

MODEL OF PROJECT (WATERFALL MODEL)

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. The Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

Waterfall Model – Design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model.

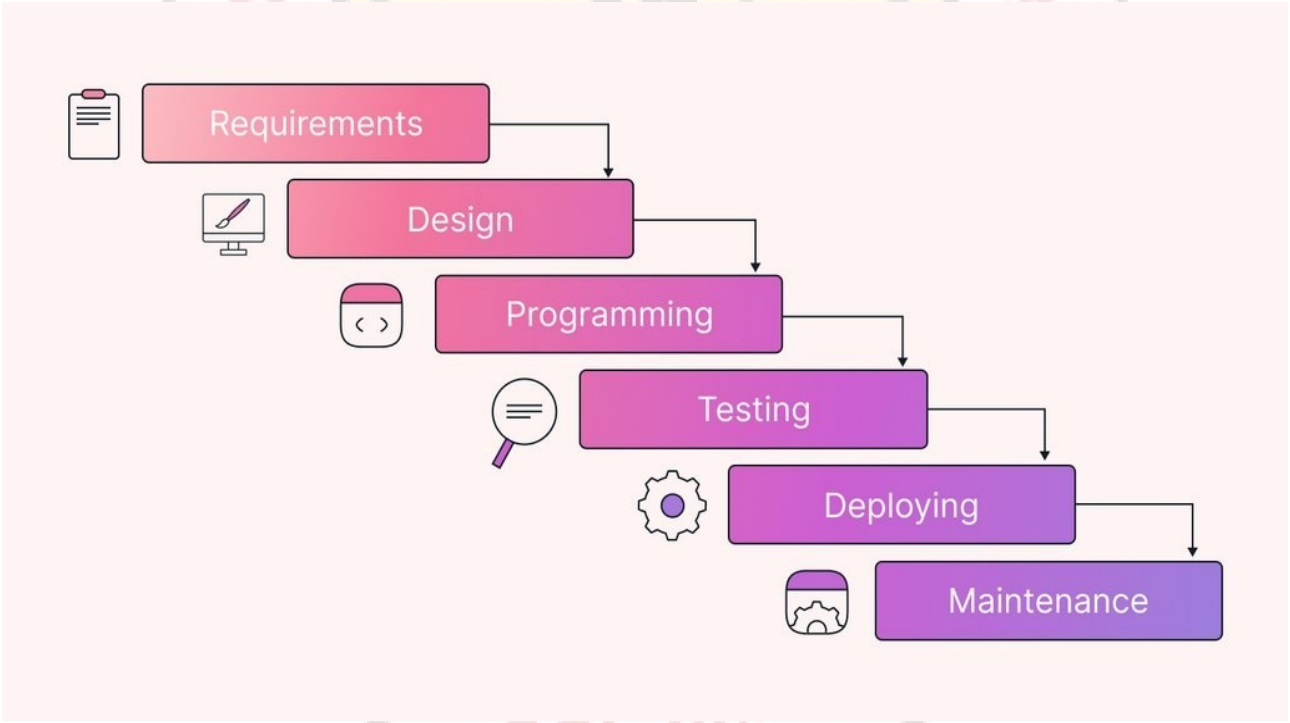

Figure S1: Phases of Waterfall Model

The sequential phases in Waterfall model are -

- **Requirement Gathering and analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design:** The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

- **Implementation:** With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system:** Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance:** There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

REQUIREMENT ANALYSIS

Requirement analysis is significant and essential activity after elicitation. We analyze, refine, and scrutinize the gathered requirements to make consistent and unambiguous requirements. This activity reviews all requirements and may provide a graphical view of the entire system. After the completion of the analysis, it is expected that the understandability of the project may improve significantly. Here, we may also use the interaction with the customer to clarify points of confusion and to understand which requirements are more important than others.

- Measurable
- Specific
- Achievable / Acceptable
- Realistic / Relevant
- Time-Bounded

REQUIREMENT ANALYSIS PROCESS

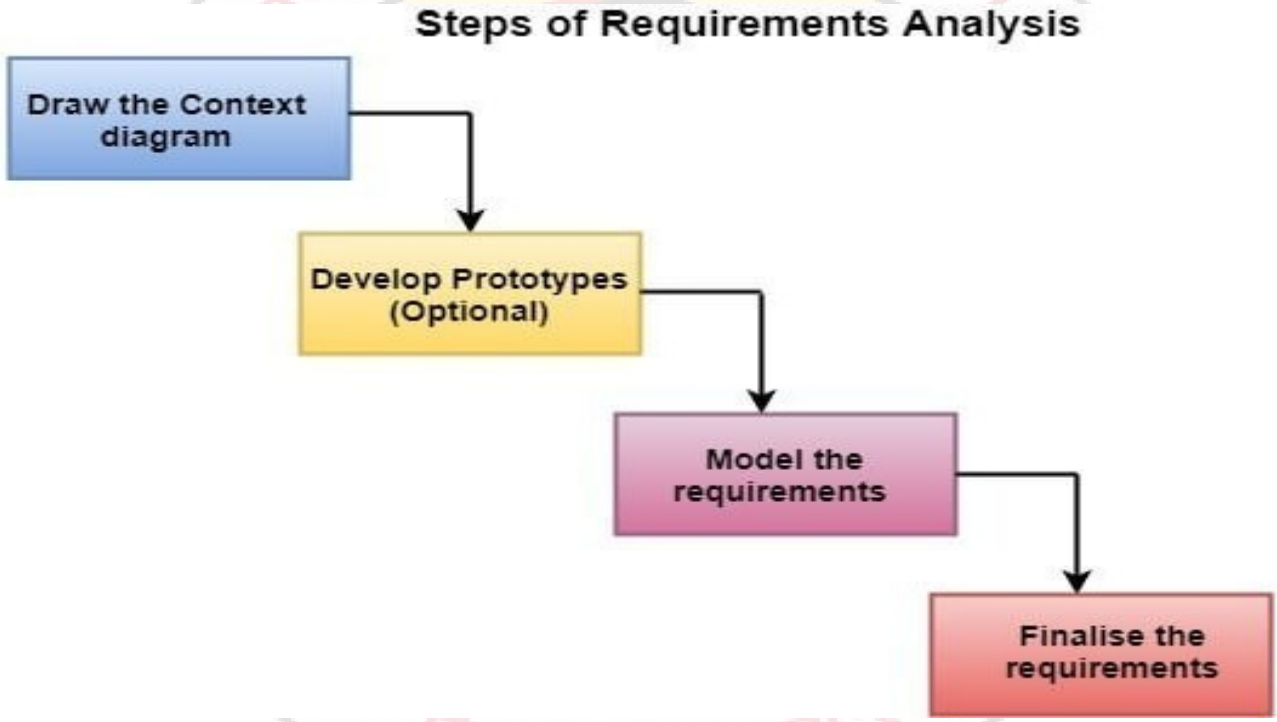

Figure S2: Steps of Requirements Analysis

Draw the context diagram: The context diagram is a simple model that defines the boundaries and interfaces of the proposed systems with the external world. It identifies the entities outside the proposed system that interact with the system.

Model the requirements: This process usually consists of various graphical representations of the functions, data entities, external entities, and the relationships between them. The graphical view may help to find incorrect, inconsistent, missing, and superfluous requirements. Such models include the Data Flow diagram, Entity-Relationship diagram, Data Dictionaries, State-transition diagrams, etc.

Finalise the requirements: After modeling the requirements, we will have a better understanding of the system behavior. The inconsistencies and ambiguities have been identified and corrected. The flow of

data amongst various modules has been analyzed. Elicitation and analyze activities have provided better insight into the system.

GATHER THE REQUIREMENTS

To begin the requirements analysis process, communicate with users to gather the requirements. Analysts can use different techniques to gather the requirements, including:

- Discussion with the team members
- Online surfing

ANALYZE THE REQUIREMENTS

In this phase, evaluate system feasibility, and confirm with the quality assurance team that the requirements are testable. The goal of this phase is to decompose, analyse and detail the requirements across the system's design. Here are the attributes of good requirements to help you study and define your list:

- Unique
- Complete
- Necessary
- Verifiable
- Consistent
- Quantifiable
- Clear and concise
- Operationally effective
- Able to be validated

IMPROVE THE QUALITY OF THE REQUIREMENTS

- Visualization: Use tools such as visualization and simulation to understand the desired product.
- Document dependencies: Document relationships and dependencies among requirements.

MODEL THE REQUIREMENTS

In this phase, it’s time to create models of the requirements, which help stakeholders and customers visualize the potential system. Then we can present the requirements using flowcharts, graphs or models to ensure the system corresponds to the business' needs.

DOCUMENT AND REVIEW THE REQUIREMENTS

Lastly, we can record the requirements in a document that is easy for both developers and users to understand. We can document the requirements in various formats, including:

- Software requirements specification
- Use cases
- Natural-language documents

- Process specification

Technologies

- Nextjs
- TailwindCss
- NextUI
- Thirdweb React SDK
- Nodejs
- MongoDB
- Mongoose
- Etherjs
- Validatorjs

DESIGN

The design of an NFT marketplace web app is a critical aspect that influences user engagement, trust, and overall usability. The user interface (UI) and user experience (UX) should be carefully crafted to provide a seamless and visually appealing environment for buyers, sellers, and enthusiasts within the NFT ecosystem.

The homepage should feature an intuitive layout that showcases trending NFTs, featured artists, and upcoming auctions. A well-organized navigation system should enable users to easily explore different categories, artists, and collections. Clear and attractive visuals, including high-resolution previews of NFTs, contribute to a visually stimulating experience, allowing users to appreciate the digital assets before making a purchase.

The individual NFT listing pages need to be designed for optimal presentation, displaying detailed information about the artwork, artist, and transaction history. Users should be able to view high-quality images or media files associated with the NFT and access essential details such as ownership history and metadata. Integrating social elements, like comments, likes, and share buttons, fosters community engagement around each NFT.

The individual NFT listing pages need to be designed for optimal presentation, displaying detailed information about the artwork, artist, and transaction history. Users should be able to view high-quality images or media files associated with the NFT and access essential details such as ownership history and metadata. Integrating social elements, like comments, likes, and share buttons, fosters community engagement around each NFT.

UML Diagram

- Access PixelVault Web App: Go to the PixelVault web app.
- Explore NFT Marketplace: Navigate the marketplace to discover featured and trending NFTs.
- View NFT Details: Click on an NFT to see its details, including description, creator info, and pricing.
- Connect Digital Wallet: Connect your digital wallet, such as MetaMask, to trade or exchange NFTs.
- Buy NFTs: Purchase NFTs directly from the marketplace. Confirm the transaction through your wallet.
- Sell NFTs: List NFTs for sale with specified prices. Wait for potential buyers to make transactions.
- Auctions and Bidding: Participate in auctions by placing bids on NFTs.The highest bidder wins the item.
- Swapping or Exchanging: Explore features for swapping or exchanging NFTs directly with other users on the platform.
- Transaction Confirmations: Confirm transactions through your connected digital wallet. Verify funds and review associated fees.
- Ownership and Transfers: NFT ownership transfers on the blockchain. Confirm changes in ownership within your wallet.

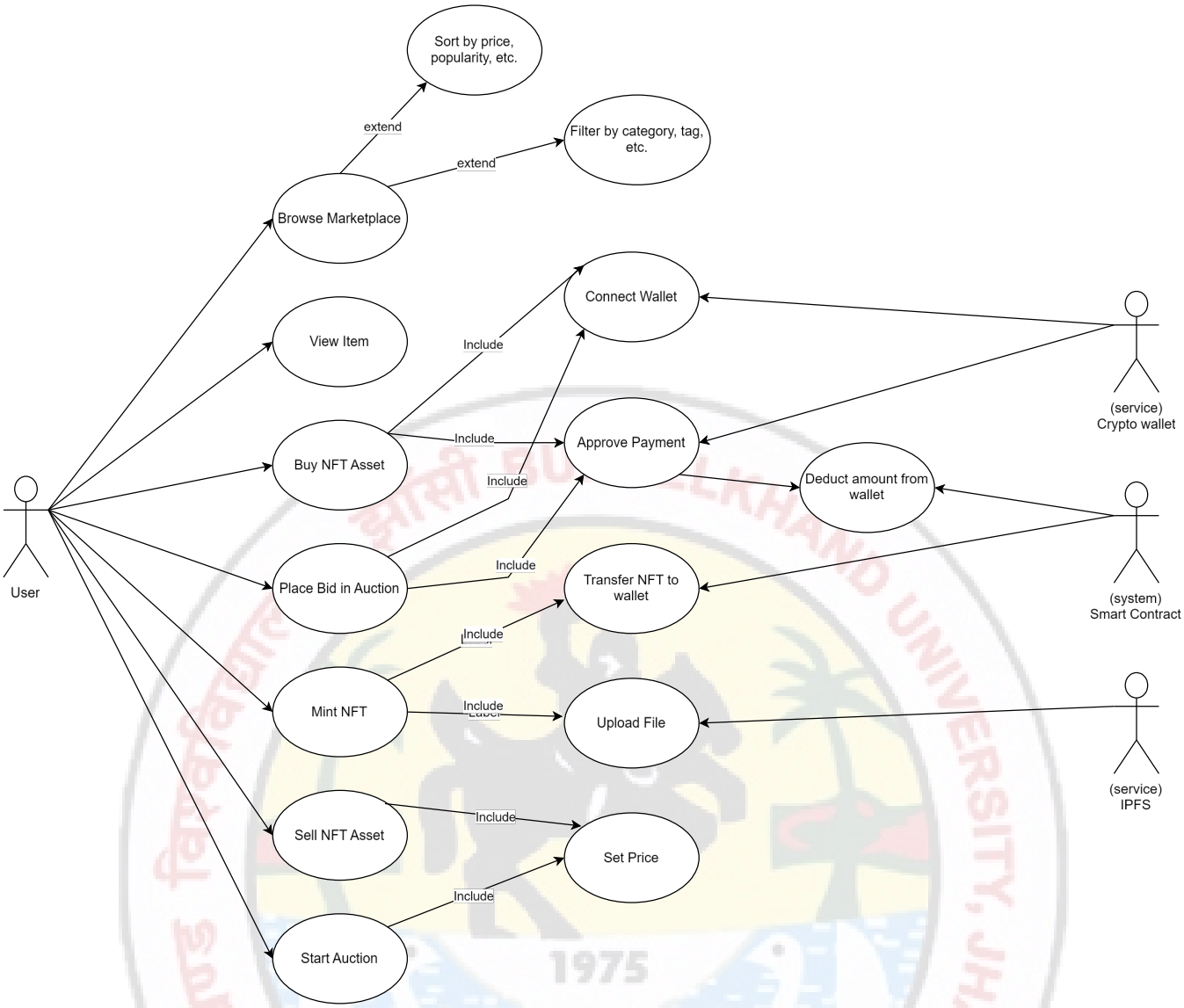

Figure S3: UML Diagram

DATAFLOW

Above Data Flow Diagram represents the flow of data during the procedure of buying NFT assets. Following are some insights about it:

- The User requests to view the Marketplace Catalogue.
- User may filter or sort the results according to their liking.
- The User chooses an item of interest and requests to view it.
- On the Item View page, the user may request to buy the item.
- Payment is approved by the User Wallet.
- Amount gets deducted from the User Wallet and gets added to Seller Wallet.
- The NFT token gets transferred to User Wallet.
- All transactions get added to Transaction Database.

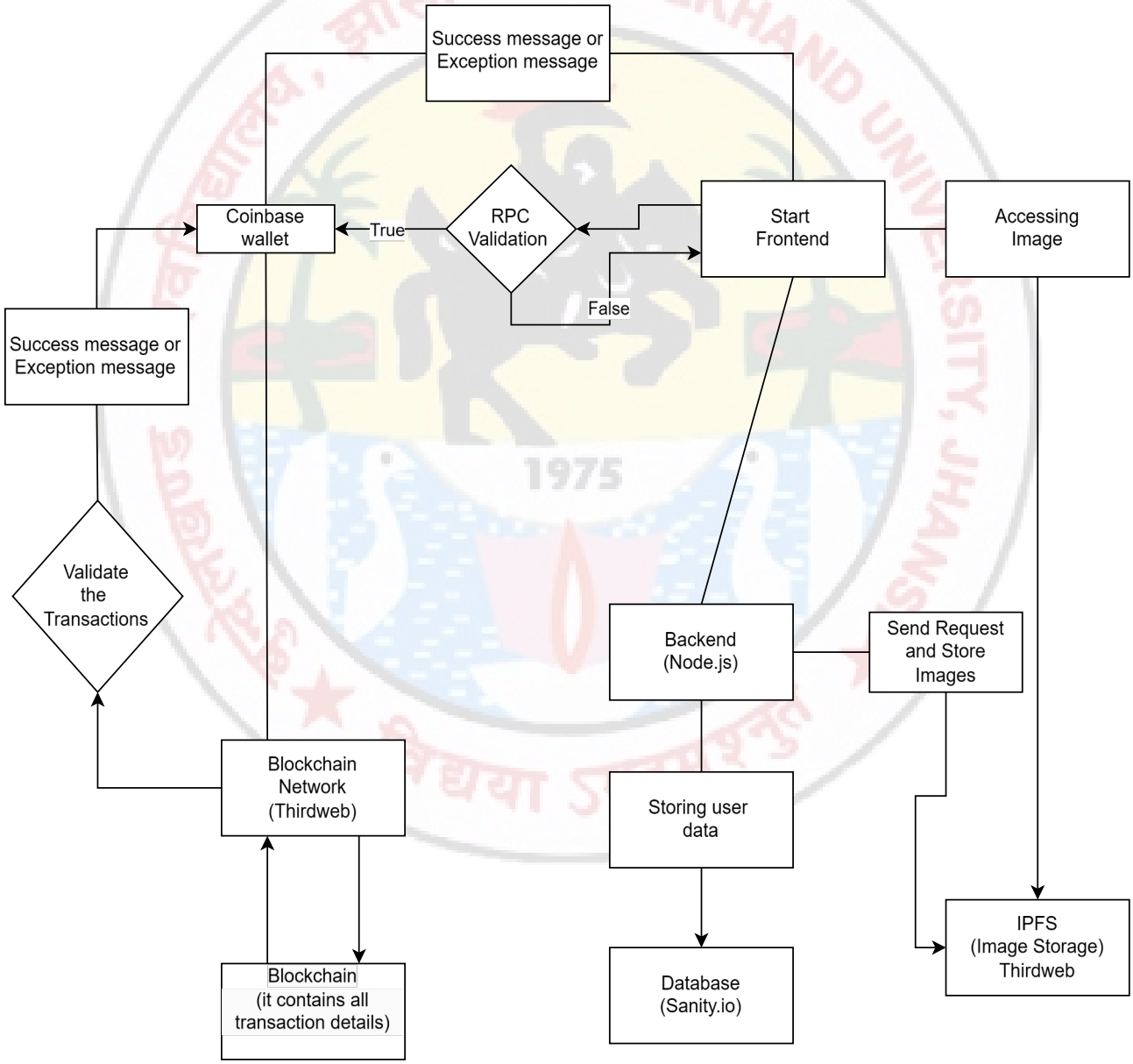

Figure S4: Dataflow of Pixelvault

IMPLEMENTATION

Implementation is the process of building the web according to its design. Implementation is an activity that is contained throughout the development phase. It is the process of bringing the designed system into operational use. The system is tested first and then turned into a working system. Every task identified in the design specification is carried out in this phase.

PLANNING THE SITE

Planning the website involves creating Application. This is an essential step because it is kind of like the skeleton of your site. The sitemap allows the developer to get an outline of what the site will look like, what pages there will be, and how they will interact with each other. This not only helps with planning but is also beneficial to the user experience. A user should be able to easily navigate a site, and this begins with the development of the sitemap. Before you begin to plan content, a sitemap lets you design what the structure will look like. Once the sitemap is completed, the other part of this step is to create a wireframe or mock-up. These are just visual representations of what the site will look like. This does not include the layout details.

Project planning

- **Project Kick-off and Planning**
  - **i.** Establish team roles and responsibilities.
  - **ii.** Set up communication and collaboration tools.
  - **iii.** Finalize project scope and objectives.
- **System Design and Architecture**
  - **iv.** Finalize system architecture and technical stack.
  - **v.** Complete database schema design.
  - **vi.** Create wireframes and mock-ups for UI/UX.

DESIGNING THE LAYOUT

The details of the layout are what will give your website character. This is the step where you get to be creative with pictures, videos, and what kinds of things the customer will notice when they come to your site. This process can take about 4-12 weeks from start to finish. The timing depends on experience, time spent on the project, and how thorough the developer is. During this step, it is especially important to keep referring back to the target audience you wish to focus on.

WRITING THE CONTENT

This step may be going on simultaneously with the other development planning steps. The written content of a website is so important to its success. The written content on a website is going to help a visitor determine their next steps. It is vital to draw customers in and keep them. There is a lot to consider when working on the content of the website. When determining what words to use, it is important that they are not too hard to understand. A general rule is that you have to assume not everyone is going to want to read words that are a higher vocabulary. A website should have a vocabulary that the average person can understand.

CODING THE WEBSITE

Now that all aspects of my website have been created, i am ready to actually begin creating the website itself. The coding typically begins with the homepage and gradually branches out to the other pages included in the site. This would be where the sitemap is followed to ensure everything is coded correctly.

We divided the coding part in different modules which are -

- **Module 1 – Software Requirement and Analysis for Pixelvault**
  - Software Requirement: Here we will find all the software required for our Pixelvault to develop like what are the technologies required for frontend and backend. Also we will find all the tools which will required in development of Pixelvault.
  - Analysis: Here we will analyze that which technology will be most suitable and work faster to make our app faster.
- **Module 2 – Architecture of Pixelvault**
  - Dataflow: Here we will design a flowchart in which we will decide how Pixelvault will work.
  - UML Diagram: Here we will design a flowchart in which we will decide how a user will interact with our Pixelvault Web App and buy and sell NFTs.
- **Module 3 – Backend Implementation**
  - NFT Schema:
    - Define the schema for NFTs (metadata, image URLs, owner details).
    - Configure relationships between users and NFTs.
  - User Management:
    - Set up user profiles.
    - Define roles and permissions.
    - Manage user authentication.
  - Transaction History:
    - Log transactions for each NFT (purchases, bids).
    - Record ownership changes.
  - MongoDB Setup
    - Configuring mongoDB using mongoose to atlas to store data.
- **Module 4 – Thirdweb setup**
  - Creating NFT Collection Contract
  - Creating NFT Marketplace Contract
  - NFT Listing:
    - Listing of NFT on thirdweb platform.
  - View Listed NFTs:
    - All the listed NFTs can be viewed on thirdweb platform.
  - Burn NFTs:
    - Burnning NFT will remove it from your wallet permanentaly.

        - This process will irreversible but you can view the NFT.
    - Ownership and transfer:
        - We can view the ownership and transfer of an NFT.
- **Module 5 – Frontend and its integration with Backend**
    - Landing Page:
        - Display featured NFTs.
        - Provide information about the platform.
        - Call to action for users to explore and create NFTs.
    - NFT Listing:
        - Paginated grid/list view of all available NFTs.
        - Filter and sort options
        - Integration with mongoDB for fetching NFT data.
    - NFT Detail Page:
        - Display detailed information about a specific NFT.
        - Option to purchase or bid on the NFT.
        - Show current owner and transaction history.
    - User Authentication and Profile:
        - User signup and login.
        - Profile page with user-specific NFTs.
        - Integration with authentication services
    - Create NFT:
        - Form to create a new NFT.
        - Upload images, set metadata, and pricing.
        - Integration with mongoDB for storing NFT data.
    - Wallet Integration:
        - Connect user wallets (e.g., MetaMask, Coinbase).
        - Check and display user's NFT holdings.
        - Enable transactions (purchase, bid) through the connected wallet.
    - Notification:
        - Notify users about successful transactions.
        - Provide updates on bids, new NFTs, etc.
        - Integration with a notification service.
    - Wallet User Interface (UI):
        - Design user-friendly interface for wallet management.
        - Include features like transaction history and balance overview.
    - Wallet Integration with NFT Marketplace:
        - Connect user's wallet to the NFT marketplace.

        - Enable wallet authentication during transactions.
    - Wallet Balances and Transactions:
        - Display wallet balance and transaction history.
        - Update balances in real-time.
- **Module 6 - Testing and Documentation**
    - Testing:
        - Implement unit and integration tests for critical functionalities.
        - Use testing libraries compatible with Next.js and mongoDB.
    - Documentation
        - Providing user manual and guide to build and run pixelvault
        - Create comprehensive documentation for developers and end-users.
- **Module 7 – Deployment and Version Controlling**
    - Deployments
        - Prepare for deployment to platforms like Vercel (Next.js) and Atlas for mongoDB database.
        - Set up continuous integration for automatic deployments.
    - Version Controlling
        - Version controlling using git and github

TESTING

Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do. The benefits of testing include preventing bugs, reducing development costs and improving performance.

ALPHA TESTING

Alpha Testing is a type of software testing performed to identify bugs before releasing the product to real users or to the public. Alpha Testing is one of the User Acceptance Testing. This is referred to as alpha testing only because it is done early on, near the end of the development of the software.

Objective of Alpha Testing

- The objective of alpha testing is to check whether the location is shown or not.
- The objective of alpha testing is to check whether the form is auto-generated or not.
- The objective of alpha testing is to check if it reads the written text or not.

Alpha Testing Process

- Review the design specification and functional requirements.
- Develop comprehensive test cases and test plans.
- Execute test plan
- Log defects
- Retest once the issues have been fixed

BETA TESTING

Beta Testing is performed by real users of the software application in a real environment. Beta testing is one of the types of User Acceptance Testing. A Beta version of the software, whose feedback is needed, is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing helps in minimization of product failure risks and it provides increased quality of the product through customer validation. It is the last test before shipping a product to the customers. One of the major advantages of beta testing is direct feedback from customers.

Characteristics of Beta Testing

- Beta Testing is performed by clients or users who are not employees of the company.
- Reliability, security, and robustness are checked during beta testing.
- Beta Testing commonly uses black-box testing.
- Beta testing is carried out in the user's location.
- Beta testing doesn't require a lab or testing environment.

Criteria for Beta Testing

- The Beta version of the software should be ready.
- Environment ready to release the software application to the public.
- Tool to capture real-time faults.

DEPLOYMENT

Deployment in software and web development means pushing changes or updates from one deployment environment to another. When setting up a website you will always have your live website, which is called the live environment or production environment. If you want the ability to make changes without these affecting your live website, then you can add additional environments. These environments are called development environments or deployment environments. The additional development environments will typically be a local environment, a development environment, and a staging environment(also known as a staging site).

VERSION CONTROLING USING GIT AND GITHUB

Version control is an essential aspect of software development, enabling teams to manage and track changes to their codebase over time. Git, a distributed version control system, has become the industry standard for source code management. GitHub, a web-based platform built around Git, enhances collaboration, code review, and project management. Together, Git and GitHub provide a powerful solution for version control in software development.

Git and Github

Git allows developers to track changes, create branches for parallel development, and merge changes seamlessly. Each commit in Git represents a snapshot of the project at a specific point in time. This decentralized nature of Git enables developers to work independently on different aspects of a project and later integrate their changes, minimizing conflicts and streamlining collaboration.

GitHub, as a hosting service for Git repositories, adds a layer of collaboration and project management on top of Git. Developers can push their local Git repositories to GitHub, making it easier to share code with others. GitHub provides features like pull requests, issues, and project boards, facilitating effective collaboration among team members. The pull request workflow allows developers to propose changes, conduct code reviews, and merge code into the main branch once approved.

Branching is a fundamental concept in both Git and GitHub. Developers can create branches to work on specific features or bug fixes without affecting the main codebase. Branches can be easily merged, allowing for a smooth integration of changes. GitHub enhances this process by providing a visual interface for comparing and merging branches, making it accessible to both technical and non- technical team members.

The version control capabilities of Git and GitHub are crucial for project stability, collaboration, and traceability. Developers can roll back to previous versions, investigate the history of changes, and identify who made specific modifications. This traceability is essential for debugging, auditing, and understanding the evolution of the codebase over time.

DEPLOYMENT USING VERCEL

Vercel is a cloud platform designed to facilitate seamless deployment and hosting of web applications. Its simplicity and integration capabilities make it a popular choice for developers seeking an efficient deployment solution. The deployment process using Vercel involves a few straightforward steps, providing an end-to-end solution for shipping web applications.

To deploy a web application with Vercel, developers typically start by linking their version control repository (e.g., GitHub, GitLab, or Bitbucket) to a Vercel project. Once connected, Vercel automatically detects the project settings and suggests a deployment configuration. Developers can

customize deployment settings, such as environment variables and build commands, to suit the specific needs of their application.

Vercel supports various frameworks and languages, enabling developers to deploy a wide range of projects, from static websites to dynamic web applications. The platform automatically builds and

optimizes the application during deployment, ensuring efficient performance and minimizing load times for end-users.

One notable feature of Vercel is its support for serverless functions, allowing developers to deploy serverless APIs alongside their web applications seamlessly. This serverless architecture simplifies scalability and reduces infrastructure management overhead.

Vercel also provides continuous deployment, automatically deploying changes to the production environment whenever new code is pushed to the linked repository. This continuous integration and deployment (CI/CD) approach ensures that the latest version of the application is always available to users.

Additionally, Vercel offers deployment previews, allowing developers to create temporary, shareable URLs for reviewing changes before merging them into the main branch. This feature facilitates collaborative development and thorough testing of new features in a production-like environment.

PROGRESS REPORT - 1

INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BUNDELKHAND UNIVERSITY, JHANSI

A Progress Report - 1

on

"PIXELVAULT – A NFT MARKETPLACE"

Submitted to the Department of Computer Science & Engineering,

Institute of Engineering and Technology

Bundelkhand University, Jhansi, for Fulfilment of the degree of B.Tech

(VIIth SEM 2023-2024)

GUIDED BY

Er. Manoj Verma

Assistant Professor

Dept. of Computer Science & Engineering

IET BU, Jhansi

COORDINATOR

Er. Priyanka Pande

Head of Department

Dept. of Computer Science & Engineering

IET BU, Jhansi

PROJECT IN CHARGE

Dr. Sadik Khan

Assistant Professor

Dept. of Computer Science & Engineering

IET BU, Jhansi

SUBMITTED BY:

1. Abhay Singh – 201381030001
2. Abhishek Singh - 201381030004 (Team Leader)
3. Aditya Gautam - 201381030007
4. Aditya Raj Pathak - 201381030008
5. Amarendra Singh Yadav - 201381030013
6. Tanu - 191381030060

PROGRESS TILL NOW

- **Module 1 – Software Requirement and Analysis for Pixelvault**
  - Software Requirement: Here we will find all the software required for our Pixelvault to develop like what are the technologies required for frontend and backend. Also we will find all the tools which will required in development of Pixelvault.
  - Analysis: Here we will analyze that which technology will be most suitable and work faster to make our app faster.
- **Module 2 – Architecture of Pixelvault**
  - Dataflow: Here we will design a flowchart in which we will decide how Pixelvault will work.
  - UML Diagram: Here we will design a flowchart in which we will decide how a user will interact with our Pixelvault Web App and buy and sell NFTs.

We have completed the above two modules.

Module 1 - Software Requirement

Requirement analysis is significant and essential activity after elicitation. We analyse, refine, and scrutinize the gathered requirements to make consistent and unambiguous requirements. This activity reviews all requirements and may provide a graphical view of the entire system. After the completion of the analysis, it is expected that the understandability of the project may improve significantly. Here, we may also use the interaction with the customer to clarify points of confusion and to understand which requirements are more important than others.

Technology Required

- Frontend Technologies
  - NextJs
  - Tailwind CSS
  - NextUI
  - Thirdweb React SDK
  - React Tostify
  - JSX
  - Framer motion
  - React icons
  - Reactjs
- Backend Technologies
  - Nodejs
  - MongoDB
  - Mongoose
  - Thirdweb
  - Validatorjs
  - etherjs

- Deployment and Version Controlling
  - Git
  - Github
  - Vercel
- Tools
  - Visual Studio Code
  - MongoDB Atlas

Module 2 - Architecture of Pixelvault

The design of an NFT marketplace web app is a critical aspect that influences user engagement, trust, and overall usability. The user interface (UI) and user experience (UX) should be carefully crafted to provide a seamless and visually appealing environment for buyers, sellers, and enthusiasts within the NFT ecosystem.

The homepage should feature an intuitive layout that showcases trending NFTs, featured artists, and upcoming auctions. A well-organized navigation system should enable users to easily explore different categories, artists, and collections. Clear and attractive visuals, including high-resolution previews of NFTs, contribute to a visually stimulating experience, allowing users to appreciate the digital assets before making a purchase.

The individual NFT listing pages need to be designed for optimal presentation, displaying detailed information about the artwork, artist, and transaction history. Users should be able to view high-quality images or media files associated with the NFT and access essential details such as ownership history and metadata. Integrating social elements, like comments, likes, and share buttons, fosters community engagement around each NFT.

The individual NFT listing pages need to be designed for optimal presentation, displaying detailed information about the artwork, artist, and transaction history. Users should be able to view high-quality images or media files associated with the NFT and access essential details such as ownership history and metadata. Integrating social elements, like comments, likes, and share buttons, fosters community engagement around each NFT.

ER Diagram

An Entity-Relationship (ER) diagram is a visual representation of the data model that depicts the entities, attributes, relationships, and constraints within a system or application. Let's break down the components of an ER diagram in detail:

- **Entities:** Entities represent real-world objects or concepts within the system being modeled. In a business context like Pixelvault, entities could include 'User', 'Product', 'Order', 'Transaction', etc. Each entity is typically represented by a rectangle in the diagram.
- **Attributes:** Attributes describe the properties or characteristics of entities. They provide additional details about each entity. For example, attributes of a 'User' entity might include 'UserID', 'Username', 'Email', 'Password', etc. Attributes are usually depicted within ovals connected to their respective entities.
- **Relationships:** Relationships define how entities are related to each other. They represent the associations and interactions between entities. In an ER diagram, relationships are illustrated using lines connecting entities. Each relationship is labeled to indicate the nature of the

association, such as 'has', 'belongs to', 'purchases', etc. Relationships can be one-to-one, one-to-many, or many-to-many, depending on the cardinality of the association.

- **Cardinality:** Cardinality specifies the number of instances of one entity that are associated with the number of instances of another entity through a relationship. Cardinality is indicated using symbols such as '1' (one), 'M' (many), or '0..1' (zero or one) placed near the ends of the relationship lines. It helps define the nature and constraints of the relationship between entities.
- **Primary Keys:** Primary keys uniquely identify each record within an entity. In an ER diagram, primary keys are denoted by underlining the attribute(s) that serve as the primary key for each entity. Primary keys are essential for ensuring data integrity and facilitating efficient data retrieval operations.
- **Foreign Keys:** Foreign keys establish relationships between entities by referencing the primary key of another entity. They are attributes within an entity that refer to the primary key of another entity. Foreign keys help maintain referential integrity and enforce constraints between related entities.

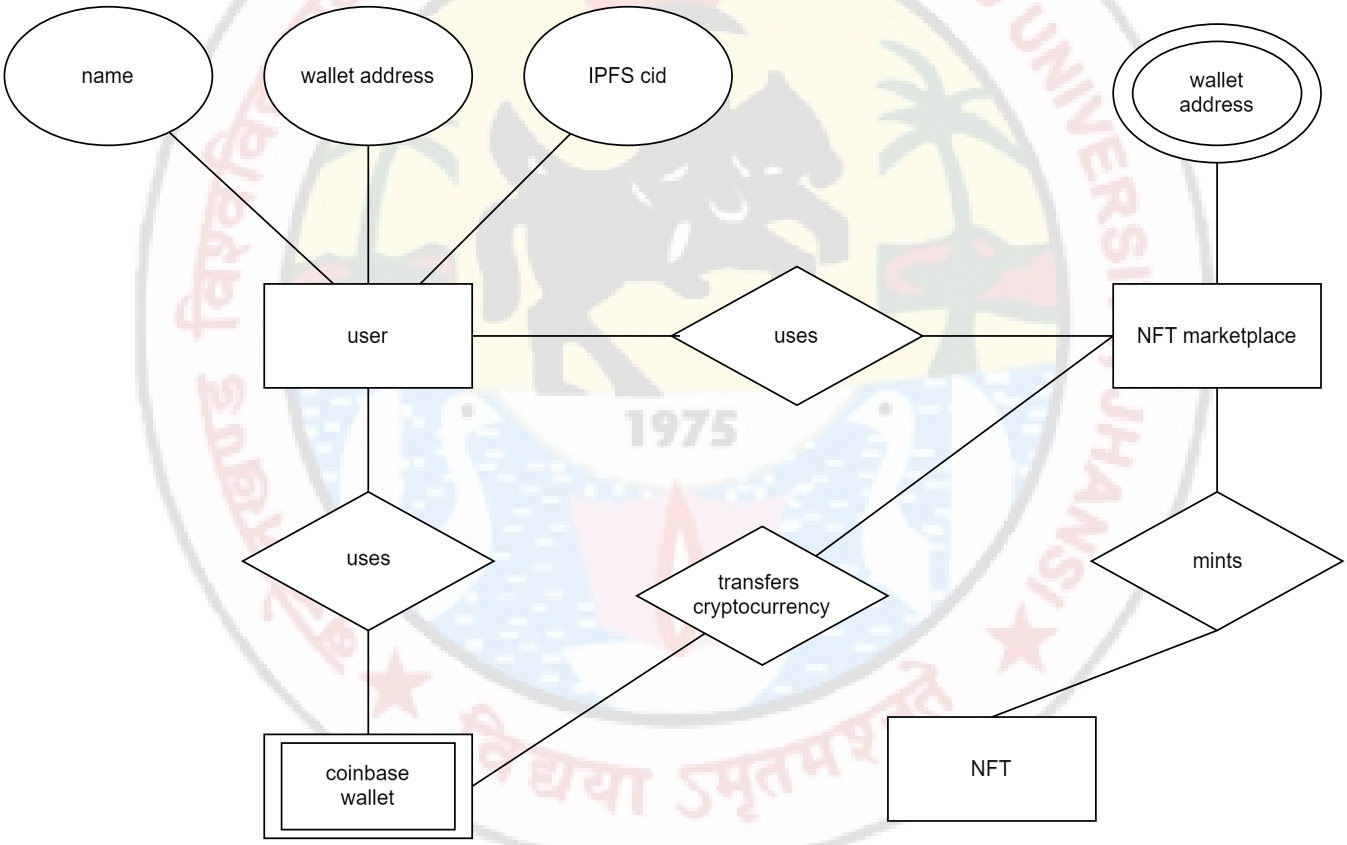

Figure PR1.1: ER Diagram

Dataflow Diagram

Data Flow Diagram represents the flow of data during the procedure of buying NFT assets. Following are some insights about it:

- The User requests to view the Marketplace Catalogue.
- User may filter or sort the results according to their liking.
- The User chooses an item of interest and requests to view it.

- On the Item View page, the user may request to buy the item.
- Payment is approved by the User Wallet.
- Amount gets deducted from the User Wallet and gets added to Seller Wallet.
- The NFT token gets transferred to User Wallet.
- All transactions get added to Transaction Database.

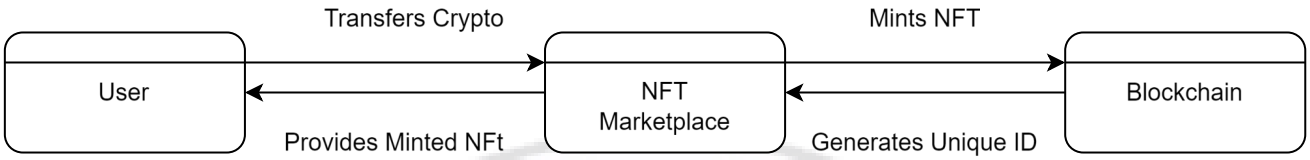

Figure PR1.2: Level 0 Dataflow Diagram

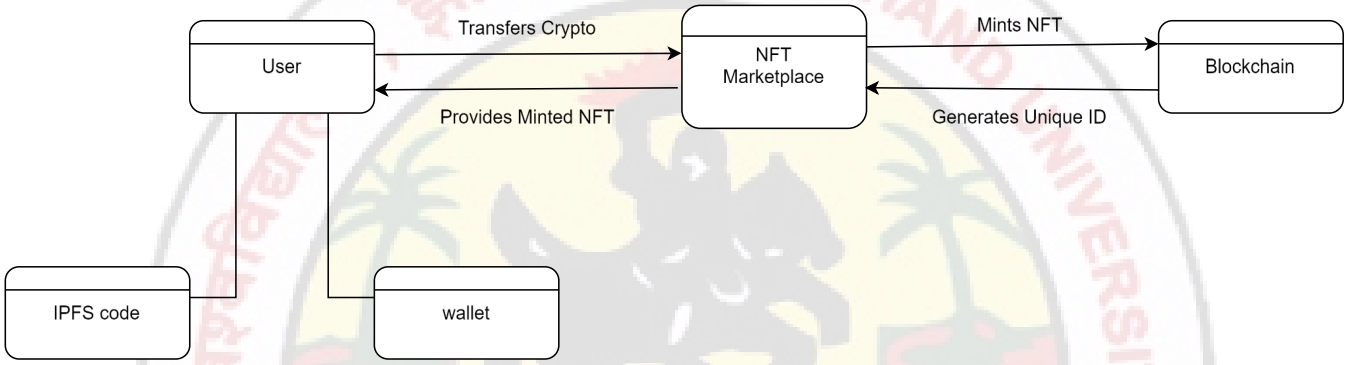

Figure PR1.3: Level 1 Dataflow Diagram

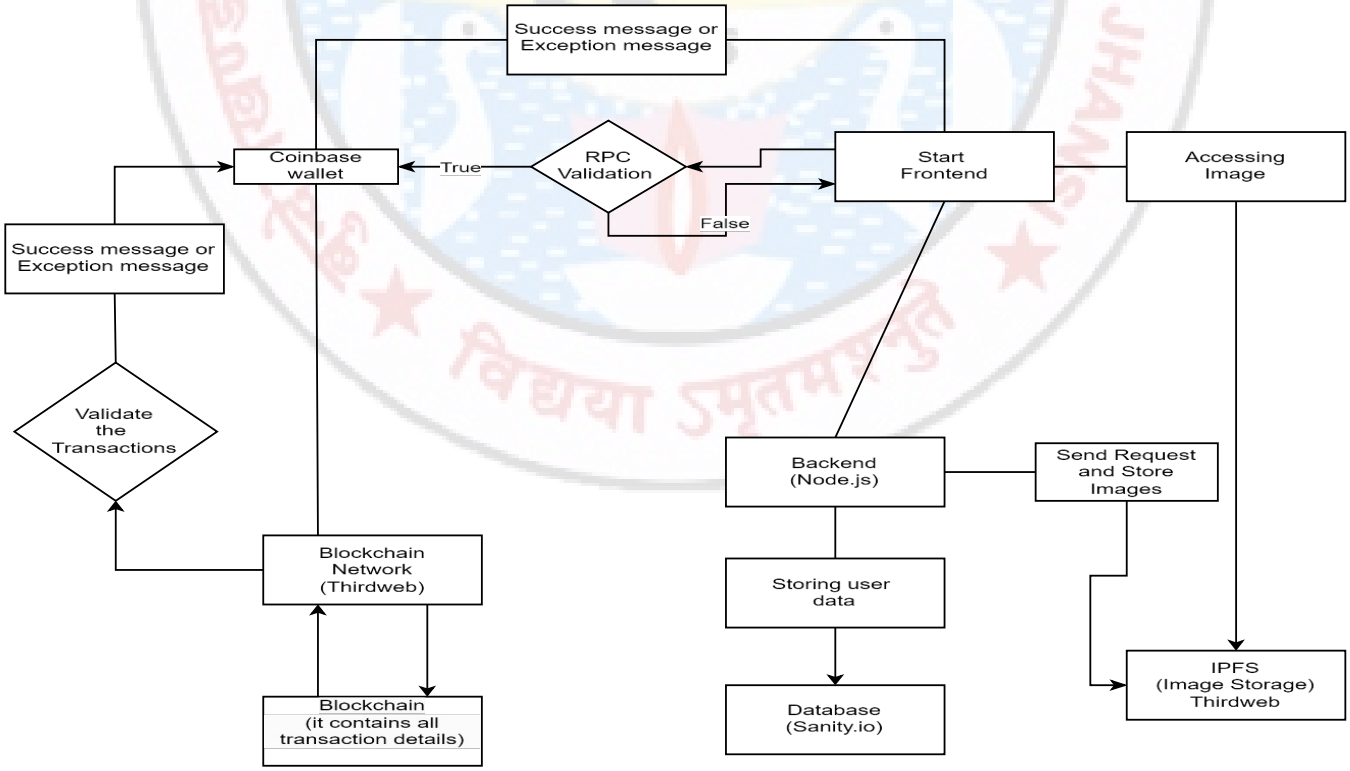

Figure PR1.4: Level 2 Dataflow Diagram

UML Diagram

- Access PixelVault Web App: Go to the PixelVault web app.
- Explore NFT Marketplace: Navigate the marketplace to discover featured and trending NFTs.
- View NFT Details: Click on an NFT to see its details, including description, creator info, and pricing.
- Connect Digital Wallet: Connect your digital wallet, such as MetaMask, to trade or exchange NFTs.
- Buy NFTs: Purchase NFTs directly from the marketplace. Confirm the transaction through your wallet.
- Sell NFTs: List NFTs for sale with specified prices. Wait for potential buyers to make transactions.
- Auctions and Bidding: Participate in auctions by placing bids on NFTs.The highest bidder wins the item.
- Swapping or Exchanging: Explore features for swapping or exchanging NFTs directly with other users on the platform.
- Transaction Confirmations: Confirm transactions through your connected digital wallet. Verify funds and review associated fees.
- Ownership and Transfers: NFT ownership transfers on the blockchain. Confirm changes in ownership within your wallet.

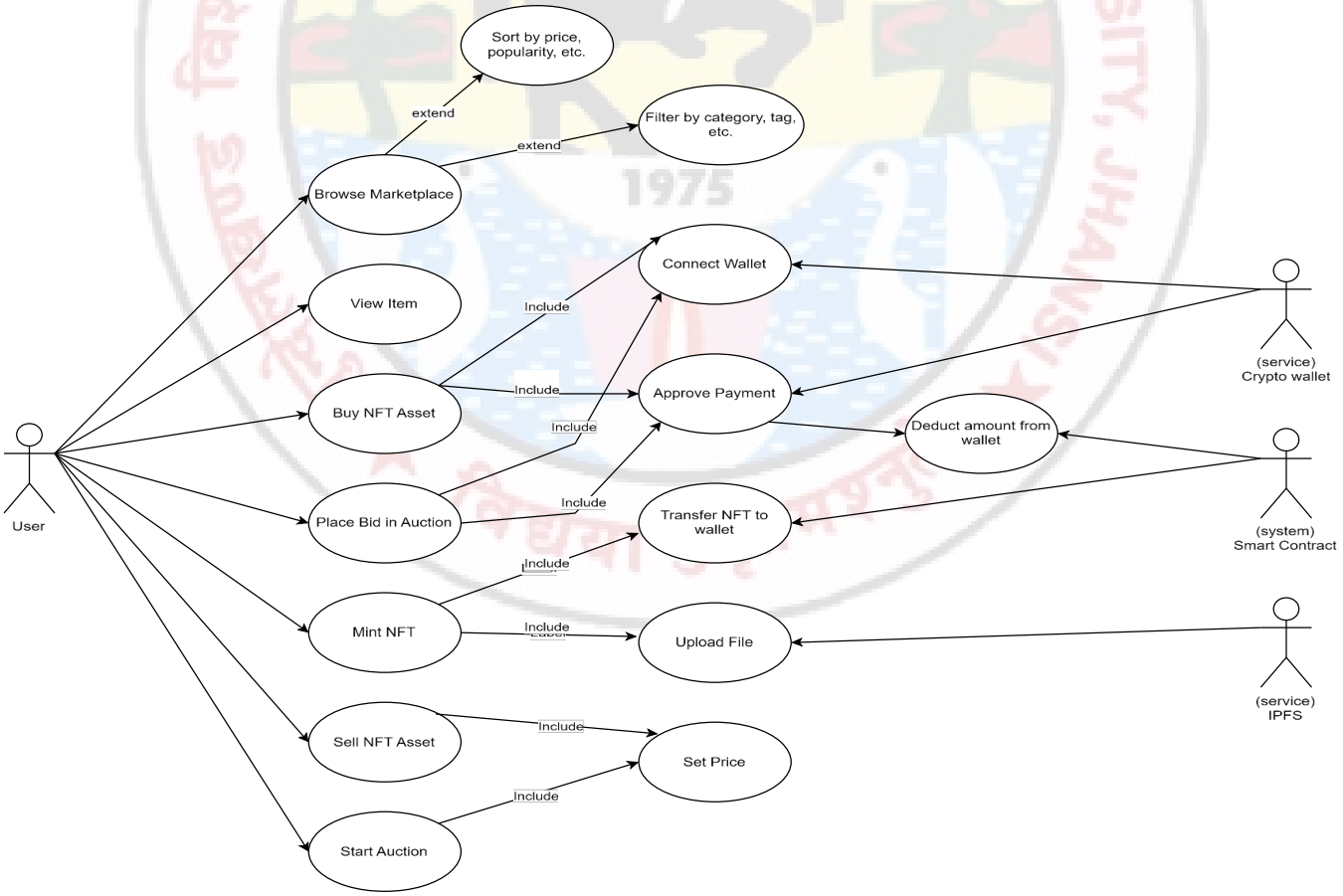

Figure PR1.5: UML Diagram

PROGRESS REPORT – 2

INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BUNDELKHAND UNIVERSITY, JHANSI

A Progress Report - 2

on

"PIXELVAULT – A NFT MARKETPLACE"

Submitted to the Department of Computer Science & Engineering,

Institute of Engineering and Technology

Bundelkhand University, Jhansi, for Fulfilment of the degree of B.Tech

(VIIth SEM 2023-2024)

GUIDED BY

Er. Manoj Verma

Assistant Professor

Dept. of Computer Science & Engineering

IET BU, Jhansi

COORDINATOR

Er. Priyanka Pande

Head of Department

Dept. of Computer Science & Engineering

IET BU, Jhansi

PROJECT IN CHARGE

Dr. Sadik Khan

Assistant Professor

Dept. of Computer Science & Engineering

IET BU, Jhansi

SUBMITTED BY:

1. Abhay Singh – 201381030001
2. Abhishek Singh - 201381030004 (Team Leader)
3. Aditya Gautam - 201381030007
4. Aditya Raj Pathak - 201381030008
5. Amarendra Singh Yadav - 201381030013
6. Tanu - 191381030060

PROGRESS TILL NOW

- **Module 3 – Backend Implementation**
  - NFT Schema:
    - Define the schema for NFTs (metadata, image URLs, owner details).
    - Configure relationships between users and NFTs.
  - User Management:
    - Set up user profiles.
    - Define roles and permissions.
    - Manage user authentication.
  - Transaction History:
    - Log transactions for each NFT (purchases, bids).
    - Record ownership changes.
  - MongoDB Setup
    - Configuring mongoDB using mongoose to atlas to store data.
- **Module 4 – Thirdweb setup**
  - Creating NFT Collection Contract
  - Creating NFT Marketplace Contract
  - NFT Listing:
    - Listing of NFT on thirdweb platform.
  - View Listed NFTs:
    - All the listed NFTs can be viewed on thirdweb platform.
  - Burn NFTs:
    - Burnning NFT will remove it from your wallet permanentaly.
    - This process will irreversible but you can view the NFT.
  - Ownership and transfer:
    - We can view the ownership and transfer of an NFT.

We have completed till module 4

Module 3 – Backend Implementation

  - NFT Schema:
    - Define the schema for NFTs (metadata, image URLs, owner details).
    - Configure relationships between users and NFTs.
  - User Management:
    - Set up user profiles.
    - Define roles and permissions.
    - Manage user authentication.

- Transaction History:
  - Log transactions for each NFT (purchases, bids).
  - Record ownership changes.
- MongoDB Setup
  - Configuring mongoDB using mongoose to atlas to store data.

```
import mongoose from "mongoose";

const UserSchema = new mongoose.Schema({
  address: {
    type: String,
    required: true,
    unique:true,
  },
  Name: {
    type: String,
    default: 'User',
  },
  Email: {
    type: String,
    default: '',
  },
  ProfileImage: {
    type: String,
    default: '',
  },
  Username: {
    type: String,
    default: '',
  },
  SocialLinks: {
    type: Object,
    default: {},
  },
});
mongoose.models = {};
export default mongoose.model("UserSchema", UserSchema);

```

Figure PR2.1: User Schema

```
_id: ObjectId('662f56b0100030f06da2faa0')
address : "0x62f50488F375ae6881Ce61F3E8adb5E98271395E"
Name : "Abhishek Singh"
Email : "abhisheksingh@gmail.com"
ProfileImage : "…"
Username : "abhishek"
▾ SocialLinks : Object
    Instagram : ""
    Linkedin : ""
    Facebook : ""
__v : 0
```


Figure PR2.2: User Data (JSON format) stored on MongoDB

```
1  import UserModel from "../../models/UserModel";
2  import connectDatabase from "../../middleware/mongoose";
3  import { isEthereumAddress } from "validator";
4
5  const handler = async (req, res) => {
6    try {
7      if(req.method == "POST"){
8        if (!isEthereumAddress(req.body.address)) {
9          res.status(400).json({ error: "Please enter valid Ethereum Address!" });
10         return;
11       }
12       let user = await UserModel.findOne({ address: req.body.address });
13       res.status(200).json({ user: user });
14     }
15     else{
16       res.status(400).json({ error: "Method not allowed" });
17       return
18     }
19   } catch (err) {
20     res.status(500).json({ error: err });
21   }
22 };
23
24 export default connectDatabase(handler);
25
```

Figure PR2.3: API endpoint to fetch user data

Module 4 – Thirdweb Setup

- Creating NFT Collection Contract
- Creating NFT Marketplace Contract
- NFT Listing:
  - Listing of NFT on thirdweb platform.
- View Listed NFTs:
  - All the listed NFTs can be viewed on thirdweb platform.

- Burn NFTs:
  - Burnning NFT will remove it from your wallet permanentaly.
  - This process will irreversible but you can view the NFT.
- Ownership and transfer:
  - We can view the ownership and transfer of an NFT.

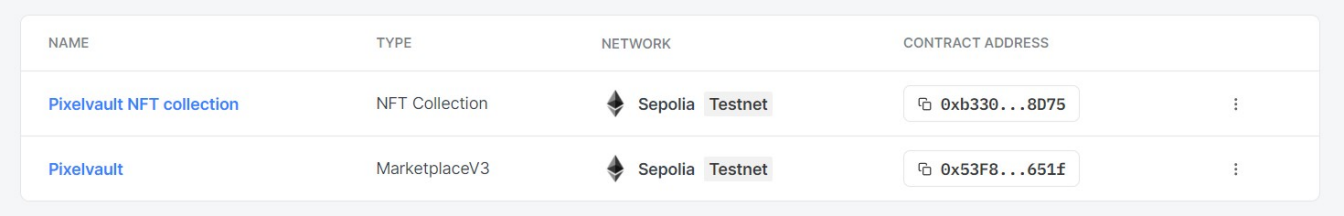

Figure PR2.4: Creating Contracts on Thirdweb

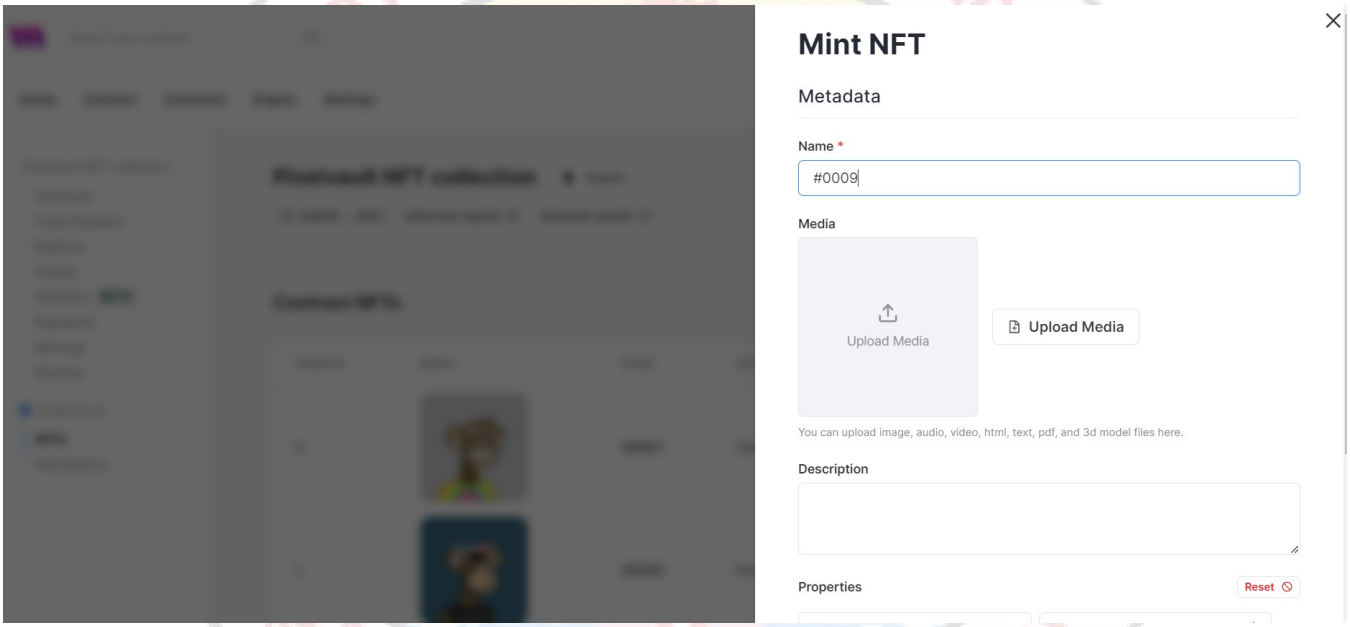

Figure PR2.5: Minting NFT on Thirdweb

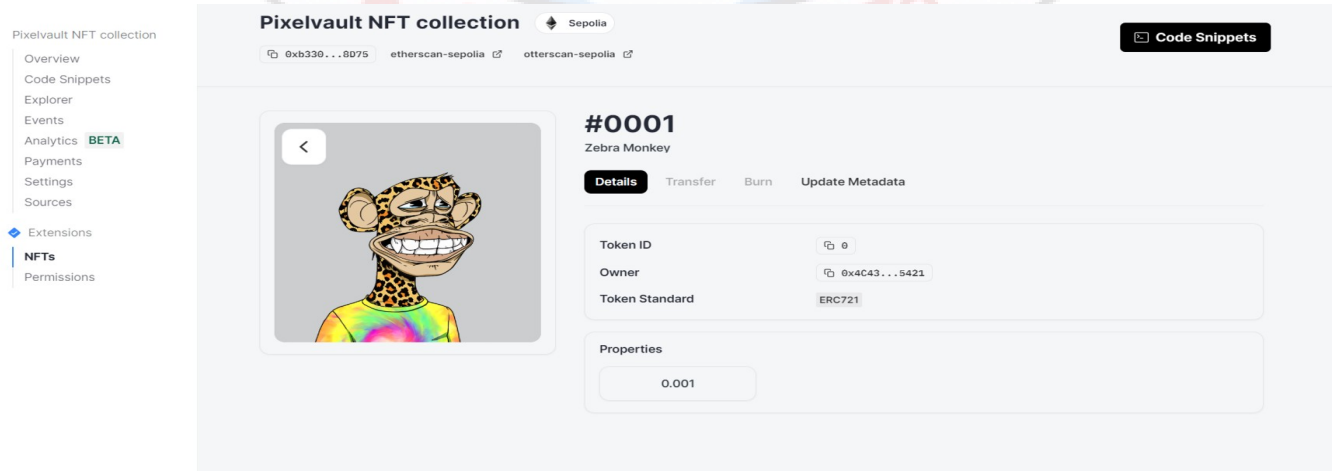

Figure PR2.6: Detail page of NFT on Thirdweb

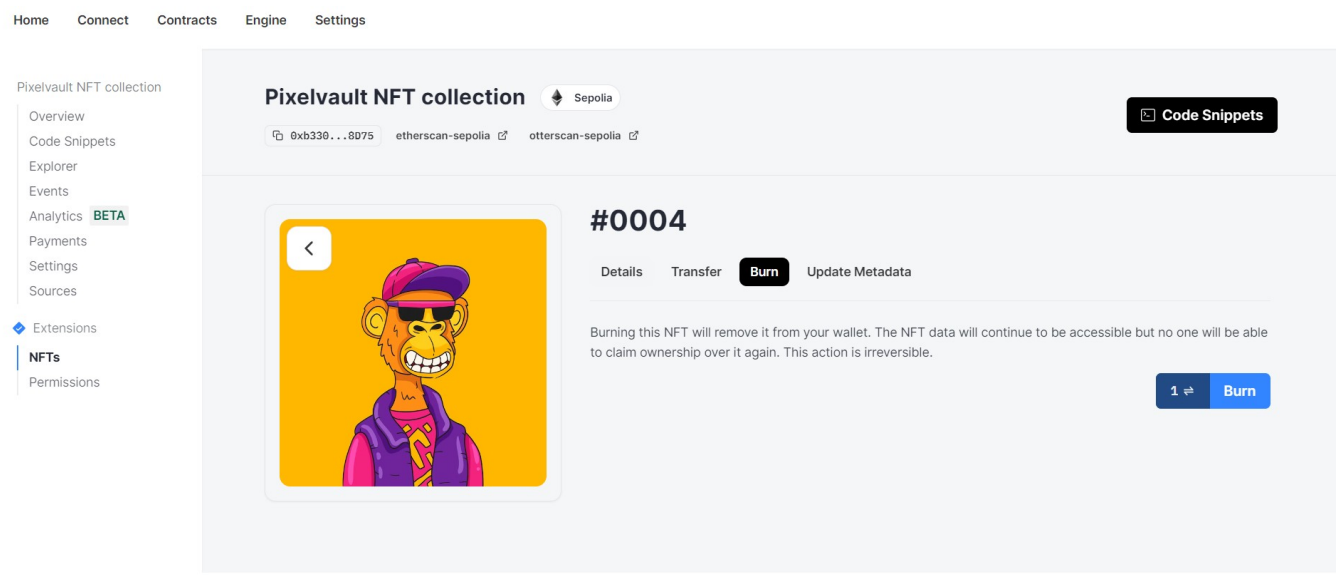

Figure PR2.7: Burn NFT on Thirdweb

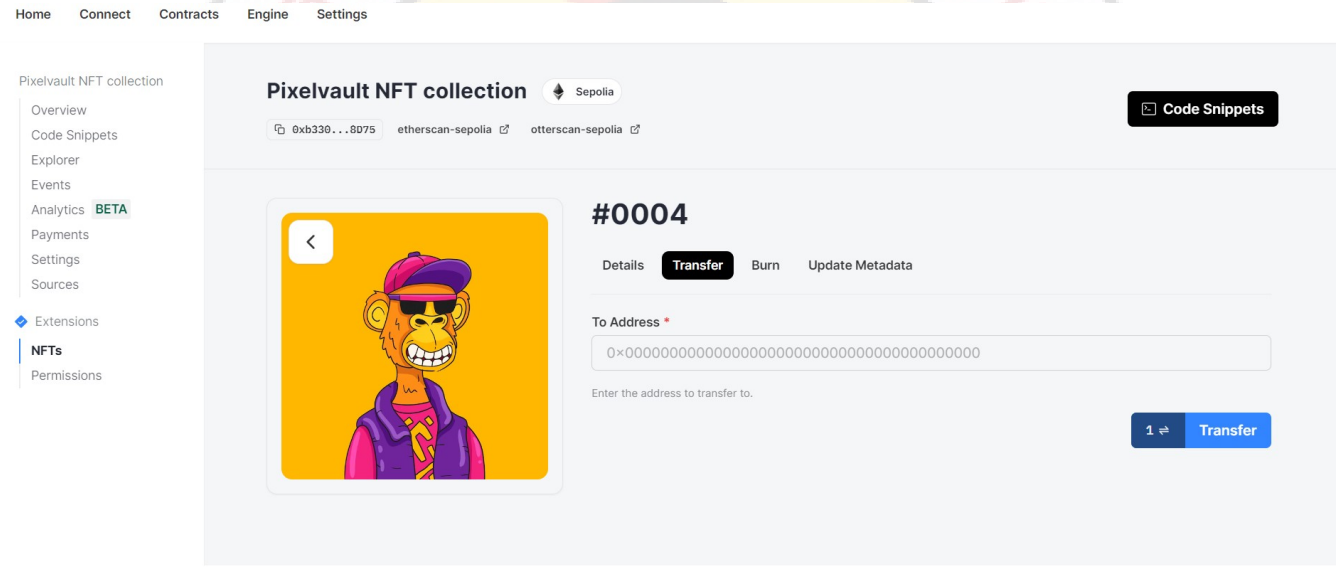

Figure PR2.8: Transfer ownership of NFT on Thirdweb

CHAPTER – 1
INTRODUCTION

1. INTRODUCTION

1.1. Problem Analysis:

In a rapidly evolving digital landscape, where creativity mingles with technology, we present an exploration into the creation and operation of an NFT Marketplace. This endeavor delves into the heart of blockchain innovation, where ownership and authenticity are redefined through the ingenious concept of Non-Fungible Tokens (NFTs).

The objective of this project is to design, develop, and deploy an NFT Marketplace named **PIXELVAULT** - a digital platform that revolutionizes the way we perceive, trade, and value digital assets. NFTs, being unique tokens secured by blockchain, grant us the ability to own and trade distinct digital items, from artwork and music to virtual real estate, backed by an immutable ledger of ownership.

Our venture into the NFT Marketplace domain involves intricate technical considerations. We dive into the mechanics of blockchain technology, analyzing its consensus protocols, cryptographic security, and decentralized architecture that ensure the authenticity and provenance of each NFT transaction. This project aims not just to understand the technology but to harness it for creating a seamless and secure platform.

1.2. Objective:

The primary objective of our project is to conceptualize, develop, and launch a cutting-edge NFT Marketplace that redefines the digital landscape. Through this venture, we aim to achieve the following goals:

- **Technical Innovation:** Design and implement an NFT Marketplace that leverages blockchain technology to ensure the security, immutability, and authenticity of NFT transactions. By employing advanced cryptographic techniques and decentralized architecture, our platform will establish a new standard for secure digital asset ownership and trading.
- **User-Centric Experience:** Craft an intuitive and user-friendly interface that caters to both creators and collectors. Our platform will provide a seamless and engaging experience, enabling users to seamlessly mint, list, and trade NFTs while fostering a sense of community within the digital art and collectibles space.
- **Ensuring Authenticity:** Establish a transparent and tamper-proof ledger through blockchain technology, assuring buyers of the authenticity and provenance of the NFTs they acquire. By implementing smart contracts, we will automate the enforcement of ownership rights and streamline the process of NFT transfer.
- **Educational Outreach:** Provide educational resources to users, creators, and collectors about NFT technology, its benefits, and potential risks. By fostering awareness and understanding, we aim to contribute to the responsible growth and adoption of NFTs.
- **Research and Analysis:** Conduct an in-depth analysis of the evolving NFT landscape, including market trends, legal considerations, and technological advancements. This research will enable us to adapt and innovate within a dynamic and rapidly changing environment.

Through the achievement of these objectives, our project seeks to contribute to the ongoing transformation of digital ownership, creative expression, and value exchange. We aim to leave a lasting

impact by creating a platform that not only embraces innovation but also empowers individuals in the realm of digital art, collectibles, and creative entrepreneurship.

1.3. Proposed System: Making NFT Marketplaces More Reliable

NFT marketplaces, where digital stuff like art and collectibles are bought and sold, have a big challenge. There's no clear way to prove if things are real or not, which makes people doubt if what they're buying is genuine. Sometimes, fake items and arguments over who owns what pop up because of this. Also, using these marketplaces can be tricky, especially for regular folks.

Our plan is to use a special kind of technology called blockchain to fix this. With blockchain, we can be sure things are real and where they came from. We want to create a marketplace that's easy to use and helps everyone feel safe when they buy or sell digital things. This way, people can trust NFTs more and use them without any confusion. Our goal is to make NFT marketplaces simple, secure, and welcoming for everyone.

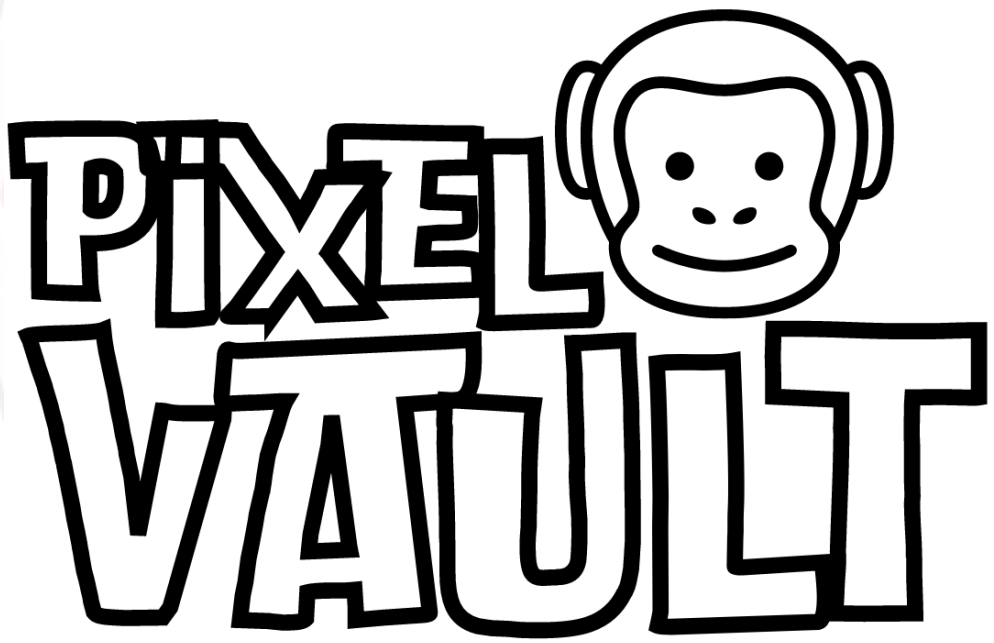

Figure 1.1: Pixelvault

CHAPTER – 2
SOFTWARE REQUIREMENT AND ANALYSIS

2. REQUIREMENT ANALYSIS

Requirement analysis is significant and essential activity after elicitation. We analyze, refine, and scrutinize the gathered requirements to make consistent and unambiguous requirements. This activity reviews all requirements and may provide a graphical view of the entire system. After the completion of the analysis, it is expected that the understandability of the project may improve significantly. Here, we may also use the interaction with the customer to clarify points of confusion and to understand which requirements are more important than others.

- Measurable
- Specific
- Achievable / Acceptable
- Realistic / Relevant
- Time-Bounded

2.1. REQUIREMENT ANALYSIS PROCESS

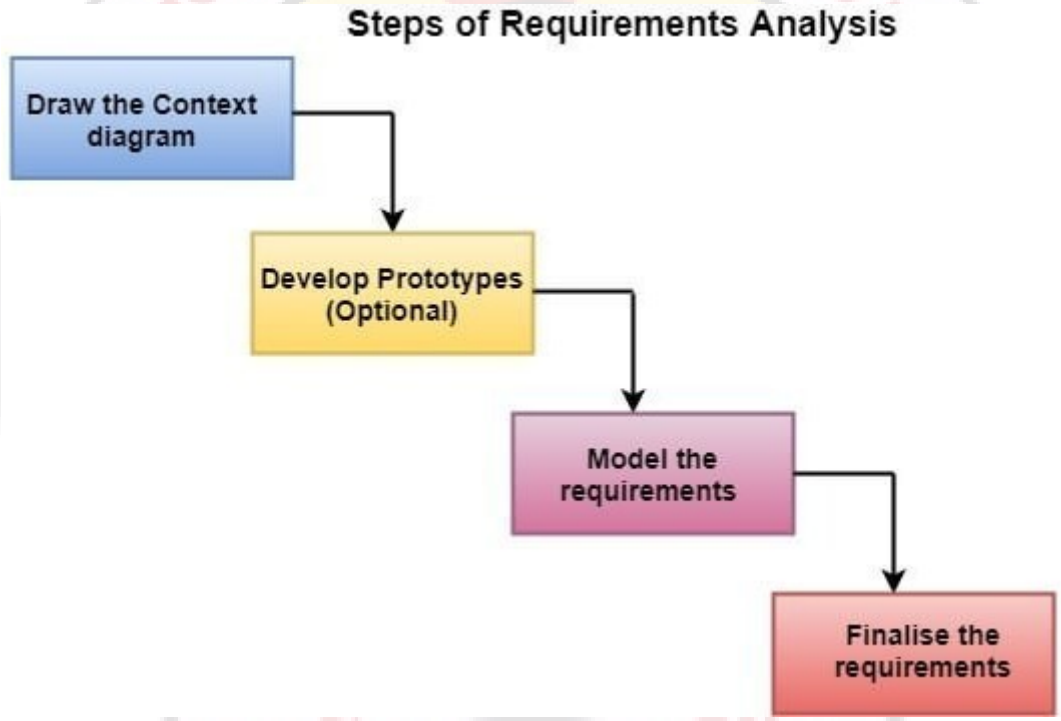

Figure 2.1: Steps of Requirements Analysis

Draw the context diagram: The context diagram is a simple model that defines the boundaries and interfaces of the proposed systems with the external world. It identifies the entities outside the proposed system that interact with the system.

Model the requirements: This process usually consists of various graphical representations of the functions, data entities, external entities, and the relationships between them. The graphical view may help to find incorrect, inconsistent, missing, and superfluous requirements. Such models include the Data Flow diagram, Entity-Relationship diagram, Data Dictionaries, State-transition diagrams, etc.

Finalise the requirements: After modeling the requirements, we will have a better understanding of the system behavior. The inconsistencies and ambiguities have been identified and corrected. The flow of

data amongst various modules has been analyzed. Elicitation and analyze activities have provided better insight into the system.

2.2. GATHER THE REQUIREMENTS

To begin the requirements analysis process, communicate with users to gather the requirements. Analysts can use different techniques to gather the requirements, including:

- Discussion with the team members
- Online surfing

2.3. ANALYSE THE REQUIREMENTS

In this phase, evaluate system feasibility, and confirm with the quality assurance team that the requirements are testable. The goal of this phase is to decompose, analyse and detail the requirements across the system's design. Here are the attributes of good requirements to help you study and define your list:

- Unique
- Complete
- Necessary
- Verifiable
- Consistent
- Quantifiable
- Clear and concise
- Operationally effective
- Able to be validated

2.4. IMPROVE THE QUALITY OF THE REQUIREMENTS

- Visualization: Use tools such as visualization and simulation to understand the desired product.
- Document dependencies: Document relationships and dependencies among requirements.

2.5. MODEL THE REQUIREMENTS

In this phase, it’s time to create models of the requirements, which help stakeholders and customers visualize the potential system. Then we can present the requirements using flowcharts, graphs or models to ensure the system corresponds to the business' needs.

2.6. DOCUMENT AND REVIEW THE REQUIREMENTS

Lastly, we can record the requirements in a document that is easy for both developers and users to understand. We can document the requirements in various formats, including:

- Software requirements specification
- Use cases
- Natural-language documents

- Process specification

2.7. TECHNOLOGIES

2.7.1. FRONTEND TECHNOLOGIES

2.7.1.1. Next.js

Next.js is a popular frontend development framework built on top of React.js. It's designed to make building React applications easier and more efficient by providing features like server-side rendering (SSR), automatic code splitting, client-side routing, and more.

Here are some key features and concepts associated with Next.js:

1. Server-side rendering (SSR): Next.js allows you to render React components on the server side before sending them to the client, which can improve performance and SEO.

2. Automatic code splitting: Next.js automatically splits your code into smaller bundles, which are only loaded when needed. This helps in reducing the initial load time of your application.

3. Client-side routing: Next.js provides a built-in routing system that allows you to define routes using the file system. This makes it easy to create dynamic and SEO-friendly URLs for your application.

4. Static site generation (SSG): In addition to SSR, Next.js also supports static site generation, where pages can be pre-built at build time. This is useful for building static websites or pages with content that doesn't change frequently.

5. API routes: Next.js allows you to create API routes within your application, which can be used to fetch data from a server or perform other server-side tasks.

6. Built-in CSS support: Next.js comes with built-in support for CSS modules, CSS-in-JS libraries like styled-components, and global CSS files.

7. TypeScript support: Next.js has built-in TypeScript support, allowing you to write type-safe code and catch errors early in the development process.

Overall, Next.js provides a powerful and flexible framework for building modern web applications with React.js, offering features that improve performance, SEO, and developer experience.

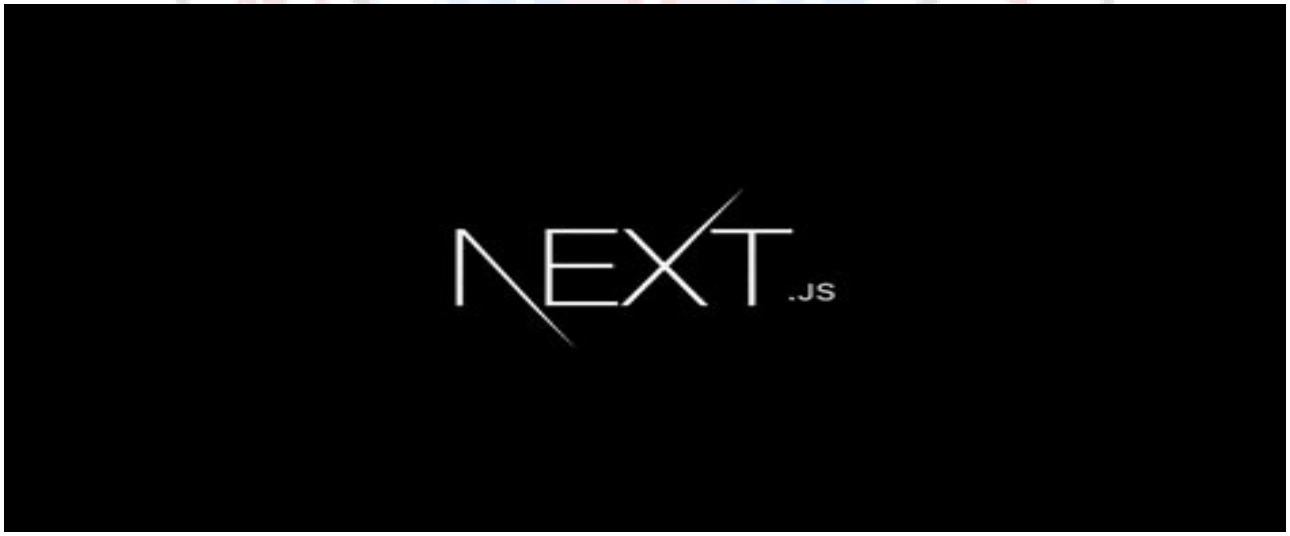

Figure 2.2: Nextjs

2.7.1.2. Reactjs

REACT could refer to several things depending on the context. If you're referring to React.js, it's a popular JavaScript library for building user interfaces. Some of its key features include:

1. Component-Based Architecture: React allows developers to build encapsulated components that manage their state, which makes it easier to build complex UIs.

2. Virtual DOM: React uses a virtual DOM to improve performance by minimizing the number of direct manipulations to the actual DOM. It updates only the parts of the DOM that have changed, resulting in faster rendering.

3. JSX: JSX is a syntax extension for JavaScript that allows developers to write HTML-like code within JavaScript. This makes it easier to write and understand the structure of UI components.

4. Declarative: React follows a declarative programming paradigm, where developers describe the desired UI state, and React takes care of updating the DOM to match that state. This leads to more predictable and maintainable code.

5. Unidirectional Data Flow: React follows a unidirectional data flow, where data flows only in one direction—from parent to child components. This makes it easier to understand how data changes propagate through the application.

6. React Hooks: Introduced in React 16.8, hooks are functions that allow developers to use state and other React features without writing a class. Hooks enable functional components to have state and lifecycle methods, simplifying code and promoting reusability.

7. Context API: React provides a Context API, which allows data to be passed through the component tree without having to pass props down manually at every level. This is particularly useful for global state management.

8. Reusable Components: React promotes the creation of reusable UI components, which can be composed together to build complex user interfaces. This component-based approach enhances code modularity and maintainability.

9. React Router: While not part of React core, React Router is a popular library for handling routing in React applications. It allows developers to define routes and navigate between different parts of the application without reloading the page.

These are just some of the features and concepts associated with React.js, which has become a go-to choice for building modern web applications due to its efficiency, flexibility, and large ecosystem of libraries and tools.

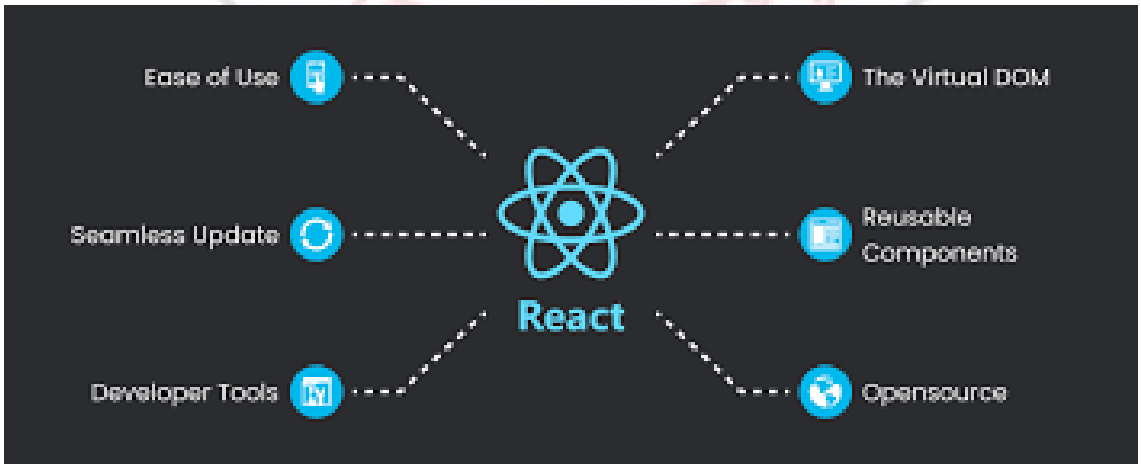

Figure 2.3: Reactjs

2.7.1.3. JSX

JSX, or JavaScript XML, is a syntax extension for JavaScript that allows you to write HTML-like code directly within JavaScript. It's most commonly associated with React.js, where it's used to describe what the UI should look like. Here's a breakdown of JSX:

1. Syntax: JSX looks similar to HTML but is actually JavaScript. It allows you to write HTML-like syntax within JavaScript files, making it easier to describe the structure of your UI components.

2. Embedding Expressions: JSX allows you to embed JavaScript expressions within curly braces `{}`. This allows you to dynamically generate content or insert variables into your JSX code.

3. Components: JSX allows you to define custom components using a syntax that resembles HTML tags. These components can accept props (properties) and can be composed together to build complex UIs.

4. Event Handling: JSX allows you to attach event handlers directly to elements using standard HTML event attributes like `onClick`, `onChange`, etc. These event handlers can be functions defined in JavaScript.

5. Expressions and Statements: JSX supports JavaScript expressions and statements, allowing you to use conditionals (`if` statements), loops (`for` and `while` loops), and other JavaScript constructs directly within JSX.

6. Babel Transformation: JSX code is transformed into regular JavaScript code by tools like Babel before it's executed in the browser. This transformation typically involves converting JSX elements into `React.createElement()` calls.

Overall, JSX is a powerful and expressive way to describe UI components in React.js applications, allowing developers to write code that is more readable and maintainable compared to using pure JavaScript to create DOM elements.

Figure 2.4: JSX

2.7.1.4. Tailwind CSS

Tailwind CSS is a utility-first CSS framework that provides low-level utility classes to quickly build custom designs without having to write CSS from scratch. Here's an overview of Tailwind CSS:

1. Utility-first approach: Unlike traditional CSS frameworks that provide pre-designed components, Tailwind CSS focuses on providing a set of utility classes that you can apply directly to your HTML

elements. These utility classes handle common styling tasks like margins, padding, colors, typography, and more.

2. Atomic CSS: Tailwind CSS follows the atomic CSS approach, where each utility class represents a single CSS property. This allows for highly granular control over the styling of elements, and enables rapid prototyping and development.

3. Responsive design: Tailwind CSS includes responsive utility classes that allow you to easily create responsive layouts. You can apply different styles to elements based on screen size using classes like `sm:`, `md:`, `lg:`, and `xl:`.

4. Customization: Tailwind CSS is highly customizable. You can configure the framework to generate only the utility classes you need, which helps in keeping the final CSS file size small. Additionally, you can extend or override existing utility classes to match your project's specific design requirements.

5. Plugin ecosystem: Tailwind CSS has a vibrant ecosystem of plugins that extend its functionality. These plugins can add new utility classes, integrate with popular design systems, or provide additional features like dark mode support.

6. Just-in-time (JIT) mode: Tailwind CSS introduced a JIT mode that dynamically generates CSS on-demand based on your HTML templates. This can significantly reduce the final CSS file size and improve build times, especially in large projects.

Overall, Tailwind CSS is a modern CSS framework that offers a pragmatic approach to styling web applications. It's particularly well-suited for rapid prototyping, small to medium-sized projects, and teams that value consistency and maintainability in their CSS codebase.

Figure 2.5: Tailwind CSS

2.7.1.5. NextUI

NextUI is a UI library for React that helps you build beautiful and accessible user interfaces. Created on top of Tailwind CSS and React Aria.

NextUI's primary goal is to streamline the development process, offering a beautiful and adaptable system design for an enhanced user experience.

NextUI is a UI library for React that combines the power of TailwindCSS with React Aria to provide complete components (logic and styles) for building accessible and customizable user interfaces. Since NextUI uses TailwindCSS as its style engine, you can use all TailwindCSS classes within your NextUI components, ensuring optimal compiled CSS size.

TailwindCSS components libraries such as TailwindUI, Flowbite, or Preline, just to name a few, only offer a curated selection of TailwindCSS classes to style your components. They don't provide any React specific components, logic, props, composition, or accessibility features.

In contrast to these libraries, NextUI is a complete UI library that provides a set of accessible and customizable components, hooks, and utilities.

Figure 2.6: NextUI

2.7.1.6. FRAMER MOTION

We use Framer Motion to animate some components due to the complexity of the animations and their physics-based nature. Framer Motion allows us to handle these animations in a more straightforward and performant way. In addition, it is well tested and production ready.

Most animations will be performed with the motion component and the animate prop.

When any value in animate changes, the component will automatically animate to the updated target.

Framer Motion is a popular open-source library for React that makes it easy to create fluid animations and interactive UI components. Some of its key features include:

1. Declarative Syntax: Framer Motion uses a declarative syntax to define animations and interactions, making it easy to understand and use.

2. Spring Physics: It provides a physics-based animation system, including spring animations, which create smooth and natural-looking motion.

3. Variants: Variants allow you to define sets of properties for different states or variants of a component, such as hover, tap, or drag states, and easily switch between them.

4. Gesture Recognition: Framer Motion includes built-in support for gesture recognition, allowing you to create interactive components that respond to user input like dragging, tapping, and swiping.

5. Layout Animation: It supports layout animations, allowing components to smoothly transition between different sizes and positions.

6. Server-Side Rendering (SSR) Compatibility: Framer Motion is compatible with server-side rendering, making it suitable for building universal React applications.

7. Custom Transitions: You can create custom transition animations using Framer Motion, allowing for unique and creative effects.

8. Accessibility: Framer Motion provides built-in accessibility features, ensuring that your animations and interactive components are usable by everyone.

These features make Framer Motion a powerful tool for creating rich and engaging user interfaces in React applications.

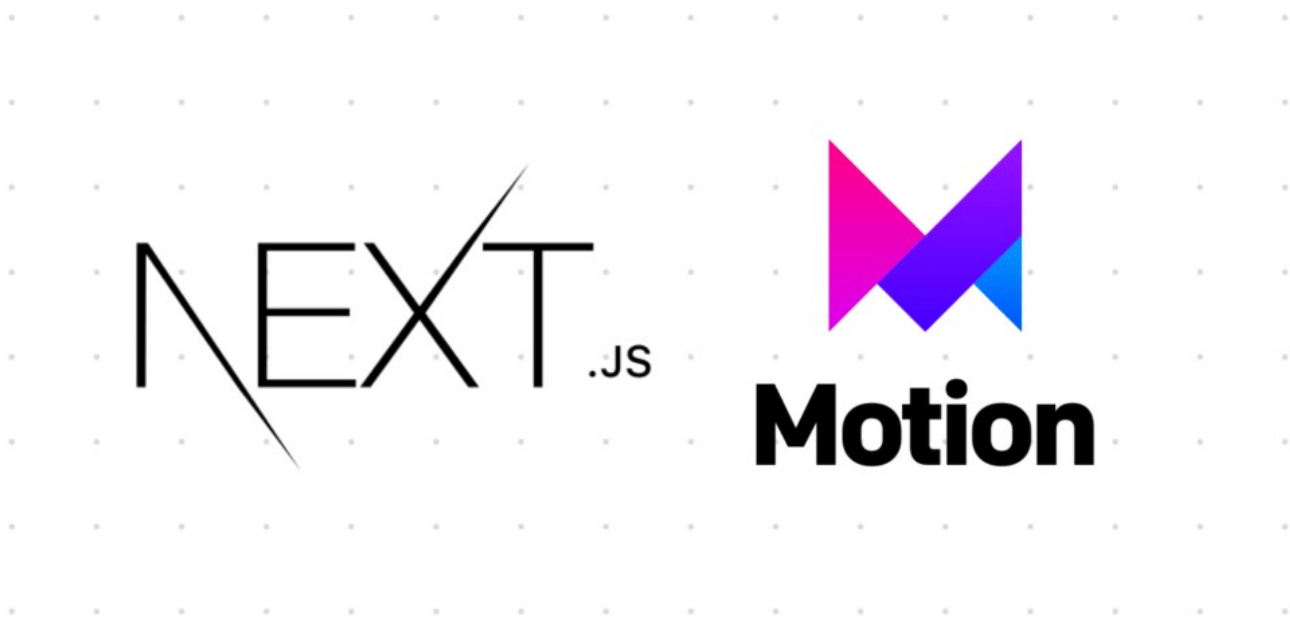

Figure 2.7: Framer Motion

2.7.1.7. REACT ICON

The React Icons library is a popular npm package that provides a collection of commonly used icons for React applications. It allows you to easily include icons from popular icon libraries like Font Awesome, Material Icons, and many others, directly into your React components.

The react-icons npm library is a popular choice for integrating icon sets into React applications. It provides a vast collection of icons from various icon libraries like Font Awesome, Material Icons, etc., allowing developers to easily include icons in their React projects without having to manually import individual SVGs or icon font files. Here are some of its key features:

1. Easy Integration: react-icons simplifies the process of integrating icons into React components. You just need to install the library via npm and import the desired icon components.

2. Wide Variety of Icon Sets: It offers a wide range of icon sets, including popular ones like Font Awesome, Material Icons, Ionicons, Feather Icons, and more. This gives developers flexibility in choosing the icon style that best fits their project's design.

3. Customization: The library provides options for customizing icons, such as adjusting size, color, and other properties using props. This allows developers to seamlessly integrate icons into their UI design.

4. Component-Based Approach: Icons are exposed as React components, making them easy to use within JSX. This approach enables developers to treat icons as regular React components, allowing for dynamic rendering and easy integration with other React elements.

5. Optimized for Performance: react-icons is designed to be lightweight and optimized for performance. It ensures that the inclusion of icons in your React application doesn't significantly impact load times or overall performance.

6. Active Maintenance: The library is actively maintained and updated, ensuring compatibility with the latest versions of React and continued support for new icon sets.

7. Community Support: react-icons has a large community of users and contributors, which means you can find plenty of resources, tutorials, and community-driven enhancements to help you use the library effectively.

Overall, react-icons simplifies the process of adding icons to React applications by providing a comprehensive collection of icon sets and easy-to-use components, making it a go-to choice for many React developers.

Figure 2.8: React Icons

2.7.1.7. THIRD WEB REACT SDK

The React SDK uses React Query under the hood to expose a collection of query and mutation hooks, each with built-in caching, query invalidation, query retries, and more.

Each hook (except for wallet/network management) wraps some functionality of the TypeScript SDK, which are made available as either a query hook to read data, or as a mutation hook to write transactions to the blockchain.

When mutations are called (when a user executes a transaction), query invalidation is automatically triggered to update the relevant queries that depend on the data that was changed. For example, when minting a new NFT, queries that view information about NFTs are re-fetched to load the new NFT automatically.

Queries

All query hooks are used to read data from the blockchain, smart contracts, a user's wallet, etc.

Each one comes with some helpful properties to create a powerful user experience:

data - The data returned from the query (e.g. the NFTs of a contract).

isLoading - Whether the query is currently loading.

error - The error returned from the query, if any.

Figure 2.9: Thirdweb SDK

2.7.1.8. REACT TOSTIFY

React Toastify is a popular library for creating toast notifications in React applications. Here are some of its key features:

1. Customizable Toasts: You can customize the appearance of the toast notifications according to your application's design requirements.

2. Positioning Options: Toastify offers various positioning options such as top-right, top-left, bottom-right, and bottom-left, allowing you to place notifications wherever you prefer on the screen.

3. Toast Types: You can create different types of toasts, including success, error, warning, and info, to convey different types of messages to users.

4. Auto-Close: Toasts can be set to automatically close after a specified duration, providing a non-intrusive way to display information to users.

5. Controlled and Uncontrolled Components: Toastify supports both controlled and uncontrolled components, giving you flexibility in managing the state of toast notifications.

6. Event Handling: You can attach event handlers to toast notifications, allowing you to execute custom logic when a toast is clicked or closed.

7. Transition Effects: The library provides built-in transition effects for showing and hiding toast notifications, enhancing the user experience.

8. Accessibility: Toastify is designed with accessibility in mind, ensuring that toast notifications are accessible to all users, including those using assistive technologies.

9. Easy to Use: With a simple and intuitive API, Toastify makes it easy to integrate toast notifications into your React application with just a few lines of code.

Overall, React Toastify offers a comprehensive solution for adding toast notifications to your React applications with flexibility, customization options, and accessibility in mind.

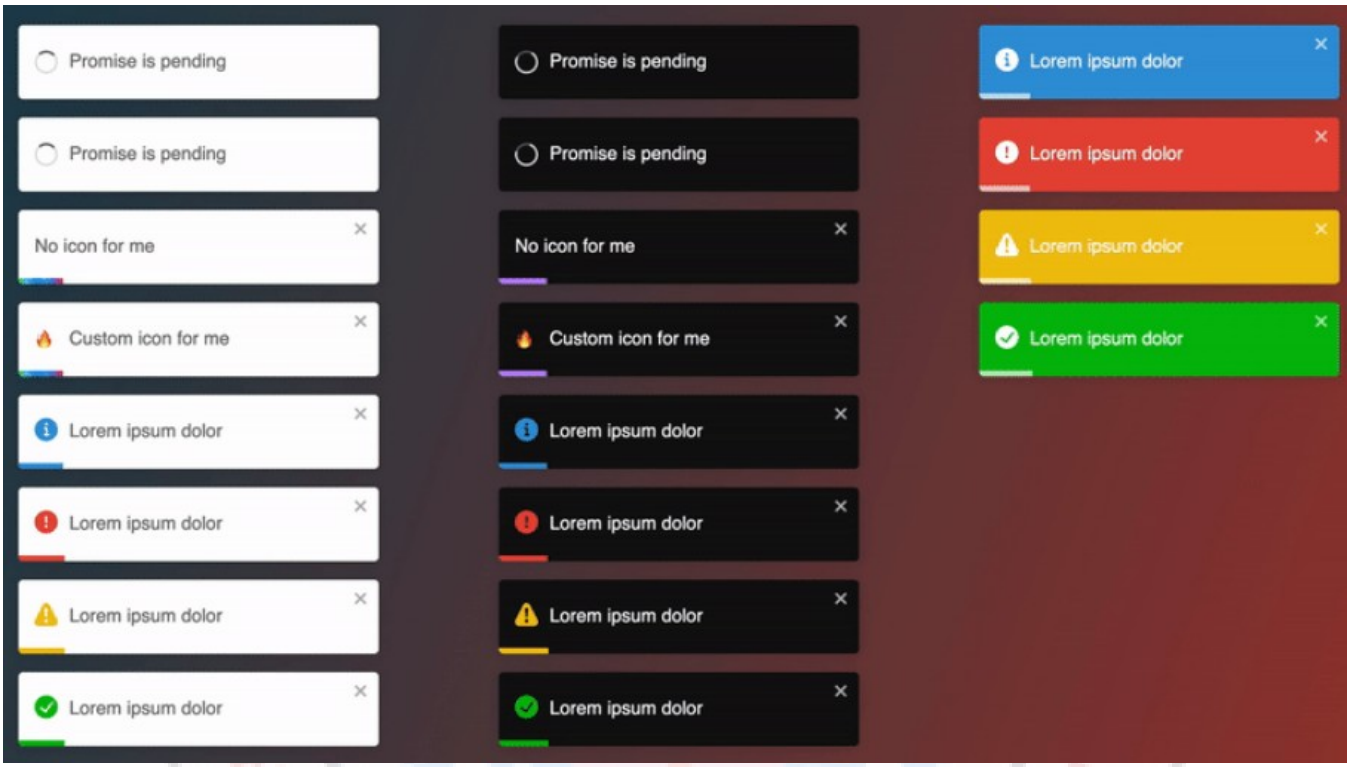

Figure 2.10: React Tostify

2.7.2. BACKEND TECHNOLOGIES

2.7.2.1. MongoDB

MongoDB is a powerful NoSQL database that has become a cornerstone in modern web development, offering unparalleled flexibility, scalability, and performance. Unlike traditional relational databases, MongoDB utilizes a document-oriented data model, storing data in flexible JSON-like documents that can vary in structure, making it ideal for handling diverse and dynamic datasets. One of MongoDB's key strengths lies in its ability to scale effortlessly, both horizontally and vertically. Horizontal scaling, or sharding, allows MongoDB to distribute data across multiple servers, enabling seamless scalability as data volumes grow. Vertical scaling, on the other hand, involves increasing the resources of individual servers, providing additional capacity to handle increased workload and throughput. Moreover, MongoDB's flexible schema design enables developers to iterate quickly and adapt to changing requirements without the need for complex migrations. This agility is particularly beneficial in agile development environments where requirements are subject to frequent changes.

Figure 2.11: MongoDB

2.7.2.2. Mongoose

Mongoose is a powerful Node.js library for MongoDB that provides a straightforward, schema-based solution for modelling data. It offers a higher level of abstraction over MongoDB's native driver, simplifying the process of interacting with MongoDB databases and enabling developers to build scalable and maintainable applications with ease. At its core, Mongoose enables developers to define data schemas that describe the structure of their data, including the types of fields, constraints, and validation rules. This schema-based approach promotes consistency and clarity in the data model, making it easier to understand and maintain over time. One of the key benefits of using Mongoose is its built-in support for data validation. Developers can define validation rules for each field in the schema, ensuring that data stored in the database meets specific criteria. This helps prevent data inconsistencies and ensures data integrity, improving the overall quality of the application. Furthermore, Mongoose provides a rich set of features for interacting with MongoDB databases, including support for CRUD (Create, Read, Update, Delete) operations, data querying, aggregation, and indexing. Its intuitive API and query builder make it easy to perform complex operations while maintaining readability and expressiveness in the code.

Figure 2.12: Mongoose

2.7.2.3. Validator.js

Validator.js is a versatile library for validating and sanitizing input data in JavaScript applications. It provides a comprehensive set of functions for validating various types of data, including strings, numbers, dates, and emails, as well as custom validation rules. One of the key features of Validator.js is its simplicity and ease of use. Developers can quickly integrate Validator.js into their applications and start validating input data with minimal effort. The library provides intuitive functions for common validation tasks, such as checking if a value is empty, validating email addresses, or ensuring that a string meets certain length requirements.Validator.js also offers robust sanitization capabilities, allowing developers to clean input data and remove potentially harmful characters or content. This helps prevent

security vulnerabilities such as cross-site scripting (XSS) attacks and SQL injection, ensuring that applications remain secure and resilient to malicious input. Furthermore, Validator.js supports both synchronous and asynchronous validation methods, making it suitable for a wide range of use cases, from simple form validation to complex data validation tasks in server-side applications. Its flexible API allows developers to customize validation rules and error messages to meet the specific requirements of their applications. Moreover, Validator.js is lightweight and dependency-free, making it easy to integrate into any JavaScript project without adding unnecessary overhead. It is compatible with both browser-based and server-side JavaScript environments, allowing developers to use it in frontend and backend applications alike.

Figure 2.13: Validator.js

2.7.2.4. Ehters.js

Ethers.js is a powerful JavaScript library that provides a simple and intuitive interface for interacting with the Ethereum blockchain. Whether you're a seasoned blockchain developer or just starting out, Ethers.js makes it easy to build decentralized applications (dApps), interact with smart contracts, and manage Ethereum accounts programmatically. One of the key features of Ethers.js is its comprehensive API, which abstracts away the complexities of interacting with the Ethereum blockchain, allowing developers to focus on building their applications rather than dealing with low-level protocol details. The library provides intuitive methods for common tasks such as sending transactions, querying blockchain data, and deploying smart contracts, making it accessible to developers of all skill levels. Moreover, Ethers.js offers robust support for Ethereum's JSON-RPC API, allowing developers to interact with any Ethereum node that exposes this interface. This flexibility enables developers to choose the Ethereum node that best suits their needs, whether it's a public node like Infura or a self-hosted node running Geth or Parity. Furthermore, Ethers.js includes powerful utilities for working with Ethereum accounts and keys, including generating new accounts, signing transactions, and encrypting private keys. This makes it easy to integrate Ethereum wallet functionality into dApps, allowing users to securely manage their funds without relying on third-party services. Additionally, Ethers.js provides comprehensive support for Ethereum smart contracts, including compiling Solidity code, deploying contracts, and interacting with contract instances. Its robust type system and smart contract ABI (Application Binary Interface) decoder make it easy to work with complex contract interfaces and data structures, reducing the likelihood of errors and simplifying development.

CHAPTER – 3
ARCHITECTURE OF PIXELVAULT

3. ARCHITECTURE OF PIXELVAULT

The design of an NFT marketplace web app is a critical aspect that influences user engagement, trust, and overall usability. The user interface (UI) and user experience (UX) should be carefully crafted to provide a seamless and visually appealing environment for buyers, sellers, and enthusiasts within the NFT ecosystem.

The homepage should feature an intuitive layout that showcases trending NFTs, featured artists, and upcoming auctions. A well-organized navigation system should enable users to easily explore different categories, artists, and collections. Clear and attractive visuals, including high-resolution previews of NFTs, contribute to a visually stimulating experience, allowing users to appreciate the digital assets before making a purchase.

The individual NFT listing pages need to be designed for optimal presentation, displaying detailed information about the artwork, artist, and transaction history. Users should be able to view high-quality images or media files associated with the NFT and access essential details such as ownership history and metadata. Integrating social elements, like comments, likes, and share buttons, fosters community engagement around each NFT.

The individual NFT listing pages need to be designed for optimal presentation, displaying detailed information about the artwork, artist, and transaction history. Users should be able to view high-quality images or media files associated with the NFT and access essential details such as ownership history and metadata. Integrating social elements, like comments, likes, and share buttons, fosters community engagement around each NFT.

3.1. ER Diagram

An Entity-Relationship (ER) diagram is a visual representation of the data model that depicts the entities, attributes, relationships, and constraints within a system or application. Let's break down the components of an ER diagram in detail:

- **Entities:** Entities represent real-world objects or concepts within the system being modeled. In a business context like Pixelvault, entities could include 'User', 'Product', 'Order', 'Transaction', etc. Each entity is typically represented by a rectangle in the diagram.
- **Attributes:** Attributes describe the properties or characteristics of entities. They provide additional details about each entity. For example, attributes of a 'User' entity might include 'UserID', 'Username', 'Email', 'Password', etc. Attributes are usually depicted within ovals connected to their respective entities.
- **Relationships:** Relationships define how entities are related to each other. They represent the associations and interactions between entities. In an ER diagram, relationships are illustrated using lines connecting entities. Each relationship is labeled to indicate the nature of the association, such as 'has', 'belongs to', 'purchases', etc. Relationships can be one-to-one, one-to-many, or many-to-many, depending on the cardinality of the association.
- **Cardinality:** Cardinality specifies the number of instances of one entity that are associated with the number of instances of another entity through a relationship. Cardinality is indicated using symbols such as '1' (one), 'M' (many), or '0..1' (zero or one) placed near the ends of the relationship lines. It helps define the nature and constraints of the relationship between entities.
- **Primary Keys:** Primary keys uniquely identify each record within an entity. In an ER diagram, primary keys are denoted by underlining the attribute(s) that serve as the primary key for each

entity. Primary keys are essential for ensuring data integrity and facilitating efficient data retrieval operations.

- **Foreign Keys:** Foreign keys establish relationships between entities by referencing the primary key of another entity. They are attributes within an entity that refer to the primary key of another entity. Foreign keys help maintain referential integrity and enforce constraints between related entities.

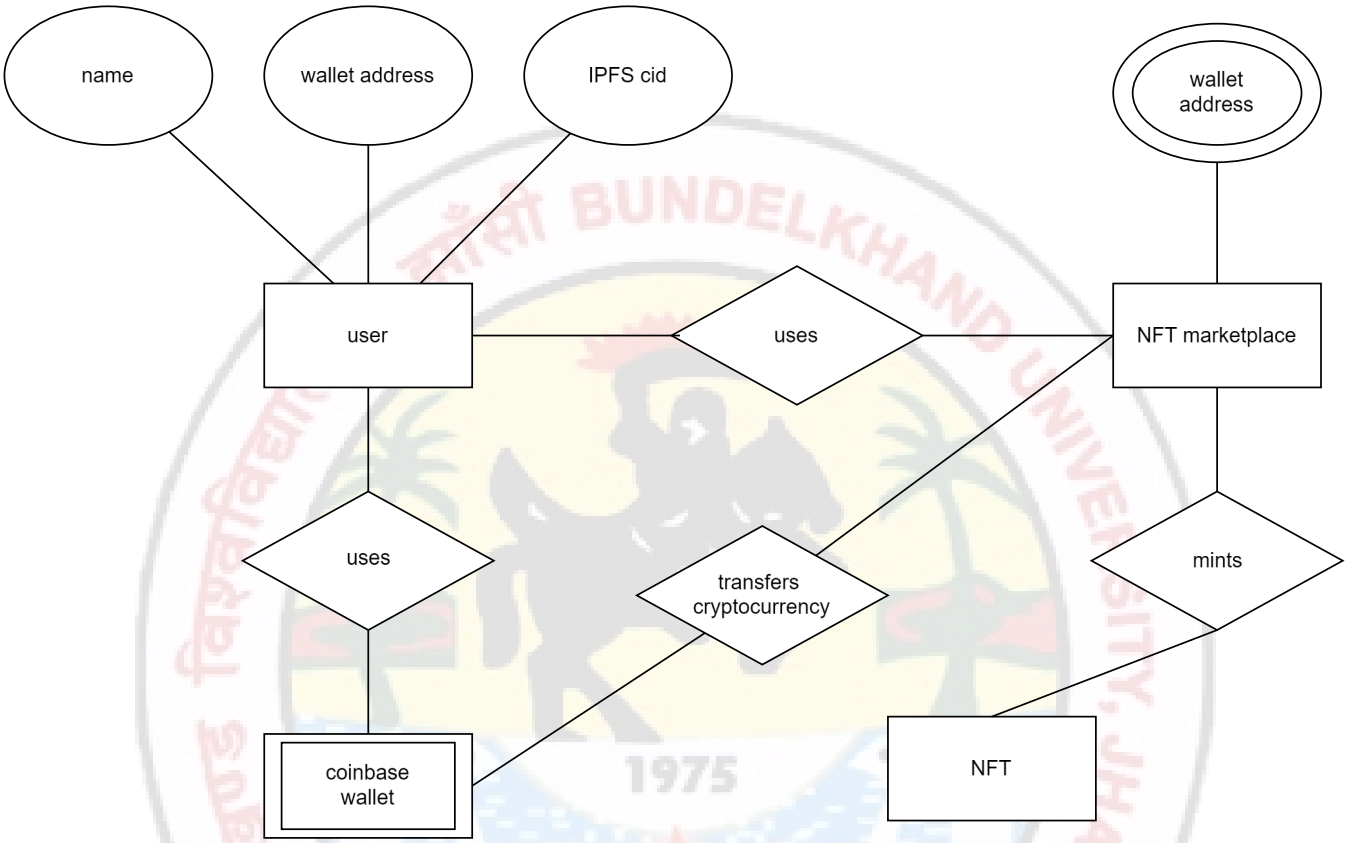

Figure 3.1: ER Diagram

3.2. Dataflow Diagram

Data Flow Diagram represents the flow of data during the procedure of buying NFT assets. Following are some insights about it:

- The User requests to view the Marketplace Catalogue.
- User may filter or sort the results according to their liking.
- The User chooses an item of interest and requests to view it.
- On the Item View page, the user may request to buy the item.
- Payment is approved by the User Wallet.
- Amount gets deducted from the User Wallet and gets added to Seller Wallet.
- The NFT token gets transferred to User Wallet.
- All transactions get added to Transaction Database.

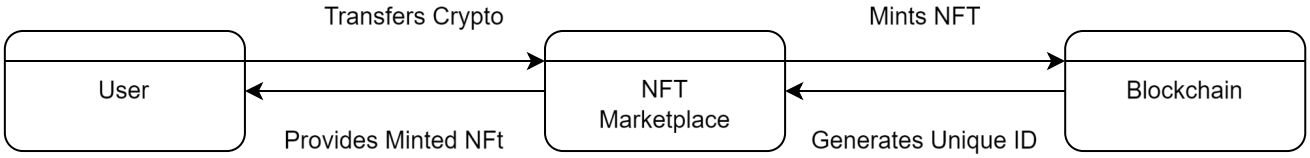

Figure 3.2: Level 0 Dataflow Diagram

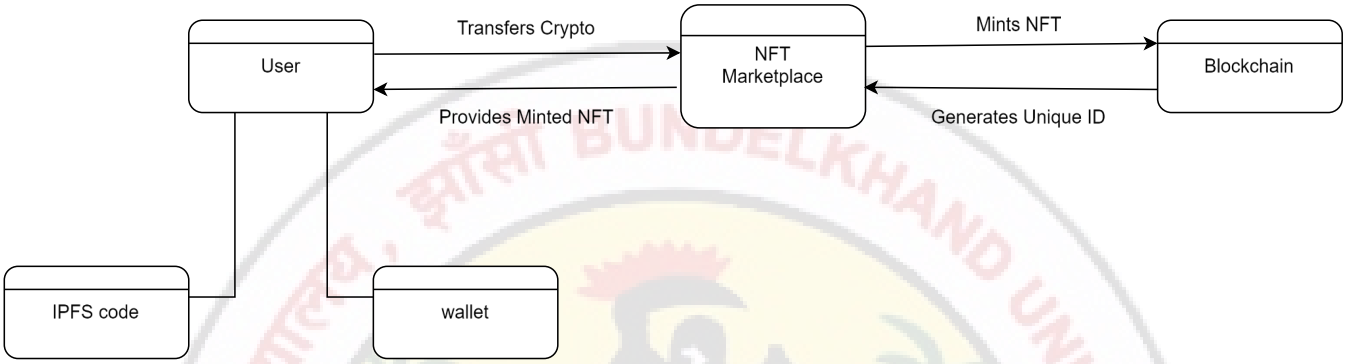

Figure 3.3: Level 1 Dataflow Diagram

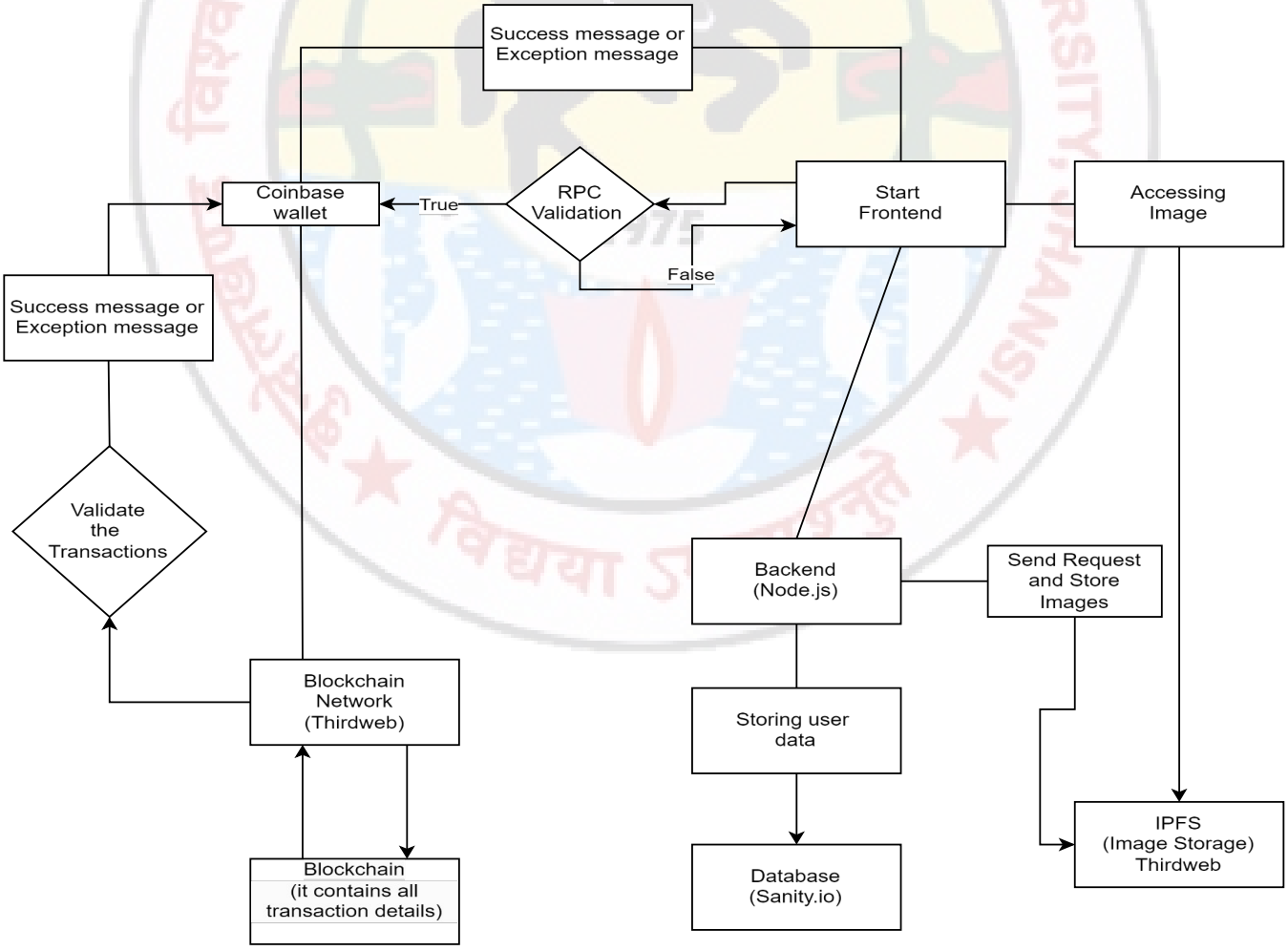

Figure 3.4: Level 2 Dataflow Diagram

3.3. UML Diagram

- Access PixelVault Web App: Go to the PixelVault web app.
- Explore NFT Marketplace: Navigate the marketplace to discover featured and trending NFTs.
- View NFT Details: Click on an NFT to see its details, including description, creator info, and pricing.
- Connect Digital Wallet: Connect your digital wallet, such as MetaMask, to trade or exchange NFTs.
- Buy NFTs: Purchase NFTs directly from the marketplace. Confirm the transaction through your wallet.
- Sell NFTs: List NFTs for sale with specified prices. Wait for potential buyers to make transactions.
- Auctions and Bidding: Participate in auctions by placing bids on NFTs.The highest bidder wins the item.
- Swapping or Exchanging: Explore features for swapping or exchanging NFTs directly with other users on the platform.
- Transaction Confirmations: Confirm transactions through your connected digital wallet. Verify funds and review associated fees.
- Ownership and Transfers: NFT ownership transfers on the blockchain. Confirm changes in ownership within your wallet.

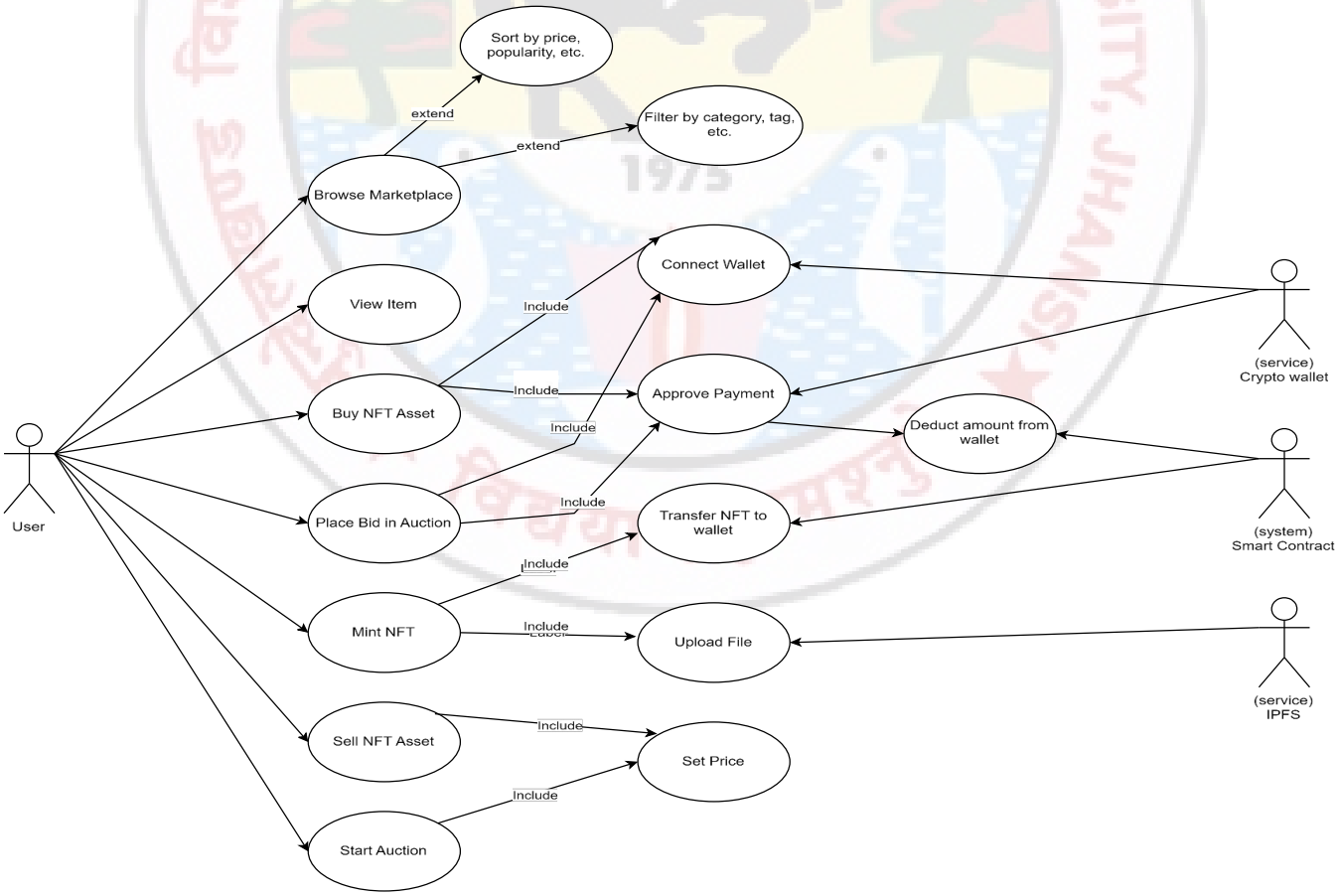

Figure 3.5: UML Diagram

CHAPTER – 4
BACKEND IMPLEMENTATION

4. BACKEND IMPLEMENTATION

In the digital age, businesses are increasingly relying on sophisticated backend systems to power their online platforms and services. Pixelvault, a leading e-commerce platform, is no exception. At the heart of Pixelvault's operations lies a powerful backend system built using MongoDB and Node.js, two cutting-edge technologies that enable seamless data storage, retrieval, and processing.

MongoDB, a NoSQL database, offers Pixelvault the flexibility and scalability needed to manage vast amounts of dynamic data efficiently. With its document-oriented data model, MongoDB allows Pixelvault to store complex data structures in a format that closely resembles JSON, facilitating agile development and adaptation to evolving business needs. As Pixelvault continues to grow and expand its product offerings, MongoDB provides the foundation for a scalable and responsive data storage solution.

Complementing MongoDB is Node.js, a server-side JavaScript runtime renowned for its asynchronous and event-driven architecture. By leveraging Node.js, Pixelvault can build a high-performance backend system capable of handling concurrent requests and delivering real-time updates to users. With JavaScript serving as the lingua franca for both frontend and backend development, Pixelvault benefits from streamlined collaboration between development teams and accelerated time-to-market for new features and enhancements.

Together, MongoDB and Node.js empower Pixelvault to create a backend system that not only meets the demands of today's dynamic e-commerce landscape but also lays the groundwork for future innovation and growth. In this exploration, we will delve into the intricacies of how Pixelvault harnesses MongoDB and Node.js to build a robust, scalable, and feature-rich backend infrastructure that drives its success in the competitive world of online retail.

4.1. Connecting Database

```
1  import mongoose from "mongoose";
2
3  const connectDatabase = handler=> async (req,res)=>{
4      if(mongoose.connections[0].readyState){
5          return handler(req,res)
6      }
7      await mongoose.connect(process.env.connectionString)
8      return handler(req,res);
9  }
10 export default connectDatabase;
```

Figure 4.1: Function to connect database

4.2. Creating User Schema

```
import mongoose from "mongoose";

const UserSchema = new mongoose.Schema({
  address: {
    type: String,
    required: true,
    unique:true,
  },
  Name: {
    type: String,
    default: 'User',
  },
  Email: {
    type: String,
    default: '',
  },
  ProfileImage: {
    type: String,
    default: '',
  },
  Username: {
    type: String,
    default: '',
  },
  SocialLinks: {
    type: Object,
    default: {},
  },
});
mongoose.models = {};
export default mongoose.model("UserSchema", UserSchema);

```

Figure 4.2: User schema

4.3. Creating endpoint to fetch user data

```
import UserModel from "../../models/UserModel";
import connectDatabase from "../../middleware/mongoose";
import { isEthereumAddress } from "validator";

const handler = async (req, res) => {
  try {
    if(req.method == "POST"){
      if (!isEthereumAddress(req.body.address)) {
        res.status(400).json({ error: "Please enter valid Ethereum Address!" });
        return;
      }
      let user = await UserModel.findOne({ address: req.body.address });
      res.status(200).json({ user: user });
    }
    else{
      res.status(400).json({ error: "Method not allowed" });
      return
    }
  } catch (err) {
    res.status(500).json({ error: err });
  }
};

export default connectDatabase(handler);

```

Figure 4.3: Endpoint to fetch user data

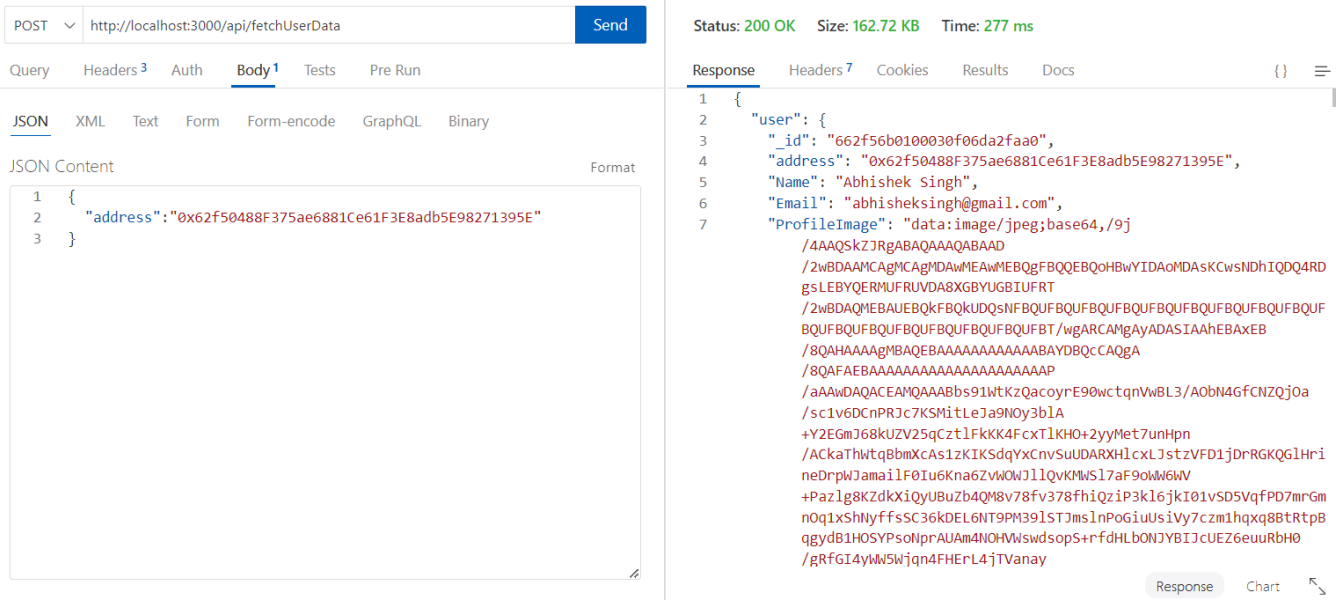

Figure 4.4: Response of endpoint to fetch user data

4.4. Creating endpoint to post user data

```
import UserModel from "../../models/UserModel";
import connectDatabase from "../../middleware/mongoose";
import isEmail from "validator/lib/isEmail";
import { isBase64, isEthereumAddress, isJSON, isLength } from "validator";

const handler = async (req, res) => {
  try {
    if (req.method == "POST") {
      if (!isEthereumAddress(req.body.address)) {
        res.status(400).json({ error: "Please enter valid ethereum address" });
        return;
      }
      const isUserExist = await UserModel.findOne({
        address: req.body.address,
      });
      let user
      if (isUserExist) {
        if (!isLength(req.body.Name, { min: 2, max: undefined })) {
          res
            .status(400)
            .json({ error: "Please enter atleast 3 characters in name" });
          return;
        }
        if (!isEmail(req.body.Email)) {
          res.status(400).json({ error: "Please enter valid email!" });
          return;
        }
        if (!isLength(req.body.Username, { min: 2, max: undefined })) {
          res
            .status(400)
            .json({ error: "Please enter atleast 3 characters in username" });
          return;
        }
        if (!isJSON(JSON.stringify(req.body.SocialLinks))) {
          res
            .status(400)
            .json({ error: "Please enter social links in valid format" });
          return;
        }
         user = await UserModel.findByIdAndUpdate(
          isUserExist._id,
          {
            Name: req.body.Name,
            Email: req.body.Email,
            ProfileImage: req.body.ProfileImage,
            Username: req.body.Username,
            SocialLinks: req.body.SocialLinks,
          },
          {
            new: true,
          }
        );
      } else {
         user = new UserModel({
          address: req.body.address,
          Name: req.body.Name,
          Email: req.body.Email,
          ProfileImage: req.body.ProfileImage,
          Username: req.body.Username,
          SocialLinks: req.body.SocialLinks,
        });
        await user.save();
      }
      res.status(200).json({ success: user });
    } else {
      res.status(400).json({ error: "This method is not allowed" });
    }
  } catch (err) {
    res.status(500).json({ error: err });
  }
};

export default connectDatabase(handler);

```

Figure 4.5: Endpoint to post user data

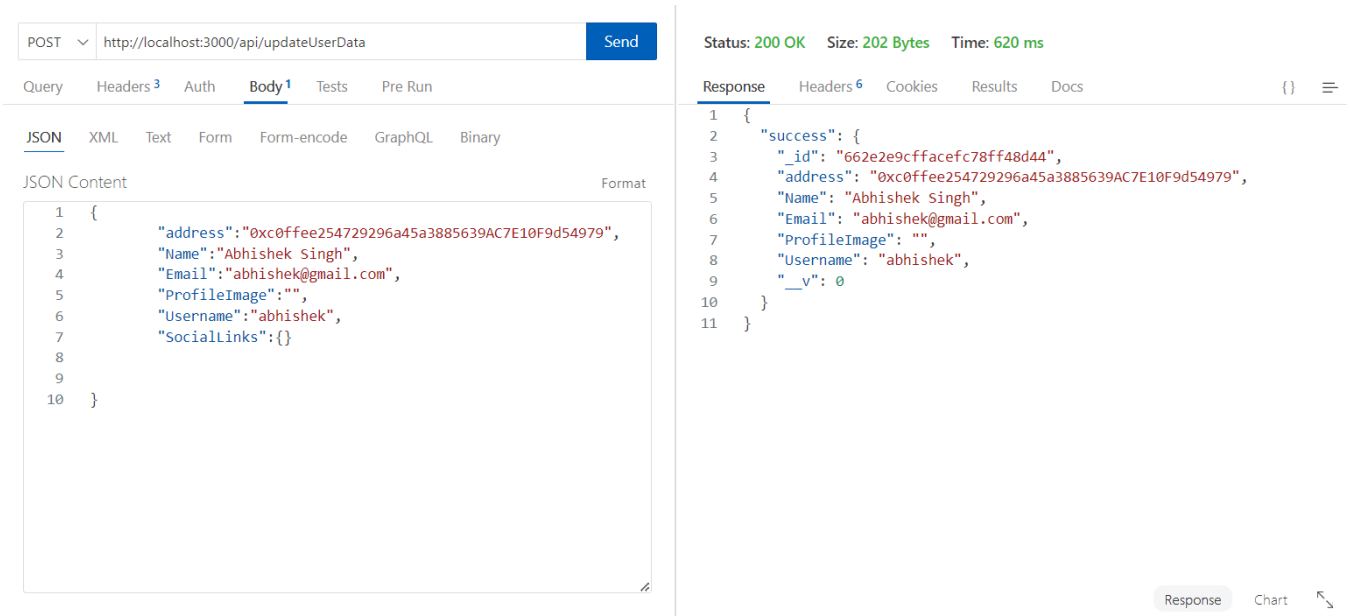

Figure 4.6: Response of endpoint to post user data

4.5. Storing User data on MongoDB

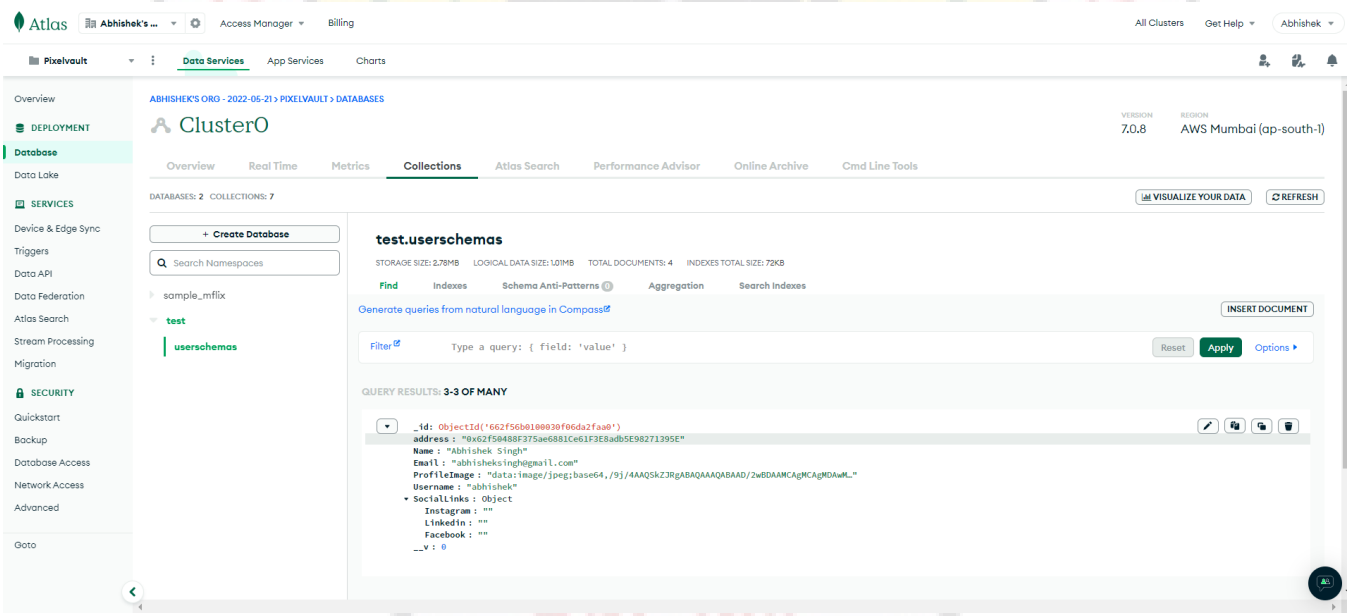

Figure 4.7: Storing User data on MongoDB

CHAPTER – 5
THIRDWEB SETUP

5. THIRDWEB SETUP

The Thirdweb React SDK uses React Query under the hood to expose a collection of query and mutation hooks, each with built-in caching, query invalidation, query retries, and more.

Each hook (except for wallet/network management) wraps some functionality of the TypeScript SDK, which are made available as either a query hook to read data, or as a mutation hook to write transactions to the blockchain.

When mutations are called (when a user executes a transaction), query invalidation is automatically triggered to update the relevant queries that depend on the data that was changed. For example, when minting a new NFT, queries that view information about NFTs are re-fetched to load the new NFT automatically.

Queries

All query hooks are used to read data from the blockchain, smart contracts, a user's wallet, etc.

Each one comes with some helpful properties to create a powerful user experience:

data - The data returned from the query (e.g. the NFTs of a contract).

isLoading - Whether the query is currently loading.

error - The error returned from the query, if any.

5.1. Deploying Contracts

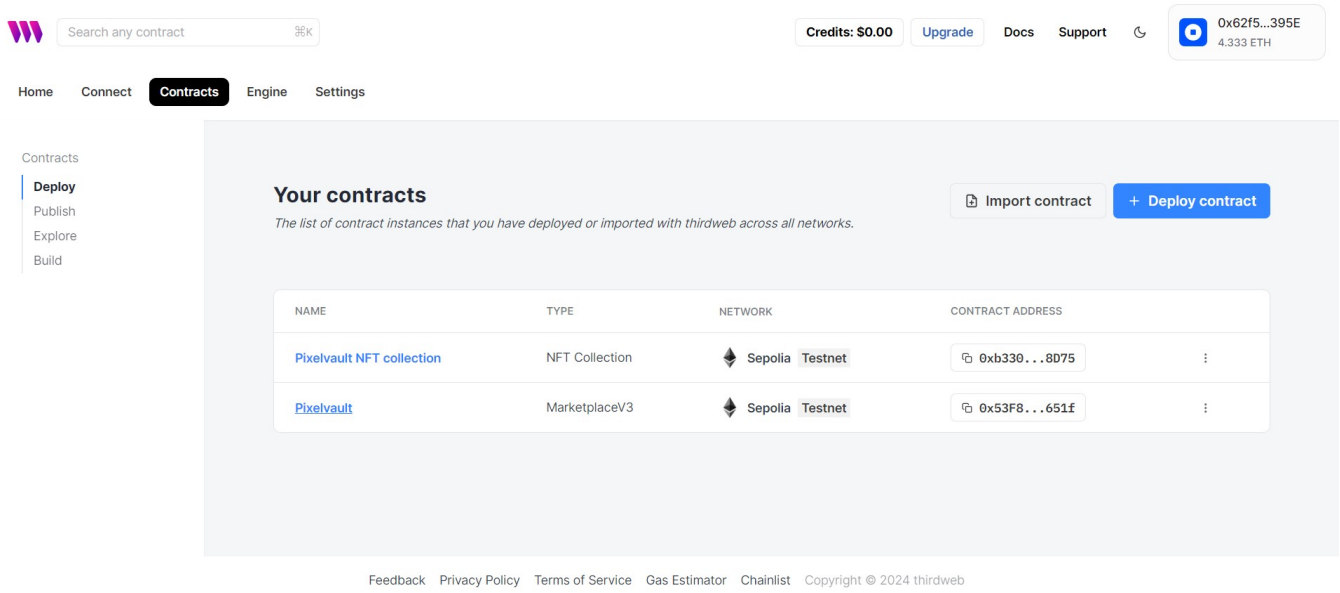

Figure 5.1: Deploying Contracts

5.2. Minting NFT

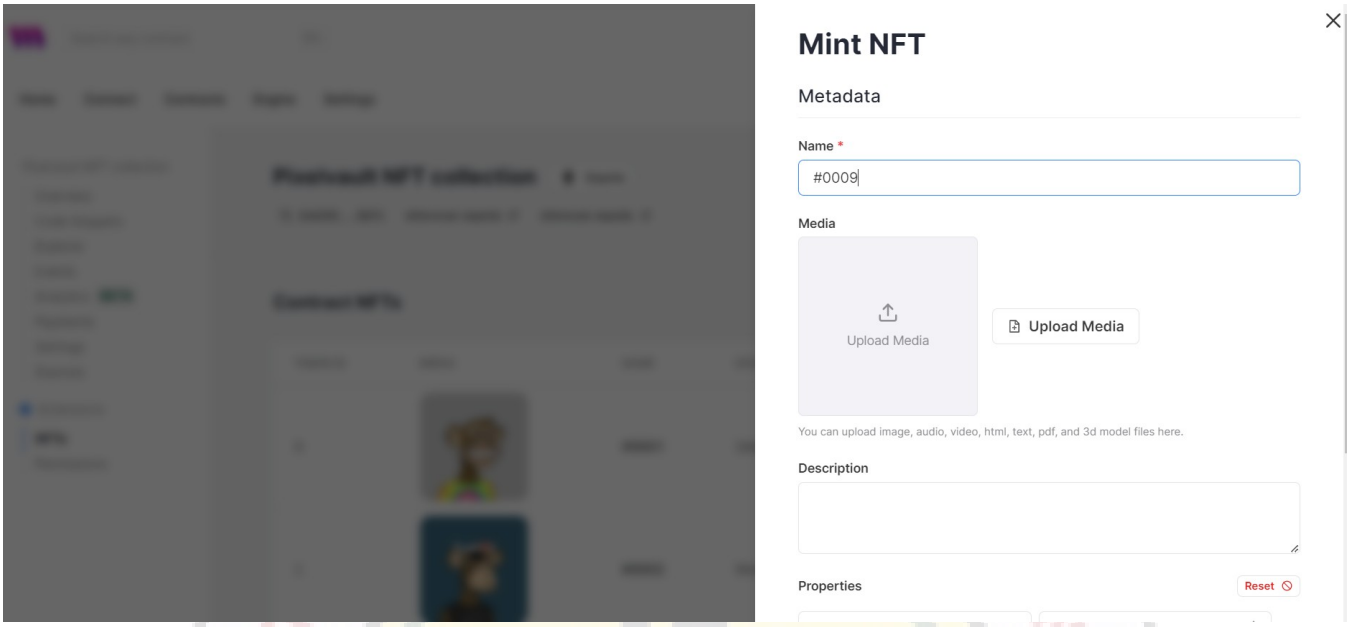

Figure 5.2: Minting NFT directaly on Thirdweb

5.3. NFT Detail page on Thirdweb

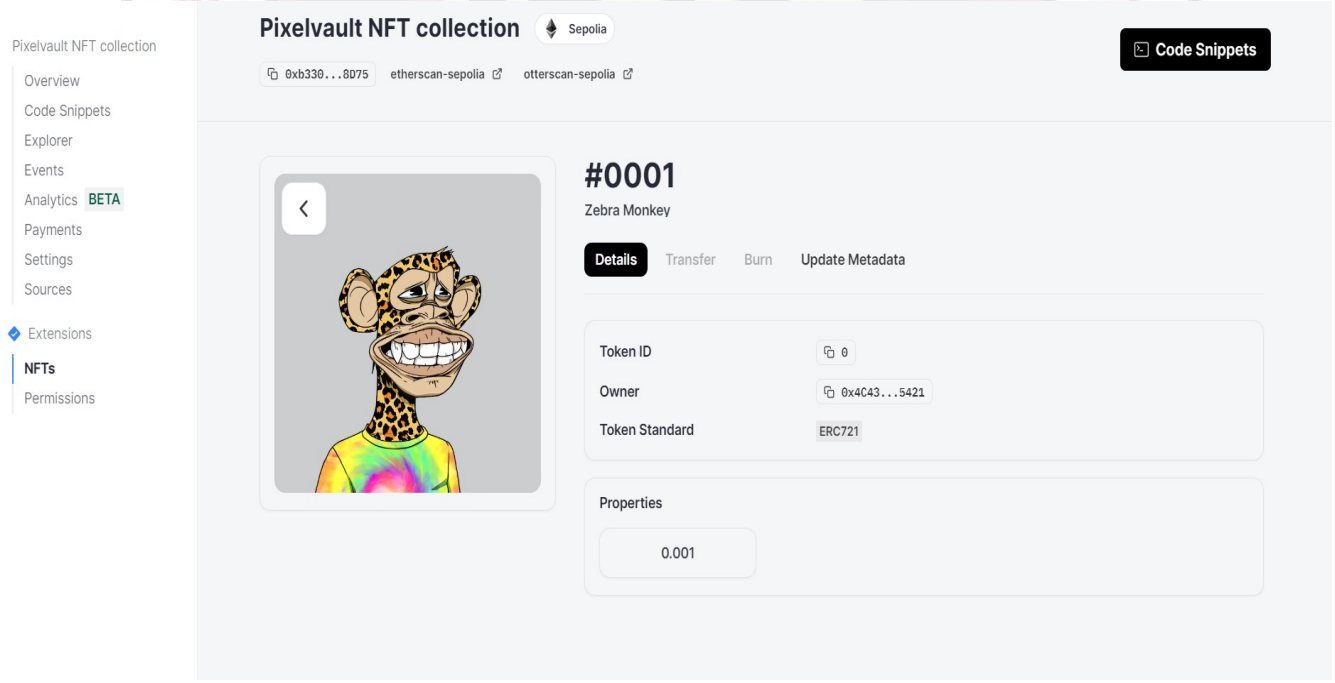

Figure 5.3: NFT Detail page on Thirdweb

5.4. Burn NFT on Thirdweb

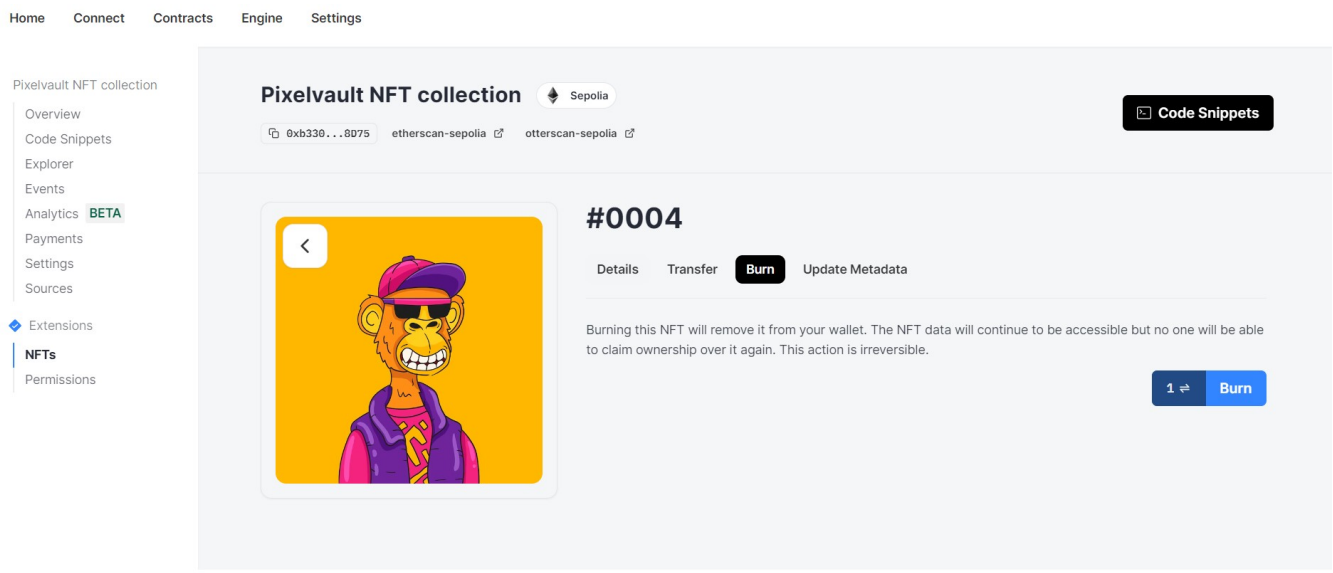

Figure 5.4: Burn NFT on Thirdweb

5.5. Transfer ownership of NFT on Thirdweb

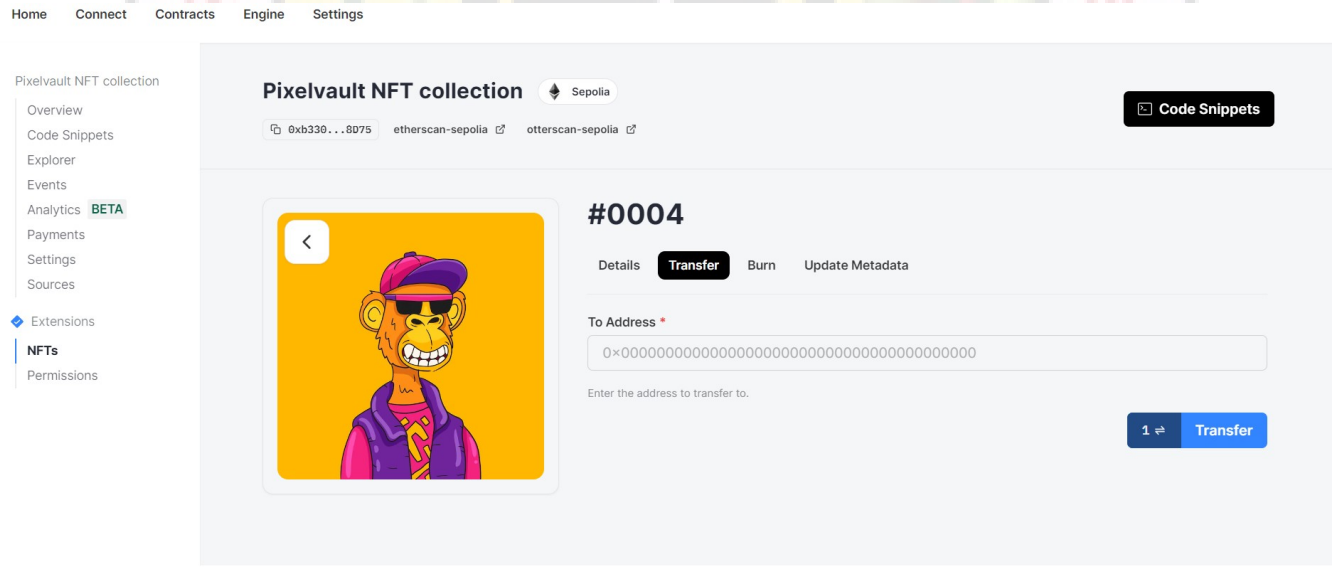

Figure 5.5: Transfer ownership of NFT on Thirdweb

5.6. Direct Listing NFTs on Thirdweb

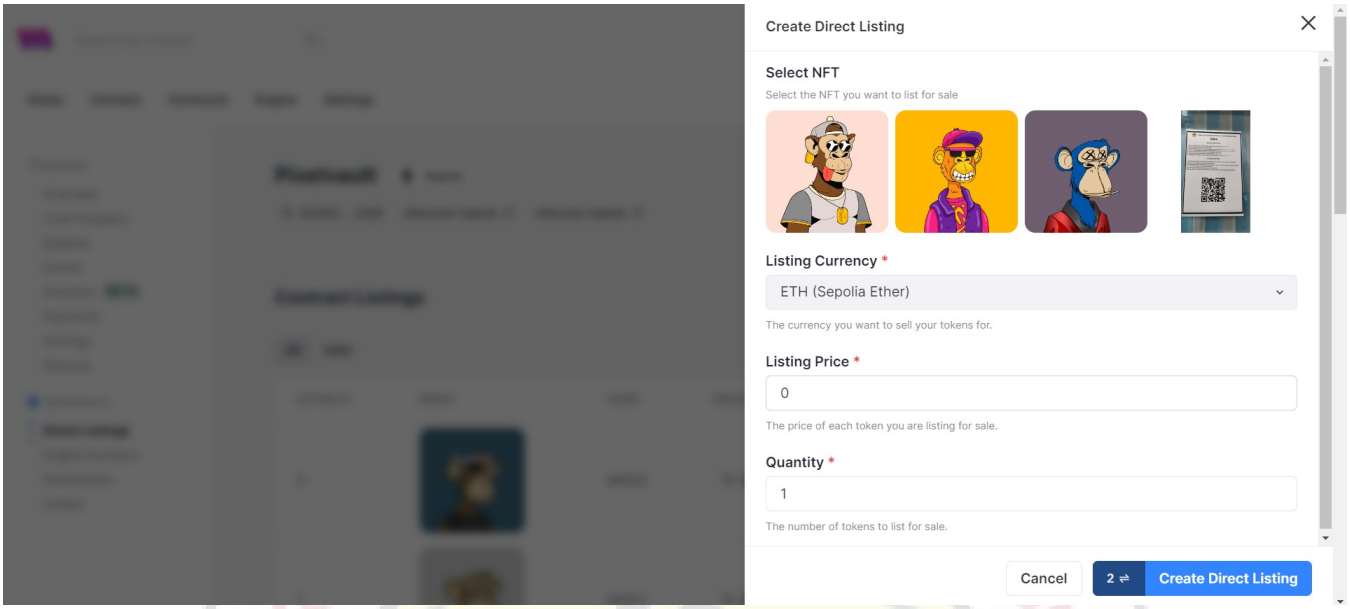

Figure 5.6: Direct Listing NFTs on Thirdweb

CHAPTER – 6
FRONTEND AND ITS INTEGRATION WITH BACKEND

6. FRONTEND AND ITS INTEGRATION WITH BACKEND

The frontend module of Pixelvault's platform embodies a harmonious blend of creativity, functionality, and responsiveness, meticulously crafted to cater to the diverse needs and preferences of its user base. Powered by cutting-edge technologies and design principles, the frontend module serves as the visual manifestation of Pixelvault's brand identity, showcasing its products, services, and value proposition in an intuitive and compelling manner.

6.1. Landing Page

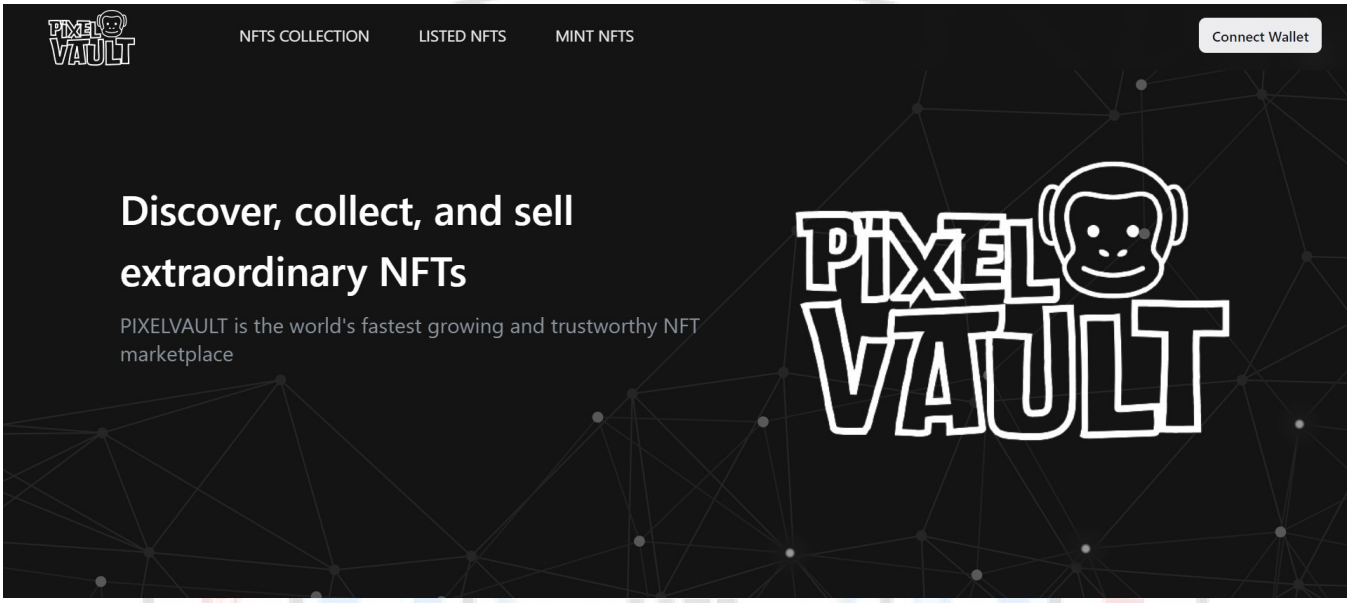

Figure 6.1: Landing Page

6.2. Login Page

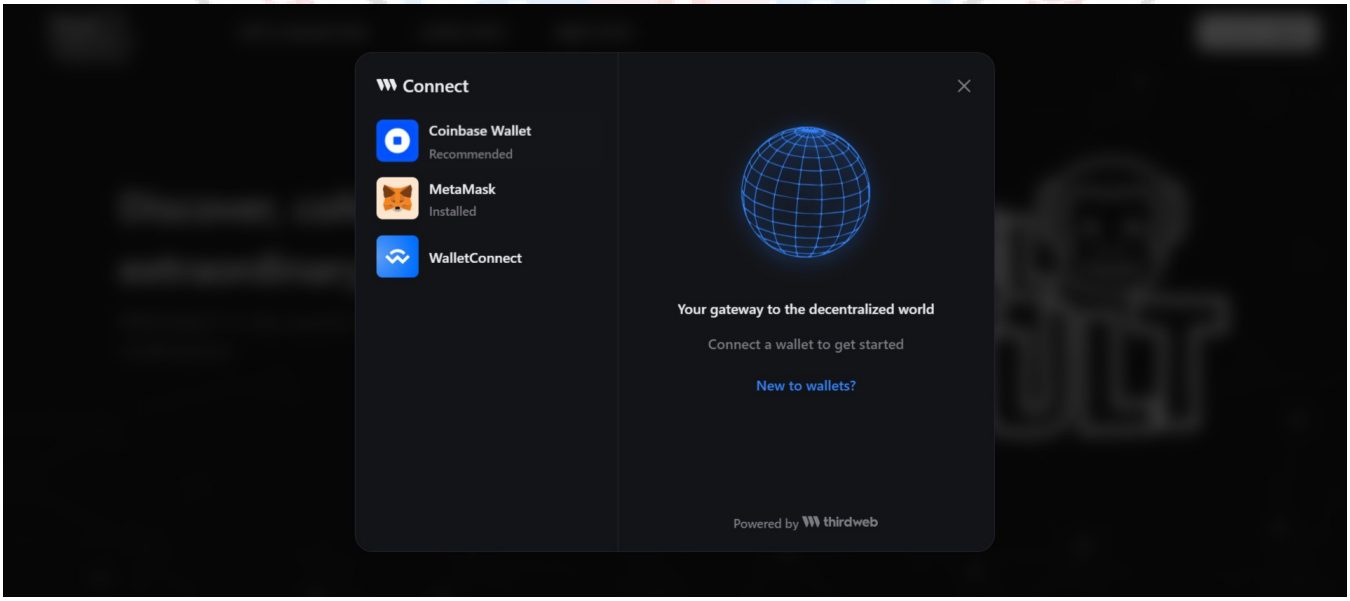

Figure 6.2: Login Page

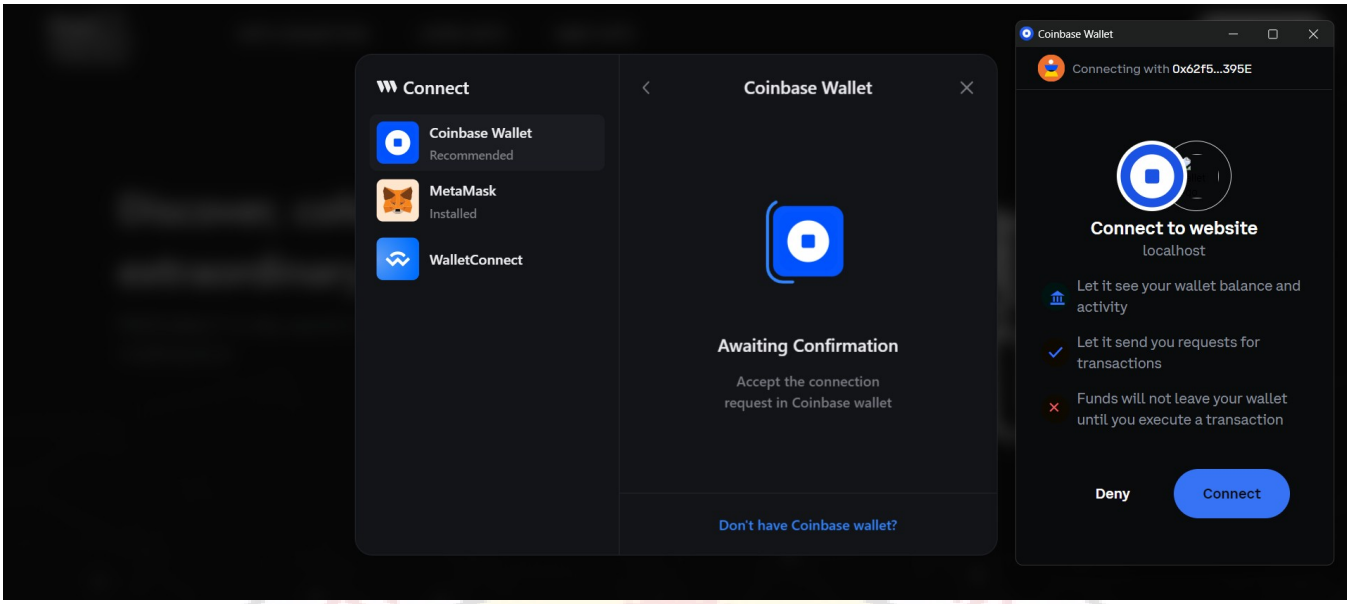

Figure 6.3: Login using Wallet

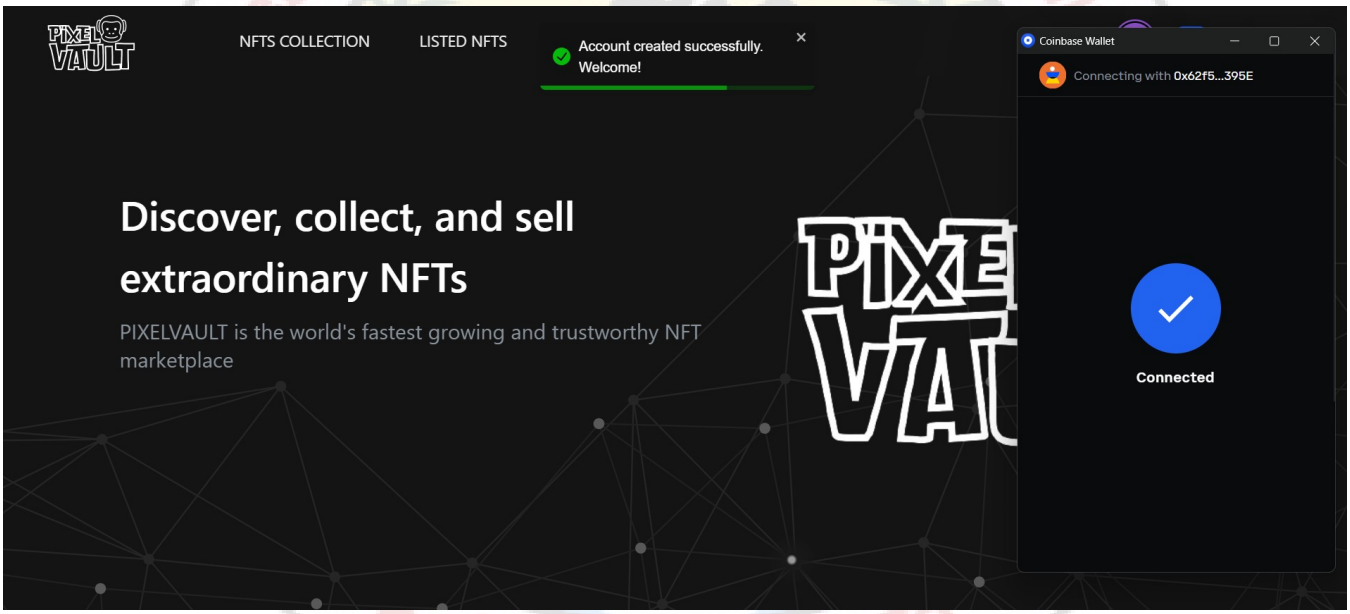

Figure 6.4: Creation of User Account using Wallet address

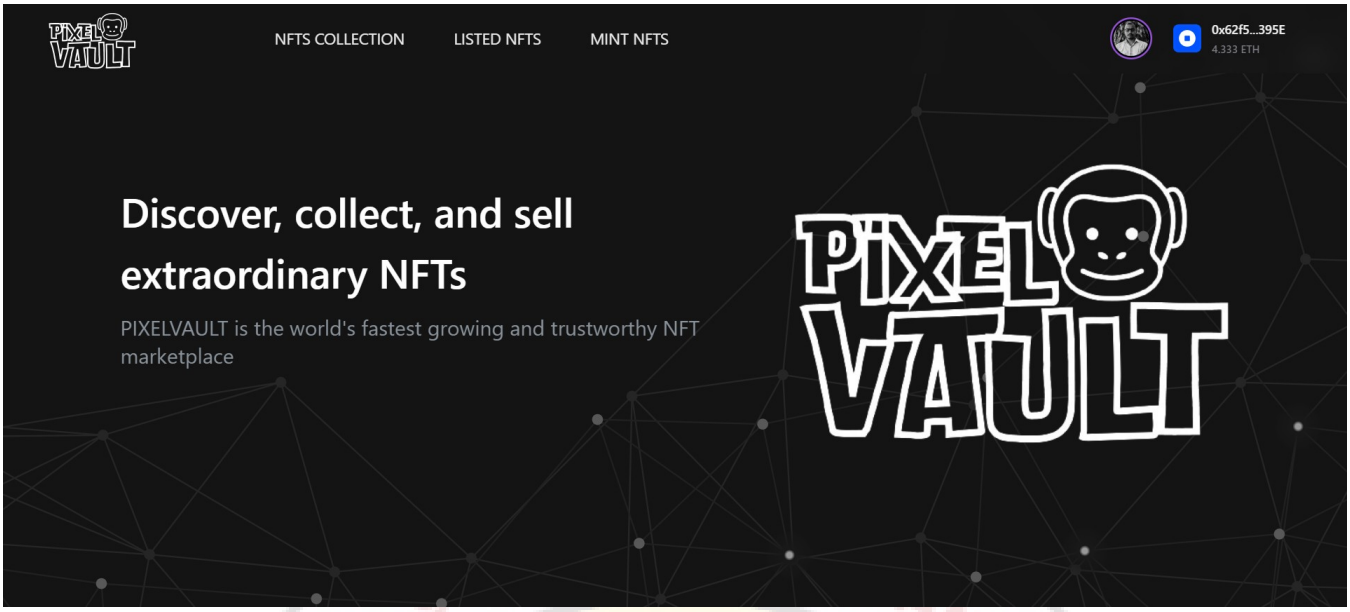

Figure 6.5: Home Page after user logged in

6.3. User Profile Page

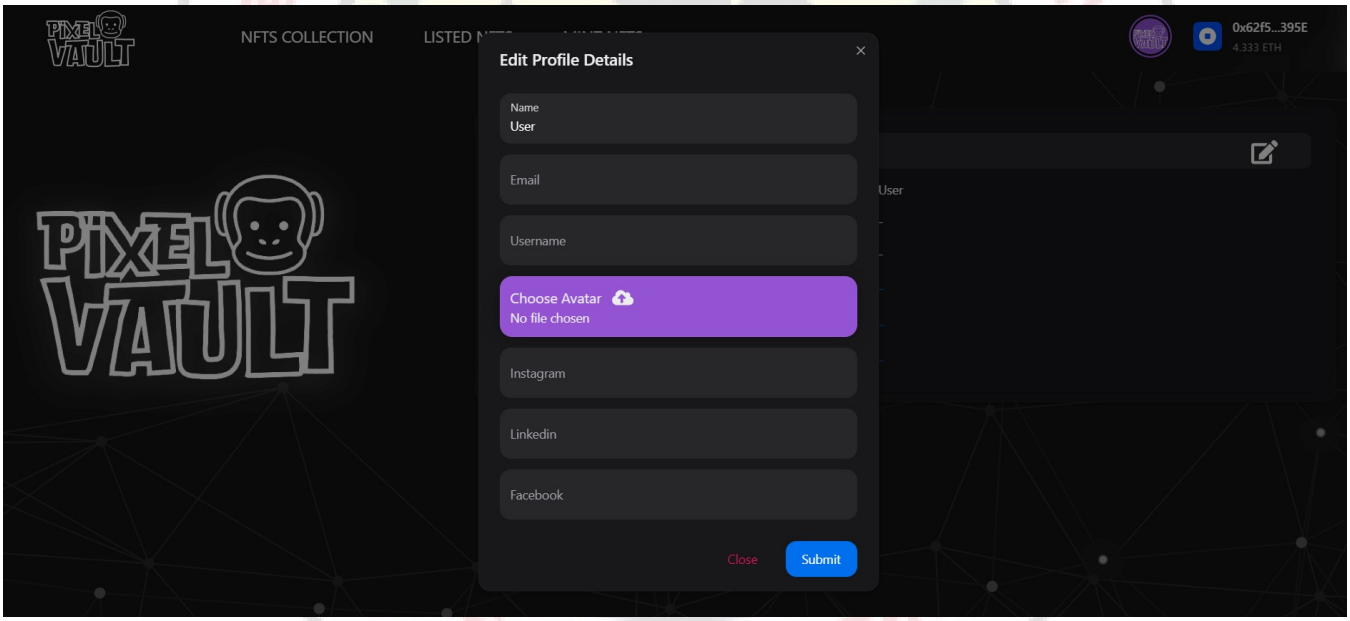

Figure 6.6: Updating User profile details

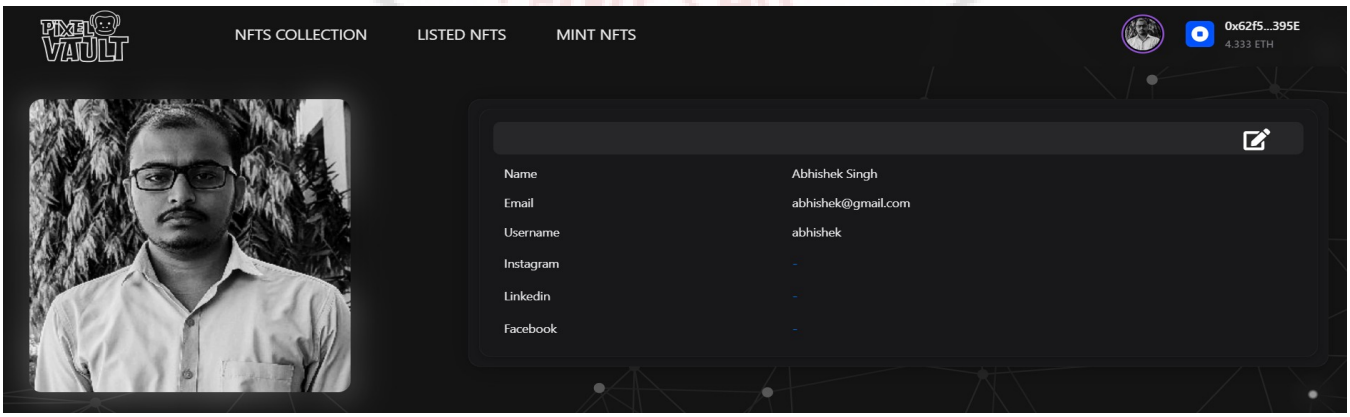

Figure 6.7: User Profile

6.4. User NFTs Page

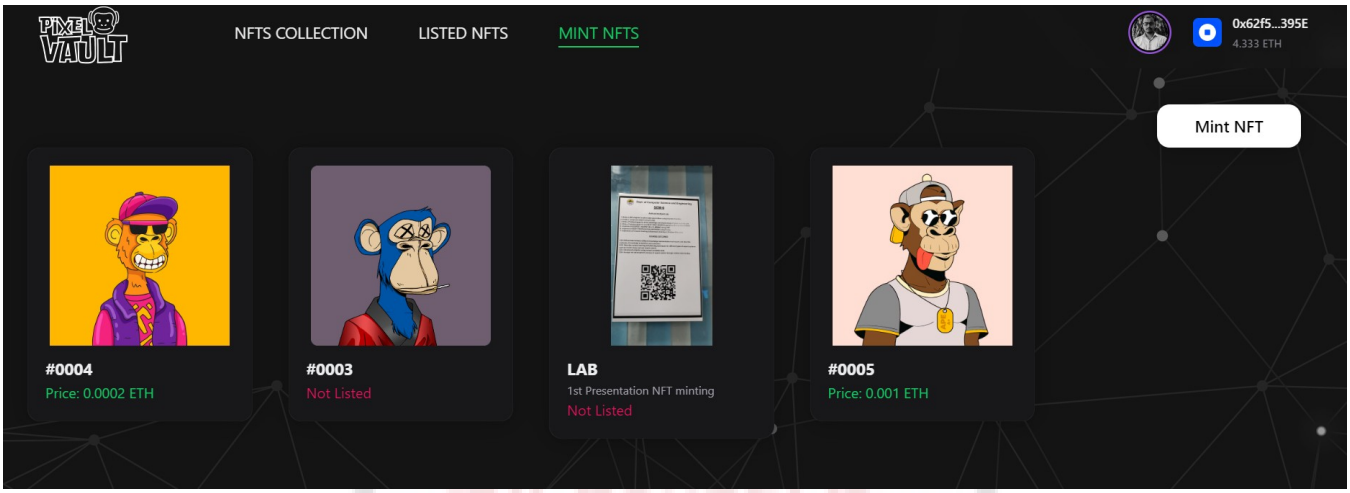

Figure 6.8: User NFT Page

6.5. NFTs Collection Page

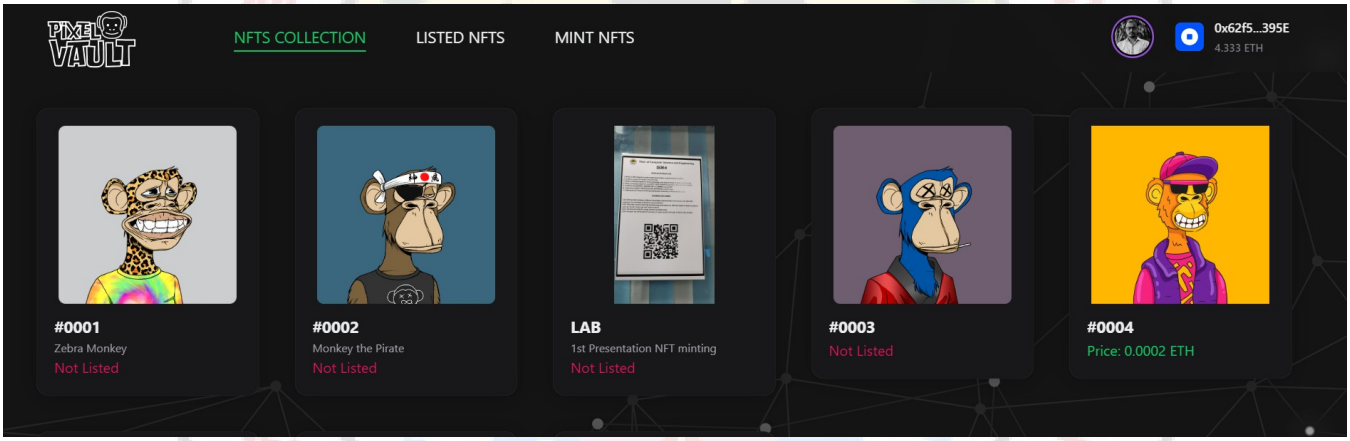

Figure 6.9: NFTs Collection Page

6.6. Listed NFTs Page

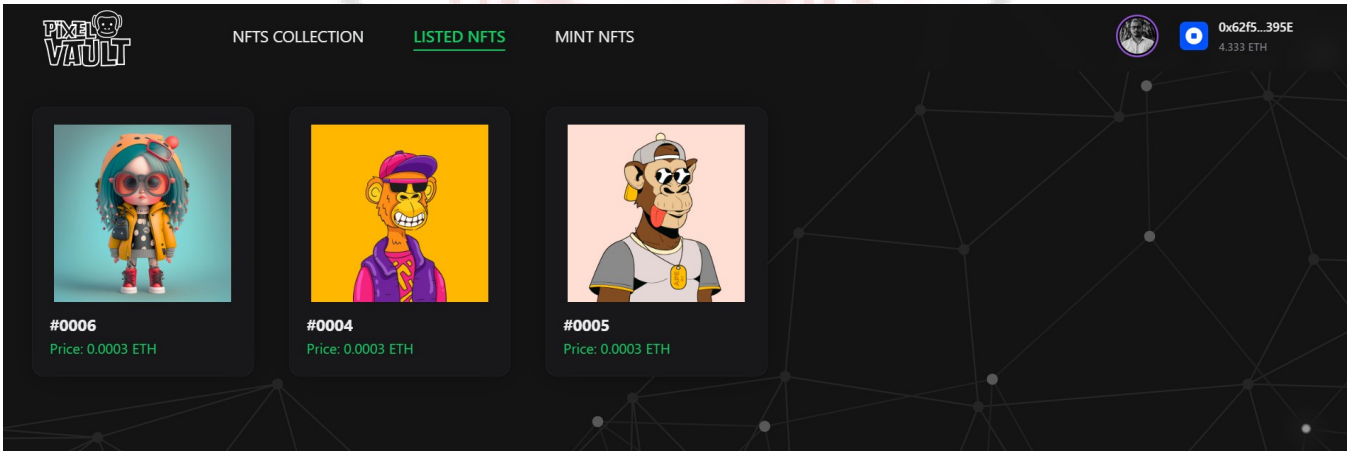

Figure 6.10: Listed NFTs Page

6.7. Mint NFTs Page

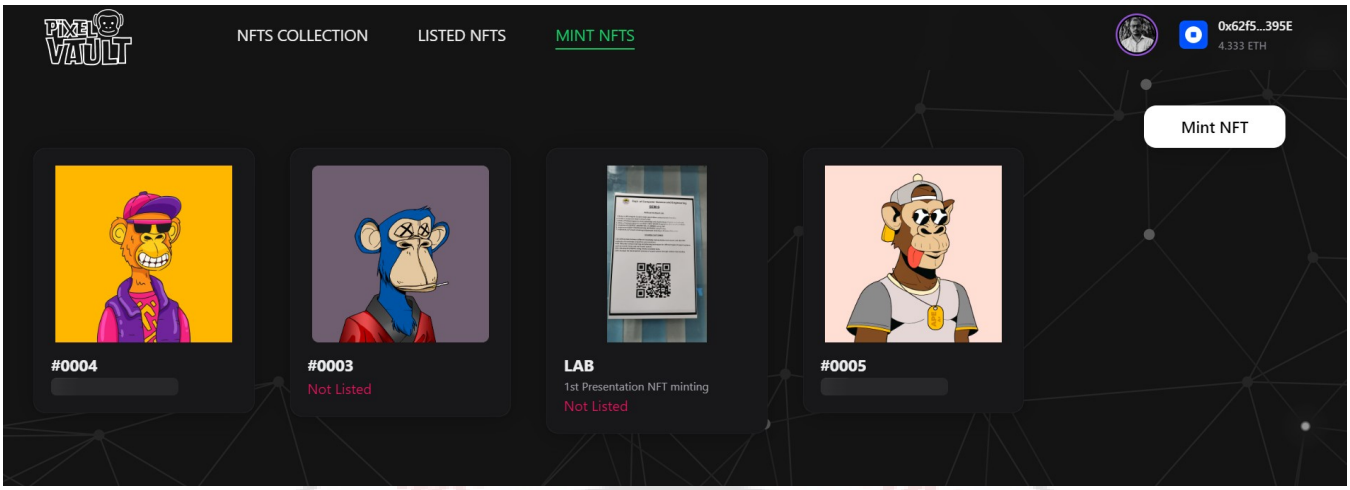

Figure 6.11: Mint NFTs Page

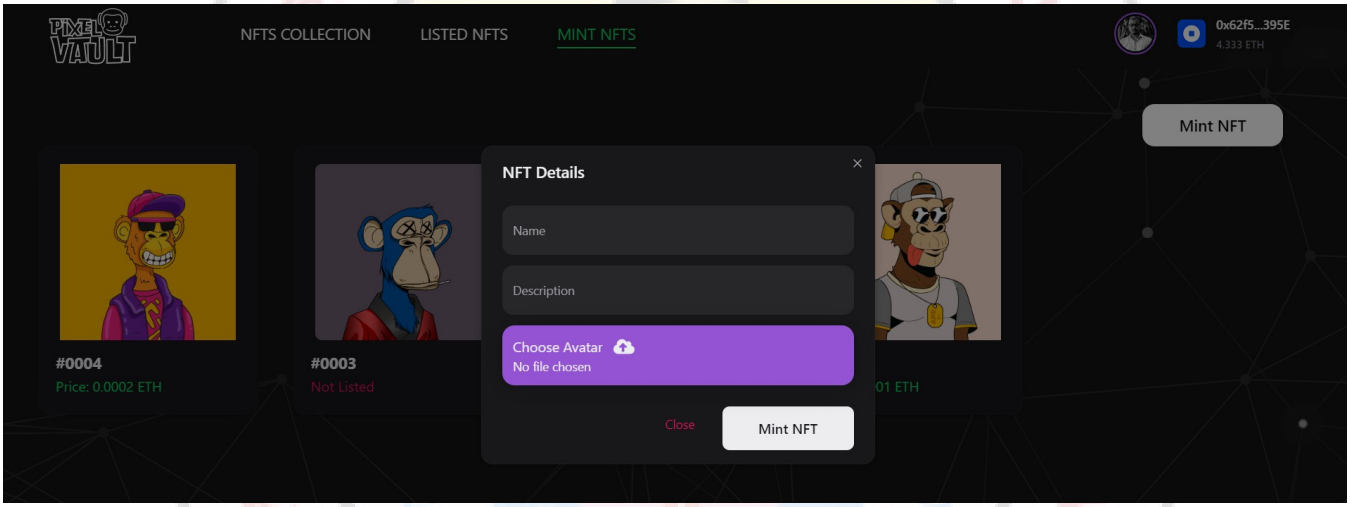

Figure 6.12: Mint NFT form page

6.8. NFT Detail Page

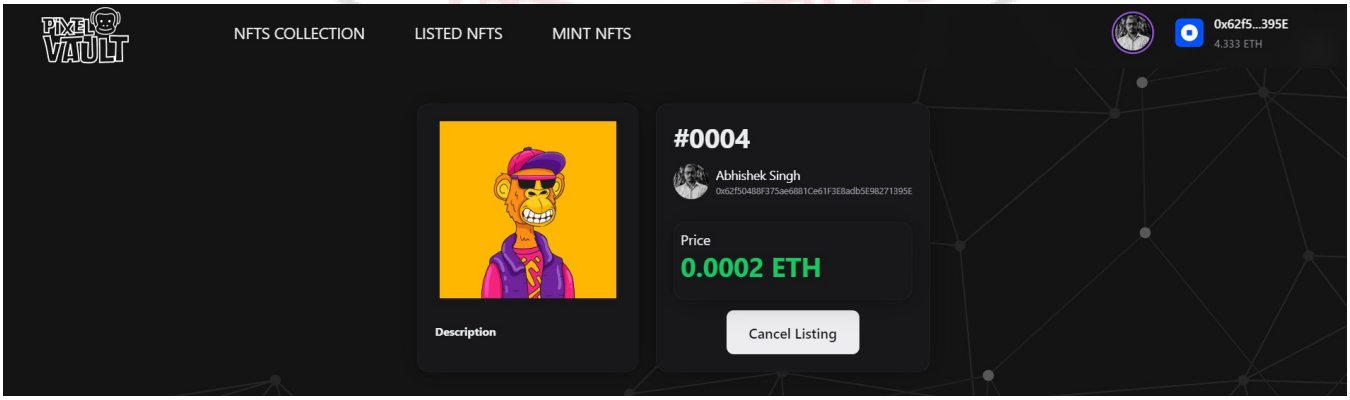

Figure 6.13: NFT Detail Page

6.9. Cancel Listing Page

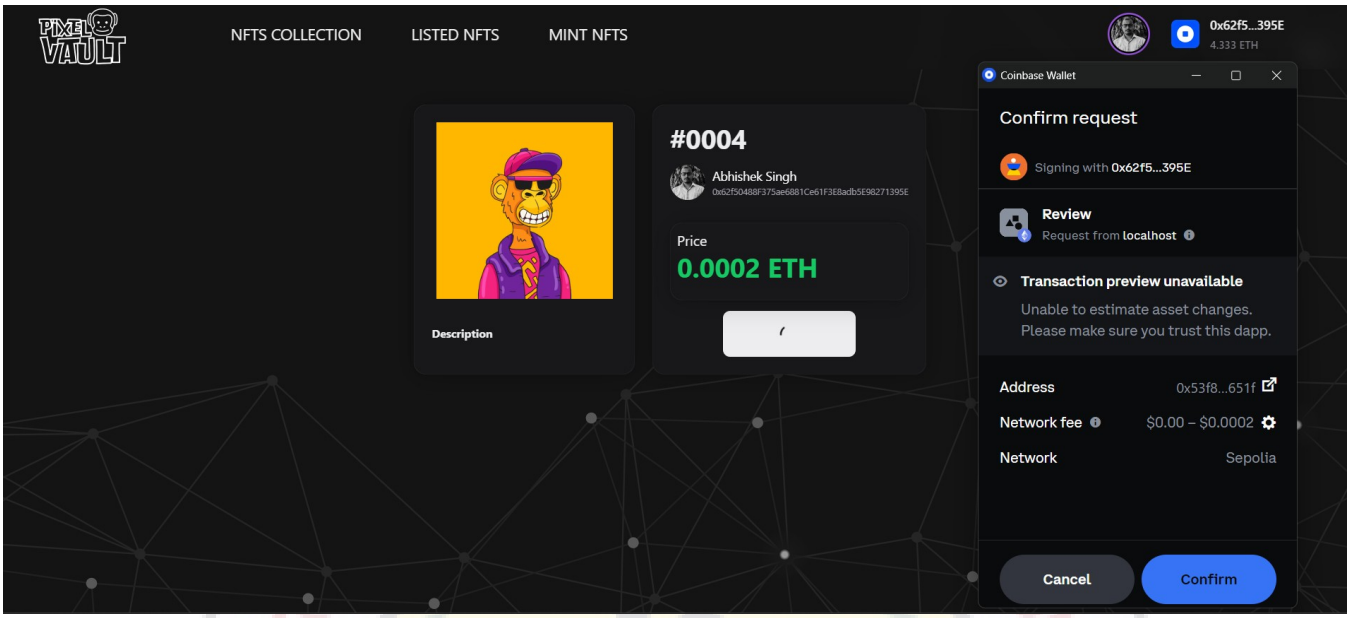

Figure 6.14: Cancelling Listing

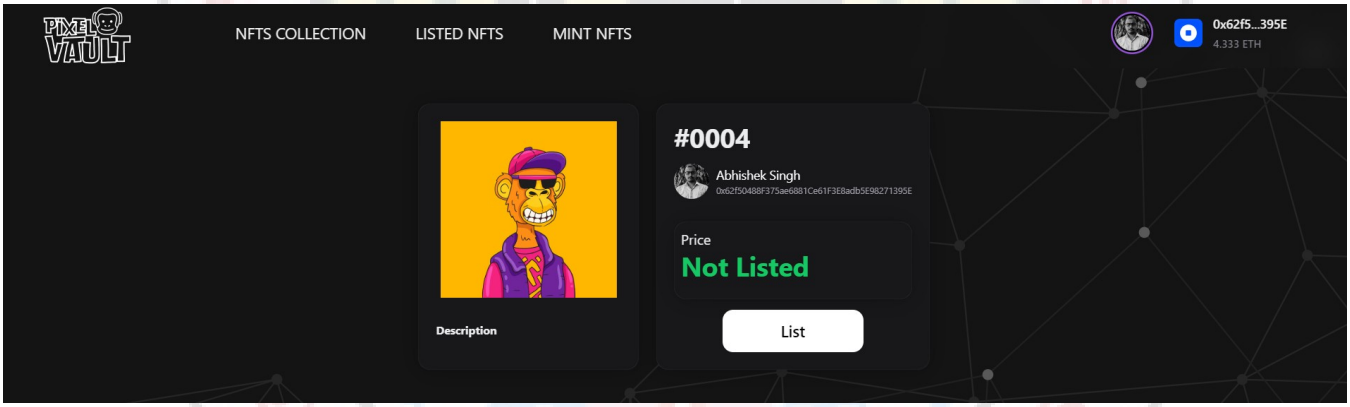

Figure 6.15: Listing Cancelled

6.10. Listing Page

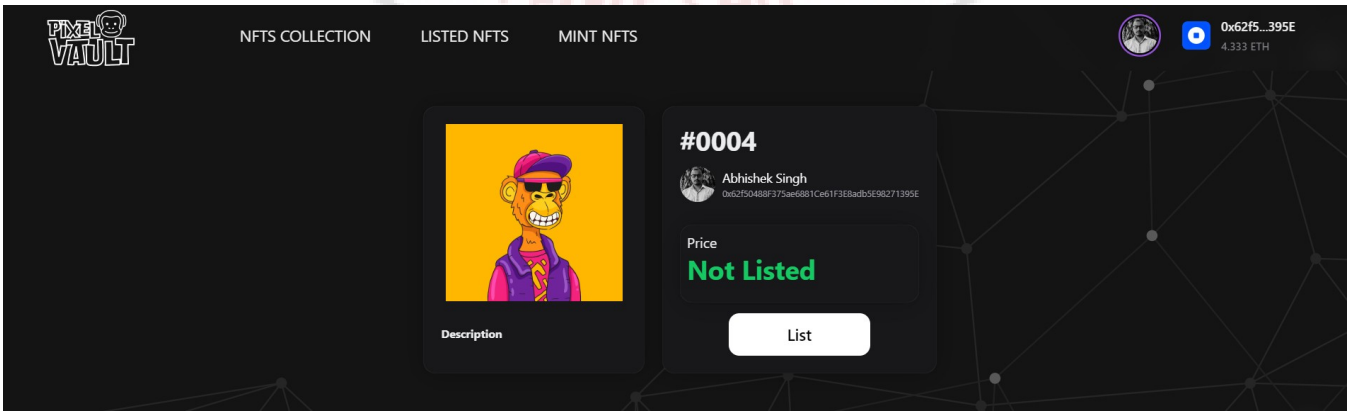

Figure 6.16: Listing Page

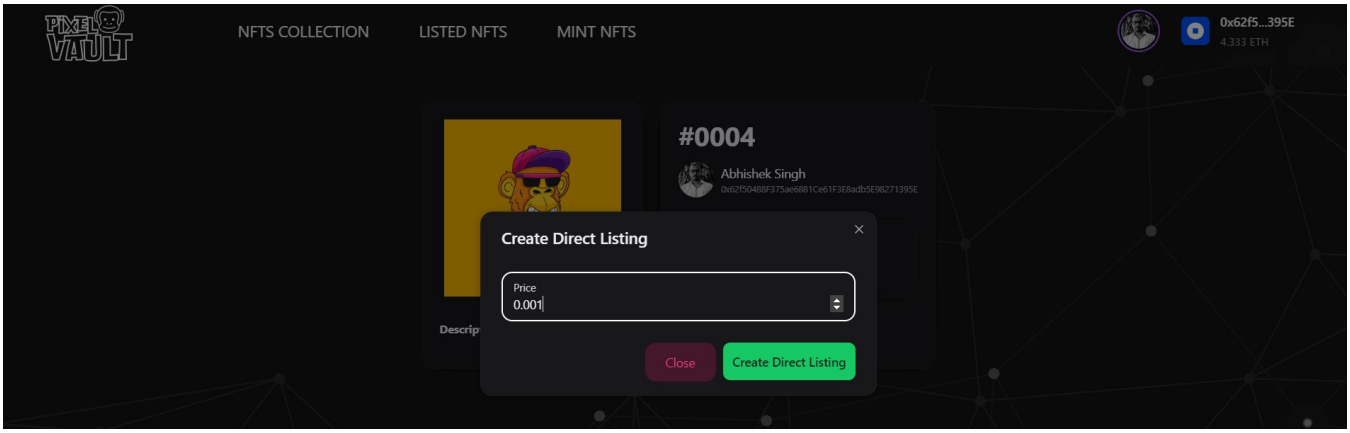

Figure 6.17: Listing Price form

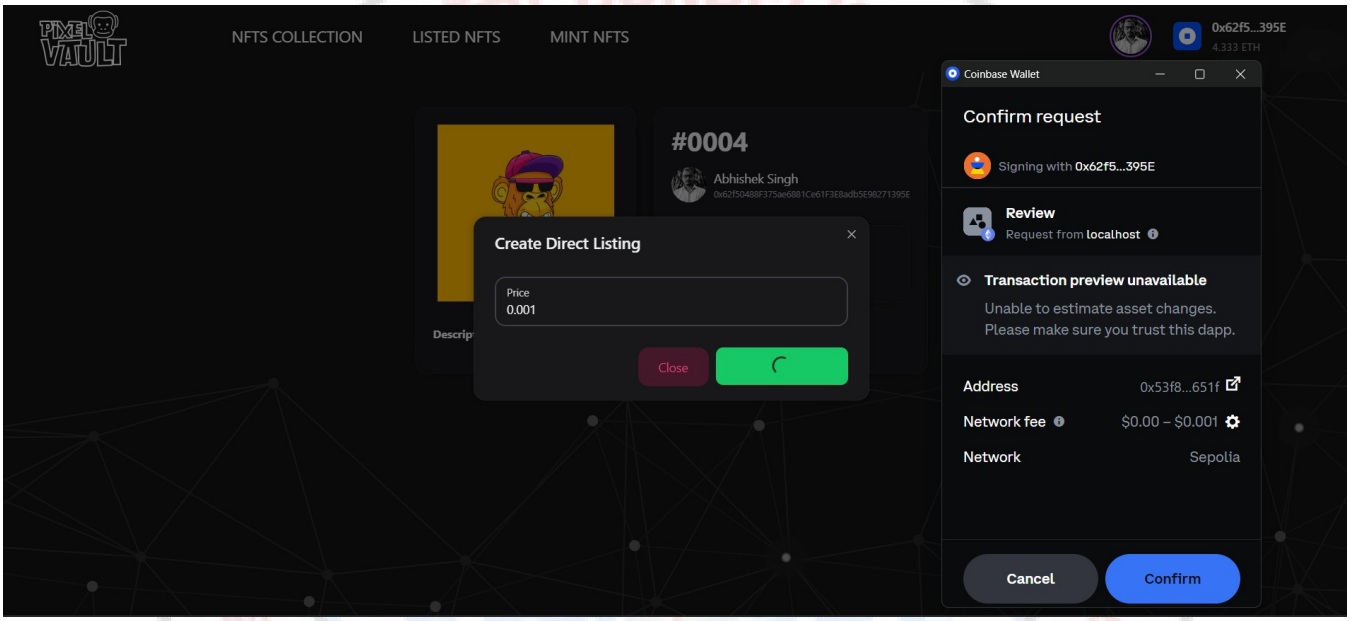

Figure 6.18: Listing NFT to marketplace

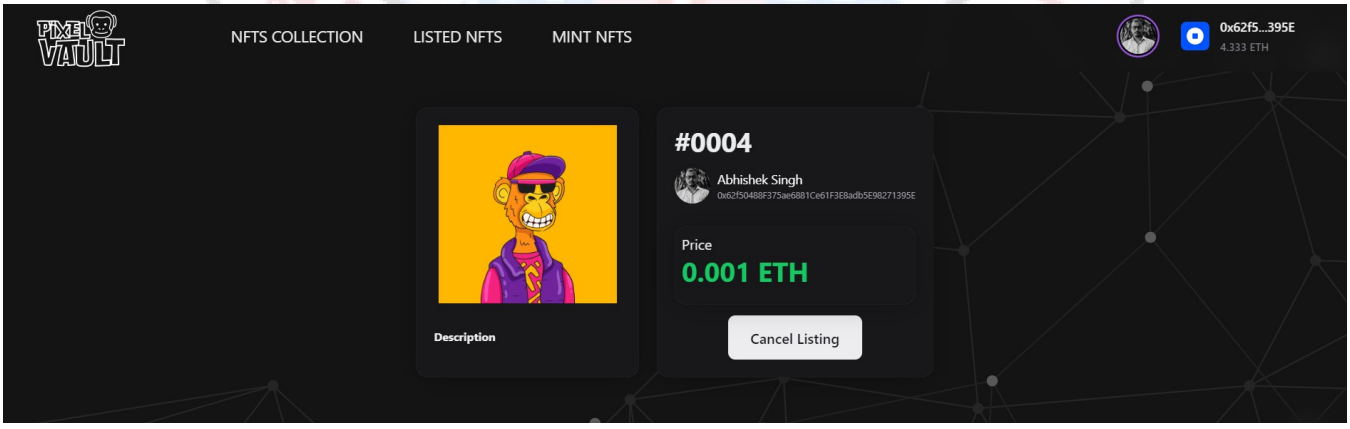

Figure 6.19: NFT Listed to marketplace

6.11. Buy NFT Page

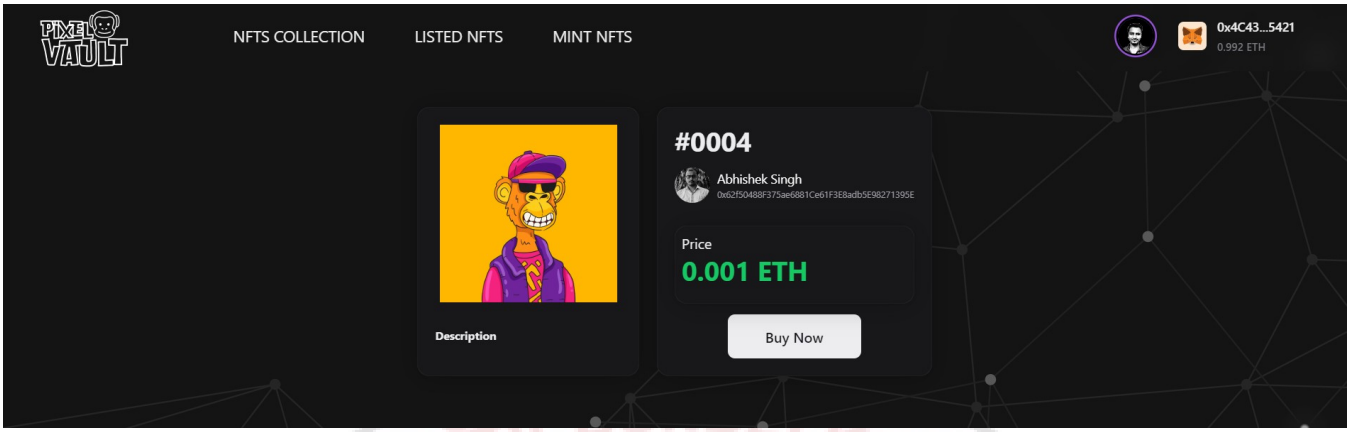

Figure 6.20: Buy NFT Page

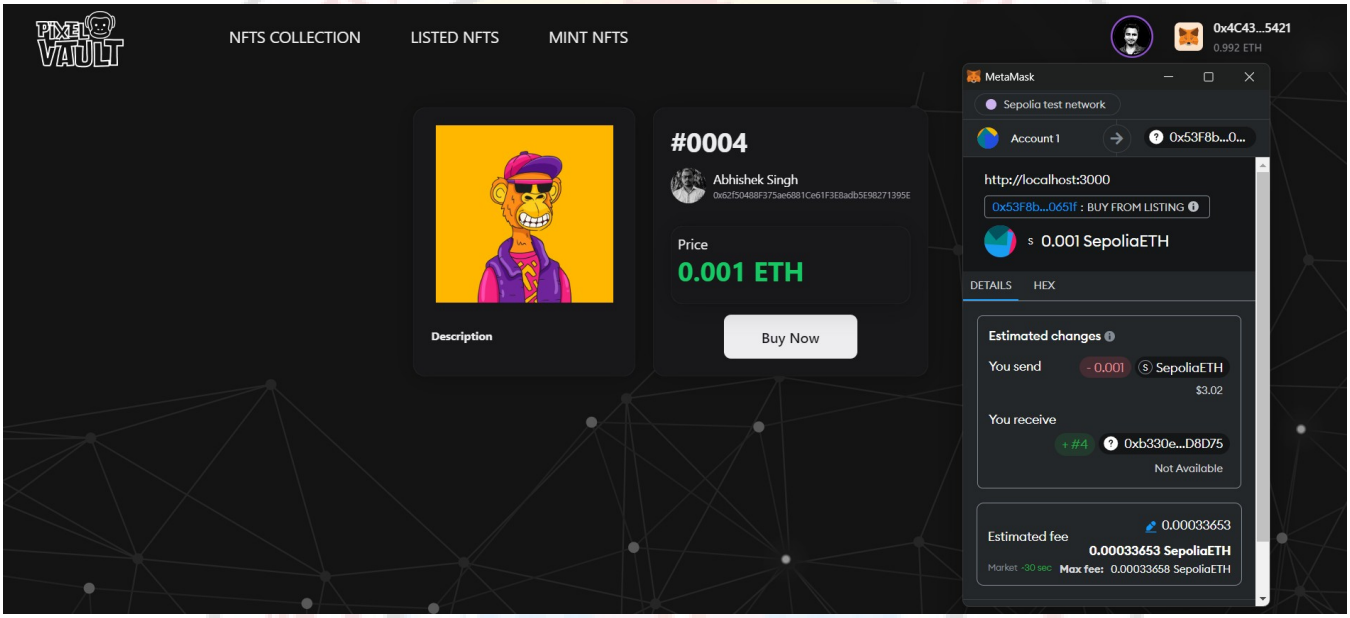

Figure 6.21: Buying NFT

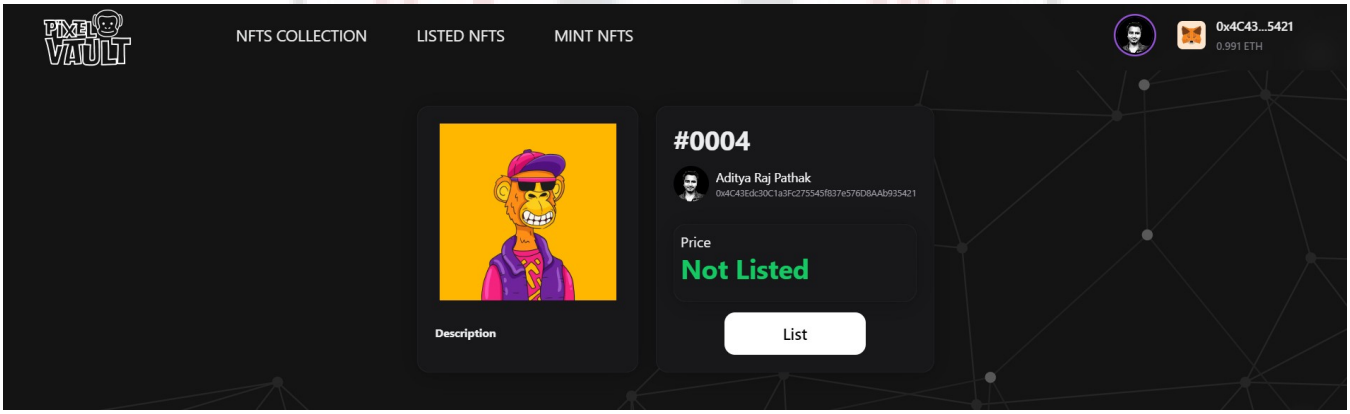

Figure 6.22: NFT Purchased

6.12. Logout Page

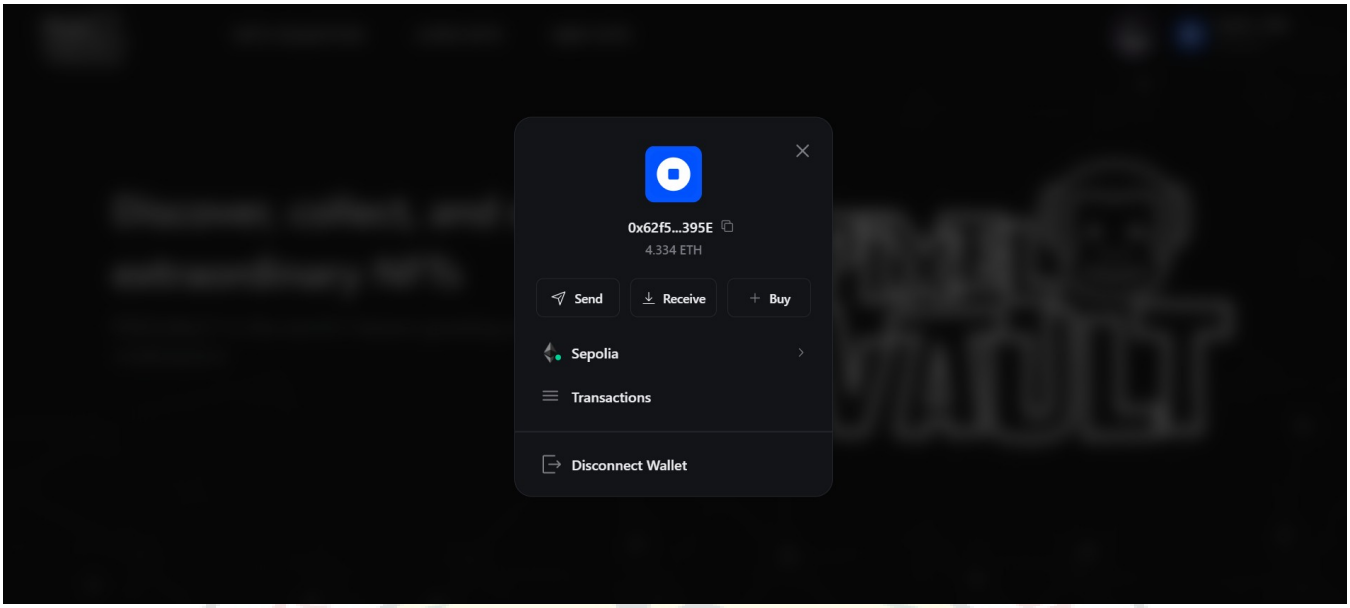

Figure 6.23: Logout Page

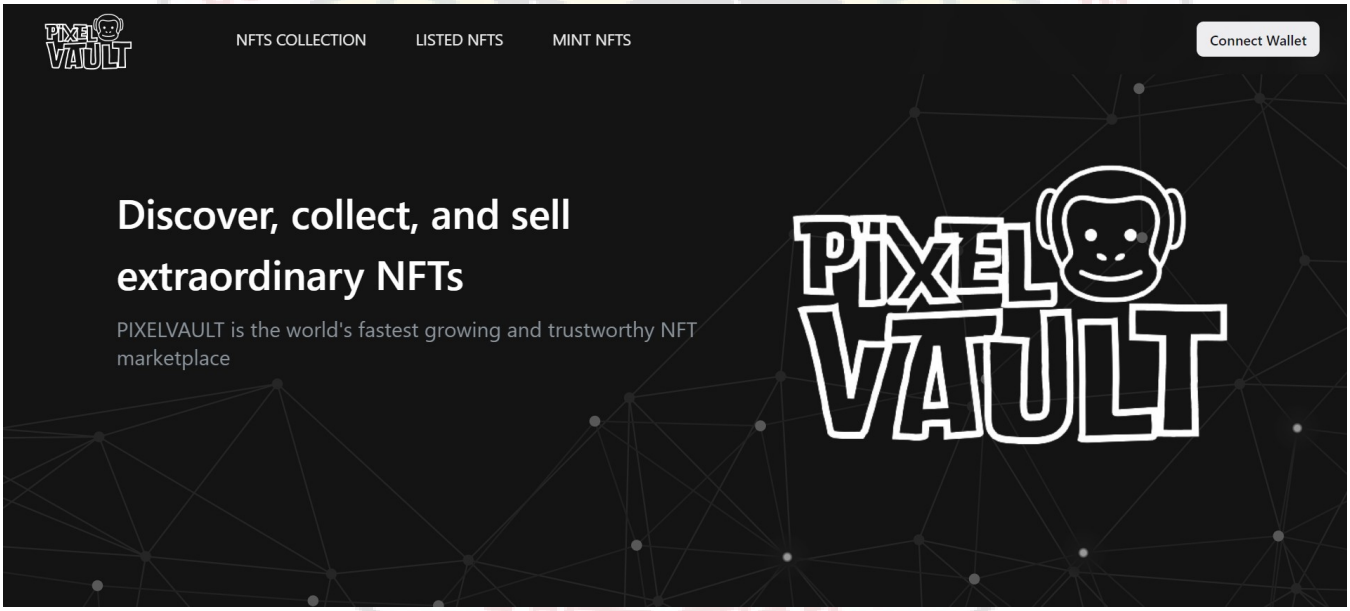

Figure 6.24: User Loged Out

CHAPTER – 7
TESTING AND DOCUMENTATION

7. TESTING

Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do. The benefits of testing include preventing bugs, reducing development costs and improving performance.

7.1. ALPHA TESTING

Alpha Testing is a type of software testing performed to identify bugs before releasing the product to real users or to the public. Alpha Testing is one of the User Acceptance Testing. This is referred to as alpha testing only because it is done early on, near the end of the development of the software.

7.1.1. Objective of Alpha Testing

- The objective of alpha testing is to check whether the location is shown or not.
- The objective of alpha testing is to check whether the form is auto-generated or not.
- The objective of alpha testing is to check if it reads the written text or not.

7.1.2. Alpha Testing Process

- Review the design specification and functional requirements.
- Develop comprehensive test cases and test plans.
- Execute test plan
- Log defects
- Retest once the issues have been fixed

7.2. BETA TESTING

Beta Testing is performed by real users of the software application in a real environment. Beta testing is one of the types of User Acceptance Testing. A Beta version of the software, whose feedback is needed, is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing helps in minimization of product failure risks and it provides increased quality of the product through customer validation. It is the last test before shipping a product to the customers. One of the major advantages of beta testing is direct feedback from customers.

7.2.1. Characteristics of Beta Testing

- Beta Testing is performed by clients or users who are not employees of the company.
- Reliability, security, and robustness are checked during beta testing.
- Beta Testing commonly uses black-box testing.
- Beta testing is carried out in the user's location.
- Beta testing doesn't require a lab or testing environment.

7.2.2. Criteria for Beta Testing

- The Beta version of the software should be ready.
- Environment ready to release the software application to the public.
- Tool to capture real-time faults.

CHAPTER – 8
DEPLOYMENT AND VERSION CONTROLLING

8. DEPLOYMENT

Deployment in software and web development means pushing changes or updates from one deployment environment to another. When setting up a website you will always have your live website, which is called the live environment or production environment. If you want the ability to make changes without these affecting your live website, then you can add additional environments. These environments are called development environments or deployment environments. The additional development environments will typically be a local environment, a development environment, and a staging environment(also known as a staging site).

8.1. VERSION CONTROLLING USING GIT AND GITHUB

Version control is an essential aspect of software development, enabling teams to manage and track changes to their codebase over time. Git, a distributed version control system, has become the industry standard for source code management. GitHub, a web-based platform built around Git, enhances collaboration, code review, and project management. Together, Git and GitHub provide a powerful solution for version control in software development.

8.1.1. Git and Github

Git allows developers to track changes, create branches for parallel development, and merge changes seamlessly. Each commit in Git represents a snapshot of the project at a specific point in time. This decentralized nature of Git enables developers to work independently on different aspects of a project and later integrate their changes, minimizing conflicts and streamlining collaboration.

GitHub, as a hosting service for Git repositories, adds a layer of collaboration and project management on top of Git. Developers can push their local Git repositories to GitHub, making it easier to share code with others. GitHub provides features like pull requests, issues, and project boards, facilitating effective collaboration among team members. The pull request workflow allows developers to propose changes, conduct code reviews, and merge code into the main branch once approved.

Branching is a fundamental concept in both Git and GitHub. Developers can create branches to work on specific features or bug fixes without affecting the main codebase. Branches can be easily merged, allowing for a smooth integration of changes. GitHub enhances this process by providing a visual interface for comparing and merging branches, making it accessible to both technical and non- technical team members.

The version control capabilities of Git and GitHub are crucial for project stability, collaboration, and traceability. Developers can roll back to previous versions, investigate the history of changes, and identify who made specific modifications. This traceability is essential for debugging, auditing, and understanding the evolution of the codebase over time.

Figure 8.1: Git and Github

8.2. DEPLOYMENT USING VERCEL

Vercel is a cloud platform designed to facilitate seamless deployment and hosting of web applications. Its simplicity and integration capabilities make it a popular choice for developers seeking an efficient deployment solution. The deployment process using Vercel involves a few straightforward steps, providing an end-to-end solution for shipping web applications.

To deploy a web application with Vercel, developers typically start by linking their version control repository (e.g., GitHub, GitLab, or Bitbucket) to a Vercel project. Once connected, Vercel automatically detects the project settings and suggests a deployment configuration. Developers can

customize deployment settings, such as environment variables and build commands, to suit the specific needs of their application.

Vercel supports various frameworks and languages, enabling developers to deploy a wide range of projects, from static websites to dynamic web applications. The platform automatically builds and optimizes the application during deployment, ensuring efficient performance and minimizing load times for end-users.

One notable feature of Vercel is its support for serverless functions, allowing developers to deploy serverless APIs alongside their web applications seamlessly. This serverless architecture simplifies scalability and reduces infrastructure management overhead.

Vercel also provides continuous deployment, automatically deploying changes to the production environment whenever new code is pushed to the linked repository. This continuous integration and deployment (CI/CD) approach ensures that the latest version of the application is always available to users.

Additionally, Vercel offers deployment previews, allowing developers to create temporary, shareable URLs for reviewing changes before merging them into the main branch. This feature facilitates collaborative development and thorough testing of new features in a production-like environment.

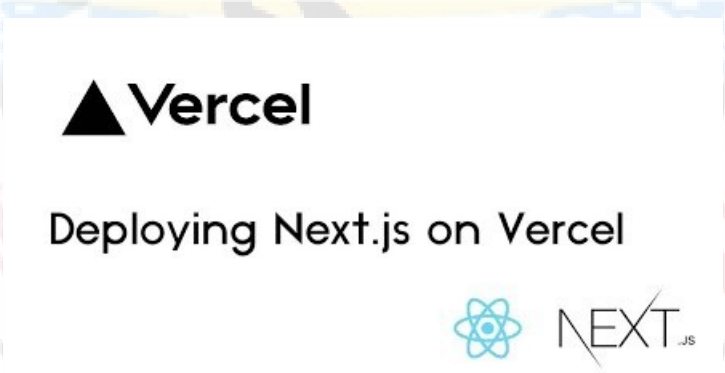

Figure 8.2: Vercel Deployment

CHAPTER – 9
CONCLUSION

9. CONCLUSION

In conclusion, "Pixel Vault" stands as a pioneering venture aimed at reshaping the digital landscape by introducing a state-of-the-art NFT marketplace. With a steadfast focus on technical innovation, user-centric design, authenticity assurance, educational outreach, and rigorous research and analysis, our project endeavors to redefine the very essence of digital asset ownership and trading.

Harnessing the power of cutting-edge technologies, "Pixel Vault" has leveraged a variety of tools and platforms to bring its vision to life. Utilizing a sophisticated content management system for seamless NFT uploads, seamless integration with third-party web services, including the trusted Coinbase cryptocurrency wallet for secure payments, and the versatile Next.js framework for frontend development, "Pixel Vault" showcases a fully functional NFT marketplace where users can effortlessly mint, sell, and purchase NFTs with confidence and ease.

Yet, amidst our accomplishments, "Pixel Vault" has also illuminated areas for growth and refinement. Challenges such as the current inability to conduct auctions and the need for optimization in the minting process have emerged as opportunities for future development and enhancement.

Looking forward, "Pixel Vault" is poised for further evolution and expansion. Plans for the introduction of auction functionality and streamlining the minting process underscore our commitment to continuous improvement. Moreover, our project suggests exciting avenues for future research, including enhancing user experience, ensuring scalability, and exploring deeper integration of blockchain technology.

Ultimately, "Pixel Vault" aspires to be more than just a marketplace; it aims to be a catalyst for change in the realm of digital ownership, creative expression, and value exchange. By championing innovation and empowering individuals within the digital art, collectibles, and creative entrepreneurship communities, "Pixel Vault" seeks to leave an indelible mark on the ever-evolving landscape of NFTs, forging a path toward a more inclusive, vibrant, and sustainable digital future.

APPENDIX-A

GLOSSARY

Blockchain Technology: A decentralized, distributed ledger technology that records transactions across multiple computers in a way that makes them tamper-resistant and transparent. Each transaction is verified by network participants and added to a chain of blocks, creating a permanent and immutable record of transactions.

Cryptography: The practice and study of techniques for secure communication and data protection in the presence of adversaries. Cryptography involves creating and analyzing protocols that prevent third parties from accessing or modifying data without proper authorization.

Data Privacy: The protection of sensitive information from unauthorized access, disclosure, alteration, or destruction. Data privacy measures aim to ensure that individuals have control over how their personal data is collected, used, and shared by organizations.

Decentralization: The distribution of authority, control, and decision-making across multiple entities or nodes, rather than relying on a single centralized authority. Decentralization reduces the risk of censorship, single points of failure, and abuse of power.

Ethereum: A decentralized platform that enables the creation and execution of smart contracts and decentralized applications (dApps). Ethereum's native cryptocurrency, Ether (ETH), is used to facilitate transactions and incentivize network participants.

Hash Function: A mathematical function that converts an input (or "message") into a fixed-size string of characters, which is typically a hexadecimal format. Hash functions are commonly used in cryptography for data integrity verification, password hashing, and digital signatures.

IPFS: InterPlanetary File System (IPFS) is a protocol and network designed to create a decentralized method of storing and sharing hypermedia in a distributed file system. IPFS aims to improve data availability and reduce reliance on centralized servers.

Listing: The process of adding an asset, such as a cryptocurrency or token, to an exchange or marketplace, making it available for trading with other assets. Listings are typically subject to approval by the exchange and may involve meeting certain criteria and compliance requirements.

Marketplace: An online platform or venue where buyers and sellers can exchange goods, services, or assets. Marketplaces facilitate transactions between parties and often provide tools and services for product discovery, payment processing, and dispute resolution.

Mining: The process of validating and adding transactions to a blockchain by solving complex mathematical puzzles. Miners compete to find a valid solution to the puzzle and are rewarded with cryptocurrency or other incentives for their efforts.

Mint: The process of creating or issuing new units of a cryptocurrency or token. Minting typically involves generating new coins or tokens according to predefined rules and distributing them to users or network participants.

NFT: Non-Fungible Token (NFT) is a unique digital asset that represents ownership or proof of authenticity of a specific item or piece of content, such as digital art, collectibles, or virtual real estate. Unlike cryptocurrencies, NFTs are not interchangeable and have unique properties.

NFT Schema: The structure or blueprint that defines the properties and attributes of a non-fungible token (NFT), including metadata, ownership information, and other relevant data. NFT schemas help standardize the representation and interoperability of NFTs across different platforms and ecosystems.

Peer-to-peer: A decentralized communication model where participants interact directly with each other, without the need for intermediaries or centralized servers. Peer-to-peer networks enable distributed sharing of resources, information, and services among network participants.

Smart Contract: Self-executing contracts with the terms of the agreement between buyer and seller being directly written into code. Smart contracts automatically enforce and execute the terms of the contract when predefined conditions are met, without the need for intermediaries.

Third Web: A term used to describe the evolution of the internet towards a more decentralized, peer-to-peer architecture, enabled by technologies such as blockchain, decentralized storage, and distributed computing. The Third Web aims to provide greater control, privacy, and autonomy to users.

Transaction: An exchange of value or information between parties, recorded and verified on a blockchain or distributed ledger. Transactions involve transferring ownership of assets, executing smart contracts, or updating the state of the blockchain in some way.

Trust less System: A system or protocol that operates without the need for trust between participants. Trust less systems leverage cryptographic techniques and decentralized architectures to ensure that transactions and interactions are secure and verifiable without relying on trusted intermediaries.

Verification: The process of confirming the accuracy, authenticity, or validity of data, transactions, or identities. Verification mechanisms are used to ensure that information is true and reliable, often involving cryptographic signatures, digital certificates, or consensus algorithms.

APPENDIX-B

REFERENCES

[1] Monika Sharma, Priyansh, Anjali Mathur, Anmol Rajoura, Aakansha Tyagi 2023. "Infinity Diaries: A Nft Marketplace " 2023 C2-F3.pdf (ipec.org.in)

[2] Mr. Umesh Mohite, Mr. Om Bhosale, Mr. Rohan Doshi, Mr. Jainil Modi, Mr. Fenil Moradiya 2023. Decentralized NFT Marketplace: Design, Implementation, and Challenges Involved JETIR Research Journal

[3] Mieszko Mazur (ESSCA School of Management), "Non-Fungible Tokens (NFT) The Analysis of Risk and Return", Mieszko Mazur, IESEG School of Management, 2021

[4] Matthieu Nadini, Laura Alessandretti, Flavio Di Giacinto, Mauro Martino, Luca Maria Aiello & Andrea Baronchellii, "Mapping the NFT revolution: market trends, trade networks, and visual features", 22nd International Arab Conference on Information Technology (ACIT), 2021

[5] Wajiha Rehman, Hijab e Zainab, Jaweria Imran, Narmeen Bawany, "NFTs: Applications and Challenges", 22nd International Arab Conference on Information Technology (ACIT), 2021

[6] Lennart Ante, "The non-fungible token (NFT) market and its relationship with Bitcoin and Ethereum", Blockchain Research Lab, Max-Brauer-Allee 46, 22765 Hamburg, 2021

[7] Pavel Kireyev, "NFT Marketplace Design and Market Intelligence", INSEAD Working Paper No. 2022/03/MKT, Business school of the world, 2022

[8] Charles Jaisurya, "Developing an NFT Marketplace - The Ultimate Guide | NFT Marketplace Development", Conference: 22nd International Arab Conference on Information Technology (ACIT), 2022

[9] Shayel Shams, "NFT Market Research: A Statistical Overview based on Digital Assets under the Crypto Space", Shandong Institute of Business and Technology, 2022

[10] Dinuka Piyadigama, Guhanathan Poravi, "An Analysis of the Features Considerable for NFT Recommendations", NFT-RecSys: A Trading Recommendations System for Non-fungible Tokens, Computer Science and Engineering University of Westminster, 2022

[11] Qin Wang, Rujia Li, Qi Wang, Shiping Chen, "Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges", Cryptography and Security (cs.CR), Southern University of Science and Technology, 2021