Web3DP: A Crowdsourcing Platform for 3D Models Based on Web3 Infrastructure

Lehao Lin
The Chinese University of Hong Kong, Shenzhen
Shenzhen, Guangdong, China
lehaolin@link.cuhk.edu.cn

Haihan Duan
The Chinese University of Hong Kong, Shenzhen
Shenzhen, Guangdong, China
haihanduan@link.cuhk.edu.cn

Wei Cai
The Chinese University of Hong Kong, Shenzhen
Shenzhen, Guangdong, China
caiwei@cuhk.edu.cn

ABSTRACT

Recently, the concept of metaverse has been rapidly emerging, which highly expands the human living space. Specifically, 3D models are at the heart of building a vast metaverse space, so a massive number of 3D models are needed. Existing 3D model libraries and platforms have achieved great results. However, most of them are unscalable, insufficiently open, inefficient to collect, and at risk of service disruption and data corruption. Therefore, we propose and implement Web3DP, a crowdsourcing platform for 3D models based on Web3 (a.k.a. Web 3.0) infrastructure. By using the decentralized blockchain technology, Web3DP has the advantages of transparency, auditability, traceability, data tamper-proof, high file transfer efficiency, and service stability. Experiments are conducted to validate the performance of the proposed platform. It illustrates that Web3DP shows better file transmission capabilities with an acceptable transaction fee to facilitate 3D model collecting and managing for metaverse, games, cultural heritage, etc.

CCS CONCEPTS

• **Information systems** → **Open source software**; **Crowdsourcing**; • **Applied computing** → **Digital libraries and archives**.

KEYWORDS

Web3, Decentralized Application, 3D Model, Crowdsourcing, Metaverse

ACM Reference Format:
Lehao Lin, Haihan Duan, and Wei Cai. 2023. Web3DP: A Crowdsourcing Platform for 3D Models Based on Web3 Infrastructure. In *Proceedings of the 14th ACM Multimedia Systems Conference (MMSys '23), June 7–10, 2023, Vancouver, BC, Canada.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3587819.3592549

1 INTRODUCTION

In recent years, the concept of metaverse [21] attracts a lot of concentration. In the expected metaverse, the 3D model becomes the core material to build the metaverse space. Large as complex accessible buildings, small as insects and floral decorations, 3D models are not only for the use of metaverse facilities, but for the innovative user-generated content (UGC). In 2021, Duan *et al.* [9] listed the features of representative metaverse examples, in which 12/13 metaverse examples integrate UGC as a part of their interaction layer. Following the trend, UGC will take an increasingly important role in the metaverse [8]. However, the UGC creation of 3D models meets some problems remained to be solved: (1) The 3D modeling process is much more difficult for normal users. For 2D pictures, copying and modifying existing similar images is a reasonable way for 2D UGC creation; however, the resource of 3D models is not as abundant as 2D images, so the users need an open repository of 3D models to learn 3D modeling or to be inspired for creation. (2) With the development of deep learning [16], many generative models, like variational auto-encoder (VAE) [15] and generative adversarial network (GAN) [10] get amazing success. For example, Text2shape [7], Pixel2Mesh [23], PolyGen [18], etc., could be helpful to the 3D model UGC creation. (3) It is not easy to transfer the 3D model from one metaverse to another. Usually, the users need to export it to the local computer and then upload it again to another metaverse space.

However, the existing 3D model libraries and datasets are not open enough in terms of uploading and downloading. Most platforms require users to register an account and wait for approval by administrators. The process is complicated and consumes a lot of time and human resources. Therefore, creating a new platform to collect and manage 3D models as a resource library for the metaverse is very needed.

Generally, there are two common forms of organizing and implementing the 3D model platform: non-crowdsourcing and crowdsourcing. In the Web 2.0 era, users get the right to data writing, and it brings the success of the crowdsourcing revolution [1]. In 2006, Jeff Howe [12] introduced the term crowdsourcing and said it is for the masses; where the author mentioned that the cost of crowdsourcing compared with any other forms of organization is "Incredibly cheap". In crowdsourcing activities, such as contributing content on Wikipedia, developing open source software, uploading videos on YouTube, etc., crowds are intrinsically motivated [11]. Consequently, due to the advantages of high efficiency and low costs, we choose to build a crowdsourcing platform.

During the past decades, network platforms and applications building on Web 2.0 technology frameworks have been sound and mature. So, when creating the crowdsourcing platform, using the centralized Web 2.0 frameworks and the traditional client-server (C/S) architecture can reduce the difficulty and cost of development and operation and lower the threshold for platform users to get started quickly. However, Web 2.0 based applications are ultimately characterized by centralization and possess the risks and flaws of

MMSys '23, June 7–10, 2023, Vancouver, BC, Canada

ACM ISBN 979-8-4007-0148-1/23/06…$15.00
https://doi.org/10.1145/3587819.3592549

a centralized platform that threaten the security of systems and data. Inspired by the success of Bitcoin[1], Vitalik Buterin proposed a decentralized platform Ethereum[2] in 2013, which can deploy and execute the smart contract. The smart contract is written in a new programming language called Solidity, and its functions can be invoked on the Ethereum Virtual Machine (EVM) after being deployed [5]. Thus, applications that run on smart contracts are called decentralized applications (DApps). Besides, the ecosystem on this decentralized infrastructure is named Web3 [3] (a.k.a. Web 3.0). Unlike Web 2.0 applications, the code of Web3 applications is open source passively, and the user interface (UI) code, frontend logic, and deployed smart contracts are all auditable. Moreover, everyone can check the chain record using blockchain explorers, like Etherscan for Ethereum and Polygonscan for Polygon. The open-source feature ensures that the program is not fraudulent, does not overcollect user information, and does not redundantly perform harmful operations on users. Such an open source advantage is not available in traditional centralized architecture platforms. Therefore the decentralized architecture of the platform has the characteristics of true open source, full stack auditable, high stability, and able to run continuously for a long time.

By using the emerging Web3 infrastructure, we design and implement a decentralized crowdsourcing platform for collecting 3D models called Web3DP. We transform every uploaded 3D model to a non-fungible token (NFT) based on the ERC-721 Standard[3]. The platform's frontend files are hosted on Fleek[4], and the backend code is written in Solidity, developed as a smart contract, and deployed on Polygon Network. After being stored by NFT.storage, the free and decentralized storage and bandwidth for NFTs on InterPlanetary File System (IPFS) and Filecoin, the 3D models can be minted to NFTs. Users can upload their 3D model to NFT.storage by the frontend JavaScript function and then call the write function of the smart contract to mint the model to NFT. After that, we can read the contract to get the 3D model's store link and the file. What's more, we also make an extra smart contract to store the NFT and string labels, where the users can add labels to 3D model NFTs for future filtering and description.

Our contributions can be concluded as follows:

- We design and implement a full-stack open source crowdsourcing platform based on the smart contract and the NFT-based database for collecting and managing 3D models, which can upload and view 3D models and also add labels to models for searching and filtering.
- We provide the performance evaluation to the system, such as file transmission time, and gas cost comparison. The experimental results illustrate that Web3DP gets better transmission efficiency with an acceptable transaction fee.

2 RELATED WORK

2.1 Centralized 3D Dataset and Platform

Over the last few years, examined works on 3D model datasets and platforms emerged to feed the state-of-the-art deep learning technology. Some representative datasets and platforms will be briefly described in the following.

ShapeNet [6] is a famous indoor 3D CAD models repository with a large scale, which contains over 3 million models, and 220000 of them are categorized into 3135 classes. It is a centralized collaborative project between researchers at Princeton, Stanford, and Toyota Technological Institute at Chicago (TTIC). However, if users want to download ShapeNet data, they need to create an account as registered users and the account should be verified and approved by website administrators.

Qiao *et al.* [19] proposed a parameterized crowdsourcing method and platform for indoor 3D furniture model data. The work meets the needs of high-precision indoor location services and makes the 3D indoor collection and expression methods simple and efficient. Shishido *et al.* [20] and Inzerillo *et al.* [14] both proposed to use the crowdsourcing method and 3D reconstruction technology to organize a proactive preservation project for the cultural heritage.

Though the afore-mentioned works achieve high efficiency in collecting 3D models by crowdsourcing, the above platforms and projects are centralized, vulnerable to a single point of failure and possibly corrupting the data. Besides, their project relies on team organization. Once the team dismisses, the project and the platform stop running. Furthermore, incentives stated on centralized crowdsourcing platforms may carry the risk of platform modification of incentive rules, dishonesty, and non-delivery.

2.2 Decentralized Crowdsourcing Platform

In order to solve the problems mentioned in the last section, many crowdsourcing projects have integrated blockchain and built their decentralized crowdsourcing platforms with different aims.

In the federated learning (FL) field, FL-MAB [4] can solve the client straggler and monetization incentive problems. It utilized a multidimensional auction mechanism for selecting users.

There are also more relevant to us that do not set the domain and scope and do crowdsourcing platforms and frameworks. The difference between them and us is the translation of the task domain into a 3D model, but we can learn from and reference their experience. CrowdBC [17] conceptualized a decentralized framework for crowdsourcing, in which a group of workers can solve a requester's task without relying on any third-party trust, the user's privacy can be guaranteed, and only low transaction fees are required. However, the framework's state machine construction contains both canceled and completed states. And it requires workers to deposit some token at registration to ensure the quality of work. So it is more suitable to meet the requester-worker relationship under the single-task crowdsourcing situation, but not suitable for continuous running, long-term, with a large amount of data tasks. Zhu *et al.* [24] presented zkCrowd, a hybrid blockchain crowdsourcing platform. They used dual ledgers and dual consensus protocols with a public chain and multiple private subchains, while delegated proof of stake (DPoS) and practical Byzantine fault tolerance (PBFT) consensuses are implemented on the public chain and subchains, respectively. However, DPoS is less centralized than other consensus protocols, where it remains centralization risks. Thus, for welfare open-source crowdsourcing projects, it is no need to use the PBFT sub-chain to isolate every task, especially for collecting 3D models.

[1] https://bitcoin.org/en/
[2] https://ethereum.org/en/
[3] https://ethereum.org/en/developers/docs/standards/tokens/erc-721/
[4] https://fleek.co/

3 WEB3DP

3.1 Architecture

Fig. 1 shows the architecture of Web3DP. inspired by Blockstack [2], we divide Web3DP into three main parts, the frontend part, the storage part, and the Polygon Network part.

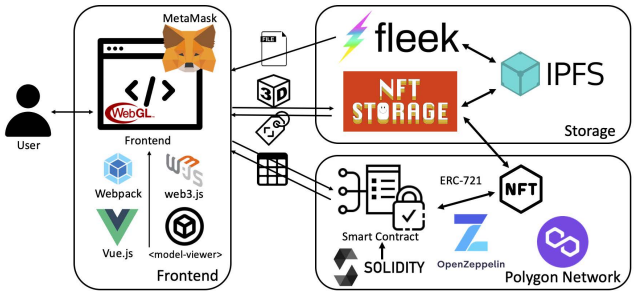

Figure 1: Architecture of Web3DP.

3.1.1 Frontend. We use Vue.js 3 as the frontend framework, and Vue-CLI (@vue/cli) is a globally installed npm package that provides the vue command in the terminal. Moreover, we use Webpack to pack all of the frontend materials. We utilized the model-viewer[5], which can easily display interactive 3D models on the web and in AR, powered by Google. Thus, we use web3.js to connect to the blockchain network and the smart contract. In order to call the write function in the smart contract from the browser, we apply the browser extension wallet, MetaMask[6], to send transactions.

3.1.2 Storage. Fleek hosts our static frontend files, such as HTML, JavaScript, and CSS, on the IPFS network. It also provides IPNS domain name mapping for the file hash value [13]. Additionally, Fleek offers CI/CD and automatic deployment services for easy integration with GitHub.

We use NFT.storage to store 3D models. We call the function, especially for the NFT digital asset storage. Each ERC-721 token contains a "metadata" string for the definition, which shows what the NFT is. It is in a standardized JavaScript Object Notation (JSON) format, and have three fixed string type attributes, "name", "description", "image". In the "image" attribute, there is no limitation saying that only image files such as JPEG and PNG are required. Any visual or auditory digital asset like audio and video is welcome, as well as 3D model file types like GLB and glTF.

3.1.3 Polygon Network. In platform implementation, we choose Polygon Network as the blockchain platform to deploy and execute our smart contract. It is an EVM-compatible Ethereum sidechain, so Solidity code can run on it. We use OpenZeppelin, a library for secure smart contract development, to assist in writing smart contracts. The ERC-721 standard is more complex with splitting across several contracts and multiple optional extensions than the ERC-20, which is for fungible tokens.

[5]https://modelviewer.dev/
[6]https://metamask.io/

3.2 Workflow

The proposed platform Web3DP consists of four phases, i.e., the initial, the model, the upload, and the label phase, as illustrated in Fig. 2, a system sequence diagram of Unified Modeling Language (UML). Furthermore, we split the platform into four main entities, users, frontend, blockchain, and IPFS. Here we denote IPFS as unified Fleek and NFT.storage usage. We will discuss the details of these four phases as follows.

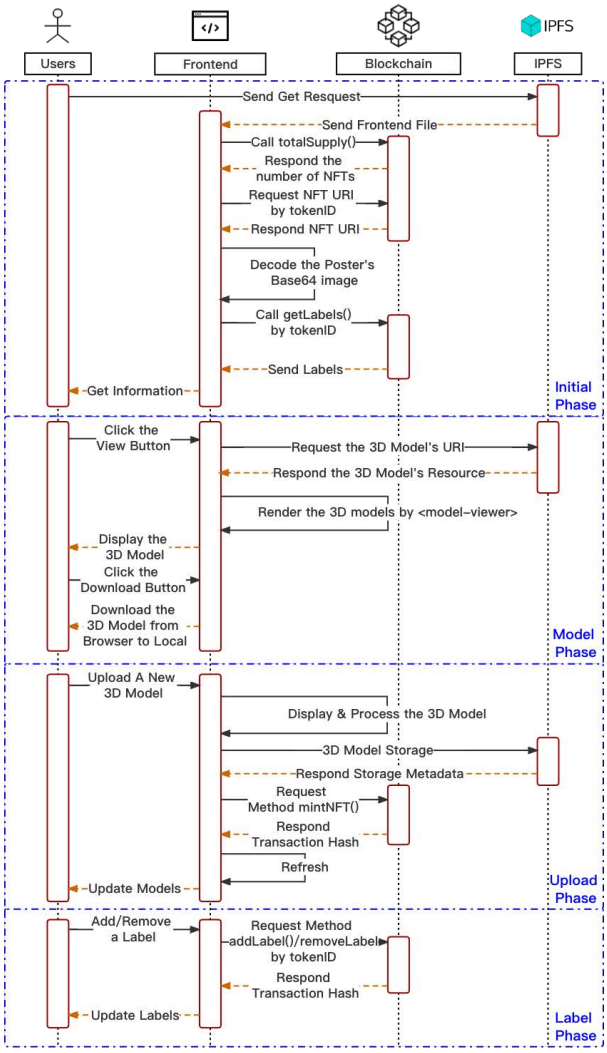

Figure 2: Sequence UML of Web3DP.

3.2.1 Initial Phase. When we enter the URL link in the browser, we will send an HTTP GET request to the hosting provider Fleek. Fleek will respond to us with the frontend code and assets stored in IPFS. Then the frontend will run the JavaScript program automatically fetching the number of NFTs in the contract by calling the function *totalSupply*(). After we get the total number of NFTs, we can enumerate token IDs from 1 to the total number to easily get a

list of NFT URIs. To avoid the network crowd, we fetch 10 latest NFTs by default. Users can click the button to fetch more data from the chain if they want to explore more. Once we have the URI, we can get the name and the poster of the 3D model. By decoding the Base64 string, the poster will be shown on the page. At the end of the initial phase, Web3DP will call the *getLabels*() function of the label contract by enumerating token IDs and updating the label column in the data table.

3.2.2 Model Phase. The model phase is about 3D model visualization and download. In the previous initial phase, we already got a list of metadata for every NFT. By tooling the model-viewer, we insert the model's IPFS link so the model can be rendered in the browser smoothly. After viewing and checking the model, users can click the download button to store the 3D model file.

3.2.3 Upload Phase. To contribute a new 3D model to Web3DP, users should switch the view mode to the upload mode by the switch button. Afterward, users can select their local 3D model in the frontend and preview the model on the page's right side. Then, users should prepare their API KEY from the official NFT.storage website. We use cookies to memorize the KEY after successfully uploading the model so that users don't have to copy and paste the key repeatedly. After uploading the model file to NFT.storage, the frontend will receive a string of the metadata URI. With our wallet address and the URI, we call *mintNFT*() function to make a new NFT. Next, the page will turn back to the view mode.

3.2.4 Label Phase. Compared to other phases, the label phase in Web3DP can be seen as a side phase. However, it is an important aspect as it allows users to edit labels on NFTs, making it the icing on the cake.

We don't design de-duplication detection when adding or removing labels to save computational resources and avoid high gas waste. We leave a zero state value to the contract for removing records. If there are multiple labels with the exact text and different statuses, Web3DP will count the numbers of both two sides and compare which side is more numerous to display. For example, users get a label list with two exact label texts but various statuses 0 and 1. The label will not show up on the table. However, the label will appear if there are three label texts with status 0, 1, and 1.

4 IMPLEMENTATION

4.1 Development

In the development process, we split the whole project code into the backend and frontend parts, as shown in Fig. 3. Web3DP sources are available publicly on GitHub: https://github.com/LehaoLin/web3dp. The external software sources and versions can be found in Table 1.

The backend part emphasizes smart contract coding and deployment. In the *artifacts* folder, there are intermediate and final compiled products like building information, solidity compiler debug information and the contract application binary interface (ABI) in JSON format. And for compilation and deployment purposes, we place the solidity code files in the *contracts* folder. Moreover, we put the module script for contract deployment under the *scripts*. The file of *.env* is to store the private key of the deploy wallet, which is not open to the public. Besides, the file *hardhat.config.js* is to set

Table 1: Main Software Packages and Versions.

Software	Version	Comments
Node.js	16.14.0	JavaScript runtime
Yarn	1.22.17	Package manager
Vue.js	3.2.13	Frontend framework
element-plus	2.2.17	UI component library
model-viewer	2.0.0	3D models displayer
maticjs	3.5.0	Interact with Matic Network
web3.js	1.8.0	Interact with a Ethereum node
Solidity	0.8.12	Language for smart contracts
ethers	5.7.1	Interact with Ethereum ecosystem
nft.storage	7.0.0	Free Storage for NFTs
contracts	4.7.3	Secure smart contract development
hardhat	2.11.1	EVM development environment

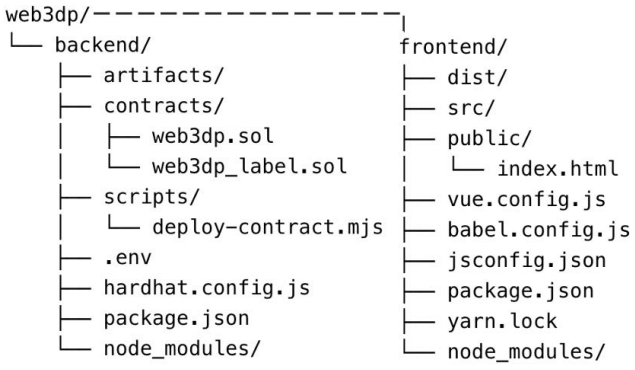

Figure 3: The Brief Structure of Web3DP Source Code.

up the environment of connecting the blockchain network and the solidity compiler version by using hardhat.

The frontend part is a general Vue3 project, as shown configure files *vue.config.js*, *babel.config.js*, *jsconfig.json*. The Vue source code i.e. **.vue* files are under the *src* path. Furthermore, compiled, packed, and zipped frontend files will be placed in the *dist*. Since we use Yarn to handle the frontend project, a *yarn.lock* is generated to fix versions of dependencies.

4.2 Demonstration

We deploy two smart contracts to *Polygon's Mumbai testnet*. The deploy script uses a remote procedure call (RPC) to interact with the Mumbai testnet[7]. Users can easily visit *faucet*[8] and enter their wallet address to get the test MATIC token. Except for reading data from the blockchain, any writing data behavior like minting NFTs and operating labels need MATIC to pay the gas fee of smart contract operation, known as the transaction fee, even though there is no purchase item in the platform.

Next, users can check addresses, transactions, and blocks from the Mumbai Polygonscan[9]. Users also can check the balance of their

[7] https://rpc-mumbai.maticvigil.com
[8] https://faucet.polygon.technology
[9] https://mumbai.polygonscan.com

wallet after getting test tokens from *faucet*, and check our smart contracts before using Web3DP:

NFT Contract Address:
0x14C77219F67E84Ab098aa08f93f984B31038ceB7
Label Contract Address:
0xA9aA9A5cDeDBd8144EF352346e810E2A5097EBF8

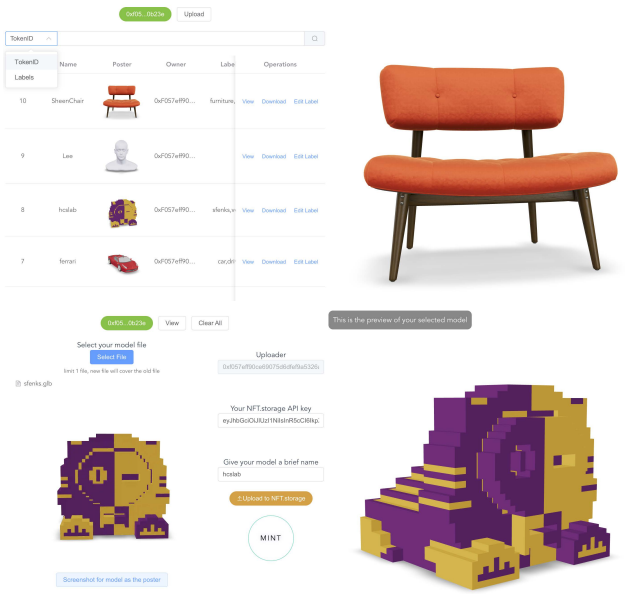

Figure 4: Up: View Mode of User Interface; Down: Upload Mode of User Interface.

We deploy the frontend on Fleek, and the platform website is https://dry-smoke-4056.on.fleek.co/. As shown in Fig. 4, the user interface (UI) is divided into two areas. The left area is for user operations, while the right area is for 3D models displaying. Users can zoom, rotate and pan 3D models in the model viewer.

5 PERFORMANCE EVALUATION

In this section, we conduct experiments to validate functionalities and evaluate the performance of the proposed platform.

5.1 Experimental Settings

To verify that the platform and model storage on IPFS possesses advantages in enhancing user experience in addition to fighting various risks, we designed experiments on the system loading and file transmission. Furthermore, we conducted experiments on the gas cost to verify whether the transaction fee will hinder users.

Figure 5: Four Screenshots of the Experiment 3D Models

Table 2: Sizes of selected Models.

File Name	Model Size	Poster Image Size
sfenks.glb	83 KB	21.7 KB
Lee.glb[10]	405 KB	19.5 KB
ferrari.glb[11]	1.7 MB	28.8 KB
SheenChair.glb[12]	4.4 MB	32.9 KB

For the experiments, we used four 3D models (shown in Fig. 5) ranging from 83 KB to 4.4 MB in size. These models were selected for their simplicity in terms of faces and textures, as well as their diverse file size distribution.

Table 2 shows file size and poster image for each item. The poster size does not seem to have any correlation with the file size. Therefore, using the transformed poster's Base64 string to fill the NFT's description field should not lead to rejection by NFT.storage.

5.2 File Transmission

Comparing the upload speed and the download speed of the platform, this section uses the four mentioned 3D models to upload and download them 10 times to derive the average and standard deviation of the whole network time.

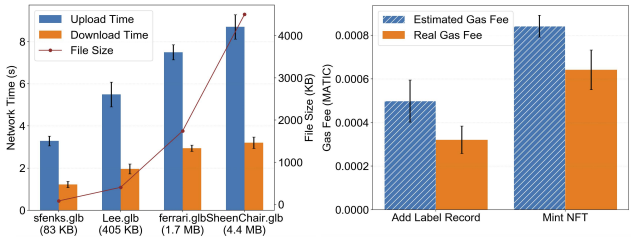

Figure 6: Left: 3D Model File Network Time on IPFS. Right: Gas Fee Comparison.

As shown in the left part of Fig. 6, the larger the file size, the longer the network time required for uploading and downloading. However, the download time increases more gradually than upload time. In this experiment, the upload time was roughly three times longer than the download time. The reason for this difference is that IPFS uses block storage for files, and blocks are spread across different nodes for parallel transmission, resulting in faster download speeds. However, during the upload process, files must first go through NFT.storage to generate NFT metadata, and cannot be sharded transferred in parallel, unlike the download process. This ensures file integrity and coherence, but significantly increases upload time. In the Seshat experiment [22], file fetching time did not increase significantly with increasing files.

[10]Lee Perry-Smith head. From the three.js example. https://threejs.org/examples/webgl_materials_matcap.
[11]Ferrari 458 Italia model. From the three.js example. https://threejs.org/examples/webgl_materials_car.html.
[12]Sheen Chair. From the three.js example. https://threejs.org/examples/webgl_loader_gltf_sheen.html.

5.3 Gas Cost

We compared the gas cost for two functions, the label record addition, and NFT mint. We repeat these two functions 5 times to record every time's estimated gas fee noted by MetaMask and the actual gas fee queried from the Mumbai Polygonscan. Then, we get averages and standard deviations of the gas fee, shown in Fig. 6.

In order to ensure smooth transaction processing, the estimated gas fee is always slightly higher than the actual gas fee. The gas fee for minting an NFT is almost twice as much as adding a label record. During the experiment (conducted on September 29th, 2022 with MATIC/USD at 0.75), minting one NFT on Web3DP cost approximately 0.0005 USD, which is acceptable for normal users in terms of user experience to participate in the crowdsourcing project. We compared the gas cost for two functions, label record addition and NFT mint, by repeating these functions five times and recording both the estimated gas fee noted by MetaMask and the real gas fee queried from the Mumbai Polygonscan.

5.4 Comparison with Other Architectures

When comparing Web3DP with existing centralized and decentralized crowdsourcing systems, Web3DP stands out in terms of open-source transparency and auditability. Its blockchain and IPFS technology provide an advantage in terms of persistence. User access and permission are streamlined in Web3DP, as it directly connects user MetaMask wallet and smart contract, eliminating the need for registration and administrator authorization, as well as pledge Token and sub-chain support. Centralized systems and Web3DP system are equally proficient in 3D model tasks. However, smart contracts on public chains require gas to invoke write methods, which leads to higher usage costs compared to other architectures. Additionally, once smart contracts and data are deployed and uploaded, the system becomes challenging to modify and upgrade, making it a disadvantage compared to other architectures.

Table 3: Comparison with Other Architectures.

	Centralized	Other DApps	Web3DP
Open-source	★	★ ★	★★★
Persistence	★ ★	★★★	★★★
User Privileges	★	★ ★	★★★
3D Task Expertise	★★★	★ ★	★★★
Ease of Costs	★★★	★ ★	★ ★
Upgrade Ease	★★★	★	★

6 CONCLUSION AND FUTURE WORK

In this paper, we presented Web3DP, a crowdsourcing platform for collecting and managing 3D models based on Web3. Users can upload their collecting model to contribute to the project and also view others' work. The proposed Web3DP is genuinely open and transparent. The users can fully trust the platform since they can audit the source code and contracts to check the security and vulnerability of functions or query the on-chain data to inspect data corruption. Moreover, the service downtime and data storage can be well guaranteed by the applied decentralized technologies.

ACKNOWLEDGMENTS

This work was supported by Shenzhen Science and Technology Program (Grant No. JCYJ20210324124205016).

REFERENCES

[1] Adedamola Adepetu, Ahmed Altaf Khaja, Yousif Al Abd, Aaesha Al Zaabi, and Davor Svetinovic. 2012. Crowdrequire: A requirements engineering crowdsourcing platform. In *2012 AAAI Spring Symposium Series*.
[2] Muneeb Ali, Jude Nelson, Ryan Shea, and Michael J Freedman. 2016. Blockstack: A global naming and storage system secured by blockchains. In *2016 USENIX annual technical conference (USENIX ATC 16)*. 181–194.
[3] Joost Bambacht and Johan Pouwelse. 2022. Web3: A Decentralized Societal Infrastructure for Identity, Trust, Money, and Data. *arXiv preprint arXiv:2203.00398* (2022).
[4] Zahra Batool, Kaiwen Zhang, and Matthew Toews. 2022. FL-MAB: client selection and monetization for blockchain-based federated learning. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*. 299–307.
[5] Vitalik Buterin et al. 2014. A next-generation smart contract and decentralized application platform. *white paper* 3, 37 (2014), 2–1.
[6] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. 2015. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012* (2015).
[7] Kevin Chen, Christopher B Choy, Manolis Savva, Angel X Chang, Thomas Funkhouser, and Silvio Savarese. 2018. Text2shape: Generating shapes from natural language by learning joint embeddings. In *Asian conference on computer vision*. Springer, 100–116.
[8] Haihan Duan, Yiwei Huang, Yifan Zhao, Zhen Huang, and Wei Cai. 2022. User-Generated Content and Editors in Video Games: Survey and Vision. In *2022 IEEE Conference on Games (CoG)*. IEEE, 536–543.
[9] Haihan Duan, Jiaye Li, Sizheng Fan, Zhonghao Lin, Xiao Wu, and Wei Cai. 2021. Metaverse for social good: A university campus prototype. In *Proceedings of the 29th ACM International Conference on Multimedia*. 153–161.
[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).
[11] Mokter Hossain. 2012. Users' motivation to participate in online crowdsourcing platforms. In *2012 International Conference on Innovation Management and Technology Research*. IEEE, 310–315.
[12] Jeff Howe et al. 2006. The rise of crowdsourcing. *Wired magazine* 14, 6 (2006), 1–4.
[13] Tam T Huynh, Thuc D Nguyen, and Hanh Tan. 2019. A decentralized solution for web hosting. In *2019 6th NAFOSTED Conference on Information and Computer Science (NICS)*. IEEE, 82–87.
[14] Laura Inzerillo and Cettina Santagati. 2016. Crowdsourcing cultural heritage: from 3D modeling to the engagement of young generations. In *Euro-Mediterranean Conference*. Springer, 869–879.
[15] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.
[17] Ming Li, Jian Weng, Anjia Yang, Wei Lu, Yue Zhang, Lin Hou, Jia-Nan Liu, Yang Xiang, and Robert H Deng. 2018. CrowdBC: A blockchain-based decentralized framework for crowdsourcing. *IEEE Transactions on Parallel and Distributed Systems* 30, 6 (2018), 1251–1266.
[18] Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. 2020. Polygen: An autoregressive generative model of 3d meshes. In *International conference on machine learning*. PMLR, 7220–7229.
[19] Biyun Qiao, Qun Sun, and Hongyuan He. 2021. A crowdsourcing method for 3D furniture based on parameterized template. In *Journal of Physics: Conference Series*, Vol. 1952. IOP Publishing, 032029.
[20] Hidehiko Shishido, Yutaka Ito, Youhei Kawamura, Toshiya Matsui, Atsuyuki Morishima, and Itaru Kitahara. 2017. Proactive preservation of world heritage by crowdsourcing and 3D reconstruction technology. In *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 4426–4428.
[21] Neal Stephenson. 2014. *Snow crash*. Bragelonne.
[22] Lin Wang, Lehao Lin, Xiao Wu, and Rongman Hong. 2021. Seshat: Decentralizing Oral History Text Analysis. In *2021 17th International Conference on Mobility, Sensing and Networking (MSN)*. IEEE, 812–817.
[23] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. 2018. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European conference on computer vision (ECCV)*. 52–67.
[24] Saide Zhu, Zhipeng Cai, Huafu Hu, Yingshu Li, and Wei Li. 2019. zkCrowd: a hybrid blockchain-based crowdsourcing platform. *IEEE Transactions on Industrial Informatics* 16, 6 (2019), 4196–4205.