Decentralized Application for Rating Internet Resources

Andreea Buţerchi[a] and Andrei Arusoaie[b]
Alexandru Ioan Cuza, University of Iaşi, Romania

Keywords: Blockchain, Decentralized Application, Rating.

Abstract: A lot of existing web applications use systems for rating internet resources (e.g., YouTube uses like/dislike-based rating, IMDb uses star-based rating). Given the received ratings, the most popular internet resources can generate large amounts of money from advertising. One issue arising from this is that the existing rating systems are entirely controlled by a single entity (i.e., the owner of the web application). In this paper we present a blockchain-based decentralized application for rating internet resources. The proposed solution provides a transparent rating mechanism, given that no central authority is involved and the rating operations are handled by a specialized smart contract. We present an implementation of our solution, where we combine authentication methods with blockchain specific features, so that the users' anonymity is preserved. We show that our approach is better than the existing rating systems in terms of transparency and reliability.

1 INTRODUCTION

Various web applications use rating systems to establish specific trends or trending posts, videos, movies or products. There are several categories of such systems, which are used for content or products evaluation: *like/dislike-based rating systems*, *star-based rating systems*, *review-based rating systems* and others.

It is worth noting that the most web applications are in control of the underlying rating systems. Moreover, the algorithms involved in the rating process are not publicly available, fact that contributes to the lack of transparency and affects the trust of the users. Since the web applications access and control the ratings, the users do not have the certainty that the owners of the web applications did not alter the rating record in their favor, especially when the ratings are a criteria for producing money from advertising.

In contrast to the traditional web applications, the decentralized applications are not controlled by a single authority. They run on a peer-to-peer network, which makes them more flexible, secure and reliable than the conventional web applications.

In order to highlight the need of introducing a decentralized rating system, we present some real scenarios, where conventional rating systems are lacking of transparency.

YouTube is considered to be one of the most popular media content providers[1]. Despite the fact that YouTube informs its users with respect to the number of views, likes or dislikes received by the video resources, it doesn't specify exactly the manner in which other factors (e.g., the watch time, the session time or the popularity of a specific channel) influence these numbers[2]. Moreover, given that the popularity of the posted content is a criteria for being incentivized by YouTube, the users may be determined to use fraudulent means (e.g., click farms) to increase their incomes. On this note, YouTube claims that its rating algorithms handle these kinds of fraudulent activities, but there is no certain proof in this respect.

Similarly to YouTube, IMDb does not disclose the used rating algorithms. The resources on IMDb do not have associated the raw average, but the weighted average of the received ratings. As expected, IMDb does not mention which factors influence the computing of the final (i.e., user accessible) ratings[3].

[a] https://orcid.org/0000-0001-7258-0620
[b] https://orcid.org/0000-0002-2789-6009

[1] https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/

[2] There are plenty of web pages where users complain (e.g., https://support.google.com/youtube/thread/11131851?hl=en). Even if the data on these pages cannot be necessarily trusted, it is hard for YouTube to guarantee that its rating algorithms are transparent and work properly.

[3] https://help.imdb.com/article/imdb/track-movies-tv/ratings-faq/G67Y87TFYYP6TWAV#

Buţerchi, A. and Arusoaie, A.
Decentralized Application for Rating Internet Resources.
DOI: 10.5220/0010573405130520
In *Proceedings of the 16th International Conference on Software Technologies (ICSOFT 2021)*, pages 513-520
ISBN: 978-989-758-523-4

1.1 Related Work

To our best knowledge, at this moment there is only one solution that has the purpose of moving rating systems into the blockchain. The solution is reported in (Shaker et al., 2021), but it seems to reiterate the ideas from (Buterchi and Arusoaie, 2020). Despite the similar motivation, the authors approach decentralized rating in a different manner. In the system proposed by the authors, the users need to register using their Ethereum addresses. It is worth noting that the authors use these Ethereum addresses to validate the rating process and, consequently, to test if the users have already rated. Even though it seems to be a suitable approach, in fact, it introduces vulnerabilities. Given that the users can generate an unlimited number of Ethereum addresses, they can register using multiple addresses. By expressing their ratings multiple times, the users alter the integrity and the credibility of the rating record. In the current paper we address this problem by using an authentication mechanism, which allows us to reduce the fraudulent activities and to preserve the integrity of the rating data. In the context of our solution, the Ethereum addresses are used only for the interaction with the smart contract and for payment purposes (i.e., the users need to pay for the rating operations). The authentication mechanism will be discussed in detail in Section 3.3. Another problem concerning the solution proposed in (Shaker et al., 2021) is that the rating method does not guarantee the relevance of the final ratings. Given that it simply computes the average of the ratings (i.e., the sum of the ratings divided by the number of raters), the final ratings could drop considerably when the users submit ratings close to 0. In the current paper, we avoid this problem since we propose a like/dislike-based rating system. The users have access to the final ratings, which consist of the total number of likes and dislikes for each rated resource.

Besides the aforementioned solution, decentralized rating can be associated with decentralized voting (Bocek et al., 2009; Khader et al., 2012; Ayed, 2017; Hsiao et al., 2018; Khoury et al., 2018). In the case of voting, the users express their vote for a specific candidate. The action of voting is irreversible and the users cannot vote more than once. In the case of rating, the users can rate internet resources and their actions are not limited to only one rating operation. Despite the visible differences, voting and rating are similar, since both systems need the same security properties, including privacy (i.e., the identity of the voter/rater should not be disclosed) and integrity (i.e., the voting/rating system should be fraud-resilient).

1.2 Contributions

In order to address the transparency issues of the traditional rating systems, we propose a blockchain-based decentralized application for rating.

Our solution provides a trustworthy and transparent rating system, given that the rating record is stored on the blockchain and the smart contract's codebase is publicly available. The proposed rating system simulates the functionalities of a *like/dislike-based rating system*. The *smart contract* handles the rating operations and prevents multiple rating situations (i.e., when a user attempts to submit multiple likes or multiple dislikes for a resource). Besides the smart contract, our solution comes with an additional *authentication mechanism*. The authentication mechanism is helpful due to the following reasons: guarantees the uniqueness of the users, reduces fraudulent actions (i.e., it does not allow unauthenticated users to rate) and provides various authentication options. In the current form of the application, the users can rate resources from Google (e.g., YouTube), GitHub and Spotify. The decentralized application comes with an intuitive *user interface*, which is divided into three sections: authentication, rating and data visualization.

1.3 Paper Organization

In Section 2, we present the concepts, the tools and the technologies used in the current paper. Our contribution, i.e., decentralized application for rating, is detailed in Section 3. In Section 4 we present the experiments that we have performed, along with an overview of their execution costs. Finally, we conclude in Section 5.

2 PRELIMINARIES

This section contains a brief presentation of the tehnologies, tools and programming languages that we used in the development of our solution.

The *blockchain* is a new technology that records data across a peer-to-peer network in a distributed shared ledger. The technology was initially designed for cryptocurrency trading (Nakamoto, 2009), and nowadays it also supports smart contracts.

Ethereum (Wood, 2014; Buterin, 2013) is a popular blockchain platform, which enables the use of *smart contracts*. Smart contracts can simulate real-world agreements for different kinds of assets. Usually, they are written in high-level languages (e.g., Solidity, Vyper) and compiled into Ethereum bytecode

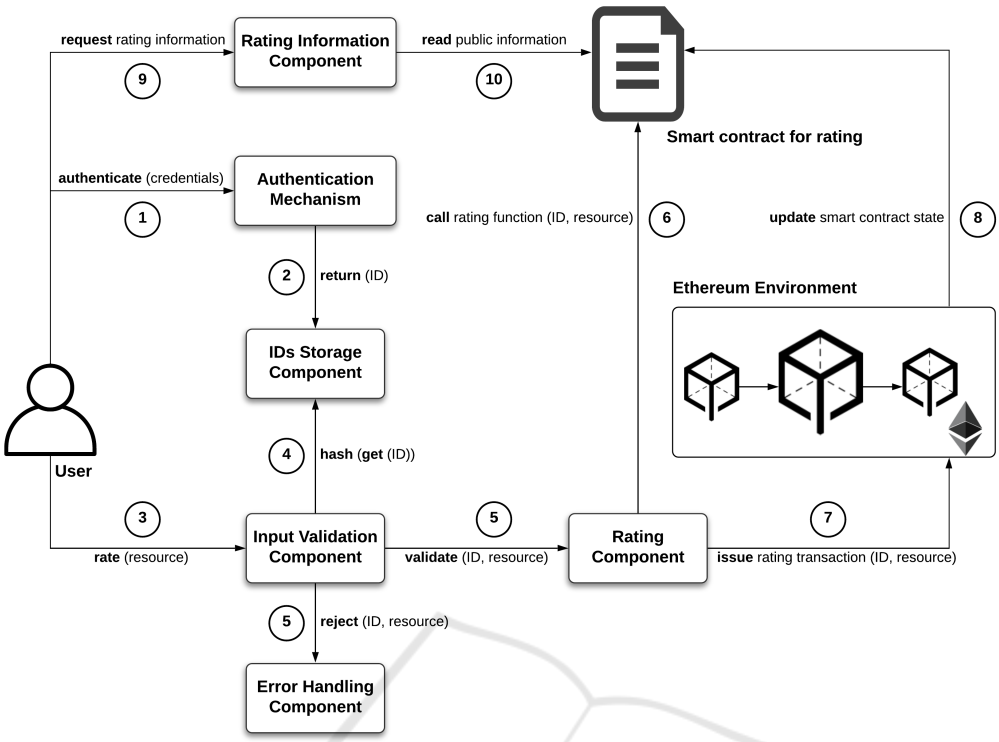

Figure 1: A use case diagram of the decentralized application for rating.

(Wood, 2014; Buterin, 2013). The bytecode is then executed using the *Ethereum Virtual Machine* (EVM).

Solidity (Ethereum-Foundation, 2021) is a statically-typed, high-level and contract-oriented programming language, designed for implementing smart contracts that run on the EVM.

Remix (Remix, 2021) is a powerful tool that allows its users to design, deploy and test smart contracts directly in the browser.

Metamask (Metamask, 2021) is a cryptocurrecy wallet that can be used as a browser extension on various existing browsers (e.g., Chrome, Firefox and Brave). It connects to the Ethereum blockchain and it facilitates testing. For this, we use *Ganache* (Suite, 2021) which provides a local Ethereum network.

React (React, 2021) and *Material UI* (Material-UI, 2021) are JavaScript libraries used for creating user interfaces for web applications.

Passport (Passport, 2021) is a JavaScript library, which provides a wide range of authentication strategies that can be embedded into web applications.

Provable (Provable, 2021) and *Chainlink* (Chainlink, 2021) are blockchain oracle services. An oracle service allows Ethereum smart contracts to access data sources outside the blockchain.

3 A DECENTRALIZED APPLICATION FOR RATING

A decentralized application (i.e., known as a DApp) is a web application that runs on a blockchain network, instead of a single computer. The logic of a decentralized application is represented by one or more smart contracts that interact with the blockchain network. In this section we present the decentralized application for rating. We start with the description of the workflow, which consists of the interaction between several components. Each component has a specific role (e.g., authentication, input validation, error handling, rating). Next, we discuss the smart contract, the authentication mechanism and the user interface.

3.1 Workflow

In Figure 1 we illustrate the workflow of our DApp.

First, the user needs to authenticate (step ①) with one of the available providers (i.e., Google, GitHub and Spotify). The authentication succeeds when the user has valid credentials. If successful, the authentication mechanism returns the user's ID (step ②). The ID is stored at the application level and it is used when the user initiates a rating operation.

Now, the user is allowed to rate internet resources

(step ③) based on their ID (step ④). In order to be recorded, the rating action needs to be valid. The input validation component performs two verifications (step ⑤): check that there is not an attempt of multiple rating (i.e., the user is allowed to switch from like to dislike and vice versa, but the attempts to submit multiple likes or dislikes for the same resource are forbidden) and assure that the resource is hosted by the provider chosen for authentication. If the requirements are satisfied, the process continues, otherwise a corresponding error is thrown (step ⑤).

Based on the validated input, the rating component interacts with the smart contract (step ⑥). The smart contract exposes a rating function, which has the role of storing the rating record on the blockchain. The function initiates a new transaction (step ⑦), which is sent across the network for validation. If the process succeeds, the state of the smart contract is updated with the new rating (step ⑧).

Finally, the user can access the updated rating record. The component responsible for the rating information interacts with the smart contract and retrieves the updated rating data (steps ⑨ and ⑩).

3.2 Smart Contract for Rating

The implemented smart contract simulates the behaviour of a *like/dislike-based rating system*. Therefore, the users can submit positive ratings (i.e., likes) and negative ratings (i.e., dislikes) for a specific resource. For the development process of the smart contract we used the contract-oriented language *Solidity*. We tested the functionalities of the smart contract using *Remix* and *Ganache*.

Both the users and the resources are uniquely identified. The ratings stored on the blockchain consist of tuples of the following form:

$$\langle \text{ user ID, resource ID, boolean value } \rangle,$$

where the boolean value indicates a positive or a negative rating. In order to interact with the smart contract, the users need to pay a fee. Our purpose is to reduce the rating costs as much as possible. To minimize the deployment and execution costs, we made our smart contract as slim as possible and put the complex logic in the web application. However, this simplification does not impact the accuracy of the rating operations.

3.2.1 Data Structures

The smart contract encapsulates a set of data structures, which facilitate the process of rating. Given that the data structures are public, the information stored inside the smart contract is also publicly available. There are two types of operations that can be performed on the data structures: read operations and write operations. When a user submits a rating, a write operation is performed and the rating is stored in the smart contract. For instance, if a user request information about the rating record, the decentralized application interacts with the smart contract and reads the data structure that stores the ratings.

Next, we provide a brief description of the most relevant data structures:

- `resourcesInformation` records the number of likes and dislikes of the rated resources.
- `ratedResources` contains tuples formed of the resource ID and a boolean value, where the boolean value indicates if the corresponding resource was previously rated or not.
- `ratingsInformation` contains tuples of the following form: user ID, resource ID and boolean value. Each time a rating operation is successful, this data structure stores the user ID, along with the rated resource and a positive or a negative rating value.
- `usersToResources` stores tuples of the following form: user ID, resource ID and boolean value. In this case, the boolean value indicates if the resource was previously rated by a specific user.

These data structures can be both externally and internally accessed. Based on the information stored inside these data structures, the decentralized application can read the rating record and, moreover, discover fraudulent behaviours.

3.2.2 Smart Contract Functionalities

The main functionalities of our smart contract are listed below:

- The users can rate positively or negatively (i.e., only one like/dislike per resource).
- The users can change their rating options for the previously rated resources.

The `rate` function handles accurately each possible use case of a *like/dislike-based rating system*. It is called each time a new rating operation is issued. A detailed workflow of the `rate` function can be inspected in Figure 2.

3.3 Authentication Mechanism

Our solution includes an authentication mechanism that guarantees the uniqueness of the users. In order to rate resources, the users need to have valid credentials for the providers that share the resources. Even if the decentralized application has access to the IDs

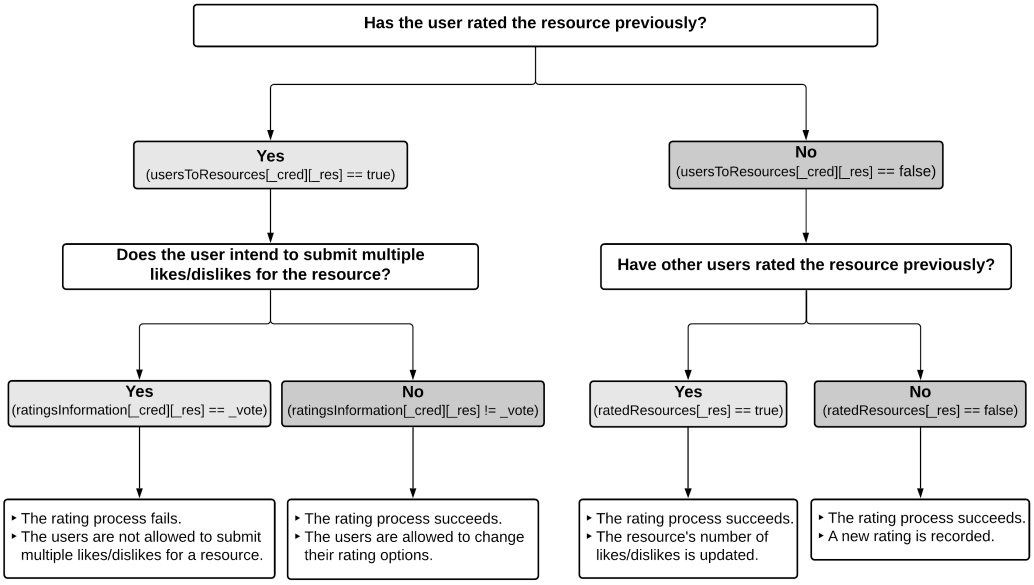

Figure 2: The diagram highlights the conditions that are evaluated within the rating function.

of the users (i.e., the IDs provided by the authentication mechanism), the IDs are not plainly stored in the smart contract, but they are previously hashed with `md5`. In this way, no one can find the credentials of the users by simply inspecting the transactions on the blockchain. Moreover, each call to the `rate` function includes a signature of the hashed ID, which is then checked in the smart contract. So, only the calls done through the DApp will be registered. Without this signature, anyone can call the smart contract with any value for the ID and rate a resource without being authenticated to an existing platform.

Our approach prevents the cases of multiple rating. For instance, if the Ethereum addresses of the users are used for authentication purposes as in (Shaker et al., 2021), then the situations of multiple rating can not be avoided. Since the users can generate an unlimited number of Ethereum addresses associated with their accounts, they can use a new identity each time they interact with the decentralized application, thus altering the rating record.

The advantage of our solution is precisely the additional authentication layer. Even so, the users can rate multiple times a resource, but they need to have multiple active accounts on the platform that share that resource. This is more difficult to achieve than the generation of Ethereum addresses, because the platforms themselves implement algorithms for detecting and removing fake accounts. Therefore, the rating process is trustworthy w.r.t. the existing accounts.

For the authentication mechanism we used *Passport*. In the current version of our application we provide support for the authentication with Google, GitHub and Spotify, but other authentication strategies can be easily added.

3.4 User Interface

The user interface of the decentralized application consists of three sections: authentication, rating and data visualization.

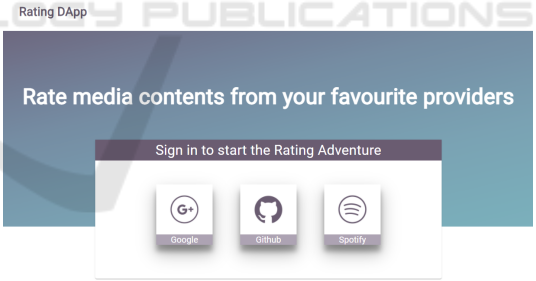

Figure 3: The authentication page.

In the authentication section (Figure 3), the user can choose one of the available authentication strategies: Google, GitHub and Spotify. The user clicks the desired authentication method and signs in with their own credentials. Once the authentication succeeds, the user can start to rate resources.

The rating section (Figure 4) encapsulates the components for input validation and error handling. The user needs to provide an URL of the resource and to select a rating option (i.e., like or dislike). Before the DApp calls the rating function of the smart contract, two input validation steps are performed:

1. Check the resource's provenience.

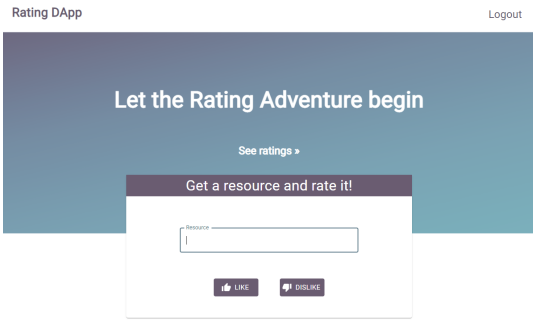

Figure 4: The rating page.

2. Check the rating record.

If one of the following situations occurs, the error handling module throws a corresponding error:

- If the user attempts to rate an invalid resource (e.g., the resource is hosted by another provider, the URL is not reachable), then the error handling module returns a notification message with the following content: *Invalid resource.*
- If the user attempts to submit multiple likes for the same resource, the rating process will stop and the error handling module returns a notification message with the following content: *Multiple ratings for the same resource are not allowed.* Analog for multiple dislikes.

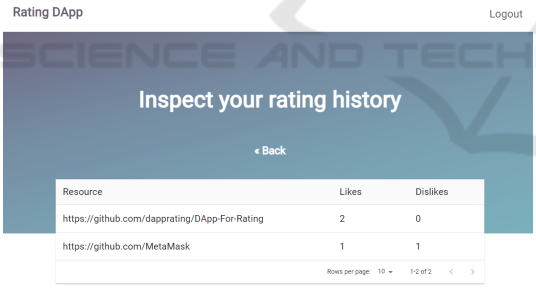

Figure 5: The data visualization page.

If all requirements are met, the rating process continues and *Metamask* informs the user w.r.t. the rating costs. When the user confirms the transaction, the rating function is called. The state of the smart contract will be updated after the transaction is mined. The URL-based resources are validated using API calls to the host services.

The data visualization section accesses the state of the smart contract, such that the user can inspect the rating record within the application.

We designed the decentralized application using *React* and *Material UI*.

4 TOOL EVALUATION & EXPERIMENTS

In this section we present an analysis of the costs required by our DApp. There are several categories of costs: deeployment costs, the costs to maintain the smart contract, and rating costs. The application presented so far was not our first attempt. We discuss below our approaches and we compare them.

Our initial approach was to encapsulate the entire rating logic in the smart contract. Since our rating system aims to avoid multiple rating situations and to validate the resources' provenience, the smart contract gained complexity. This led to increased execution costs and made our initial attempt unfeasible.

Moreover, smart contracts can not access information outside the network. Since we needed confirmations that the rated resources are hosted by the corresponding providers, we initially used *Provable* and *Chainlink*. This approach led to different kinds of problems. First of all, the deployment costs and the costs supported by users for the rating operations (i.e., a call of the `rate` function) increased considerably, because the off-chain requests require extra fees. The off-chain requests involved also the usage of custom cryptocurrencies (e.g., LINK for the *Chainlink*).

In order to stimulate the interest for the proposed rating system, we tried to minimize the costs[4]. Based on the observation that off-chain interactions are not suitable, we moved the authentication, input validation and other side logic outside the smart contract. The complexity of the smart contract is now considerably reduced. Except the minimal information (i.e, IDs, resources and their associated ratings) required to keep the ratings in a decentralized manner, the rest of the logic is directly handled at the application level. Thus, the off-chain dependencies were removed from the smart contract.

As expected, the smart contracts with external providers have higher deployment costs, due to additional functions which facilitate the off-chain communication. The smart contract with no external providers has significantly lower deployment costs. This is the one we have currently implemented[5].

The smart contracts which use external providers have a higher cost for the rating operation too. For instance, the rating operation for the smart contract which uses *Provable* is expensive, since the external

[4]These are computed in a testing environment provided by Remix, Metamask and the Kovan Test Net.

[5]The code is available on GitHub: https://github.com/dapprating/DApp-For-Rating/blob/main/contracts/Rating.sol.

Table 1: A comparison between our approaches: simple smart contract vs. smart contracts with external providers. We show both the deployment and rating costs.

Costs analysis		
Smart Contract version	**Deployment**	**Rating operation**
Simple smart contract (no external providers)	0.091400 ETH (1538986 gas)	0.003548 ETH (59754 gas)
Smart contract with Provable	0.218751 ETH (3683309 gas)	0.006162 ETH (44014 + 59754 = 103768 gas)
Smart contract with Chainlink	0.185437 ETH (3122376 gas)	0.008805 ETH (148268 gas)

requests charge the user with additional costs[6]. We estimate that the total costs for a rating operation for this type of smart contract is the sum between the costs for the off-chain interaction and the execution costs for the `rate` function. For the smart contract that uses *Chainlink*, the costs for the rating operation can be higher than the ones displayed in the table. This is expected, since the smart contract needs an amount of LINK[7] in order to execute external requests.

We conclude that our current implementation is the most efficient one w.r.t. the costs involved. The Github repository contains the source code and installation instructions. The smart contract code is written in `Rating.sol` under the `contracts` folder, while the UI is mainly in the `client` folder.

5 CONCLUSIONS

The solution we propose in this paper is a decentralized application that stores the ratings in the blockchain using a smart contract. In order to rate an internet resource, users need accounts on the platform that host that resource. Obviously, their accounts or login data are not stored in the blockchain, therefore we use their hashed value. Note that this approach keeps the benefits brought by the algorithms for eliminating or detecting fake accounts currently implemented in the existing media platforms. In this context, our rating application cannot accept more fake ratings than the existing approaches, as shown in Figure 6. The transparency and decentralization are guaranteed by the use of the blockchain.

To summarize, our approach provides a decentralized solution to keep ratings for internet resources identified by an URL. The approach is based on the blockchain technology that provides anonymity and immutability of data besides decentralization. We use authentication with third parties in order to set an inferior limit to the number of ratings. The limit is the number of accounts controlled by a user.

5.1 Future Work

A possible future work is to add more extensions, so that the users can authenticate with other accounts. Currently we support only Google, GitHub and Spotify, but our architecture allows any extension that is compatible with *Passport*. Another improvement is related to the various types of ratings. Now, our application only allows users to rate resources using a two choice (i.e., like/dislike) option. However, other rating systems allow scores or systems based on stars. Adding reviews is also an interesting feature, but this requires a very strong analysis on the costs involved to keep the reviews on the blockchain. One idea is not to keep the reviews on the blockchain, but only a hashed value of the review so that it can be easily checked if a review retrieved from a different resource is indeed the one whose hash is stored in the blockchain.

The solution that we are currently investigating is trying to keep the rating process out of the blockchain, and in the same time keep a trace of them in the blockchain. Basically, various platforms can connect their rating mechanism to the blockchain (i.e., to a smart contract) and submit a hash of the current status of the ratings. This hash should be exactly the hash of a ledger of their rating records. In order to check the authenticity of the data provided by a rating platform (e.g., YouTube), one can inspect the ledger of rating records provided by the platform, compute the hash, and compare it to the existing hash value stored in the blockchain. The advantage of this idea is twofold: first, one does not need to pay in order to rate a resource, and second, the interaction with the blockchain is significantly reduced. However, this idea requires a deeper investigation, because transparency and reliability might be affected.

[6]These can be inspected here: https://docs.provable.xyz/#pricing-advanced-datasources-call-fee.

[7]The exchange rate between LINK and ETH can be found at: https://currencio.co/link/eth/.

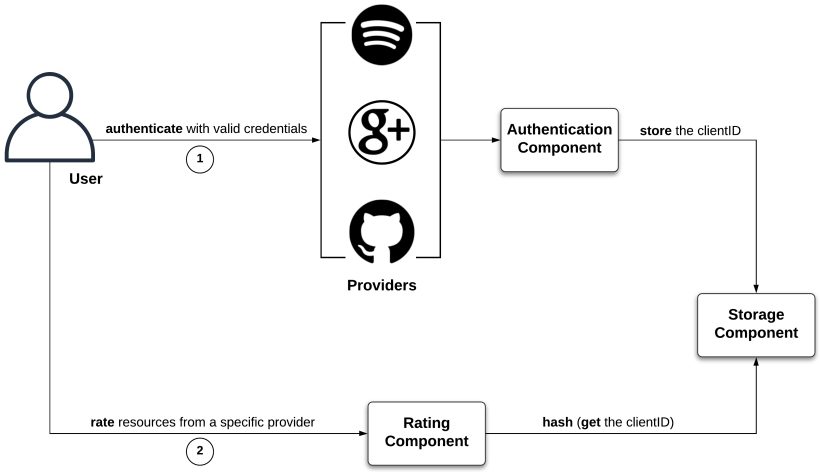

Figure 6: Authentication and rating. The users can not rate a resource until the authentication process is completed.

REFERENCES

Ayed, A. B. (2017). A conceptual secure blockchain based electronic voting system. *International Journal of Network Security & Its Applications*, 9:01–09.

Bocek, T., Peric, D., Hecht, F., Hausheer, D., and Stiller, B. (2009). Peervote: A decentralized voting mechanism for p2p collaboration systems. In Sadre, R. and Pras, A., editors, *Scalability of Networks and Services*, pages 56–69, Berlin, Heidelberg. Springer.

Buterchi, A. and Arusoaie, A. (2020). Dapp for rating. *ArXiv*, abs/2009.03253.

Buterin, V. (2013). Ethereum : A next-generation smart contract and decentralized application platform.

Chainlink (2021). Chainlink docs. https://chain.link/.

Ethereum-Foundation (2021). Solidity docs. https://docs.soliditylang.org/en/v0.8.0/.

Hsiao, J.-H., Tso, R., Chen, C.-M., and Wu, M.-E. (2018). Decentralized e-voting systems based on the blockchain technology. In Park, J. J., Loia, V., Yi, G., and Sung, Y., editors, *Advances in Computer Science and Ubiquitous Computing*, pages 305–309, Singapore. Springer Singapore.

Khader, D., Smyth, B., Ryan, P. Y. A., and Hao, F. (2012). A fair and robust voting system by broadcast. In Kripp, M. J., Volkamer, M., and Grimm, R., editors, *5th International Conference on Electronic Voting 2012 (EVOTE2012)*, pages 285–299, Bonn. Gesellschaft für Informatik e.V.

Khoury, D., Kfoury, E. F., Kassem, A., and Harb, H. (2018). Decentralized voting platform based on ethereum blockchain. In *2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET)*, pages 1–6.

Material-UI (2021). Material ui docs. https://material-ui.com/.

Metamask (2021). Metamask docs. https://docs.metamask.io/guide/.

Nakamoto, S. (2009). Bitcoin: A peer-to-peer electronic cash system. *Cryptography Mailing list at https://metzdowd.com*.

Passport (2021). Passport docs. http://www.passportjs.org/docs/.

Provable (2021). Provable docs. https://provable.xyz/.

React (2021). Github react. https://reactjs.org/docs/getting-started.html.

Remix (2021). Remix docs. https://remix-ide.readthedocs.io/en/latest/.

Shaker, M., Aliee, F. S., and Fotohi, R. (2021). Online rating system development using blockchain-based distributed ledger technology. *ArXiv*, abs/2101.04173.

Suite, T. (2021). Ganache docs. https://www.trufflesuite.com/docs/ganache/overview.

Wood, G. (2014). Ethereum: a secure decentralised generalised transaction ledger. https://gavwood.com/paper.pdf.