Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

Smart Contract Implementation for milestone based digital contract and financial transactions in the Sponsee Application

Arpit Shukla | Joel Lo Zhao Cheng | Yan Chengxun Rupert

[1]Sponsee Pte Ltd, Singapore
(arpit@sponsee.sg), (joel@sponsee.sg),
(rupert@sponsee.sg)

Abstract

The aim of this paper is to introduce and examine the ways in which smart contracts are used to overcome workflow obstacles in the Sponsee application, a free-for-all influencer monetization platform. Through the integration of 3 unique smart contracts (token factory, registration contract & digital agreement contract), Sponsee has managed to circumvent key issues in dispute resolution, contract legalities and financial transaction fees, to create an influencer based economy.

KEYWORDS:
Social Media, Blockchain, Sponser, Sponsee, Smart Contract, IPFS, Gig Agreement

1 | INTRODUCTION

The Sponsee is a cross platform mobile and web application that facilitates organic connections between social media influencers and businesses around the world. Influencer marketing is a thriving and growing industry that adds value to both traditional and modern businesses[1]. However, a large portion of small and medium enterprises do not utilize social media influencers to market their services or products[2]. This can be attributed to a high barrier of entry in the form of :

1. Not knowing which social media platform to use
2. No benchmark on the return of investment per influencer shoutout
3. No rating system
4. No assurance on delivery of media content to justify pre payment of services
5. No reference on rates per shoutout
6. No information on an influencer's willingness to participate
7. High cost per brand deal for accessible influencers (large follower count)

In more detail, the Sponsee application is a fee-free peer-to-peer marketplace that removes the traditional middleman (in influencer marketing agencies) and allows for influencers and businesses to converse directly with the aim of establishing a brand deal or sponsorship deal. Additionally, Sponsee contains a full suite of features designed to enhance and compliment the influencer marketing process for both sponsors (businesses) and "sponsees" (influencers). These include:

1. Cross social media platform filtering and sorting system
2. Direct access to listed brand deals and jobs up for browse

3. Rating system for both sponsors and sponsors
4. Digital contract system to facilitate brand deals and shoutouts
5. Fully transparent financial transaction system
6. Dispute resolution
7. Decentralised Governance

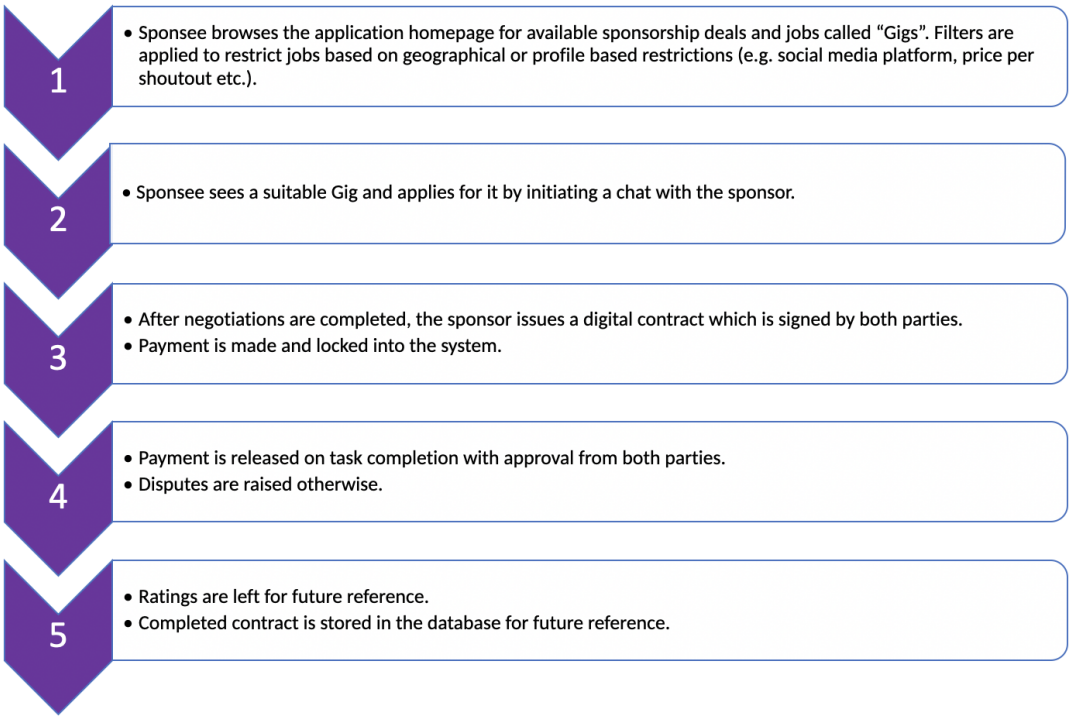

FIGURE 1 Simplified workflow for sponsees

Another problem faced in the traditional influencer marketing landscape is that it is a largely mega-influencer dominated economy. The difference in revenue generated per shoutout between micro/nano-influencers and large influencer celebrities is not linear to the increase in their respective outreach[3]. This means that larger influencers are overcharging businesses for brand endorsements while smaller businesses with less financial backing do not have access to the influencer group more suited for their needs. In summary, Sponsee hopes to change the nature of this economy, by connecting smaller sized influencers with smaller businesses.

Through the use of a full suite of features as mentioned above, Sponsee's vision is to be the one stop solution platform for all influencers and businesses. Influencers with a low follower count should be able to sieve out and connect with appropriate businesses within their budget range and likewise for smaller businesses.

The workflow for the use of the Sponsee application can be represented with the following flow diagrams in figures 1 and 2.

2 | PROBLEM FORMULATION

The Sponsee application faces several challenges in its workflow and API based implementation of certain features.

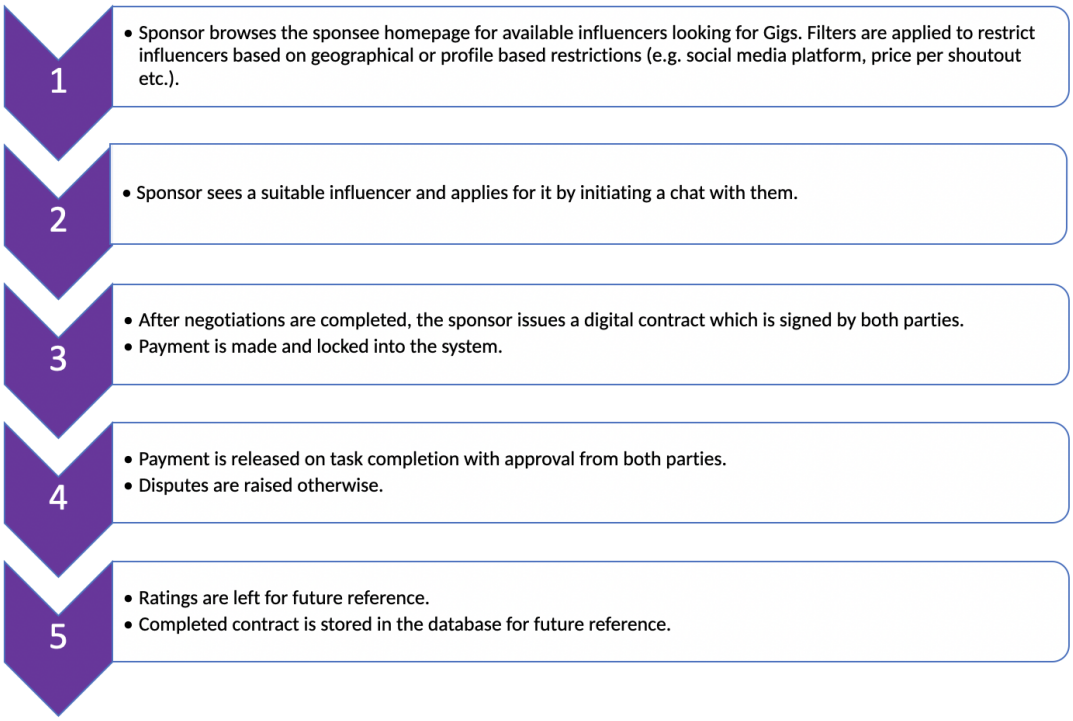

FIGURE 2 Simplified workflow for sponsors

2.1 | Digital Contract

First, a major problem surfaces in the formulation of a digital contract between the sponsor and sponsee. With the generation of a pdf based digital contract, there is little legal weight to the terms and conditions agreed by both parties. Essentially, once payment is made, there is no mechanism to prevent the Sponsee from breaking the contract. A third party solution proposed was through an intermediary such as Docusign. However, since the Sponsee app caters to an international audience, legally binding contracts are tedious to verify and hard to enforce with little to no consequences on breakage.

2.2 | Dispute Resolution

Second, the dispute resolution process is entirely subjective in this traditional application workflow. When disputes are raised, the details of the dispute are revealed to the administrators or to a third party dispute resolution service. This introduces the element of human error, and is forecasted to cause dissatisfaction to the party losing the dispute. As such, this motivates the need for an objective system of dispute resolution that does not rely on human judgement.

2.3 | Financial Transactions

Third, the traditional financial transaction process between sponsors and sponsees is a trust based system. Standard practice requires sponsors to make payment for shoutouts before the task is completed. This is an inherently flawed practice. Sponsors face a high barrier of entry to influencer marketing because of the uncertainty in the results sponsees might provide. Sponsees also have little to no incentive to do a good job of their shoutout. This prompted the use of smart contracts to create a decentralised method of rewarding sponsees based on performance, which shall be elaborated on in the later section.

2.3.1 | Data Storage & Extraction

Fourth, the storage and extraction of media content and digital contracts is an expensive task, due to high frequency and high bandwidth communications with our server. Even with compression and caching, it is not sustainable when scalability is factored

into the growth of user base. This motivated the use of the IPFS peer to peer network system for storage of data and extraction through blockchain hash keys.

2.4 | Lack of Complexity

Fifth, the traditional workflow of the Sponsee application does not allow for complex and conditional agreements to be made between parties involved. It does not allow for partial/conditional reward tiers based on milestones completed. The use of smart contracts changes this as it allows for complex agreements to be subdivided into subtasks and validated on the blockchain in a procedural manner.

2.5 | Centralised Governance

Sixth, centralised management of platforms is one of the biggest obstacle when it comes to transparency and trust. Due to the centralised governance, the users are forced to trust and believe in the authorities decision. This also limits the power of the end users as they are unable to express themselves in such a centralised system. The use of decentralized autonomous organization (DAO) in the Blockchain Technology serves as a solution to this problem by providing the end users and the community power for decision making and provide their comments in the decision making for the ecosystem.

2.6 | Validity & Security

The final and largest problem faced with the traditional application implementation is the lack of measures in place to ensure ratings, accounts and agreements are verified and recorded without possibility of interference. For example, scammers might use bots to increase the rating of a sponsee account to receive jobs without actually fulfilling the contract in the end. Contract details are also centralised on the Sponsee server, and are inaccessible to the public which raises questions about transparency. The use of blockchain technology solves all these issues as it provides a concrete and transparent manner of data and contract storage with little interference or tampering possible, once the smart contracts involved are written per negotiation.

3 | SYSTEM ARCHITECTURE

The technical workflow of Blockchain Technology in Sponsee can be visualised by the System Architecture model in Fig. 3. This section provides description of procedure and steps involved to propose the blockchain based solution and communicate with blockchain using the Sponsee mobile application.

3.1 | Non - Custodial Wallet Integration

A cryptocurrency wallet is a machine [5], physical medium [6], software, or service that holds public and/or private keys used in cryptocurrency transactions. A cryptocurrency wallet also has the capability of encrypting and/or signing data in addition to the basic feature of holding keys. Signing will lead to the execution of a smart contract, a crypto exchange, authentication, or the legitimately signing of an 'article'. Every transaction in the Blockchain gets signed using these keys stored in the wallet to authorise the account from which the transaction is initiated[4].

A non-custodial trading platform is a cryptocurrency exchange system where only the customer has complete leverage of their wallet. People with their own wallets have complete discretion of their tokens, passwords, and keys, since no single authority stores their passwords, keys, or coins. Non-custodial wallets can take several different forms, including smartphone apps, web applications, and browser extensions. Non-custodial trading sites will connect easily to non-custodial wallets by pressing a quick "connect" icon.

All the Sponsors and Sponsees using the application will be provided with a non-custodial Crypto Wallet. This wallet will enable users to communicate with the Smart Contracts deployed on the Binance Smart Chain (BSC). The Non-Custodial Wallet will be integrated with the front-end application.

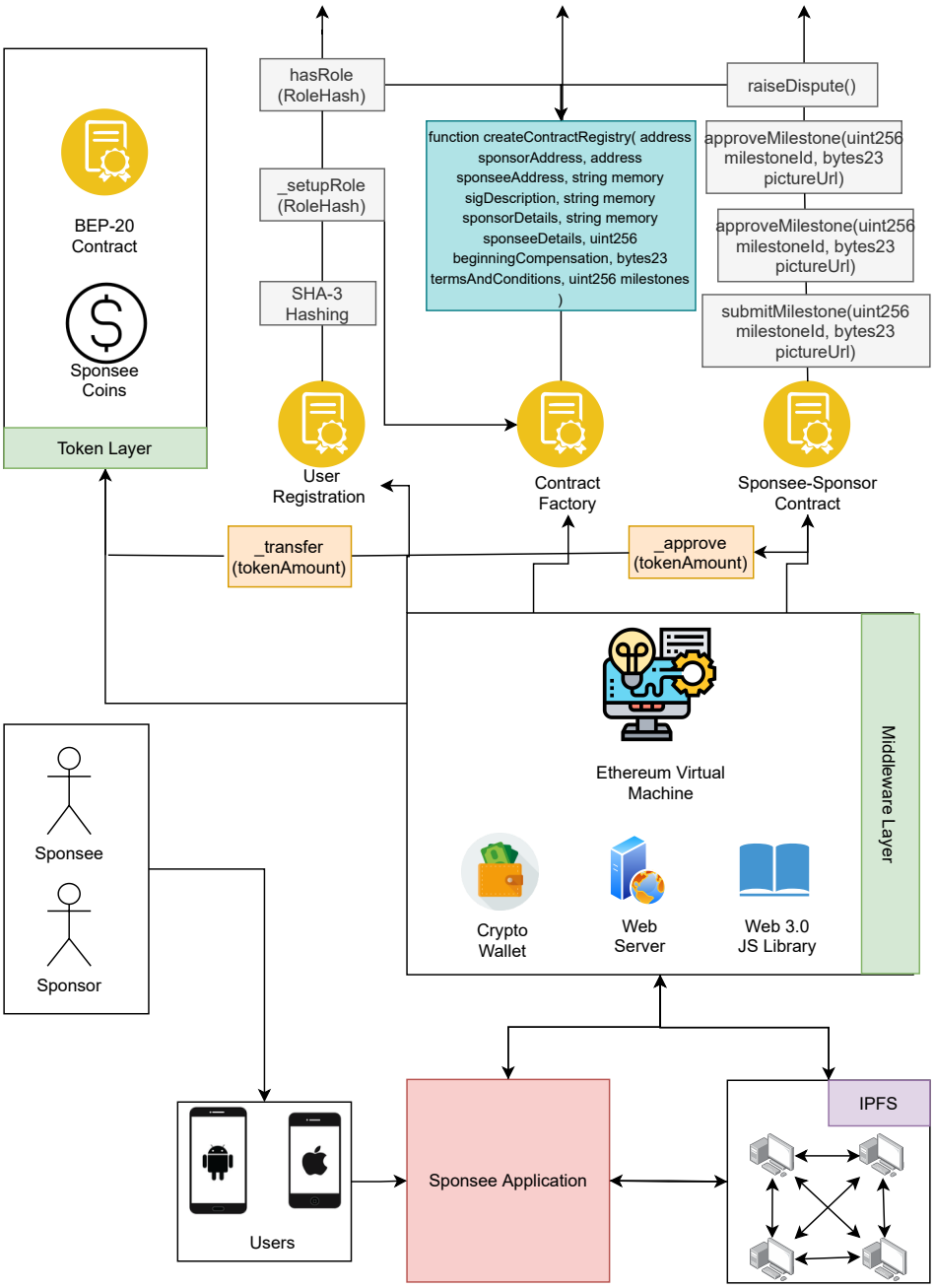

FIGURE 3 Sponsee System Architecture

For the non-custodial wallet integration, the procedures involved for the experimental setup are setting up the web3 provider, network configuration to set up RPC connections, using the wallet X modal for sending the transactions and using the non-custodial signature requests custom UI for signing the transactions and data. The specification of the network provider will be done using the provider javascript SDK. Moreover, the swapping of existing web3 provider ID is done with custom non-custodial wallet service providers, and then continues developing using the same version of web3 to be utilised. Then, the custom node switching is done by changing the network configurations as visualised in the following pseudocode:

For connecting the users, wallet sign-up is implemented. This authentication can be done in various ways. The login modal is made robust using the EIP-1102 industry standard defined by the blockchain researchers in Ethereum documentation. EIP-1102 defines a network communication across decentralized applications and EVM-enabled DOM ecosystems that enables the Ethereum-enabled DOM environment to decide what data to provide the dapp and how often.

Algorithm 1 Network Configuration with Non-Custodial Wallet

Input: $RPC_{url}, Chain_{id}, Custom_{api}$
Output: Connection of Blockchain Node with non-custodial wallets through RPC calls.
Initialization: Here, web3 is imported from the npm library and set according to the environment in which the app is used. Also, 3rd party non-custodial wallet library is imported and integrated in the backend.

```
import Service from 'Non-Custodial-Provider';

import Web3 from 'web3';

const BSCOptions = { rpcUrl :' https : //bsc − dataseed.binance.org/', chainId : 56 // Smart Chain mainnet chain id }

// Setting network to Smart Chain

const fm = new Service('CUSTOM_API_KEY', BSCOptions);

window.web3 = new Web3(fm.getProvider());

```

3.2 | User Registration Smart Contract

The user registration smart contract is used to serve the purpose of user onboarding and registration. Moreover, it is also used to include several features like calculating the crypto hash to create unique identifiers for mapping the stakeholder categories. The role based access control mechanism is implemented for governance within the system and to include the constraints to limit smart contract functionalities. The code reusability is done by importing standard libraries such as open-zeppelin functions.

Although the ease of acquisition is helpful for basic structures or rapid prototyping, various types of approval are often required. A user account can be allowed to bar users from a scheme, but it cannot produce new tokens. In this respect, Role-Based Access Control (RBAC) provides stability.

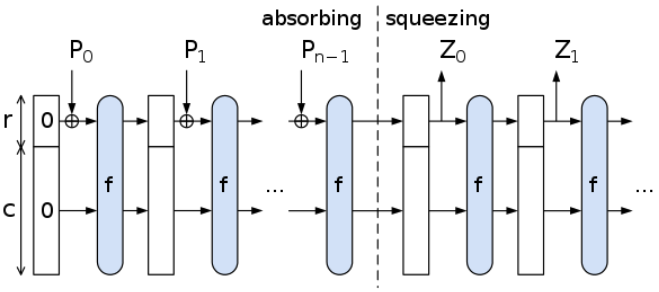

FIGURE 4 Construction of sponge function to calculate the hash[5]

In essence, we define various functions, each of which is able to execute separate sets of actions. Instead of using onlyOwner anywhere, we use onlyAdminRole in some areas and onlyModeratorRole in others. Sponsee developers would be able to specify regulations according to how identities should be allowed to execute a function.

The majority of software production employs role-based access management systems: some clients are casual users, others may be supervisors or administrators, and a few may have managerial rights. Roles are provided by Sponsee Pte Ltd by applying role-based access management. Its use is simple: for each role that you choose to specify, you can store a parameter of type Role which holds a statement of account that has that role.

The procedure to assign a new role, i.e., Sponsee Role / Sponsor Role to a particular user, has a crypto-hash that is calculated using the keccak256 hash function. NIST launched Secure Hash Algorithm 3 on August 5, 2015, as the newest member of the Secure Hash Algorithm family of standards[6 7]. While part of the same set of specifications, SHA-3 differs from the MD5-like form of SHA-1 and SHA-2 internally[8]. SHA-3 is a subset of the larger cryptographic primitive family Keccak, which was developed by Guido Bertoni, Joan Daemen, Michal Peeters, and Gilles Van Assche, and is based on RadioGate[9]. The researchers of Keccak also suggested alternate functions for the feature that have not (yet) been standardised by NIST, such as a stream

cypher, an authenticated encryption method, a "tree" hashing mechanism for accelerated hashing on some architectures, and the AEAD cyphers Keyak and Ketje [10].

Keccak is built using a novel technique known as sponge building. Sponge construction is based on a large nonlinear system or random permutation, and it allows inserting ("absorbing") and extracting ("squeezing") any volume of data when serving as a pseudo - random function with respect to every previous input [13]. As a result, there is a lot of leeway. For limited message sizes, the developers of the Keccak implementations and the SHA-3 functions recommend using the smoother feature KangarooTwelve with modified parameters and a latest tree hashing feature with no additional processing [11].

In the Sponsee application, DEFAULT ADMIN ROLE is a key role throughout the AccessControl mechanism that serves as the primary administrative position across all functions. Unless _setRoleAdmin has been utilised to pick a new admin position, a wallet for this role would be capable of handling some other role. The user registration smart contract then stores this hash in the form of bytes in the blockchain storage. The following line shows the example to calculate the Sponsee_Role SHA-3 hash [12]:

Algorithm 2 SHA-3 crypto hashing for role-based governance in Sponsee

Input: $E, License, CR_i, Score_i^k$
Output: Farmer credit purchase successful or not
Initialization: Here, AFRMP are available in the marketplace and have their prices registered over SC. Here, *sender* is the company.

```
1:
2: bytes32 public constant SPONSEE_ROLE = keccak256("SPONSEE_ROLE");
3:
4: // Grant the sponsee role to a specified account
5:
6: _setupRole(SPONSEE_ROLE, sponsee);
7:
8: // Check that the calling account has the sponsee role
9:
10: require(hasRole(SPONSEE_ROLE, msg.sender), "Caller is not a sponsee");
11:
12: bytes32 public constant SPONSOR_ROLE = keccak256("SPONSOR_ROLE");
13:
14: // Grant the sponsor role to a specified account
15:
16: _setupRole(SPONSOR_ROLE, sponsee);
17:
18: // Check that the calling account has the sponsor role
19:
20: require(hasRole(SPONSOR_ROLE, msg.sender), "Caller is not a sponsor");
21:
```

3.3 | BEP-20 Spon Token Smart Contract

The $SPON tokens are minted using the custom BEP-20 smart contract. BEP-20 is the Binance Smart Chain token standard that extends ERC-20. BEP-20 has emerged as a technological benchmark; it could be used for token execution across all smart contracts upon the Binance Smart Chain and includes a set of guidelines that all BSC-based tokens should obey. The BEP-20 proposes a specification for Fungible Tokens, which would have the feature of being precisely the same (in type and value) as another Token. For instance, an BEP-20 Token functions similarly to ETH in that one Token has and always will be equivalent to certain other Tokens.

BEP-20 tokens are also blockchain-based properties with values that can be submitted and obtained. The primary distinction is that BEP-20 tokens are distributed on the BSC network rather than any private blockchain.

The $SPON tokens will be the fuel for the system workflow and serve as the economical perspective of the contract cycle. The contract agreement between the sponsee and sponsor involves the price finalised after the completion of the negotiation phase. This compensation provided to the sponsee after the completion of a successful gig will be done using these BEP-20 tokens. Once the contract is finalised and digitally signed by both the parties after the negotiation, the smart contract will be initialised, a new contract address is created for each agreement between the stakeholder. Now, the sponsor needs to provide the tokens with the same value of the price agreed upon and stored in the blockchain through the sponsee - sponsor smart contract. These tokens will be put on a hold state into the contract wallet until the lifecycle of the contract gets finished. Once the gig is

completed successfully, then the tokens will be released from the smart contract into the wallet of the sponsee. This makes the economic management of the sponsor-sponsee contract autonomous, immutable, transparent and decentralised.

The BEP-20 token smart contract includes all the standard functions that are verified by the BEP-20 interface. Following are the functions included along with their ip / op :

Methods

- function name() public view returns (string)
- function symbol() public view returns (string)
- function decimals() public view returns (uint8)
- function totalSupply() public view returns (uint256)
- function balanceOf(address _owner) public view returns (uint256 balance)
- function transfer(address _to, uint256 _value) public returns (bool success)
- function transferFrom(address _from, address _to, uint256 _value) public returns (bool success)
- function approve(address _spender, uint256 _value) public returns (bool success)
- function allowance(address _owner, address _spender) public view returns (uint256 remaining)

Events

- event Transfer(address indexed _from, address indexed _to, uint256 _value)
- event Approval(address indexed_owner, address indexed _spender, uint256 _value)

3.4 | Agreement Contract Factory Smart Contract

The Agreement Contract Factory of the Sponsee Application manages the initialisation of every sponsee-sponsor agreement contract. It functions as the registry that is used to register the data and then accordingly generate a new smart contract for every gig.

The application front-end interface will be used to provide all the input data from the contract creation form that is entered by the end-users (i.e. sponsees and sponsors). Following are the data to be included in the registry to initialise the sponsee-sponsor agreement smart contract:

- Gig description
- Sponsor Details
- Sponsee Details
- Final Compensation
- Terms and Conditions
- Milestones

The gig description input would contain the details involved with regards to the task to be done, as well as if there are certain conditions to be made and other rules and regulations regarding the task. The sponsor and sponsee details would involve basic details such as full name, username, contact details, etc. The final compensation amount is keyed as an integer value into the contract to make it stored immutably over the distributed ledger of BSC. This amount is by default valued in global fiat currency (like US Dollars) which will eventually be converted to the $SPON token valuation in the back-end using a dynamic price fetching API from the supported crypto-exchanges of Sponsee. An approval implementation will trigger between the Sponsor and the contract address where the Sponsor will approve to transfer to the Contract address the agreed sum.

The terms and conditions represent the extra details and clarifications that are required to be mentioned and agreed upon by both the stakeholders before starting the task. The terms and conditions between the two parties will be saved in the IPFS (Inter Planetary File System) and correspondingly in the Blockchain as Uri. Finally, the milestones are the most crucial part of the whole task life-cycle. The milestones primarily represent the sub-tasks involved to complete the entire task so that the Sponsee can successfully receive the full compensation. This will be further discussed in detail in the Section 5.5 as drafted below.

3.5 | Sponsee - Sponsor Agreement Contract

The sponsee-sponsor agreement smart contract is the most crucial smart contract in the system workflow. Once, the final price and other terms and conditions are finalised between both the parties, both parties need to provide their approval, and a new instance of this smart contract is created with all the details entered in its constructor.

Now, once the smart contract is created, an approval implementation will trigger between the Sponsor and the contract address where the Sponsor will approve to transfer to the Contract address the allocated sum. This approval will be verified to make sure that the amount approved matches the final compensation price to be paid to the sponsee on successful completion of the task.

On successful verification, the sponsee will be notified to begin performing the task to be done on a milestone basis.First, the sponsee will perform the initial milestone and eventually jump to the next one in an order mentioned in the smart contract. On the completion of a milestone, the sponsee will update the status to milestone completed along with the proof attached. The sponsor will on the other hand be notified of the milestone being completed. Now, the sponsor will verify its completion and if everything is fine then, the sponsor will sign the milestone to be completed. This signature will be updated on the smart contract and the transaction will be stored immutably on the blockchain which can be seen and verified anytime on a later stage if needed.

At the end of the contract if every milestone is done and approved by the sponsor, 100% of the Price/Reward will be transferred to the Sponsee, else if a percentage of the milestones is done at the end of the contract, then that percentage will be multiplied with Price/Reward and transferred to the Sponsee. After the agreement between the two parties ends, if everything is fulfilled then the contract will be stated as completed. A transferFrom function will trigger to transfer the Price/Reward from the smart contract to the Sponsee address. This is the workflow if all goes successful.

The dispute resolution module of Sponsee is activated when there is some error in the ongoing task initiated by either the Sponsee or the Sponsor. In such cases, blockchain technology is used to partially automate the dispute resolution process by verifying the digital signatures to determine successfully verified milestones. The amount valued for the completion of the verified milestones is first released to the sponsee to make sure that the sponsee at least gets the compensation of what all has been done. The representative of the dispute resolution team of Sponsee analyses the actual milestone on which the issue had occurred. The committee provides a decision and resolves the dispute by manually verifying the proof provided by both the party and checking real time the sub-task done by the sponsee. Depending on the result by the committee, the one who cheated is charged with the fine and the contract is closed.

In any case, once the contract gets finished, either successfully or unsuccessfully, both the sponsor and sponsee need to rate their experience of the opposite party. These ratings according to the reviews will update the aggregated rating score of both of them. Hence, the ratings will be stored on the blockchain to keep it secure as well as transparent from other alterations or illegal manipulation of ratings by bots in the system. The contract will be stated in the Registry contract as completed and will be saved in mapping for later queries.

3.6 | Communication Layer

The SPON token will be used as a layer of communication between the stakeholders of the ecosystem like the development team, community, end-users as well as 3rd party companies in the future.

The decentralised governance will be implemented by using SPON tokens. After deployment of SPON, its control would be handed over to the ecosystem once its established using a Multi-Sig wallet. Every $SPON token holder (by staking some min tokens) can pitch the idea, suggestion, new plan etc into the ecosystem. There will be a period of 7 days for example where there can be open discussion over community, devs, and all the stakeholders. During these 7 days there will be an open vote where the public can stake their tokens (with a minimum value) and then vote for that proposal or against it. After 7 days, the results would be announced and changes will be made accordingly in the roadmap. After the vote, using Multi-Sig wallets (eg., 3 multi signers: Dev team, community and core team), the changes will be signed and stored into blockchain. However, we also

plan to add veto power for emergencies to reject a decision that has been voted successfully but is harmful for the future of the ecosystem and this veto power can only be used by multi-sig holders.

4 | CONCLUSION

The novel use of smart contracts to facilitate peer to peer digital contracts and agreements is unique to the Sponsee application, and definitely one of its kind. Through harnessing blockchain technology, Sponsee is able to overcome obstacles faced in traditional workflow processes, ranging from dispute resolution to secure and bot-free ratings. This puts Sponsee on the front-lines of technological change and serves as a prime example for other organizations to follow suit. As the use of blockchain technology evolves, there is a transition from simply using the blockchain as a means of data storage and transparency. The next natural step in Sponsee's research would then be the use of smart contracts for passive revenue generation purposes. In time to come, this would be another powerful tool added to Sponsee's arsenal.

References

1. Wielki J. Analysis of the Role of Digital Influencers and Their Impact on the Functioning of the Contemporary On-Line Promotional System and Its Sustainable Development. *Sustainability* 2020; 12(17). doi: 10.3390/su12177138
2. Kadekova Z, Holienčinová M. Influencer marketing as a modern phenomenon creating a new frontier of virtual opportunities. *Communication Today* 2018; 9: 90-104.
3. Hughes C, Swaminathan V, Brooks G. Driving Brand Engagement Through Online Social Influencers: An Empirical Investigation of Sponsored Blogging Campaigns. *Journal of Marketing* 2019; 83(5): 78–96. doi: 10.1177/0022242919854374
4. Cryptocurrency wallet. *Wikipedia* 2021.
5. Sponge function. 2020. Page Version ID: 962234349.
6. EIP-20: ERC-20 Token Standard. .
7. How to send bitcoin to a hardware wallet. .
8. Before Buying Bitcoin, Choose Where to Store It. Here Are Some Crypto Wallet Options. .
9. swenson . NIST Selects Winner of Secure Hash Algorithm (SHA-3) Competition. 2012
10. Chang SH, Perlner RA, Burr W, et al. Third-Round Report of the SHA-3 Cryptographic Hash Algorithm Competition. In: ; 2012
11. Dworkin MJ. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. 2015.
12. Computer Security Division ITL. Hash Functions | CSRC | CSRC. 2017.