Towards Web3 Applications: Easing the Access and Transition

GUANGSHENG YU, Data61, CSIRO, Australia
XU WANG, GBDTC, University of Technology Sydney, Australia
QIN WANG, Data61, CSIRO, Australia
TINGTING BI, Data61, CSIRO, Australia
YIFEI DONG, GBDTC, University of Technology Sydney, Australia
REN PING LIU, GBDTC, University of Technology Sydney, Australia
NEKTARIOS GEORGALAS, Applied Research, British Telecom, UK
ANDREW REEVES, Applied Research, British Telecom, UK

Web3 is leading a wave of the next generation of web services that even many Web2 applications are keen to ride. However, the lack of Web3 background for Web2 developers hinders easy and effective access and transition. On the other hand, Web3 applications desire encouragement and advertisement from conventional Web2 companies and projects due to their low market shares. In this paper, we propose a seamless transition framework that transits Web2 to Web3, named Web3Communicator, after exploring the connotation of Web3 and the key differences between Web2 and Web3 applications. We also provide a full-stack implementation as a use case to support the proposed framework, followed by performance evaluation and surveys with ~1,000 participants that show ~80% positive and ~20% neutral responses. We confirm that the proposed framework Web3Communicator addresses the defined research question, and the implementation well satisfies the framework Web3Communicator in terms of strong necessity, usability, and completeness based on the survey results.

CCS Concepts: • Information systems → Web interfaces;

Additional Key Words and Phrases: Web3, Web2, DApp, Blockchain, Service Management System

1 INTRODUCTION

Web3 has drawn intensive attention from communities and investors. As an umbrella term, Web3 covers a series of blockchain-based decentralized applications (DApps), services, and economies [1–5] that bring significant impacts on both traditional finance and cryptocurrency markets. To date (as of Mar 2023), over 12,143 DApps have been developed on-chain and 285,152 smart contracts are deployed across 48 protocols. A total of 1.67M users are actively interacting with the smart contracts within 24 hours, as evidenced by their wallet addresses. In this sense, Web3 impresses users by providing such a connect the wallet button on the upper-right corner of each webpage. Users can use DApps through embedded wallet entries by invoking specific functions that are deployed on blockchain-engined platforms (e.g., Ethereum [6]). The shift of backend servers from centralized clouds to decentralized chains has mostly distinguished Web3 and previous web styles.

1Web3Communicator stands for Web2 (Two)–Web3 (Three) Communicator.
2Data source [Mar 2023]: DappRadar https://dappradar.com/industry-overview.
Following its narrative connotation, we observe that Web3 users still occupy a pretty small percentage (appr. 0.03% of 4.95B Internet users) over the Internet. The constraints majorly come from its incompatibility: (i) a typical Web3 application cannot be smoothly applied in a traditional web context due to the absence of blockchain engines; conversely, (ii) a traditional Web2 application can hardly be integrated with blockchain due to the lack of proper Application Programming Interfaces (APIs). Rational developers would start their work on wide-adoption applications, namely Web2 Apps, for higher user exposure and more potential revenues rather than sparing efforts on Web3 applications that are uncertain. Such concerns motivate this work:

How to ease the usage of Web3 applications and achieve a smooth transition of applications between the Web2 and Web3 space?

We investigate the bottlenecks of the transition between Web2 and Web3 by diving into their distinctions in design and implementation. For most Web2 applications, user management, data manipulation, and private-preserving policies are constructed upon centralized databases in a closed manner. In contrast, Web3 DApps usually apply asymmetric-encryption-based identity to establish more secure and robust user management in a decentralized manner among many parties without prior trustworthiness [7–10]. Data manipulation differs from that of Web2 applications due to the immutability of data storage in Web3, and access control also requires new approaches to partition the visibility. In addition, existing Web3 DApps lack flexible mechanisms to smooth the workflow of documenting Restful APIs and test suites during the development phase. Therefore, the incompatible user management strategies, execution procedures for data manipulation, private-preserving policies, and the lack of Web3-compatible API tools have greatly retarded the smooth transition of applications in different domains.

To fill the above gaps, in this work, we propose a practical solution to integrate both Web3 and Web2 applications and related services seamlessly. We deconstruct the Web2 architecture and extract three major components: frontend APIs, backend servers, and supplementary databases. Aligning with the core principles of Web3, we accordingly modify these components to enable seamless integration with blockchain engines. Specifically, we design three types of adjustable components: a SaaS module for integrating Web2 software by providing generalized APIs, a backend interpreter that interprets and forwards requests from Web2 to Web3, a blockchain layer that configures self-governed policies via smart contracts as well as computes on-chain calculations through chain Software Development Kits (SDKs). Our proposed solutions can greatly promote the transition from classic Web2 applications to the Web3 space without redundant development or complicated middleware. In short, we highlight the contributions as follows.

- We explore the connotation of Web3 by investigating plenty of in-the-wild Web3 projects. As a new concept, we compare existing Web2 solutions and so-called Web3 projects to determine the root features of Web3 applications and their dependencies, extracting their differences from classic Web2 applications.
- Based on comprehensive investigation, we propose a seamless transition framework that transits Web2 to Web3, named W/ebTCom. Our proposed solution establishes an interpreter to bridge the Web2 applications and the Web3 backend engines. In particular, the proposed framework WEBTCom can offer effective and reliable access control and user management across the decentralized Web3 and centralized Web2 and provide an approach to conduct transition with existing popular SaaS and the framework. While operating, WEBTCom can also effectively improve production by automatically generating API documents for developers and communities.

\[^3\text{Data source: Digital 2022: Global Overview Report}\ ] https://datareportal.com/reports/digital-2022-global-overview-report.
• We provide a full-stack implementation ranging from front and backend APIs, structure design, and smart contract programming to the on-cloud dockerized deployment. Code size reaches up to 11,351 lines\textsuperscript{4}. The system is applied to daily service management processes and promotes the establishment of effective, flexible, reliable, and trustworthy Web3-driven applications. Our prototype has been inspected by British Telecom (BT) in practice.

• We further conducted a quantitative performance evaluation and a qualitative evaluation from the perspectives of managers, investors, and developers through surveys targeting 1,000 practitioners. The survey feedback shows that the research question is well satisfied by the proposed framework WEBTTCOM, and the presented full-stack implementation proves its strong capability of smoothing the transition in terms of necessity, usability, and completeness.

The remainder of this paper is organized as follows. Section 2 provides the Web3 basics. Section 3 proposes the research methodology and presents our research question. Section 4 introduces a use case that implements the new pattern, followed by discussions about the implementation notices and limitations shown in Section 5. Section 6 provides existing studies related to this work. Section 7 concludes this paper and highlights our contributions.

2 APPROACHING WEB3: A PRELIMINARY

In this section, we show the Web3 basics by comparing with Web2 and providing a typical Web3 instance.

2.1 Differences between Web 1/2 and Web3

Traditional Internet, including so-called Web1 and Web2, has been developed for decades. Web1 is regarded as a suite of read-only protocols that contain static sites to present images, text, and videos. Users search for the targets by accessing web portals. Web2 changes the way of interaction by enabling user-generation content (UGC). Users can publish their original content, such as images, reviews, testimonials, or even podcasts, on social media websites (e.g., Facebook, Twitter). In this sense, Web2 is regarded as read-write. Web3 differs from previous styles by adding features of ownership and transfer. Users will create self-controlled accounts, generally in the forms of wallets, to manage digital assets and virtual data. Rather than relying on centralized servers, Web3 users can freely transfer their assets under the governance of smart contracts, which brings the advantages of auto-execution, being accountable, and being globally verified. These smart contracts connect both upper-layer DApps and underlying blockchain platforms.

| Functions | Architecture | Instance |
|-----------|--------------|---------|
| **Web1** | read         | client-server | Yahoo |
| **Web2** | read/write   | client-server | Facebook, Google |
| **Web3** | read/write/own/transfer | client-SC-chain | Ethereum, BSC |
| **Web2→3** | read/write/own/transfer | client-Int.-SC-chain | WEBTTCOM |

\textsuperscript{4}Specifically, we present the detailed size distribution. The front end takes 4 MB with image resources, while 253 KB with code only (5367 lines). The backend is a 1.3 MB project and 249 KB for codes (4831 lines). Web3 takes 265 KB (1153 lines). Note that the source code is set confidential due to the non-disclosure agreement with British Telecom (BT).
2.2 Typical Web3 Architecture

Tradition web architecture is based on a client-server model. The client is used for sending and receiving requests, while the server is used to process these requests and corresponding logic. The server side, also known as the backend, covers many fundamental aspects like operating systems (Windows, Linux), platforms (.Net, LAMP), and storage. API is to connect the application tier to servers. In contrast, the Web3 architecture replaces centralized backend servers with distributed ledgers. The backend contains two sectors, smart contracts (SC) for defining logic and rules, and blockchain platforms for processing transactions and achieving consensus. Web3 is more complex than traditional Web2 due to its complete decentralization, which requires dealing with the consistency problem [11]. In this work, we aim to ease the transition between the Web2 application tier and blockchain-backend systems. We establish an interpreter (short for Int.) to connect them seamlessly.

2.3 Essential Component

We provide basic building blocks to construct Web3 applications: a series of blockchain-specific components covering blockchain, smart contracts, on-chain applications, and clients.

Blockchain. Blockchain is a digital ledger that operates in a decentralized manner to securely and transparently record transactions [12]. Transactions are recorded as blocks, which are subsequently organized into a hierarchical structure. The blocks are arranged in a chronological and unalterable sequence to form the blockchain. To add a new block to the chain, a process called “mining” is used, which involves solving complex mathematical problems to validate transactions and add new blocks to the blockchain. This mining process is regulated by a consensus mechanism, which sets rules to ensure that all participants in the blockchain network agree on the validity of the transactions and their order in the blockchain.

Smart contract. Smart contracts are computer programs that automatically enforce the terms of an agreement between parties. They are used to speed up, verify, or execute digital negotiations. Ethereum developed smart contracts on the blockchain system by using Turing-complete scripting languages to achieve complex functionalities and execute thorough state transitions via consensus algorithms, resulting in final consistency. Smart contracts enable unfamiliar parties to conduct fair exchanges without the need for a trusted third party. They have a broad range of applications, including financial services [13][14], security protocols [15], and decentralized governance [16]. Smart contracts are viewed as a disruptive technology that could revolutionize many industries by improving efficiency, reducing costs, and eliminating the need for intermediaries.

On-chain application. On-chain applications, also known as DApps, are applications that run on blockchain systems [17]. Unlike traditional applications, which are centralized and controlled by a single entity, on-chain applications are decentralized and operate on distributed networks. The applications are designed to be transparent, secure, and trustless. They use smart contracts to execute code and transactions, and the consensus mechanism of the underlying blockchain network to verify and validate these transactions. Thanking the nature of decentralization, they are resistant to censorship, tampering, and other forms of malicious activity [18]. For specific usage, on-chain applications include various types such as NFT [19], gaming, social networking, etc.

Light client. A light client [20] plays a crucial role in the world of Web3 and blockchain technology, acting as an intermediary between user requests and the back-end servers or blockchains. Its primary function is to forward user requests to their appropriate destination without engaging in any logic processes. Generally, a light client is represented by a wallet [21] in the context of Web3 or blockchain. It is supported by locally running light nodes, which synchronize information with full nodes. This makes using a light client highly advantageous in resource-constrained
environments, such as different hardware devices, as it can reduce the costs of performing complex computations on-chain [22–24]. In the Web3 ecosystem, it is typical for users to connect their wallets to perform interactive actions on the website. By understanding the role of a light client, users can navigate the complexities of Web3 and blockchain technology with ease.

3 RESEARCH DESIGN

In this section, we conduct an exploratory study [25], which proposes a new Web3 driven framework, named WEBTTCOM, for implementing applications; and we define the research question (RQ) for identifying and extracting the evidence (e.g., the benefits of Web3 applications) for evaluating our proposed Web3 framework.

3.1 Research Question

This study aims to design and analyze whether our Web3 framework is effective and practical regarding quality attributes (i.e., smooth transition) and development productivity. Such characterization of Web3 will shed light on future Web3 applications and developments. Specifically, this work aims to address the Research Question (RQ) as:

Given the proposed Web3 framework (i.e., WEBTTCOM), is it effective, regarding the transition of Web2 to Web3 application development?

- **Web2 to Web3 transition**: Given that many developers do not have relevant development background for Web3 applications, is the proposed framework practical and helpful for them to transit the Web2 to Web3 applications for meeting and implementing the specific requirements?

- **Data privacy and governance**: In the evolution from Web2 to Web3, various issues related to data privacy and security, data privacy and governance in Web3 is one of the most discussed challenges for different stakeholders. As such, to address this concern, our RQ can explore whether the proposed Web3 framework (i.e., WEBTTCOM) ensures data privacy and governance.

- **Development productivity**: If the proposed Web3 framework impacts developers’ productivity? For example, (1) practitioners’ daily development tasks and documentation. (2) problem-solving for Web3 application development.

3.2 Study Design Process

Our research methodology consists of three stages, as depicted in Fig. 1. In the first stage, we proposed a Web3 transition framework (i.e., WEBTTCOM), which includes an interpreter to help practitioners transit and bridge the Web2-based to Web3-based applications. We implemented a full-stack project based on the proposed WEBTTCOM in the second stage. In the last stage, to evaluate the effectiveness of the proposed WEBTTCOM and the feasibility of the full-stack project, we conducted a quantitative performance evaluation and carried out a qualitative survey with 1,000 practitioners, including Venture Capitals (VCs), business managers, and developers, to gather their feedback and opinions.

**Phase 1: Web3 driven framework.** We proposed a Web3 framework (WEBTTCOM), which allows developers to implement effective and reliable applications. The framework guarantees:

- Web2 to Web3 smooth transition: our framework provides a trustworthy transition.

- Blockchain backend: the framework ensures the decentralization of Web3-driven applications.

The details of the framework are described in Section 4.1.
Phase 2: Web3 application implementation. In this phase, we implemented a full-stack application, which is based on the Web3 framework that we proposed. The details of the application are described in Section 4.2.

Phase 3: Performance evaluation and surveys. As depicted in Fig. 1, in this phase, we gave the overall quantitative performance evaluation. At the same time, we invited 1,000 practitioners to participate in qualitative surveys to confirm the effectiveness of our proposed Web3-driven framework and implementation.

- We invited 1,000 practitioners from BT, universities, and VC members in Australia and China, as well as developers from GitHub and those on the mailing list, to participate in our surveys. These surveys were anonymous and the survey results were based on the valid respondents.
- We designed a questionnaire consisting of three main questions (see Section 4.4.1) to get developers’ opinions on the effectiveness of our proposed framework and the implementation we developed.
- We applied qualitative data analysis and consistent comparison to summarize the statements from the survey result regarding the productivity of using our framework and application.

The questionnaire consists of three parts of questions:

- Part 1: We asked demographic questions, such as the surveyees’ experience in Web3 development and management.
- Part 2: We asked open-ended questions to understand their opinions on Web3 design and development in practice.
- Part 3: We prepared candidate topics by carefully reading the contents of representative textbooks. We picked a list of topics not explicitly mentioned in the open discussion and asked the participants to discuss those topics further. At the end of each survey, we thanked the participants and briefly informed them what we planned to do with his/her response.

4 RESEARCH RESULT

In this section, we propose a new framework, named WEB3COM, which not only smooths the transition between Web2 and Web3 but also achieves high data privacy and data governance and improves development productivity. A full-stack implementation is presented as a use case to
support the framework. We also conducted surveys where the intended 1,000 practitioners were invited to confirm the effectiveness of the proposed framework and implementation.

4.1 A New Framework: WEBTTCOM

A new framework, named WEBTTCOM (Web2 to Web3 Communicator), is proposed as a result. WEBTTCOM provides trusted and reliable service management across multiple organizations in a decentralized manner and promotes the transition from the existing Web2 form to Web3. Note that "transition" does not mean the existing Web2 applications must be completely reconstructed. Instead, a new interpreter is attached in the middle to offer reliable communication, smoothly integrating Web2 applications to Web3 platforms and transiting Web2 applications that would have been slowly drifting apart from decentralization into a Web3-empowered structure. In particular, the structure of WEBTTCOM consists of a blockchain layer, a backend interpreter, and a Web2 Software as a Service (SaaS), as in Fig. 2.

The blockchain layer stores service management data and supportive data and controls access to the service data. To this end, the blockchain layer contains a smart contract for data provisioning, a private data repository for service data, and a public data repository for data access policies and data hash checksum. In the smart contract, the policy resolver module extracts attributes of requests, such as requested data, request location, and user affiliation, and identifies effective access control policies. The write control module enforces data storage policies, such as data storage location and data expiration. The update control module takes charge of data updates, while the read control module enforces data access policies. The Physical-to-Logical (P2L) mapping module updates data presentation according to predefined physical data formats for data storage and logical data formats required by the SaaS. Note that the blockchain layer is designed to be blockchain-agnostic, allowing for seamless integration with any blockchain platform empowered by automated smart contracts and applicable functionalities such as user management, access controls, external integration, etc. This design ensures flexibility and adaptability, as the system can leverage the strengths of various blockchain platforms without being constrained by any single platform.

The backend, functioning as a Web2-Web3 interpreter, bridges the gap between traditional Web2 SaaS platforms and the emerging Web3 ecosystem by facilitating seamless communication between the two. This is achieved through the Web2-Web3 transition module, which is responsible for forwarding requests originating from Web2 SaaS applications to Web3 smart contracts and subsequently formatting the resulting Web3 responses in a manner that can be easily understood.
by Web2 SaaS applications. In order to effectively manage users during the transition phase, the backend interpreter maintains separate user management systems for both Web2 and Web3 users, with a completed synchronization at the end. This ensures that each user category’s unique attributes and requirements are adequately addressed while preserving the overall user experience. For the Web2 side, the backend interpreter offers a comprehensive REST API for SaaS applications to interact with. This API enables SaaS platforms to send requests and receive responses from the Web3 environment without directly interacting with Web3 technologies. This simplifies the integration process and allows developers to focus on their core application logic. To further streamline the development process, the backend interpreter incorporates automated development technology that automatically generates routes and documentation. This significantly reduces the time and effort required by developers to create, maintain, and update their APIs, allowing them to concentrate on delivering high-quality services to their users. When it comes to communicating with blockchain networks, the backend interpreter leverages blockchain Software Development Kits (SDKs). These SDKs provide a set of pre-built tools and libraries that make it easier for developers to interact with various blockchain protocols and smart contracts. By utilizing these SDKs, the backend interpreter can efficiently query, submit transactions to, and retrieve data from the blockchain while maintaining high security and reliability. In other words, the backend interpreter plays a crucial role in bridging the divide between Web2 SaaS applications and the emerging Web3 ecosystem. By offering a seamless communication channel between the two environments, managing user transitions effectively, and providing robust development tools, the backend interpreter enables the integration of Web3 technologies into existing Web2 applications, paving the way for a decentralized and interconnected digital landscape.

Web2 SaaS platforms offer sophisticated service management interfaces that allow users to easily submit, view, and process service tickets after authenticating themselves on the platform. These interfaces are designed to provide a seamless user experience, making it simple for users to access and manage their service requests. To ensure adaptability and compatibility with the emerging Web3 technologies, the service management functionality is implemented as a modular, plug-and-play component. This design approach enables the service management module to be easily integrated with popular service management platforms, such as ServiceNow React Container [26], while allowing for incorporating Web3-enhanced features.

4.2 Implementing WEBTTCom: A Use Case in a Service Management System

An implementation of the proposed framework WEBTTCom is discussed in which a trusted and reliable service management platform across BT Global, BT Australasia, and BT Australasia customers is demonstrated. Managing the services (e.g., inventory/ordering/ticketing) that BT deploys is challenging due to a lack of trust in the data to execute these processes. Conflicting information, lack of real-time parameters of the state of the various devices, and complex disputes between two or more parties are just a few ingredients of these challenges. BT’s services can consist of equipment from different manufacturers deployed in different countries, managed by different secondary service providers, operating under different SLAs, and serving different customers. Correct management of these complex services has proven to be one of the key challenges for BT’s engineering teams and is one of the key factors for the company’s international reputation and success. Currently, these services are managed by legacy and non-integrated systems that often cannot operate in real-time to meet global customers’ demands.
Fig. 3. The class diagram of the instantiated service management platform.

The HyperLedger Fabric [27] is set up with one Fabric ordering service\(^5\), and organizations representing BT Global, BT Australasia, and BT Australasia customers in the same channel\(^6\) for the Web3 service. Each ordering service and organization has one Certificate Authority (CA) node and one peer node. Service ticket data are recorded in the Hyperledger Fabric in real time and are certified by the data hash. With Web3 technology, all organizations have a consistent view of the service data. The platform is developed in TypeScript 4.4 using Visual Studio Code 1.69.1 and runs on the elastic servers of Amazon Web Service (AWS). MySQL 5.7 is selected to handle the Web2-based registries, whereas the Web3-based registries rely on the native CouchDB of HyperLedger Fabric 1.4.

The implementation design of a service management system, as a use case of the proposed WEBTTCOM, is illustrated in Fig.3 using the class diagram. Blockchain maintains a blockchain network with several registered SmartContract. SmartContract is inherited by three classes, i.e., Web3UserRegistry, Web3ObjectRegistry, and Web3PolicyRegistry. The Web3UserRegistry registers the user Web3 information in Hyperledger Fabric CAs and interacts with the corresponding customizable off-chain-stored Web2UserRegistry in which the local user management is conducted. All objects, including the product inventory, product order, service inventory, and trouble ticket service, are on-chain-stored in Web3ObjectRegistry with dedicated policies of access control stored in Web3PolicyRegistry. All registries comprise Record in which the user-defined data record attributes are stored. Accessing the attribute values in Record relies on Web3Config in which the settings of Blockchain are defined. AutoOpenAPIGenerator auto-generates the OpenAPI standard [28] documentation for modules including Web2UserRegistry, Web3UserRegistry, Web3ObjectRegistry, and Web3PolicyRegistry.

4.2.1 Attribute-based Access Control in Web3. In the Hyperledger Fabric, each organization has one private data collection [29], which excludes other organizations, such that private business data are only saved and managed by the permitted organization. All blockchain members can

---

\(^5\) The ordering service validates transactions and assembles valid transactions into ordered blocks.

\(^6\) A Fabric channel is a private sub-network for permitted members.
access public on-chain data. Web3 users are registered to CAs with their attributes, including organizations, departments, and user types. The attributes are used for the attribute-based access control to the service data. Each Web3 user has a unique private key to sign Web3 transactions and access Web3 services.

**Smart Contract:** A service smart contract is developed to manage service tickets. The exposed entry points of the smart contract are shown in Listing 1. The smart contract extends the Fabric `Contract` class and exports functions as APIs.

In the smart contract, create/update/read functions are developed for `Web3PolicyRegistry` and `Web3ObjectRegistry`, as shown in Fig.3. `Web3PolicyRegistry` is implemented with the Fabric public data scheme where all blockchain peers have a copy of the public data, while `Web3ObjectRegistry` is realized with the Fabric private data scheme where only authorized peers have data copies. Variable `ctx` collects the context of the transaction calling the smart contract, including user attributes and timestamp for access control and transient service data. Variable `attributes` provides Web2 attributes for access control, such as geolocation information. Variable `publicPayload` gives the data requested by smart contracts and recorded on the Fabric blockchain, such as access control policies.

**Access Control:** The smart contract enables attribute-based access control (ABAC), where access control policies are designed based on user and environment attributes and organized as the `Web3PolicyRegistry`. A policy is either `writePolicy`, `readPolicy`, or `updatePolicy`, and each of them has unique fields describing access control requirements. All the policy types also have general metadata, including an updated time, an effective time, and a priority level, for policy conflict resolver. All policies are created by organization admins and saved in the public data repository such that policies can be accessed by all Web3 users and applied to all private data collections.

The policy resolving process is shown in Listing 2. The policy resolver first extracts the Web3 user attributes and environment attributes of the transaction, such as transaction timestamp, from the `ctx` variable. Next, the policy resolver module retrieves all related policies according to the policy IDs embedded in the request data. Then, the policy resolver identifies effective policies by checking Web3 user roles, effective timestamps, policy priority, and geolocations. At last, the policy resolver returns effective policies for the request.

The write control, update control and read control modules then process the request according to the request data in the request transactions and effective policies. The request is rejected if no effective policy is returned from the policy resolver module. For a data write/update request, the write control and update control modules first identify the right private data collection from effective `writePolicy` and `updatePolicy`. The modules also check request attributes against the requirements specified in the effective policies and terminate the request for failed checks. The modules then extract transactional data and transient data from the request transaction and create a Web3 object with object information and related access policy IDs. Then, the modules can write/update the Web3 object to the data collection. For a data read request, the read control module gets the private data collection from the request transaction and reads the requested data out following the effective `readPolicy`. The read control module also implements the P2L mapping on the data keys following the mapping rules stated in effective `readPolicy`.

**4.2.2 Web2-Web3 Transition.** Recall that "transition" does not lead to a completed reconstruction of existing Web2 applications. It is the interpreter that establishes the communication and smoothly makes the Web2 applications empowered by Web3. Each organization has an interpreter to convert Web2 requests to Web3 requests and forward requests to specific Web3 nodes according

---

7Transient data is a type of data for Fabric private data collection, which can be read by smart contracts but does not be recorded in on-chain transactions.
to request attributes for Web3 processing. The interpreter then monitors Web3 processing results and creates returns for Web2 requests.

**Web2 APIs:** The interpreter provides REST APIs supporting create, read, and update operations in Web2 applications and implements the Bearer Authentication scheme [30].

A Web2 request includes payloads describing create, read, and update operations on Web3 objects and policies. Besides that, the Web2 request contains a bearer token for authentication by the interpreter. The bearer token also allows the interpreter to identify the corresponding Web3 identity and private key for signing Web3 transactions. The Web2 request also gives details for the corresponding Web3 request, including the channel name, smart contract name, and targeted private data collection, such that the interpreter can create correct Web3 request transactions and send them to the right Web3 node for processing.

**Web3 Interaction:** Interpreters manage user identities and interact with the service smart contract using the Hyperledger Fabric SDK [31], as shown in Listing 3. The channel and service smart contract is specified by variables `channelName` and `scName` in lines 10-11.

For every Web3 request, the interpreter needs to create a request transaction and sign the transactions with a valid Web3 user identity. To this end, the interpreter saves the Web3 user identities of all Web2 users in the organization, i.e., `walletPath` and `walletID` in Listing 3, and maintains the mapping between Web2 and Web3 identities. The interpreter is an organization admin user who can register and revoke users. During the user registration process, a user provides user information and credentials to the interpreter, like general Web2 user registration. The interpreter first creates a new user record on itself and then sends the registration information to the organizational CA node to create a Web3 identity. Then, the interpreter keeps the user’s Web3 wallet, including a private key, and records the mapping between the Web2 identity and Web3 identity.

The interpreter needs to connect specific Web3 nodes that have requested private data collections. To this end, the interpreter maintains connection profiles to Web3 nodes from different organizations and records the mapping between private collections and the hosting Web3 nodes. When the interpreter receives a Web2 request, the interpreter identifies the data collection related to the request. The interpreter forwards the request to any Web3 node if the requested data is public. If the request points to some private data collection, the interpreter first identifies the hosting Web3 node and then forwards the request to the Web3 node using pre-configured connection profiles.

### 4.2.3 Transition with SaaS: ServiceNow-based Web Application.

ServiceNow is a popular cloud-based workflow automation platform for enterprises by streamlining and automating routine work tasks. Companies often seek a simple solution for connecting their existing ServiceNow applications to Web3 systems. As a result, we chose ServiceNow as our front-end system. We intend to solve the following problems with our scheme:

- ServiceNow’s existing system could be transformed in an efficient manner.
- Programmers’ coding can be made easier by following a few simple steps.

To resolve these issues, we implement a ServiceNow container as shown in the left part of Fig.2. We develop a web application using React and embedded it in ServiceNow. By using REST APIs, the embedded application can smoothly communicate with the backend and the interpreter.

The web application was developed using React, an open-source JavaScript library for creating user interfaces with a large fan community. Therefore, most programmers are familiar with it and can easily obtain assistance from other community members.

---

8A connection profile describes the Web3 node to be connected, such as ID and IP/port.
A modern website deployment method - Webpack - is used to deploy this web application. Instead of deploying all resource files directly to the web server, Webpack binds multiple files into a single deployment file. Multiple HTML files are combined into one HTML file, and multiple Javascript files are combined into one Javascript file. This deployment method simplifies the resource access path, which is exactly what we require.

- The deployment utilizes two ServiceNow concepts: the UI page and the web service.
- The ServiceNow UI page allows developers to customize web pages using HTML or XML.

In this case, we create one UI Page and insert the combined HTML code using Webpack.

The ServiceNow web services provide access to resources through REST APIs. Three web services have been developed: the Asset Service, the Style Service, and the JS Service. All images are accessible through the asset service. The styling service provides all CSS access. The JS service provides access to JavaScript code.

The web application has three functions: user login, policy management, and ticket management. As part of implementing these functions, the embedded application communicates with the interpreter using HTTP REST APIs.

4.2.4 Backend and Automated Development Tools. The Express framework offers a flexible Node.js web application with MySQL being used for Web2UserRegistry and Web3Config. Sequelize is used to provide effective connections to the Web2 local registry in the manner of an Object-relational Mapper (ORM). The platform is docker-based by default, and docker-compose offers an orchestration service for distributed and flexible docker containers.

The system also offers automated generation and documentation of OpenAPI-compliant REST APIs [28] for Web2UserRegistry, Web3UserRegistry, Web3ObjectRegistry, and Web3PolicyRegistry via TSOA 3.11 [32]. The TSOA framework integrates the OpenAPI compiler to construct Node.js server-side applications in type-safe TypeScript at runtime, as TypeScript is used for programming smart contracts at the Web3 side and backend/frontend at the Web2 side. The function, routeGenerate, can easily generate the API routes automatically with no pain based on the definition of controllers of all registries above; see the lower half of each registry in Fig.3. At the same time, the YAML-based OpenAPI documentation and test suite are also generated to ease API testing and smooth the development of transition via docGenerate.

4.3 Performance at a Glance

The developed WebttCOM and Hyperledger Fabric 1.4.6 have been deployed on a Mac mini, equipped with a 3.2 GHz 6-Core Intel Core i7 and 32 GB 2667 MHz DDR4 memory, utilizing Docker. WebttCOM operates with Fabric SDK fabric-network version 1.4.0, while the Hyperledger Fabric connection profiles and the wallets of associated users are integrated into WebttCOM. Within Hyperledger Fabric, the BatchTimeout is configured to 2 seconds, establishing the maximum duration a blockchain node should wait for transactions to be mined into a block. Consequently, write operations can be confirmed in a maximum of 2 seconds.

For the performance evaluation, five APIs have been selected, encompassing read/write operations and public/private data, as detailed in Table 2. The two data objects, i.e., policy and ticket, are stored as public and private data on the blockchain, respectively. The API Get a policy retrieves a single policy, approximately a 1KB JSON object, based on the policy ID. Get all policies retrieves all policies, totaling 91 policies and 56.97KB. Get all tickets fetches all accessible tickets, which include eight tickets and amount to 5.67KB. It is noteworthy that read access control is enabled in the smart contract. Upon receiving the read request, the smart contract initially retrieves all valid readPolicy according to the user and policy metadata, subsequently identifies accessible tickets, and returns
| API             | R/W | Public/Private | Payload     | Size    |
|-----------------|-----|----------------|-------------|---------|
| Get a policy    | Read| Public         | 1 policy    | 1KB     |
| Get all policies| Read| Public         | 91 policies | 56.97KB |
| Get all tickets | Read| Private        | 8 tickets   | 5.67KB  |
| Create a policy | Write| Public         | 1 policy    | 1KB     |
| Create a ticket | Write| Private        | 1 ticket    | 2KB     |

them to WEBTTCom. Create a policy generates a single writePolicy of about 1KB, while Create a ticket creates a single ticket, approximately 2KB, adhering to the write access control.

![Graph](image)

(a) Web2 response time and Web3 processing time  
(b) Processing time on WEBTTCom

Fig. 4. Time performance of WEBTTCom

Fig. 4 illustrates the overall response time and Web3 processing time of the APIs, labeled as Web2 and Web3, respectively. During the test, each API is executed independently 20 times in a sequence, with the overall response time derived from the HTTP response and the Web3 processing time extracted from the blockchain peer log. Therefore, the overall processing encompasses HTTP request handling in WebttCom, user authentication in WEBTTCom, transaction generation in WEBTTCom, communication between WEBTTCom and the blockchain, and Web3 processing time. As depicted in Fig. 4(a), the three read APIs require less than 600 ms, and both the overall response and Web3 processing time increase with the complexity and data size of the applications. The two write APIs take approximately 2.5 seconds, attributable to the BatchTimeout of 2 seconds, which could be expedited with a reduced BatchTimeout. Operations on private data necessitate a longer processing time than those on public data. Fig. 4(b) demonstrates the processing time on WEBTTCom, i.e., the discrepancy between the overall response time and Web3 processing time. WEBTTCom takes about 150 ms for the APIs, while Get all policies takes roughly 200ms and exhibits the largest variance due to the substantial data transmission and resolution of a large number of policies. The evaluation results affirm that WEBTTCom can efficiently transition Web2 to Web3 applications.
4.4 Survey Results

We followed the same benchmarks in [33] by conducting a set of surveys to assess the Necessity, Usability, and Completeness of the proposed framework WEBTTCom and the implementation.

4.4.1 Survey Questions. The technical background and Web3 experience were asked, followed by the following questions from the domain experts to ensure the Necessity of the new proposed framework WEBTTCom:

- What do you think of the pros and cons of Web2 and Web3 by now under your background?
- What do you think of the necessity of a smooth transition between Web2 and Web3 (both Web2 to Web3 and Web3 to Web2) from your organizational and personal perspectives?
- What are the possible challenges during the transition that you think are required to be resolved immediately?

Further, we asked the following questions for feedback on WEBTTCom and overall Usability and Completeness of the implementation.

- To what extent does the Service Management System developed by UTS and BT, the implementation of the proposed framework WEBTTCom, match the principles of WEBTTCom and address the challenges above?
- What are your suggestions to enhance the suitability of the new framework WEBTTCom and its implementation?

4.4.2 Key Findings. We summarized key findings from the surveys that can support the proposed framework WEBTTCom and implementation as a use case. Out of the 1,000 practitioners we invited, we received feedback from 58 valid respondents. Approximately 80% of the responses were positive, and around 20% were neutral; see Fig. 5. For clarity, we referenced five highly representative responses in Appendix-B during our discussion of the key findings, listed as follows:

1) Necessity - enable guided and structured design: Building Web3 technology into Web2 systems can solve critical security and trust issues in Web2 systems, especially for businesses across multi-organizations. Web2 systems are centralized and can hardly achieve trust across organizations, while the inherent consensus mechanism of Web3 technology can provide verified single-ground truth and thus build trust across parties. The second surveyee stated, "A transition from Web2 to Web3 will bring the data trustworthiness and cybersecurity guarantee from Web3 to Web2 systems."

It is important to have a smooth transition between Web2 and Web3. This is because Web2 is a mature technology, but developing Web3 applications could be challenging. As stated by the
fourth surveyee with limited Web3 knowledge “The modification of the existing Web2 system should not be difficult. To avoid overloading the programmers with Web3 knowledge, they should not learn too much.”, and the fifth surveyee stated, “Web2 and Web3 should be able to coexist and interact smoothly.” Detailed challenges during the transition between Web2 and Web3 include

- The transition of two different technologies, as stated by the fourth surveyee “Due to the differences in concept, technology, and tools between Web2 and Web3, it is difficult to integrate the Web2 system with the Web3 system.”
- New access control schemes in Web3. The first surveyee stated “Web3 is transparent, how to apply flexible and feasible access control becomes important for cases where data privacy is considered”, and the third surveyee stated “DLTs are a promising solution due to the ability of smart contracts to ensure that the required country-specific data management policies are agreed and enforced.”
- Heterogeneous user management across Web2 and Web3. The first surveyee stated, “Approach to apply the user management in a shared ledger may be a challenge as Web2 parties tend to have separate user management systems locally.”
- High development and transition cost. The first and the fifth surveyees stated “Easing the transition by using automated tools is normal in Web2 applications and is also essential during the transition between Web2 and Web3.” and “Challenges include lack of experienced developers, lack of available development resources, tools.”

2) Usability - flexible access control: The developed flexible access control can implement access control policies as designed. The service provider can define access control policies, including data storage and write and read policies. The smart contract on the Hyperledger Fabric enforces all the policies and controls access to on-chain data. All surveyees have confirmed the developed access control mechanism. The second surveyee stated, “The system can enforce all the expected data governance and access control policies.”, and the third surveyee stated “In evaluating the solution we have demonstrated that a DLT Hyperledger layer can meet the required success criteria related to cross-country access control and user-management; as well as connecting to a traditional SaaS workflow management layer.”

3) Usability - compatible user management: The developed user management compatibly manages Web2 and Web3 users. When a user submits a Web2 registration request, Web2 and Web3 accounts are simultaneously created and managed by the developed framework. When users log in and submit requests, the framework can perform authentication with the users’ Web2 credentials and process Web3 requests and responses in the representation of the users. All chosen surveyees are satisfied with the developed user management. The second and fourth surveyees stated, “A user can use one identity to access Web2 and Web3 services.” and ”Web2 programmers are not required to touch too much.”

4) Usability - easy integration with existing platforms and services: The developed system seamlessly integrates with existing service management on ServiceNow. All operations are conducted in ServiceNow, including login, access control policy management, and service ticket management. The workflow remains the same with the Web2 version. Nevertheless, all data are securely saved on the Hyperledger rather than disconnected databases and can be verified with the Web3-certified data hash. As the second surveyee stated, “Users can access Web3-certified data services from the SaaS. The complicated Web3 details are transparent to users.”

5) Usability - highly automated development tools: Highly automated development tools are implemented to generate and document OpenAPI-compliant REST APIs of both Web2 and Web3 registries. This significantly improves programming efficiency by sharing the same development workflows and schema formats between Web2 and Web3. As the 1st, 2nd, and 4th surveyees
gave positive feedback by stating “Automated development tools significantly reduce the programmers’ workload and has almost become a must-use tool during the current workflow.”

6) Completeness - sufficient decentralization with robust data privacy and governance: The surveyees satisfy the framework Web3Com and the use case regarding achieving sufficient decentralization level while ensuring strong data privacy and governance; as the first surveyee stated “The framework appears to be covering most perspectives including the access control, data privacy, data governance, user management, connections to existing SaaS, and development productivity.”. This is done by successfully applying the flexible access control in a decentralized manner, i.e., smart contracts on the Hyperledger Fabric. Robust data privacy and governance can thus be offered to ensure secure and transparent data management while fostering trust among participants in a distributed and tamper-resistant manner.

7) Completeness - limited cases: The fourth and fifth surveyees stated “More commercial Web2 systems should be integrated with Web3.” and “Need to find suitable business use cases to demonstrate the benefit of Web3”. Three surveyees noted that the proposed framework Web3Com requires more business use cases to support by potentially integrating the existing commercial Web2 projects. While we admit this limitation at the time of writing, this work originates from solving the transition issue from Web2 to Web3, which came across by our business partner proceeding with their latest strategy. This work is new and lacks sufficient existing use cases. Therefore, this work focuses on demonstrating how Web3Com solves the research question, and we postpone extending use cases to improvements and future works.

8) Potential Improvements: Among the chosen representative respondents, four indicated that the framework and the provided use case seem limited to permissioned blockchain platforms such as HyperLedger Fabric. They suggested implementing this framework on other prominent platforms, such as Ethereum, and with more advanced scalability solutions such as blockchain sharding [34–36] and cross-chain designs [37], necessitates additional testing. Another highlighted respondent noted that while the integration with ServiceNow is commendable, there’s a need to further test its compatibility with services and functionalities offered by Platform as a Service (PaaS) or Infrastructure as a Service (IaaS) platforms, such as AWS and Azure. Notably, these representative insights align with the general sentiment expressed in the 58 valid responses we received.

5 LIMITATIONS AND VALIDITY

We apply the guidelines [38] to discuss key threats to the validity (construct, internal, external) of this work.

Construct validity reflects what extent the research questions and methodology are appropriately used in a study. A threat in Stage 3 (i.e., the surveys) is whether or not our surveyees are representative. To reduce this threat, we invited 1,000 practitioners who come from the company, universities, and VC members who are applying for our Web3 implementation. In addition, the areas that the participants have worked on cover a wide range of domains (e.g., blockchain and AI applications). However, they may not be representative of all practitioners, and not all the intended surveyees are expected to respond in time. To mitigate this potential bias, we have carefully chosen questions and topics. Our survey respondents completed the surveys based on their opinions and perceptions. It is possible that they conflate the skills that are very important and the skills that are very relevant to their projects or industrial contexts.

Internal validity focuses on factors that may influence the validity of the results. The main threat in our study is whether the study process we designed and the Web3 framework and application that we proposed can answer our research questions. It is also possible that we draw the wrong conclusions about respondents’ perceptions from their comments. To minimize this threat,
we read transcripts many times and checked the survey results and the corresponding comments several times.

The selection of statements produced at the end of the surveys may not be comprehensive and may be biased to the background of experts—who may not be able to articulate their own opinions. To mitigate this bias, we have taken the following steps:

- Aside from asking direct questions about their opinions about Web3 and applications that we designed and implemented, we also asked them to discuss general topics that they had not explicitly mentioned. The topics were selected from Web3 textbooks and online resources; they include concepts, comprehension, programming language, requirements, design implementation, testing, and tool usage.
- Three authors have performed data analysis to cross-check their answers by using card sorting [39], and we carefully examined and only included relevant statements.

**External validity** concerns the generality of our study results to other settings. Our results and summaries are based on the Web3 framework and application and survey participants’ opinions instead of a rigorous analysis of participants’ claims. It is possible that opinions regarding the Web3 framework and application differ from one participant to another. To improve the generalizability of our results, we invited 1,000 practitioners and obtained 58 respondents. Still, our findings may not generalize or represent the perception of all software engineers. For example, the respondents are from one company and VCs that are interested in Web3 technologies and closed universities. It can be noticed in Fig. 5(b) that a sentiment bias in certain categories. The framework was initially designed for BT Company, leading to skewed positive feedback. A similar bias exists in the Universities category, potentially due to some participants being affiliated with the authors’ institution. In contrast, feedback from VCs and other developers, sourced from GitHub and mailing lists, appears more balanced and free from conflicts of interest. It would be interesting to perform another study to investigate more software engineers to perceive the future benefits and limitations of the Web3 framework and application.

**Platform validity** refers to the extent to which the proposed framework can be universally applied across multiple blockchain platforms. While our design principle for the blockchain layer is inherently blockchain-agnostic, our case study predominantly used Hyperledger for illustration. This choice was primarily to demonstrate the capabilities of our framework in a clear and specific manner. However, it is essential to stress that the architecture and design of our framework, Web2Web3Com, do not restrict its applicability. The core principles and mechanisms of Web2Web3Com can be effectively translated to other blockchain platforms with the necessary configuration adjustments. The foundation remains consistent, versatile, and designed for broad adaptability.

To elucidate further on the adaptability of Web2Web3Com with other platforms such as Ethereum: The crux of our framework lies in its modular design. The blockchain layer, which is designed to be blockchain-agnostic, can interface with Ethereum’s smart contract system just as it does with Hyperledger. Ethereum, with its robust smart contract capabilities, can effectively replace the smart contract for data provisioning and the modules for write, update, and read controls present in our framework. Furthermore, Ethereum’s dynamic storage system can be leveraged to handle service data storage in our blockchain layer, substituting for private and public data repositories. The backend interpreter, which serves as the bridge between Web2 and Web3, remains consistent. Its Web2-Web3 transition module can be configured to forward requests to Ethereum’s smart contracts, ensuring seamless communication. Similarly, user management for Web2 and Web3 users in the backend would work harmoniously with Ethereum, ensuring an integrated user experience.
6 RELATED WORK

In this section, we give a quick overview of the notion of Web3 and then introduce ways of building DApps on top of blockchains.

Notion of Web3. The concept of Web3 was first proposed by Wood [40] with an initial discussion focusing on blockchain-based digital infrastructure. Later, Weyl et al. [1] illustrate the ways of building a decentralized society by exploring Web3-related applications, requirements, opportunities, and challenges. Wang et al. [3] provide the discussion between Web3 and blockchain from the perspective of architecture design. The work has identified a total of 12 types of designs according to the data workflow of access, computation, and storage. A series of reports from Consensys [41] investigates the economic performances created by decentralized networks. Web3 economy covers many on-chain DeFi protocols [14] such as stablecoins, borrowing, lending, and leverage, distributed governance protocols (e.g., DAO) [42][43] and many innovations combined with other technologies [44, 45]. These protocols are built on top of smart contracts. Liu et al. [46] explore three types of infrastructural enablers, including the single smart-contract powered chain, federated contracts, and interoperable blockchain platforms. However, Web3 so far still confronts high-level controversies in terms of its definition and application. We, in this paper, extend our exploration by complying with its core decentralization nature guaranteed by underlying blockchain services.

Constructing Web3 Applications. Traditional Web1 and Web2 applications rely on centralized backend servers for computations and storage. Web3 applications [3] replace these servers with decentralized blockchain platforms [6][47]. Building a Web3 application requires three phases: (i) embedding the wallet for access, (ii) connecting frontend with blockchain platforms, and (iii) operating executions on-chain. A wallet [20][21] is backed by locally running lightweight nodes and helps Web3 users to create on-chain accounts (in the form of address [12]) as their identities. A user can initiate a request by sending a transaction from wallets to the transaction pool. Smoothly processing the received requests from the frontend requires a suite of standard protocols, including token standards (e.g., EIP [48], BEP [49] or even BRC [50]) and unified APIs. All the compiled bytecodes will be executed on-chain by iterative state transitions. The consensus mechanism is critical in maintaining state consistency and chain stability [51]. In some cases, external techniques are needed for supportive functionalities such as distributed storage [52], layer-two computations [53], cross-chain bridges [54][55] or on-chain oracles [56]. Our work achieves more than building a simple DApp that is independent of operating blockchains. We, instead, develop a general interpreter to connect Web2 applications to the current leading Web3 platforms from the access layer to the chain layer. This gives an educational study for the community.

7 CONCLUSION

In this paper, we gave a research question by exploring the connotation of Web3 and the key differences between Web2 and Web3 applications. We proposed a new framework, named WEBTTCom, to enable a seamless transition from Web2 to Web3. In particular, WEBTTCom can smoothly connect traditional Web2 applications to mainstream Web3 blockchain platforms while additionally ensuring high data privacy and governance, and improving development productivity. Our design innovative introduces an interpreter mechanism that can aggregate and deal with requests between Web2 and Web3 spaces. Accordingly, we implemented a full-stack system consisting of over 11,351 lines of code, conducted a performance evaluation, and launched a survey to assess the effectiveness of our framework. Corresponding surveys with experienced participants confirmed WEBTTCom satisfies our research question with a few possible limitations of the framework and its
related business cases. We further provide our recommended improvements, including the extension of the framework to involve more blockchain platforms and search for more business cases.

REFERENCES

[1] E Glen Weyl, Puja Ohlhaver, and Vitalik Buterin. Decentralized society: Finding web3’s soul. *Available at SSRN 4105763*, 2022.

[2] Xu Wang, Xuan Zha, Wei Ni, Ren Ping Liu, et al. Survey on blockchain for internet of things. *Computer Communications*, 136:10–29, 2019.

[3] Qin Wang, Rujia Li, Qi Wang, Shuping Chen, et al. Exploring web3 from the view of blockchain. *arXiv preprint arXiv:2206.08821*, 2022.

[4] Guangsheng Yu, Xuan Zha, Xu Wang, Wei Ni, Kan Yu, Ping Yu, J. Andrew Zhang, Ren Ping Liu, and Y. Jay Guo. Enabling attribute revocation for fine-grained access control in blockchain-iot systems. *IEEE Transactions on Engineering Management*, 67(4):1213–1230, 2020.

[5] Guangsheng Yu, Litianyi Zhang, Xu Wang, Kan Yu, Wei Ni, J. Andrew Zhang, and Ren Ping Liu. A novel dual-blockchained structure for contract-theoretic lora-based information systems. *Information Processing & Management*, 58(3):102492, 2021.

[6] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[7] Yue Liu, Qinghua Lu, Guangsheng Yu, Hye-Young Paik, and Liming Zhu. A pattern-oriented reference architecture for governance-driven blockchain systems. In *IEEE International Conference on Software Architecture (ICSA)*, pages 23–34, 2023.

[8] Yue Liu, Qinghua Lu, Guangsheng Yu, Hye-Young Paik, Harsha Perera, and Liming Zhu. A pattern language for blockchain governance. In *Proceedings of the 27th European Conference on Pattern Languages of Programs, EuroPLoP ’22*, New York, NY, USA, 2023. Association for Computing Machinery.

[9] Yue Liu, Qinghua Lu, Guangsheng Yu, Hye-Young Paik, and Liming Zhu. Defining blockchain governance principles: A comprehensive framework. *Information Systems*, 109:102090, 2022.

[10] Yue Liu, Qinghua Lu, Guangsheng Yu, Hye-Young Paik, and Liming Zhu. Bgra: A reference architecture for blockchain governance, 2022.

[11] Xu Wang, Wei Ni, Xuan Zha, Guangsheng Yu, et al. Capacity analysis of public blockchain. *Computer Communications*, 177:112–124, 2021.

[12] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. SoK: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy (SP)*, pages 104–121. IEEE, 2015.

[13] Sam M Werner, Daniel Perez, Lewis Gudgeon, Ariah Klages-Mundt, Dominik Harz, and William J Knottenbelt. SoK: Decentralized finance (DeFi). *ACM Advances in Financial Technologies (AFT)*, 2022.

[14] Erya Jiang et al. Decentralized finance (DeFi): A survey. *arXiv preprint arXiv:2308.05282*, 2023.

[15] Rujia Li et al. How do smart contracts benefit security protocols? *arXiv preprint arXiv:2202.08699*, 2022.

[16] Aggelos Kiayias and Philip Lazos. SoK: Blockchain governance. *ACM Advances in Financial Technologies (AFT)*, 2022.

[17] Bhabendu Kumar Mohanta, Debashis Jena, Soumyashree S Panda, and Srichandan Sobhanayak. Blockchain technology: A survey on applications and security privacy challenges. *Internet of Things, 8*:100107, 2019.

[18] Sin Kit Lo, Yue Liu, Guangsheng Yu, Qinghua Lu, Xiwei Xu, and Liming Zhu. Distributed Trust Through the Lens of Software Architecture. *arXiv e-prints*, May 2023.

[19] Qin Wang, Rujia Li, Qi Wang, and Shuping Chen. Non-fungible token (NFT): Overview, evaluation, opportunities and challenges. *arXiv preprint arXiv:2105.07447*, 2021.

[20] Panagiotis Chatzigiannis, Foteini Baldimtsi, and Konstantinos Chalkias. SoK: Blockchain light clients. In Ittay Eyal and Juan Garay, editors, *Financial Cryptography and Data Security (FC)*, pages 615–641, Cham, 2022. Springer International Publishing.

[21] Kostis Karantias. SoK: A taxonomy of cryptocurrency wallets. *Cryptography ePrint Archive*, 2020.

[22] Xu Wang, Guangsheng Yu, Xuan Zha, Wei Ni, Ren Ping Liu, Y. Jay Guo, Kangfeng Zheng, and Xinxin Niu. Capacity of blockchain based internet-of-things: Testbed and analysis. *Internet of Things, 8*:100109, 2019.

[23] Xu Wang, Ping Yu, Guangsheng Yu, Xuan Zha, Wei Ni, Ren Ping Liu, and Y. Jay Guo. A high-performance hybrid blockchain system for traceable iot applications. In Joseph K. Liu and Xinyi Huang, editors, *Network and System Security*, pages 721–728, Cham, 2019. Springer International Publishing.

[24] Xu Wang, Guangsheng Yu, Ren Ping Liu, Jian Zhang, Qiang Wu, Steven W. Su, Ying He, Zongjian Zhang, Litaow Yu, Taoping Liu, Wentian Zhang, Peter Loneragan, Eryk Dutkiewicz, Erik Poole, and Nick Paton. Blockchain-enabled fish provenance and quality tracking system. *IEEE Internet of Things Journal*, 9(11):8130–8142, 2022.
[25] Steve Easterbrook, Janice Singer, et al. Selecting empirical methods for software engineering research. In Guide to advanced empirical software engineering, pages 285–311. Springer, 2008.

[26] Servicenow. https://www.servicenow.com/.

[27] Elli Androulaki, Artem Barger, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys Conference (EuroSys), pages 1–15, 2018.

[28] OpenAPI. The openapi specification. https://github.com/OAI/OpenAPI-Specification, 2022.

[29] Hyperledger Fabric. Private Data. https://hyperledger-fabric.readthedocs.io/en/release-2.2/private-data/private-data.html?highlight=private-data, 2022.

[30] SmartBear. Bearer Authentication. https://swagger.io/docs/specification/authentication/bearer-authentication/, 2022.

[31] Hyperledger Fabric. Hyperledger Fabric SDK for node.js. https://hyperledger.github.io/fabric-sdk-node/release-1.4/index.html,

[32] Lukeautry. Openapi-compliant rest apis using typescript and node. https://github.com/lukeautry/tsoa, 2022.

[33] Guangsheng Yu, Xu Wang, Kan Yu, Wei Ni, J. Andrew Zhang, and Ren Ping Liu. Survey: Sharding in blockchains.

[34] Xiwei Xu, H.M.N. Dilum Bandara, Qingshua Lu, Ingo Weber, Len Bass, and Liming Zhu. A decision model for choosing patterns in blockchain-based applications. In 2021 IEEE 18th International Conference on Software Architecture (ICSA), pages 47–57, 2021.

[35] Guangsheng Yu, Xu Wang, Kan Yu, Wei Ni, J. Andrew Zhang, and Ren Ping Liu. Survey: Sharding in blockchains. IEEE Access, 8:14155–14181, 2020.

[36] Zhixu Zhang, Xu Wang, Guangsheng Yu, Wei Ni, Ren Ping Liu, Nektarios Georgalas, and Andrew Reeves. A community detection-based blockchain sharding scheme. In Shiping Chen, Rudrapatna K. Shiyamasundar, and Liang-Jie Zhang, editors, Blockchain – ICBC 2022, pages 78–91, Cham, 2022. Springer Nature Switzerland.

[37] Guangsheng Yu et al. Predicting nft classification with gnn: A recommender system for web3 assets. In 2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pages 1–6. IEEE, 2023.

[38] Qin Wang, Guangsheng Yu, Yi Lin, Caijun Sun, Lam Duc Nguyen, Sherry Xu, and Shiping Chen. An empirical study on snapshot daos. arXiv preprint arXiv:2211.15993, 2022.

[39] Qin Wang, Guangsheng Yu, Yi Lin, Caijun Sun, Lam Duc Nguyen, Sherry Xu, and Shiping Chen. An empirical study on snapshot daos. arXiv preprint arXiv:2211.15993, 2022.

[40] Guangsheng Yu et al. Leveraging architectural approaches in web3 applications-a dao perspective focused. In IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pages 1–6. IEEE, 2023.

[41] Zhuotao Liu, Yangxi Xiang, et al. Make web3. 0 connected. IEEE Transactions on Dependable and Secure Computing, 2021.

[42] Guangsheng Yu et al. Predicting nft classification with gnn: A recommender system for web3 assets. In IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pages 1–5. IEEE, 2023.

[43] Qin Wang, Guangsheng Yu, Shange Fu, Shiping Chen, Jiangshan Yu, and Xiwei Xu. A referable nft scheme. In 2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pages 1–6. IEEE, 2023.

[44] Zhuotao Liu, Yangxi Xiang, et al. Make web3. 0 connected. IEEE Transactions on Dependable and Secure Computing, 2021.

[45] Binance smart chain. Accessible at https://www.bnbchain.org/en/smartChain, 2022.

[46] Ethereum improvement proposals. Accessible at https://eips.ethereum.org/, 2022.

[47] Instance: Bep-20. Accessible at https://academy.binance.com/en/glossary/bep-20, 2022.

[48] Qin Wang and Guangsheng Yu. Understanding brc-20: Hope or hype. https://hal.science/hal-04216335/document, 2023.

[49] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin backbone protocol: Analysis and applications. In Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), pages 281–310. Springer, 2015.

[50] Juan Benet. Ipfs-content addressed, versioned, p2p file system. arXiv preprint arXiv:1407.3561, 2014.

[51] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. SoK: Layer-two blockchain protocols. In International Conference on Financial Cryptography and Data Security (FC), pages 201–226. Springer, 2020.

[52] Alexei Zamytatin et al. SoK: Communication across distributed ledgers. In International Conference on Financial Cryptography and Data Security (FC), pages 3–36. Springer, 2021.

[53] Gang Wang et al. Exploring blockchains interoperability: A systematic survey. ACM Computing Surveys, 2023.

[54] Lorenz Breidenbach, Christian Cachin, et al. Chainlink 2.0: Next steps in the evolution of decentralized oracle networks. Chainlink Labs, accessible at https://naorib.ir/white-paper/chainlink-whitepaper.pdf, 2021.
APPENDIX

APPENDIX-A

The entrance of the implemented service management contract is shown in Listing 1. The service management contract extends the Hyperledger Fabric Contract class. Users can invoke the functions by sending transactions containing the function names and parameters. Users can create, update, and readWithFilter objects saved on the Hyperledger Fabric. Functions are implemented in controller files.

```javascript
import { Context, Contract, Info, Returns, Transaction } from 'fabric-contract-api';
import { Controller } from 'web3-controller';

export class ServiceContract extends Contract {
  @Transaction(true)
  @Returns('string')
  public async create(ctx: Context, attributes: string, publicPayload:string): Promise<string> {
    return await new Controller('policy|object|user').createRegistry(ctx, ctx.stub.getTxID(), attributes, publicPayload);
  }

  @Transaction(true)
  @Returns('string')
  public async update(ctx: Context, dataId: string, attributes: string, publicPayload:string): Promise<string> {
    return await new Controller('policy|object|user').updateRegistry(ctx, dataId, attributes, publicPayload);
  }

  @Transaction(false)
  @Returns('string')
  public async readWithFilter(ctx: Context, publicPayload:string): Promise<string> {
    return await new Controller('policy|object|user').readRegistryWithFilter(ctx, publicPayload);
  }
}
```

Listing 1. Entry Point of the Ticket Management Contract.

The read and update policy checking is shown in Listing 2. The function checks all policies linked to the corresponding objects, which are given by the parameter policyIDs. The function fetches the policies with the policyIDs, checks user, location, validTime, updateTime, and priority, and returns the effective policy.

```javascript
public async getActiveReadAndUpdatePolicy(ctx: Context, location: string, policyIDs: string[]): Promise<any> {
  let effectivePolicy: any = {};
  let user = lla.getUserInfo(ctx);
  let ts = ctx.stub.getTxTimestamp().getSeconds() * 1000;
  let latestTimestamp = 0;
  let highestPriority = 0;
  for (let policyID of policyIDs) {
    let policyObj: any = await new pc.policy().loadPolicyFromID(ctx, policyID);
    let updateTime = await lla.getLastUpdateTime(ctx, policyID);
    let permittedUser = new userInfo(policyObj.privilege.rowPolicy.userAffiliation)
    if (user.match(permittedUser)) {
      if (location == policyObj.privilege.rowPolicy.location.countryName) {
        let startStamp = new Date(policyObj.validFor.startDateTime).valueOf();
        let endStamp = new Date(policyObj.validFor.endDateTime).valueOf();
        if (ts >= startStamp && ts <= endStamp) {
          if (policyObj.priority > highestPriority) {
            effectivePolicy = policyObj;
            latestTimestamp=updateTime;
            highestPriority=policyObj.priority;
          }
          if (policyObj.priority == highestPriority) {
            if (updateTime > latestTimestamp) {
              effectivePolicy = policyObj;
              latestTimestamp=updateTime;
            }
          }
        }
      }
    }
  }
  return effectivePolicy;
}
```

Listing 2. Policy Checking Function.
The Web2-Web3 interpreter is shown in Listing 3. When receiving a Web2 request, the interpreter creates a Hyperledger Fabric connection, including the connection profile, Fabric user, channel name, and contract name, according to the payload of the Web2 request. The interpreter then forwards the Web2 request and parameters to the corresponding functions in the Web3 contract. When receiving Web3 results, the interpreter returns them to the Web2 request.

```javascript
export async function Trans(_fabric: ITransient, context?: Required<Pick<IUserManagement, "userID" | "userPWD">>, fnc: string, ...args: string[]) { 
  let promise: Promise<string> = new Promise(async (resolve, reject) => { 
    try { 
      const fabricConCof: FabricConfiguration = new FabricConfiguration( 
        _fabric.connection, /* The Fabric connection profile */ 
        _fabric.walletPath, /* The path to Web3 credentials */ 
        _fabric.walletID, /* The Web3 user */ 
        _fabric.channelName, /* The Fabric channel name */ 
        _fabric.scName, /* The smart contract name */ 
        _fabric.listen /* Fabric event listener*/ 
      ); 
      const [gateway, connectionProfile, connectionOptions] = await getFabricConfig(context, _fabric); 
      let network = gateway.getNetwork(fabricConCof.channelName); 
      let contract = (await network).getContract(fabricConCof.scName); 
      const channel = (await network).getChannel(); 
      let result: Buffer | any; 
      result = await contract.submitTransaction(fnc, ...args); 
      await gateway.disconnect(); 
      if (result) { resolve(result.toString()); } 
      else { resolve("Invoking ${fnc} succeeds."); } 
    } catch (error) { reject(error); } 
  }); 
  return promise; 
}
```

LISTING 3. Web2-Web3 Interpreter.

APPENDIX-B

In the pursuit of gathering comprehensive insights on the topic, we reached out to a diverse group encompassing 1,000 individuals and institutions through various channels including GitHub, mailing lists, corporations, and academic universities. Despite this extensive outreach, we received a total of 58 valid responses by the stipulated deadline. While the response rate might appear limited given the vast initial pool, it’s imperative to underscore the rigor and validity of the data collected. Every response underwent meticulous validation to ensure its relevance and accuracy. Out of these valid contributions, five responses were handpicked to be showcased in this appendix due to their exemplary representativeness of the collective sentiments and insights. These five chosen responses stand as testament to the varied perspectives and depth of knowledge within our respondent pool, further bolstering the credibility of our methodologies and the consequent findings.
# First response

1. **What is your job in the organization? Technical or non-technical?**
   I am a software developer in my organization.

2. **What is your Web3 background?**
   I have 4-year developing experience on Web3 (expert)

3. **What do you think of the pros and cons of Web2 and Web3 at present?**
   **Web2:**
   Pros: mature, design patterns, tools, and workflows have been thoroughly explored and developed, and have been proven applicable for many huge and complicated tasks.
   Cons: heavily relies on trustworthy centralized entities. Building trust among parties heavily relies on social engineering, which often causes interests dispute.
   **Web3:**
   Pros: achieve trustworthy relationships among parties without centralized entities in a decentralized manner.
   Cons: immature, lacks design patterns, tools, and workflows. Popularity is not as good as Web2.

4. **What do you think of the necessity of a smooth transition between Web2 and Web3 from your organization and your personal perspectives? Web2 to Web3 and Web3 to Web2?**
   It is of importance to have a smooth transition between Web2 and Web3. Web2 project sometimes not only needs to focus on the trust issue but also needs to have an efficient incentive mechanism to encourage the public to use the services. And Web3 fits the requirement by applying decentralized ledger technology. Many Web3 applications have proven their potential in many areas including finance, asset trading, gaming, law... However, Web3 is still new to the market and needs to increase its popularity by collaborating with those giant companies and their mature products.

5. **What possible challenges during the transition do you think are required to be resolved?**
   (1) Web3 is transparent, and how to apply flexible and feasible access control becomes important for cases where data privacy is considered.
   (2) Approach to applying user management in a shared ledger may be a challenge because in Web2 parties tend to have separate user management systems local.
   (3) How to connect with existing mature products, such as SaaS, may be a challenge.
   (4) Easing the transition by using automated tools is normal in Web2 applications and is also essential during the transition between Web2 and Web3. Having such tools that can also be seamlessly integrated into existing Web2 and Web3 frameworks
may be a challenge.

6. To what extent the Service Management System developed by UTS and BT UK matches the framework and solve the questions?
   a. Access control
      Satisfied, the data-driven policy is smart.
   b. User management
      Satisfied
   c. Connection with SaaS
      Satisfied with the combination with ServiceNow which has been widely used in many existing products.
   d. Automated development tools
      TSOA is an interesting tool as Typescript is very popular and the OpenAPI standard has almost been a must-use tool during the current workflow in Web2 development.

7. What are your suggestions to enhance the suitability of the new framework WEBTTCOM and its implementation?
   The framework appears to be covering most perspectives including access control, data privacy, data provenance, user management, connections to existing SaaS, and development productivity. And the implementation also shows a good match to the framework. One point that could be improved is the extension of Web3 platforms where other platforms can be involved other than HyperLedger. Then any possible changes to the implementation to match the framework should be considered.

Second response

1. What is your job in the organization? Technical or non-technical?
   I am a researcher and software developer in my organization.

2. What is your Web3 background?
   I have 4-year research and development experience on Web3.

3. What do you think of the pros and cons of Web2 and Web3 at present?
   Web2:
   Pros: Web2 technology has been developed for decades. There are many mature solutions for various requirements and development tools for rapid development and simple management. Consumers have been well educated on the Web2 service pattern.
   Cons: The biggest challenge of Web2 systems is the trust issue. For a single Web2 service, consumers have to trust Web2 service providers, and the service providers need to trust developers, infrastructure providers, etc. When multiple Web2 systems are connecting to
each other, they need to trust the services and data from others. In the case of untrusted relations or inconsistent data, it is challenging to solve disputes and provide services as a whole.

Web3:
Pros: Web3 can provide single ground truth to all participants in a trustless way. Different parties do not have to build trust with each other.
Cons: Web3 is a new technology. The architecture and process of Web3 services are very different from those of Web2. Deploying and maintaining Web3 infrastructure and applications could be very time-consuming and risky. Consumers need to be educated about changes from Web3.

4. What do you think of the necessity of a smooth transition between Web2 and Web3 from your organization and your personal perspectives? Web2 to Web3 and Web3 to Web2?
We are aware of the benefit of Web3 and have plan to employ Web3 technology in our business to provide trusted services and reduce business loss.
A transition from Web2 to Web3 will bring the data trustworthiness and cybersecurity guarantee from Web3 to Web2 systems. The Web3 token mechanism can support incentive schemes in Web2 applications.
The smooth transition from Web3 to Web2 applications will promote Web3 technology and applications. There have been many Web3 applications, like tokens and contracts. However, Web3 applications are only popular among the Web3 community because the requirements and access to Web3 applications are very different from widely-used Web2 applications and can be hard for general users. A Web3 to Web2 transition will simplify access and encourage more users to try Web3 technology and applications.

5. What possible challenges during the transition do you think are required to be resolved?
1) How to seamlessly integrate Web3 services into existing Web2 systems
2) How to implement data governance and privacy policies in Web3
3) How to manage Web2 and Web3 user identities simultaneously
4) How to reduce integration and development cost
5) How to efficiently manage Web3 infrastructure
6) Can Web3 technology support large-scale applications

6. To what extent the Service Management System developed by UTS and BT UK matches the framework and solve the questions?
a. Access control
Satisfied. The system can enforce all the expected data governance and access control policies.
b. User management
Satisfied. A user can use one identity to access Web2 and Web3 services.
c. Connection with SaaS
Satisfied. Users can access Web3-certified data services from the SaaS. The complicated Web3 details are transparent to users.

d. Automated development tools
Satisfied. The automatic document generation technology can save 20

7. What are your suggestions to enhance the suitability of the new framework WEBTTCom and its implementation?

The developed WebttCom is a good start for transiting Web2 to Web3. I suggest the team develop more service functions to the framework, such as managing ticket files across multiple departments and organizations, such that the system usability could be improved. The framework can also be integrated with other Web2 businesses, such as workflow management. Besides new functions, comprehensive trials need to be conducted to verify the stability and capacity.

Third response

Hence, there are two options for the paper: 1) The survey section could provide an aggregated view from the project team’s view on how well the prototype addresses the needs of Web3, without attributing views to individual organizations. Or 2) On the BT side we could write a section for the evaluation part of the paper.

To give you a flavor of what a BT-authored section would look like: It would explain that Web3 is not well defined at the moment and is currently often linked to ideological aims around openness and decentralization, compared to the current internet which is seen, by the Web3 community, as dominated by “big tech”. The Web3 concept is a very nascent area at the moment where it will be interesting to see how it evolves and fits with the current Web2 world in which we live.

Web3 needs to answer a number of core questions related to complexity, scalability cost; and, critically, also cost-effectiveness and accountability, which are often more important to commercial organizations than adherence to decentralization principles. However, the lack of well-formed answers to these high-level generic considerations should not prevent us from identifying use cases and prototyping solutions to further our knowledge of the opportunities.

Looking at the Web3 technology portfolio it already includes some useful enabling technologies, including distributed ledger technologies (DLTs) which, with their inherently decentralized architecture, can be useful for specific applications. On their own they are unlikely to form the universal technology base for the next generation of the Internet; however, there are real use cases that exist today where DLTs may have an important role to play. The work with UTS is an example of this and examines a real-world use case related to IT service management across different country boundaries where different data privacy regulations apply.

In this context, DLTs are a promising solution due to the ability of smart contracts to ensure that the required country-specific data management policies are agreed upon and enforced. For the current prototyping project, it was assumed that the transition to a DLT-based solution will require interworking with the more traditionally architected solutions which are already in common usage in commercial ecosystems. In evaluating the solution
we have demonstrated that a DLT Hyperledger layer can meet the required success criteria related to cross-country access control and user management; as well as, connecting to a traditional SaaS workflow management layer.

In meeting these criteria the prototype has demonstrated sufficient promise that we are examining the next stage in research and development, which is to run test scenarios, utilizing realistic commercial data flows, through the prototype. This measured exploration and transition to DLT technologies show the path which we believe many established commercial organizations will follow as they gain confidence that the decentralized approach can meet the required business requirements. The approach of monitoring the evolving Web3 concepts, and engaging with the core technology developments, to experiment and build early insight on use cases for these emerging capabilities (e.g. DLT, smart contracts) is a pragmatic approach for ensuring that value can be gained.

Fourth response

1. **What is your job in the organization? Technical or non-technical?**
   
   I am a software developer in my organization. Technical.

2. **What is your Web3 background?**
   
   I have a half year of Web3 background. My Web3 knowledge is limited.

3. **What do you think of the pros and cons of Web2 and Web3 at present?**

   Web2:
   
   Pros: Web2 has a large development community including programmers, tools, and solutions. Web2 has been widely applied to many organizations and it has been verified as a stable technology.
   
   Cons: In the case of multiple organizations, it is not used for cross-validation and trust in the scenario.

4. **What do you think of the necessity of a smooth transition between Web2 and Web3 from your organization and your personal perspectives? Web2 to Web3 and Web3 to Web2?**

   Since most systems are based on Web2, we must focus on the transition from Web2 to Web3. The modification of the existing Web2 system should not be difficult. In order to avoid overloading the programmers with Web3 knowledge, they should not learn too much. It is also possible to apply the existing Web2 technology to Web3-based systems. It is not necessary for normal users to recognize the transition.

5. **What possible challenges during the transition do you think are required to be resolved?**
1) Because of their limited understanding of Web3, programmers are reluctant to transfer their existing Web2 systems to Web3.
2) Due to the differences in concept, technology, and tools between Web2 and Web3, it is difficult to integrate the Web2 system with the Web3 system.

6. To what extent the Service Management System developed by UTS and BT UK matches the framework and solve the questions?
   a. Access control
      Solved. Web3 implements access control, which means that Web2 programmers are not required to touch too much.
   b. User management
      Solved. It’s similar to the above. Web3 implements access control, which means that Web2 programmers are not required to touch too much.
   c. Connection with SaaS
      Solved, the modification of the existing Web2 system is not too extensive. It is also possible to apply the existing Web2 development method and architecture to the transition.
   d. Automated development tools
      Solved, automated development tools significantly reduce the programmers’ workload.

7. What are your suggestions to enhance the suitability of the new framework and its implementation?
   Attempts to integrate the existing commercial Web2 system’s functionality with Web3.
   More commercial Web2 systems should be integrated with Web3.
   Provides more Web3 services.

---

Fifth response

1. What is your job in the organization? Technical or non-technical?
   I am a developer, technical

2. What is your Web3 background?
   3 years of exposure to Web3 development

3. What do you think of the pros and cons of Web2 and Web3 at present?
   Web2: pros – mature technology, plenty of resources available; cons – being a centralized system, trust can be an issue
   Web3: pros – has built-in trust mechanisms; cons – new tech, not many resources for developers

4. What do you think of the necessity of a smooth transition between Web2 and Web3 from your organization and your personal perspectives? Web2 to Web3 and Web3 to Web2?
Certain businesses require the trusted services provided by web3, but not all. Many existing services are fine with Web2. Only services that deal with multi-organizations require web3. Web2 and web3 should be able to coexist and interact smoothly.

5. **What possible challenges during the transition do you think are required to be resolved?**
Challenges include a lack of experienced developers, a lack of available development resources, and tools.

6. **To what extent the Service Management System developed by UTS and BT UK matches the framework and solve the questions?**
   a. Access control
      Completed.
   b. User management
      Completed.
   c. Connection with SaaS
      A good start in connection with ServiceNow
   d. Automated development tools
      Some tools were developed.

7. **What are your suggestions to enhance the suitability of the new framework and its implementation?**
   Need to find suitable business use cases to demonstrate the benefit of web3.