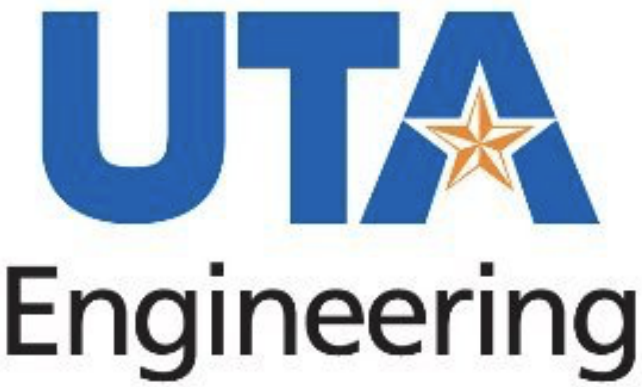
Department of Computer Science & Engineering
The University of Texas at Arlington

Architectural Design Specification
CSE 4316: Senior Design I
Spring 2024



Web3 Enjoyers
Smart Pay

Edward Alkire
Abhisek Kumar Jha
Sarah Akin
Zeynep Erguven
Kerry Pacheco

Revision History

Revision	Date	Author(s)	Description
0.1	04.04.2024	CC	document creation
0.2	04.30.2024	SA, ZE, AJ, KP, EA	complete draft

CONTENTS

List of Figures

List of Tables

1 INTRODUCTION

Smart Pay is a tool for companies to use for employment contracts. Smart pay will utilize Web3, blockchain and smart contracts. The architectural layout consists of a client, server and blockchain layer. Within those systems are many other required subsystems that all interact. Client layer servers as a front end layer and blockchain and server can be considered back end layers.

2 System Overview

1. Client Layer:

- **User Interface**: This layer provides the interface for users to interact with the Smart Pay application. It includes web pages and forms for creating and managing smart contracts, viewing organization information, checking on-chain events, and accessing user-specific data.
- **Web3 Wallet**: Users can connect their Web3 wallet to the application, allowing them to securely manage their cryptocurrency funds and interact with the blockchain.

2. Server Layer:

- **Database**: It handles user account authentication, user profile storage, payment history, and other session-related data. It stores information that is not suitable for storage on the blockchain.
- **Account Service**: Responsible for managing user accounts, including user registration, login, and profile management.
- **Web Controller**: Handles incoming requests from the user interface, processes them, and interacts with other services and the blockchain layer as necessary.
- **Authorization Service**: Ensures that users have the necessary permissions to access certain features and data within the application.
- **Blockchain Indexing Service**: Interacts with the blockchain layer to index and retrieve data from smart contracts and on-chain events. It improves application performance by storing and processing blockchain data in a format that is easily accessible to other components.

3. Blockchain Layer:

- **Smart Contract**: Contains the financial legal terms and specifications of treasury management and salary agreements translated into programming code. It defines the business logic and rules of the agreement, including fund management, salary distribution, and other financial operations. Smart contracts are executed on the blockchain, providing transparency, security, and trust-minimization for all parties involved.

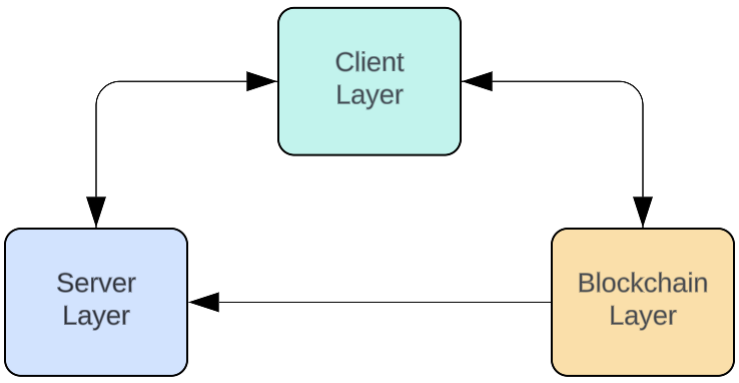


Figure 1: A simple architectural layer diagram

2.1 Block Chain Layer Systems

Smart contract encapsulates the legal terms and operational specifications of treasury management and salary agreements. These smart contracts secure financial integrity, executing predefined business logic and enforcing contractual obligations. the Smart Pay application ensures the transparency, security, and trust-minimization of all financial transactions,

2.2 Server Layer Systems

The database is serving as the repository for critical user data, including authentication credentials, profile information, and transaction histories. This layer hosts a suite of essential services, including the Account Service, responsible for managing user accounts and profiles, and the Authorization Service, ensuring secure access control to application features. The Web Controller process user requests, interfacing with external services, and coordinating interactions with the blockchain layer.

However, most vital to the efficiency and performance of the system is the Blockchain Indexing Service. By intelligently indexing and retrieving data from the blockchain, this service optimizes application responsiveness, enabling seamless access to critical on-chain information while mitigating the performance overhead associated with direct blockchain queries.

2.3 Client Layer Subsystems

The Client Layer serves as the interface through which users interact with the Smart Pay application. Here, user-friendly web interfaces empower organizations and individuals to navigate the complexities of treasury management and salary agreements with ease. Users can initiate and oversee the deployment of smart contracts, access real-time organization information, and track on-chain events pertinent to their accounts. Crucially, this layer also facilitates the integration of Web3 wallets, ensuring the seamless management of cryptocurrency funds and enabling secure interactions with the blockchain.

3 Subsystem Definitions & Data Flow

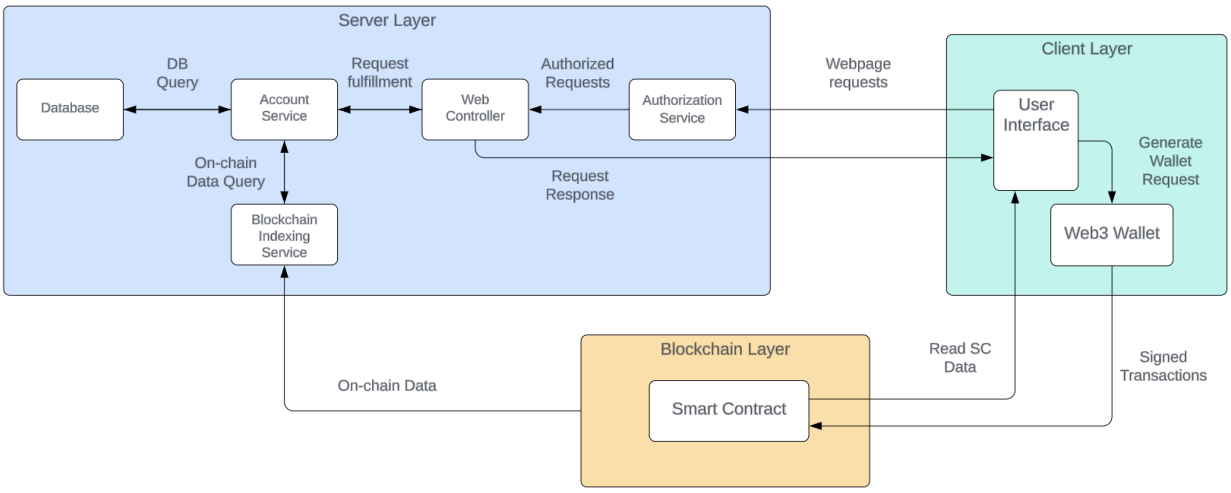


Figure 2: Smart Pay Architecture diagram

4 Blockchain Layer Subsystems

For the purposes of this project, the blockchain layer is a minor yet central component of the system. This layer is responsible for hosting smart contract deployments. A key assumption of this layer is that it will be an EVM compatible blockchain, as the other layers must be designed for compatibility with the chosen blockchain architecture. The applications server layer interfaces with the blockchain layer through the blockchain indexing service, in order to index finalized on-chain event data.

4.1 Smart Contract

Each smart contract instance will define the programmatic behavior of a financial agreement and asset management therein. The client layer may interact with a smart contract, reading the current state through RPC to a blockchain provider, or write to one by signing a transaction in their web3 wallet, submitting it to an Ethereum node.

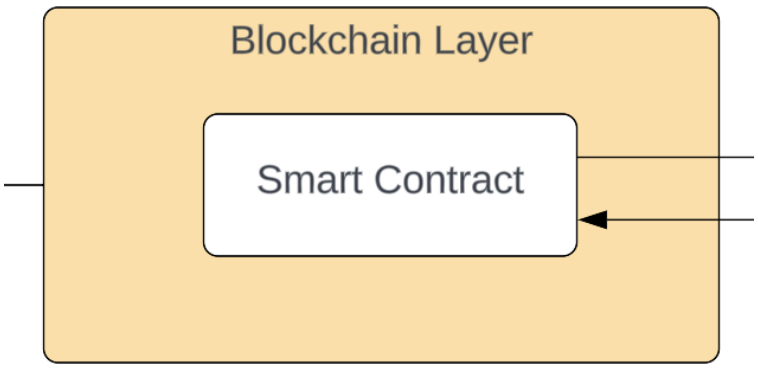


Figure 3: Blockchain layer diagram

4.1.1 Assumptions

A key assumption to the smart contract subsystem is that it will have a known or provided ABI. How the smart contract it written determines it's ABI, the ABI used by the user must be accurate to the smart contract in order to perform smart contract interactions. There are also security assumptions involved, such as smart contract code security, and security of the blockchain it is deployed to. Insecure smart contract code may result in loss of user funds, or unintended behavior. The blockchain a smart contract is deployed to is responsible for preserving integrity of SCs; as follows blockchain integrity violation such as from extended consensus failure or Sybil attack may result in loss of funds, despite smart contract security.

4.1.2 Responsibilities

Smart contract instances are responsible for managing asset custody, disbursement, and the programmed logic of financial agreement terms.

4.1.3 SUBSYSTEM INTERFACES

Table 2: Subsystem interfaces

ID	Description	Inputs	Outputs
#1	Smart Contract ABI, composed of the smart contract function signatures	function parameter input	N/A

5 Server Layer Subsystems

5.1 Database

The database stores all user profile information pertaining to individual or company users. It also has in store all financial information that will not be stored in the blockchain. The database will manage data requests from the web controller and ensure that the appropriate results are transmitted back.

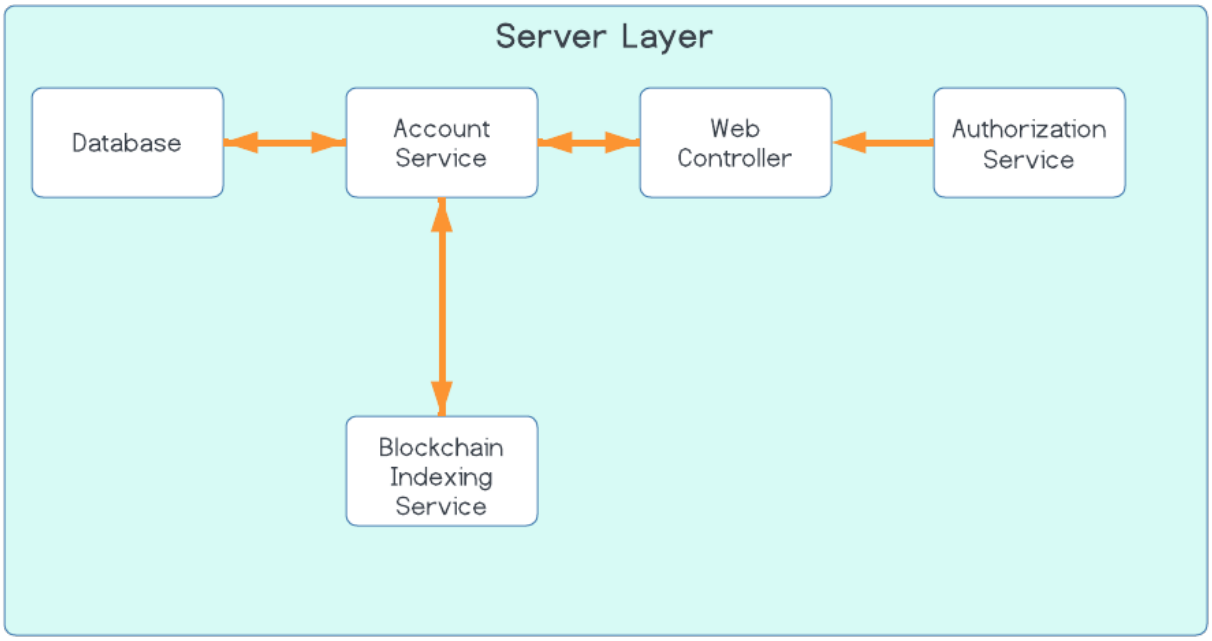


Figure 4: Database subsystem in the server layer

5.1.1 Assumptions

The Database is reliable and highly available, ensuring that salary data is always accessible when needed.
The Database preserves data integrity, it will verify authorization tokens and prevent unauthorized access and tampering.
The Database is secure, it receives previously authorized information and will follow best practices for data encryption.

5.1.2 Responsibilities

The Database will handle input queries from the Account service and process these queries; using query language it will retrieve the data requested and output it back to the account service once they are processed.
The database will store financial transaction information such as transaction IDs, recipient addresses, transaction amounts, and other relevant details. It will verify and validate these transactions before storing them on the blockchain.
The Database will ensure that user profile information and financial transaction information are recorded accurately and securely.

5.1.3 Subsystem Interfaces

Table 3: Subsystem interfaces

ID	Description	Inputs	Outputs
#2	Query requests from Account service	Data requests	Requested data response or Error message

5.2 Account Services

The account service manages previously authorized http requests that are relayed by the web controller. These requests are then delivered to the appropriate database service handlers that process these queries. The account service will also manage the Database response with the query results and relay them back to the web controller.

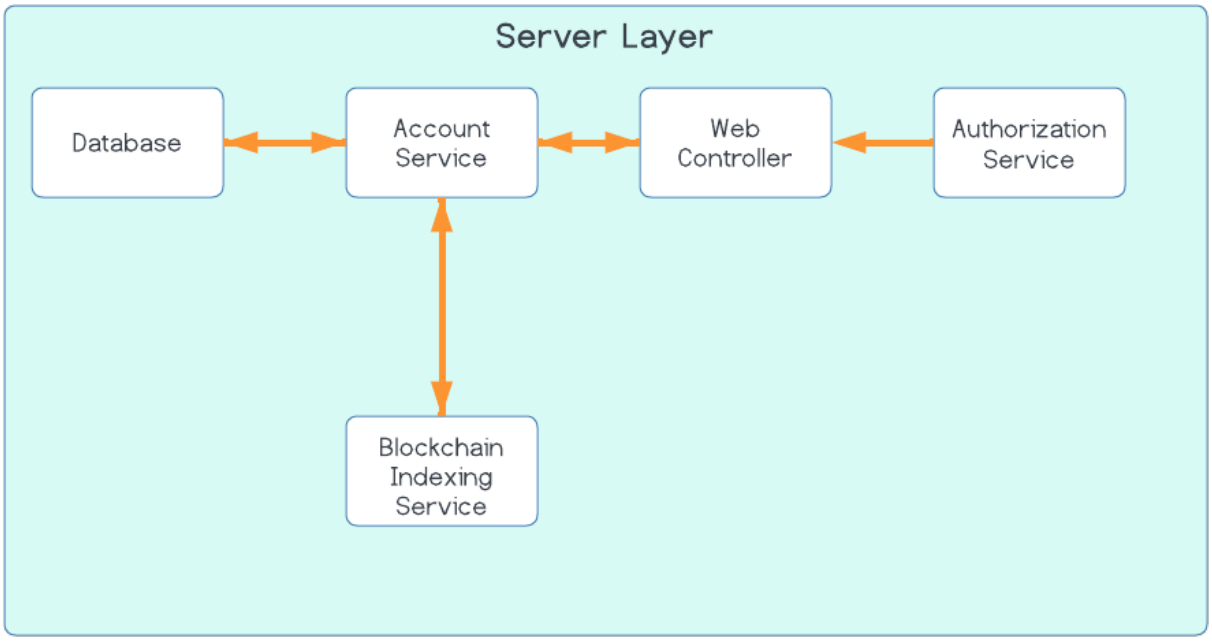


Figure 5: Account Service subsystem in the Server layer

5.2.1 Assumptions

The Account service authorizes access to user profile information and salary-related information based on user roles and permissions.
The Account service validates input data to ensure its accuracy, integrity, and compliance with predefined rules and constraints.
The Account service follows best practices for data security, protecting sensitive salary information from unauthorized access and ensuring data privacy.
The Account service will handle a large volume of salary transactions and is scale-able to accommodate future growth.

The account service interacts seamlessly with the blockchain service to record and validate salary transactions, ensuring transparency and immutability.
The account service will provide informational error messages when errors occur.

5.2.2 Responsibilities

Validate and Relay http requests previously authorized by the web controller to the appropriate database handler.
Manage user salary information, including salary amounts, payment schedules, and payment history that will be stored in the database. Validate and sanitize input data to prevent security vulnerabilities and ensure data integrity.
Handle errors and exceptions if they occur, then provide informative error messages to users and administrators.
Interface with the blockchain service to record salary transactions securely and transparently. Verify and validate salary transactions before recording them on the blockchain.
Communicate with the database subsystem to store and retrieve user account information and salary data.
Follow best practices for data encryption, user authentication, and access control.

5.2.3 Subsystem Interfaces

Table 4: Account Service Subsystem interfaces

ID	Description	Inputs	Outputs
#3	HTTP requests from Web Controller	Data request	Data requested or Error message
#4	Relay data request to Database	N/A	Data requested or Error message
#5	Relay data request to Blockchain	N/A	Data requested or Error message

5.3 Web Controller

The Web Controller manages the incoming HTTP requests, directing them to the appropriate service handlers based on the requests details. It ensures that the requests are authenticated and authorized before processing.

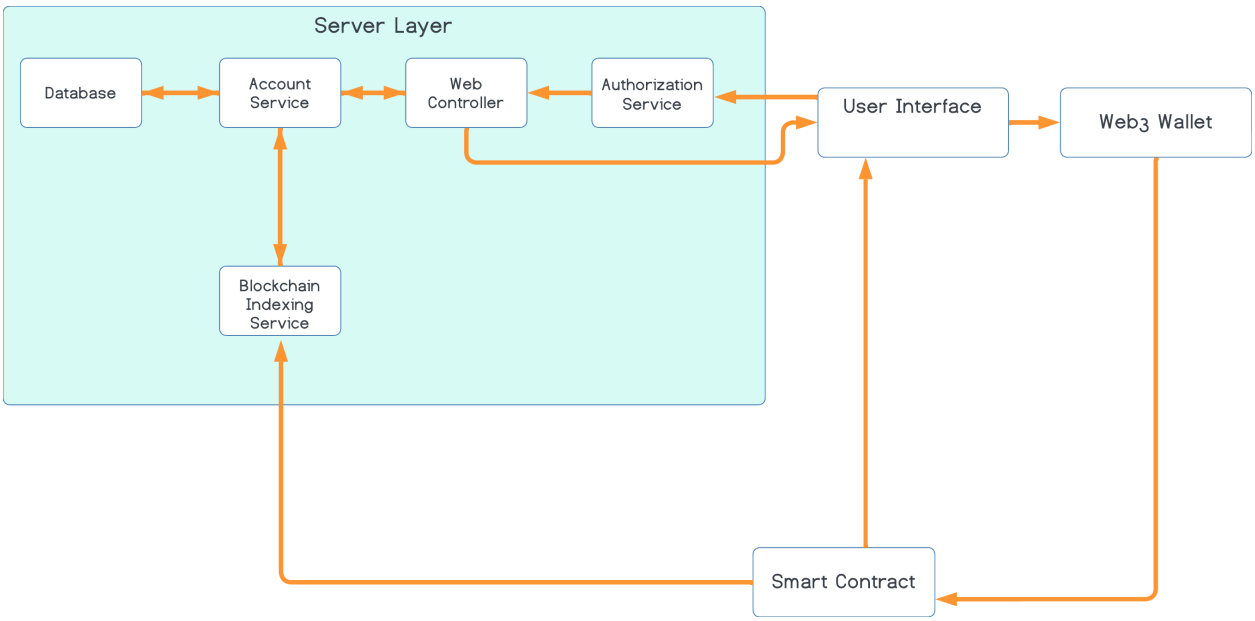


Figure 6: Web Controller in Server Layer

5.3.1 Assumptions

All the incoming requests are well-formed according to the HTTP standard. The authentication details like a tokens should accompany all requests that needs secure access. The Web Controller can handle multiple requests simultaneously without performance degradation.

5.3.2 Responsibilities

- Routing: It dynamically determines the correct service endpoints for each request based on the request URL and HTTP method.
- Authentication: The Web Controller validates access credentials provided in the headers or body of the request prior to any service-specific processing. This step is crucial for ensuring that subsequent operations are securely executed under the identity of verified users.
- Logging: It maintains the comprehensive logs of all the incoming requests and outgoing responses. This is critical for troubleshooting, analysis, and maintaining the integrity of the system against potential security threats.

5.3.3 Web Controller Interfaces

Table 5: Web Controller interfaces

ID	Description	Inputs	Outputs
#01	Request Reception Interface	HTTP Requests	Request Data
#02	Response Dispatch Interface	Processed Data	HTTP Responses

5.4 Authorization Service

The Authorization Service is responsible for verifying the user's credentials and permissions. It ensures that users can only access features and data they are permitted to.

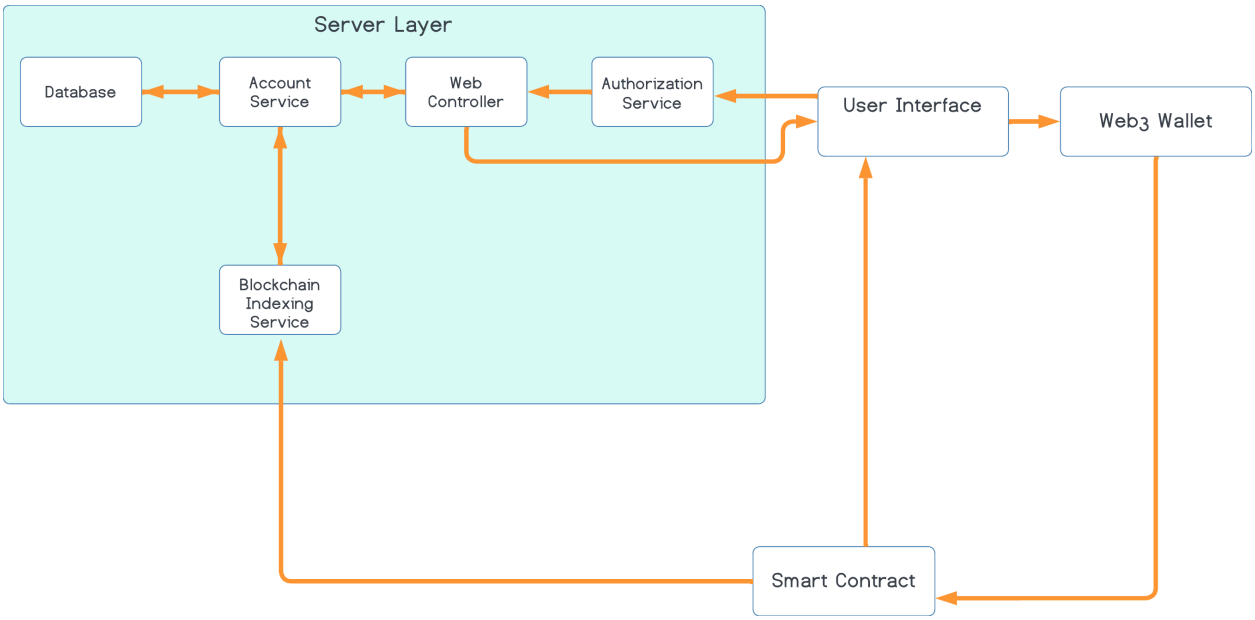


Figure 7: Authorization service in Server Layer

5.4.1 Assumptions

User roles and permissions are predefined and stored securely. Each request includes a valid authentication token or credentials. The authorization rules are complex and require dynamic checking against the user's context and requested action.

5.4.2 Responsibilities

- Token Validation: This involves decoding tokens, verifying signatures, and checking the token against a revocation list if necessary.
- Permission Checks: It determine if a user has the necessary permissions for a requested action. This include complex rule evaluation that might consider the user's role, the context of the request, and other security policies.
- Security Audits: It systematically records all the authorization decisions and the reason behind them.

5.4.3 Subsystem Interfaces

Each of the inputs and outputs for the subsystem are defined here. Create a table with an entry for each labelled interface that connects to this subsystem. For each entry, describe any incoming and outgoing data elements will pass through this interface.

Table 6: Subsystem interfaces

ID	Description	Inputs	Outputs
#03	Permission Validation Interface	User ID	Authorization Decision
#04	Token Inspection Interface	Access Tokens	Validation Results

5.5 Blockchain Indexing Service

The blockchain indexing service indexes blockchain data in a usable and queryable format.

5.5.1 Assumptions

For this project we are making the assumption that we will be able to use a third party service for blockchain indexing of smart contract data.

5.5.2 Responsibilities

This service will be responsible for aggregating and making on-chain data relative to specified smart contracts queryable. Given a specified smart contract, it retrieving true and complete data according to the requested parameters.

Table 7: Subsystem interfaces

ID	Description	Inputs	Outputs
#1	Indexed Data Query	Smart contract address Query parameters	Requested Data

6 Client Layer Subsystems

The Client Layer interacts with both the Server Layer and Blockchain Layer. The Client Layer consists of the User Interface and Web3 Wallet. The User Interface receives inputs from the User, Web Controller and Smart Contract. The User Interface outputs to Web3 Wallet and Authorization Service.

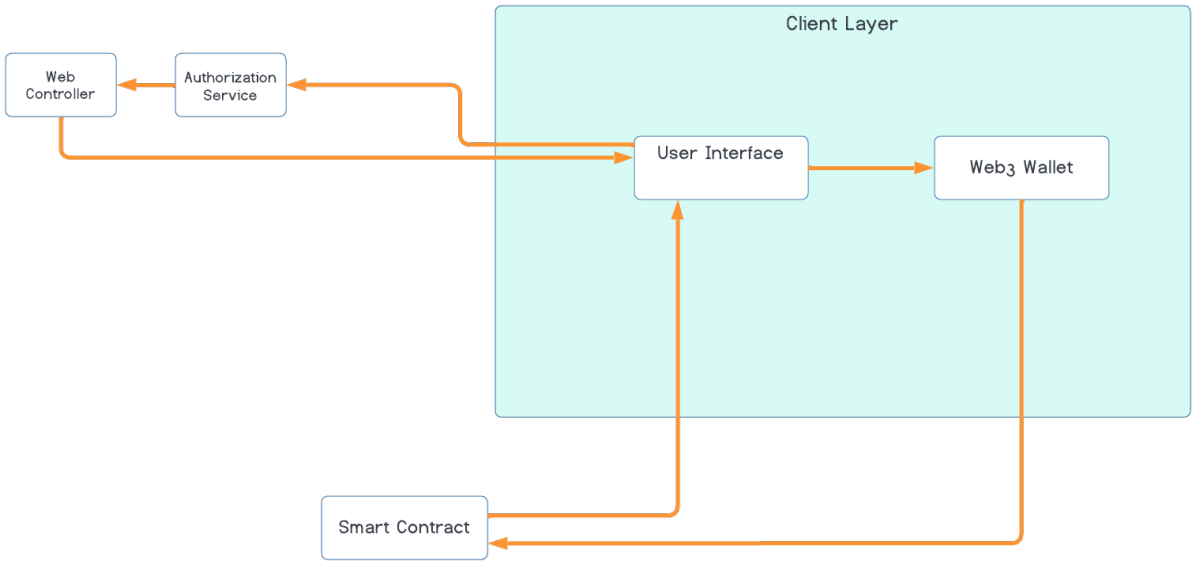


Figure 8: Client Layer diagram

6.1 User Interface

The User Interface interacts directly with Authorization Service, Web Controller, Smart Contract and Web3 Wallet.

6.1.1 Assumptions

The assumption is that the Authorization Service will be a barrier between the database, causing the Client Layer never to directly interact with it.

6.1.2 Responsibilities

The User Interface will be the only thing system collecting and sending data from the user. It will collect login information, and data requested, such as creating a new employment contract or viewing that contract later on.

6.1.3 Subsystem Interfaces

The following are the inputs and outputs received from other subsystem interfaces.

Table 8: User Interface

ID	Description	Inputs	Outputs
#7.1	User	Logins Data requested	Data requested Error messages
#7.2	Web3 Wallet		Data requested
#7.3	Authorization Service		Logins
#7.4	Web Controller		Data requested Error messages
#7.5	Smart Contract	Data requested	

6.2 Web3 Wallet

The Web3 Wallet interacts directly with the User Interface, and Smart Contract.

6.2.1 Assumptions

The assumption is that the User Interface will not allow requests for data to be sent to the Web3 Wallet if the user does not have access. The Web3 Wallet will not make authorization decisions.

6.2.2 Responsibilities

The Web3 Wallet will act as a connector to the Ethereum network and Smart Contracts. It will receive the requested data from the User Interface and pass the request on to the Smart Contract to receive.

6.2.3 Subsystem Interfaces

The following are the inputs and outputs received from other subsystem interfaces.

Table 9: Web3 Wallet

ID	Description	Inputs	Outputs
#8.1	User Interface	Data requested	
#8.2	Smart Contract		Data requested

References