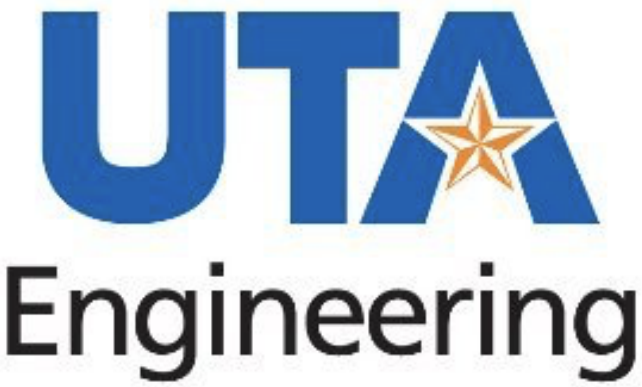
Department of Computer Science & Engineering
The University of Texas at Arlington

System Requirements Specification
CSE 4316: Senior Design I
Spring 2024



Web3 Enjoyers
Smart pay

Edward Alkire
Abhisek Kumar Jha
Kerry Pacheco
Zeynep Erguven

Revision History

Revision	Date	Author(s)	Description
0.1	03.31.2024	EA	complete draft

CONTENTS

List of Figures

1 Product Concept

This section describes the purpose, use, and intended user audience of Smart Pay. The product is designed to simplify providing financial transparency through smart contracts, in order to introduce a level of transparency unprecedented in the traditional financial system. The purpose, use, and intended audience is described here.

1.1 Purpose and Use

The product will provide a user interface for viewing financial data and insights about organizations connected to the platform. The purpose is to support the use of smart contracts for transparent financial management and trust-minimized financial agreements, in order to be used as a medium for publicly sharing organizations' "digital identity" as related to their verifiable financial status and reputation.

1.2 Intended Audience

The product will be made available publicly as a web application, with an intended audience in organizations that would benefit the most by being able to offer superior financial transparency, such as non-profit organizations, which more than other kinds of organizations, rely on trust and their reputation for their funding. The primary intended users are organization administrators and members of the public; with a potential following form of user being individuals with financial ties to an organization, such as employees with salary agreements.

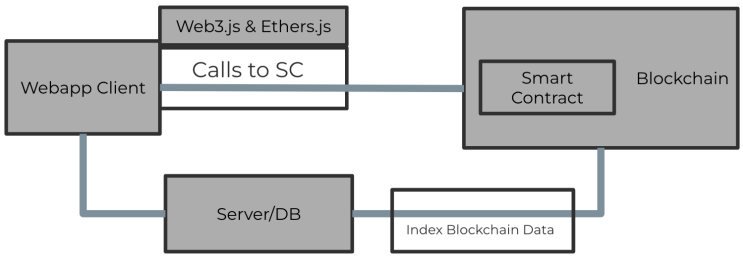


Figure 1: Conceptual diagram

2 Product Description

This section provides the reader with an overview of Smart Pay. The primary operational aspects of the product, from the perspective of end users, maintainers and administrators, are defined here. The key features and functions found in the product, as well as critical user interactions and user interfaces are described in detail.

2.1 Features & Functions

Smart Pay is a web application for monitoring and interfacing with smart contracts. The principle components and how they are connected is depicted in Figure 1.

Essential features and functionality include:

- Support account creation for user types such as individuals and organizations.
- Aggregate users on-chain financial data, including live status and historical insights.
- Publicly searchable user accounts to discover and share user accounts

To support historical data aggregation blockchain indexing will be required, which is non-trivial since blockchains are typically not designed with a standard query language to use. This challenge may be mitigated by using a third party smart contract indexing service.

2.2 External Inputs & Outputs

Name	Description	Use
User Data	Data associated to a user to display when a user account is viewed	Output
Account Address	On-chain address to associate to a user, possibly input by users wallet	Input
Blockchain Data	Data indexed from the blockchain to present smart contract history relevant to users	Input

2.3 Product Interfaces

The default user interface will display status of top or featured organizations. Logged in users will be able to access a private account interface differing by account type (such as personal or organization). Each user account will be searchable to view their public information (especially including verified financial history such as salary agreement history for personal accounts or treasury and transaction history for organizations).

3 Customer Requirements

In the development of the Smart Pay application, our goal is to meet and exceed the expectations of our users. This application is designed to provide organizations with the ability to manage their treasury and salary agreements transparently and securely using blockchain technology. Below are specified customer requirements for the Smart Pay project, each designed to fulfill a specific customer need, ensuring that users have a consistent and satisfying experience.

3.1 GUI Design Consistency

3.1.1 Description

The GUI (Graphical User Interface) of the Smart Pay web application will feature a slate blue background. This color choice is made to ensure consistency with other similar software products offered by the customer, promoting a uniform brand experience. Slate blue is specified as #007FFF, adhering to the six-digit hexadecimal color specification. Graphics may be included to enhance understanding and appeal to the user interface.

3.1.2 Source

This requirement is specified by the customer to align with their existing suite of software tools and branding guidelines.

3.1.3 Constraints

Design and development efforts must consider economic factors, such as the cost of implementing custom UI components, and sustainability, focusing on long-term maintenance and compatibility with various devices and browsers.

3.1.4 Standards

Adheres to *Web Content Accessibility Guidelines (WCAG) [3].* for color contrast and visibility, ensuring that the application is accessible to a wide range of users, including those with visual impairments.

3.1.5 Priority

High - Essential for customer acceptance and brand consistency.

3.2 Transparent Treasury Management

3.2.1 Description

The application will enable organizations to deploy smart contracts for transparent and public treasury management. Users should be able to view detailed transaction histories, current balances, and pending transactions in real-time, ensuring complete transparency.

3.2.2 Source

Derived from the core project goal to enhance transparency in financial transactions for organizations, as identified by the project team and endorsed by the customer.

3.2.3 Constraints

Must ensure user privacy and data security, comply with financial regulations and blockchain protocol limitations, and manage scalability to support a growing number of transactions.

3.2.4 Standards

Complies with blockchain industry standards for smart contracts, such as *Ethereum's ERC standards* [1], and adheres to financial regulatory standards applicable to public financial disclosures.

3.2.5 Priority

Critical - A foundational feature of the product, without which the project would not meet its intended purpose.

3.3 Real-Time Salary Agreements

3.3.1 Description

The system will support trust-minimized, smart contract-based salary agreements, allowing employees and employers to set up, manage, and view salary distributions in real time. This includes the capability for organizations to stream salaries to employees, adjusting in real-time based on agreed-upon conditions.

3.3.2 Source

This requirement stems from the need to improve trust and transparency between employers and employees regarding salary agreements, a need identified by both the project team and potential end-users.

3.3.3 Constraints

Implementation must consider the technical feasibility of real-time streaming on various blockchain platforms, economic implications for transaction fees, and legal compliance with employment laws.

3.3.4 Standards

Must comply with *employment law standards and regulations* [2] in jurisdictions where the application will be used, as well as technical standards for secure and efficient blockchain transactions.

3.3.5 Priority

Critical - Essential for achieving the project's value proposition of trust minimization in salary agreements.

4 Packaging Requirements

Include a header paragraph here. Packaging requirements are those requirements that identify how the delivered product will be packaged for delivery to the end-user; or how it will "look" when finished and delivered. For example, you might specify that the software required for operation will be pre-loaded on the hard drive, delivered on CD/DVD, or available via download. Software might be customer installable, or not, etc. Hardware components could be all in a single package, provided as a "bag of parts" to be assembled/installed by the user, painted a certain color, logos affixed, etc. Care should be taken not to duplicate requirements found in other sections of this document.

4.1 Web-app design

4.1.1 Description

The product will be delivered as a high quality and maintainable web application

4.1.2 Source

Customer expectations

4.1.3 Constraints

The application front end will be developed using industry standard web frameworks such as React

4.1.4 Priority

High

5 Performance Requirements

The performance requirements of this product revolve around user expectations. Meeting these expectations is not expected to be challenging, but necessary to document since user expectations should not be an afterthought in a user oriented product.

5.1 User Experience

5.1.1 Description

The user experience must be responsive.

5.1.2 Source

Customer expectations

5.1.3 Constraints

Latency in loading pages must be less than five seconds.

5.1.4 Priority

High

6 Safety Requirements

To make a secure and reliable product, developers must consider safety requirements that affect the overall security of the smart contract, and also target specific code security issues. Code vulnerabilities can arise from bugs in the contract's code, which can lead to security risks. It is crucial to thoroughly examine and test the code using various security tools and methods. The developers will follow the following requirements to ensure the integrity of the system:

6.1 Data Security

Ensure that all the customers sensitive data such as: salary information, bank information, and personal information are encrypted upon storage and during use to prevent unauthorized access.
Implement secure communication protocols, such as HTTPS, to protect data transmitted between users and the smart contract. Utilize encryption and hashing techniques to safeguard data integrity and confidentiality during transmission over networks.

6.1.1 Source

This requirement is specified by the development team in conjunction with the customer expectation to have data privacy and security as a priority.

6.1.2 Constraints

Data encryption requires experienced developers to program strong encryption algorithms to encrypt salary information before it is stored within the smart contract or transmitted over the network. Also they need to ensure that secure communication channels are used to transmit salary information to and from the smart contract.

6.1.3 Standards

adhere to industry standards and best practices for data encryption and security. This includes following guidelines from organizations like NIST (National Institute of Standards and Technology)

6.1.4 Priority

Critical

6.2 Account Authentication

Implement robust authentication mechanisms to verify the identity of users accessing the system. Employ role-based access control (RBAC) to restrict network availability and enforce appropriate levels of access to different functionalities and data within the system. Additionally, developers will consider implementing multi-factor authentication (MFA) for enhanced security.

6.2.1 Source

This requirement is specified by the development team in conjunction with the customer expectation to have robust methods for account authentication.

6.2.2 Constraints

The complexity of blockchain technology and smart contracts may present usability challenges for users, particularly employees and HR administrators unfamiliar with decentralized systems.

6.2.3 Standards

Developers must design and implement User-friendly interfaces essential for enhancing user experience. In addition, account authentication should employ various methods of authentication such as digital signature, Role-Based access control and multi-factor authentication to protect user accounts.

6.2.4 PRIORITY

Critical

6.2.5 STORAGE

Smart contract storage layout refers to data structure rules that will dictate how variables are laid out in long-term memory. Salary based smart contracts have state variables that need to be stored long-term. Efficient storage layout(gas efficiency) minimizes the cost of storing data in the blockchain's long-term memory, which is inherently expensive. Adhering to storage layout rules allows developers to optimize gas usage and maximize cost savings.
Different design patterns like proxy or diamond patterns utilize specific storage layout configurations. Familiarity with these patterns and their corresponding storage layouts enables developers to implement them effectively in their contracts.
External Interface Consideration: The layout of state variables contributes to the contract's external interface alongside public functions and variables. Changes in storage layout, influenced by compiler updates, impact the contract's interface and, consequently, its interoperability with other contracts and systems.

6.2.6 SOURCE

This requirement is specified by the development team to set a uniform method to store data so it will be better organized.

6.2.7 CONSTRAINTS

Creating effective storage structures to maximize memory capacity.

6.2.8 STANDARDS

Adheres to industry standards and best practices for secure coding.

6.2.9 PRIORITY

High

6.3 IMMUTABILITY

Design smart contracts with appropriate logic to ensure immutability once deployed on the blockchain. Prevent unauthorized modifications to the contract code or its state to maintain integrity and trust in the system. Utilize blockchain platforms with built-in immutability features, such as Ethereum, and employ smart contract upgrade patterns, like proxy contracts, if necessary.

6.3.1 SOURCE

This requirement is specified by the development team to make sure the smart contract cannot be altered without authorization.

6.3.2 CONSTRAINTS

Requires detailed input from the customer to make sure the regulations and specifications that are set into the smart contract are fully understood by all parties.

6.3.3 STANDARDS

Adheres to industry standards and best practices for secure coding.

6.3.4 PRIORITY

High

6.4 AUDITING AND TRACING

Incorporate features to enable auditing and tracing of salary payments and transactions. Maintain comprehensive logs of all salary-related activities, including payment initiation, approval, and disbursal, to facilitate auditing and dispute resolution. Implement event logging and blockchain explorers to provide transparent visibility into contract activities and transaction histories.

6.4.1 SOURCE

This requirement is specified by the development team in conjunction with the customer to make sure there is a history to refer to if an issue where to arise.

6.5 CONSTRAINTS

The complexity of blockchain technology and smart contracts may present usability challenges for users, particularly employees and HR administrators unfamiliar with decentralized systems. Storage size and memory capacity for data history has to be taken into account when creating a user's history.

6.5.1 STANDARDS

Developing smart contracts for salary payments requires clear legal and contractual frameworks to define rights,obligations, and dispute resolution mechanisms. Developing a product that complies with these regulations can be complex and requires thorough understanding and adherence to relevant laws in different jurisdictions.

6.5.2 PRIORITY

Critical

6.6 INPUT VALIDATION

Perform thorough validation of input data to prevent common vulnerabilities such as input validation attacks, buffer overflows, and injection attacks. Ensure that all input parameters adhere to defined constraints and standards. Implement input sanitation techniques, such as input validation libraries or regular expression checks, to mitigate the risk of malicious input manipulation.

6.6.1 SOURCE

This requirement is specified by the development team to have robust methods for input validation, this requirement is tightly related to account authentication.

6.6.2 CONSTRAINTS

Developers experience with secure methods for input validation.

6.6.3 STANDARDS

developers must adhere to general best practices and principles for secure coding. These include type checking (input parameters should match the expected type), range checking which prevents overflow, boundary testing (uncovers vulnerabilities) and immutable validation logic.

6.6.4 PRIORITY

Critical

6.7 ERROR HANDLING

Implement comprehensive error handling mechanisms in both front end and back end to manage unexpected situations and errors. Provide informative error messages to users and log detailed error information for debugging and auditing purposes. Implement fail-safe mechanisms to handle critical errors and prevent system downtime or data corruption.

6.7.1 Source

This requirement is specified by the development team to have robust methods for error handling, this requirement is tightly related to input validation.

6.7.2 Constraints

Developers experience with secure methods for error handling.

6.7.3 Standards

developers must adhere to general best practices and principles for secure coding.

6.7.4 Priority

Critical

7 Maintenance & Support Requirements

For the Smart Pay application, ensuring robust maintenance and support post-deployment is crucial for sustaining long-term user satisfaction and product reliability. These requirements aim to facilitate efficient troubleshooting, updates, and user support, ensuring that the application remains secure, functional, and responsive to user needs over time. This includes outlining clear protocols for updating the software, addressing potential security vulnerabilities, providing user support, and managing hardware or infrastructure dependencies.

7.1 Source Code Accessibility and Documentation

7.1.1 Description

The source code for the Smart Pay application will be thoroughly documented and accessible to authorized maintenance personnel. This includes inline comments explaining critical sections of the code, as well as a comprehensive README file that details the software architecture, dependencies, build instructions, and deployment processes. This requirement ensures that maintainers can understand and navigate the codebase efficiently for updates or troubleshooting.

7.1.2 Source

This requirement is specified by the development team, recognizing the importance of maintainability and ease of knowledge transfer in software projects.

7.1.3 Constraints

There may be intellectual property concerns that restrict access to certain parts of the source code to internal team members only. Additionally, ensuring the documentation remains up-to-date with the codebase can be challenging and requires dedicated resources.

7.1.4 Standards

It adheres to industry best practices for software documentation, such as those outlined in the IEEE Software Engineering Standards for documentation.

7.1.5 Priority

High - It is essential for effective maintenance and support.

7.2 Technical Support Training Materials

7.2.1 Description

Create and maintain a set of training materials for technical support staff, including troubleshooting guides, frequently asked questions (FAQs), and scenario-based resolution paths. These materials should be easily accessible, possibly through an internal wiki or knowledge base, and regularly updated to reflect new issues, features, or changes in the application.

7.2.2 Source

It is derived from anticipated customer support needs and the necessity for efficient resolution of user issues to maintain high levels of customer satisfaction.

7.2.3 Constraints

Producing and maintaining these materials requires ongoing effort and coordination between the development team and customer support personnel. Keeping the content current and comprehensive may challenge as the application evolves.

7.2.4 Standards

Compliance with internal training standards and best practices in customer support documentation.

7.2.5 Priority

Moderate - It is important for customer satisfaction but secondary to direct product functionality.

7.3 Maintenance Environment Specification

7.3.1 Description

Specify the software environment and tools required for maintaining the Smart Pay application, including development environments, version control systems, debugging tools, and any proprietary software used in the development process. This ensures that maintenance personnel have the necessary tools and environments set up for effective troubleshooting and updates.

7.3.2 Source

It is defined by the development team based on the technical stack and tools used in the application's development.

7.3.3 Constraints

The need to license proprietary software or tools for maintenance personnel, which may incur additional costs.

7.3.4 Standards

It adheres to the software development life-cycle management and IT operations management standards relevant to the application's technology stack.

7.3.5 Priority

Moderate - It is necessary for maintenance activities but adaptable based on available resources and tools.

8 Other Requirements

The Smart Pay salary-based smart contract main goal is to provide a secure, efficient, and transparent solution for automating salary payments on the Ethereum blockchain. Other requirements that can help achieve that are the following:

8.1 Salary Calculation and Disbursement

8.1.1 Description

The smart contract should calculate and disburse employee salaries based on predefined rules and parameters. Salary calculations should consider factors such as employee role, salary, bonuses, deductions, and currency conversion if applicable.

8.1.2 Source

The development team recognizes the critical importance of having accurate salary calculations to avoid conflicts.

8.1.3 Constraints

Salary information and calculation is dependent on the company policy and agreement with the employee. The developers must assure to have access to this information in order to design and implement the appropriate contract.

8.1.4 Standards

Adhere to company policy, employee agreement and adherence to labor laws.

8.1.5 Priority

High

8.2 Employee Registration

8.2.1 Description

The smart contract should allow employers to register employees by storing their relevant information, including name, address, bank account details, and employment contract terms. Employee registration should include validation checks to ensure data accuracy and prevent duplicate entries.

8.2.2 Source

Derived from the need to create unique entries in the employee database.

8.2.3 Constraints

Employers diligence to create, and update the employee database to maintain accurate employee information.

8.2.4 Standards

Adhere to company policy, employee agreement and adherence to labor laws.

8.2.5 Priority

Moderate

8.3 Payroll Period Management

8.3.1 Description

Employers should be able to define and manage payroll periods, including start and end dates, frequency (e.g., monthly, bi-weekly), and payment deadlines. The smart contract should enforce payroll period constraints to prevent overlapping or missed payroll schedules.

8.3.2 Source

The developers recognize the need for employees to be paid on time and take that into account when designing the contract.

8.3.3 Constraints

Developers are dependent on the accuracy of the information given by the company.

8.3.4 Standards

Adhere to company policy, employee agreement and adherence to labor laws.

8.3.5 Priority

High

9 Future Items

9.0.1 Description

Create and maintain a set of training materials for technical support staff, including troubleshooting guides, frequently asked questions (FAQs), and scenario-based resolution paths. These materials should be easily accessible, possibly through an internal wiki or knowledge base, and regularly updated to reflect new issues, features, or changes in the application.

9.0.2 Source

It is derived from anticipated customer support needs and the necessity for efficient resolution of user issues to maintain high levels of customer satisfaction.

9.0.3 Constraints

Producing and maintaining these materials requires ongoing effort and coordination between the development team and customer support personnel. Keeping the content current and comprehensive may challenge as the application evolves.

9.0.4 Standards

Compliance with internal training standards and best practices in customer support documentation.

9.0.5 Priority

Moderate - It is important for customer satisfaction but secondary to direct product functionality.

9.1 Maintenance Environment Specification

9.1.1 Description

Specify the software environment and tools required for maintaining the Smart Pay application, including development environments, version control systems, debugging tools, and any proprietary software used in the development process. This ensures that maintenance personnel have the necessary tools and environments set up for effective troubleshooting and updates.

9.1.2 Source

It is defined by the development team based on the technical stack and tools used in the application's development.

9.1.3 Constraints

The need to license proprietary software or tools for maintenance personnel, which may incur additional costs.

9.1.4 Standards

It adheres to the software development life-cycle management and IT operations management standards relevant to the application's technology stack.

9.1.5 Priority

Moderate - It is necessary for maintenance activities but adaptable based on available resources and tools.

9.2 Employee Registration

9.2.1 Description

The smart contract should allow employers to register employees by storing their relevant information, including name, address, bank account details, and employment contract terms. Employee registration should include validation checks to ensure data accuracy and prevent duplicate entries.

9.2.2 Source

Derived from the need to create unique entries in the employee database.

9.2.3 Constraints

Employers diligence to create, and update the employee database to maintain accurate employee information.

9.2.4 Standards

Adhere to company policy, employee agreement and adherence to labor laws.

9.2.5 Priority

Moderate

REFERENCES

[1] Ethereum Foundation. ERC-20 Token Standard, 2018. Accessed: 3/29/2024.

[2] U.S. Department of Labor. Digital Platforms and the Future of Labor Law Compliance, 2020. Accessed: 3/29/2024.

[3] Web Accessibility Initiative (WAI). Web Content Accessibility Guidelines (WCAG) 2.1, 2018. Accessed: 3/29/2024.