# Software Requirements Specification Document
for
BitAuction

Version No. 1.0 **Date: 23/2/2025**

Prepared by:

Amr Ahmed
Fareeda Ragab
Joseph Shokry
Mohamed Arous
Michael Monir
Omar Tammam

For Department of: Computer and Systems Department

Submitted to: Professors **Date: 17/3/2025**

Document Release Notice

Project Document:

Amr Ahmed

Fareeda Ragab

Joseph Shokry

Mohamed Arous

Michael Monir

Omar Tammam

Details

Document ID	**Version No.**	**Description**
	V1.0	This Document describes the SRS of our BitAuction platform

APPROVED BY: Joseph Shokry

DATE: 24 Feb 2025

DOCUMENTATION VERSION CONTROL

VERSION	DATE	DESCRIPTION	PAGE No. & Item No.	PREV. PAGE No.	REASON FOR CHANGE
1.0	23/2/2025	First Version	14	N/A	N/A
1.1	24/2/2025	Second Version	14	14	- Removed the 2-factor authentication of the system. - Add new functionality allowing the user to withdraw his bids back before the auction ends.
1.2	14/3/2025	1.2	14	14	- Removed admin role

1. Introduction.. 5
1.1 Purpose.. 5
1.2 Scope.. 5
1.3 Definitions, Acronyms, and Abbreviations.. 6
1.4 References.. 6
1.5 Overview.. 6
2. The Overall Description.. 6
2.1 Application Software Perspective.. 6
2.1.1 System Interfaces.. 7
2.1.2 User Interfaces.. 7
2.1.3 Hardware Interfaces.. 7
2.1.4 Software Interfaces.. 7
2.1.5 Communications Interfaces.. 8
2.1.6 Memory Constraints.. 8
2.1.7 Operations.. 8
2.1.8 Site Adaptation Requirements.. 8
2.2 Application Software Functions.. 9
3. Specific Requirements.. 10
3.1 External interfaces.. 10
Inputs and Outputs.. 10
3.2 Functional Requirements.. 12
Bidding Process (Two-Stage Bidding).. 12
Auction Management.. 12
Bidding Process.. 12
Transaction Settlement.. 12
Ordering Service Affairs (Timestamping & Ordering).. 12
3.3 Performance Requirements.. 13
3.4 Logical Database Requirements.. 13
3.5 Design Constraints.. 13
3.5.1 Standards Compliance.. 13
3.6 Software System Attributes.. 13
3.6.1 Reliability.. 13
3.6.2 Availability.. 13
3.6.3 Security.. 13
3.6.4 Maintainability.. 14
3.6.5 Portability.. 14
4. Change Management Process.. 14

1. Introduction

1.1 Purpose

The purpose of this **Software Requirements Specification (SRS)** is to provide a clear and detailed description of the functional, non-functional, and system requirements of the **BitAuction** platform. This document serves as a reference for developers, project managers, quality assurance teams, and stakeholders to ensure a shared understanding of the system's objectives, constraints, and functionalities.

The intended audience includes:

- **Developers** implementing the system.
- **Project Managers** overseeing the development lifecycle.
- **Quality Assurance Teams** ensuring compliance with defined requirements.
- **End Users** who need an understanding of the system's functionality.

1.2 Scope

BitAuction is a **web-based decentralized auction platform** leveraging **Hyperledger Fabric** to provide a secure and transparent environment for conducting auctions. The system will support **only open outcry auctions**, ensuring all bids are publicly declared in real-time. The primary functionalities include:

1. **Auction Management:** Sellers list items for auction with parameters such as starting bid, reserve price, and auction duration.
2. **User authentication and identity managemen**t: via Web3 wallets.
3. **Auction Creation:** for sellers to list digital assets.
4. **Bidding Process:** Buyers place bids in real-time, with validation and notifications.
5. **Transaction Settlement:** Smart contracts process payments and asset transfers securely.
6. **Security & Integrity:** Smart contract-based escrow, multi-factor authentication, and fraud prevention mechanisms.
7. **Ordering & Timestamping:** Integration with NTP servers via oracles ensures precise transaction ordering.
8. **Data Storage:** Auction records are stored immutably using Hyperledger Fabric and IPFS.

The system **will not** support private, sealed-bid, or silent auctions, as it is exclusively designed for **open outcry auctioning**.

The system **will not** handle physical goods or fiat currency transactions. It is designed to operate exclusively with **digital assets** and **cryptocurrencies.**

1.3 Definitions, Acronyms, and Abbreviations

- **NFT**: Non-Fungible Token
- **SRS**: Software Requirements Specification.
- **Smart Contract**: Self-executing contracts with the terms of the agreement directly written into code.
- **Web3**: Blockchain-based authentication framework
- **Hyperledger Fabric**: A permissioned blockchain framework
- **NTP**: Network Time Protocol (used for accurate timestamping)
- **Oracles**: External services providing real-world data to the blockchain
- **PTM**: Product Traceability Matrix
- **Open Outcry Auction**: A traditional auction format where bids are publicly declared in real-time.

1.4 References

- Hyper-ledger Fabric Documentation: link

1.5 Overview

This document is organized into the following sections:

- Section 1: Introduction, providing the purpose, scope, and definitions.
- Section 2: Overall description of the system, including product perspective, functions, and user characteristics.
- Section 3: Specific requirements, detailing functional and non-functional requirements, external interfaces, and design constraints.
- Section 4: Change management process for handling updates to the SRS.
- Section 5: Supporting information, including appendices and additional resources.

2. The Overall Description

2.1 Application Software Perspective

BitAuction is a **distributed system** built on **Hyperledger Fabric** and interacts with:

- **Smart contracts** for handling auction rules and transactions.
- **Oracles** for integrating with **NTP servers** to ensure accurate timestamps.
- **Web3 wallets** for user authentication.
- **Distributed storage (Hyperledger Fabric's state database)** to store transaction history.
- **Only supports open outcry auctions**, requiring real-time bidder interactions

The system is designed to be independent and self-contained, with no direct dependencies on external systems other than the blockchain network. However, it will use external libraries for wallet management and payment processing. Additionally, it may interact with external oracles for timestamp synchronization and dispute resolution.

2.1.1 System Interfaces

- **Blockchain Network**: The system interfaces with **Hyperledger Fabric** to execute smart contracts.
- **Web3 Wallets**: The system supports MetaMask, WalletConnect, and similar wallets for authentication and transactions.
- **External Oracles**: Used for fetching timestamps from **NTP servers** to ensure fair bid ordering.
- **Payment Processors**: Integrates with third-party libraries to manage cryptocurrency payments securely.

2.1.2 User Interfaces

The system will provide:

- **Graphical User Interface (GUI)**: A web-based UI for users to participate in auctions.
- **Real-Time Bid Tracking**: A live auction page displaying bids as they occur.
- **Notifications Panel**: Real-time updates on bid status and auction outcomes.

2.1.3 Hardware Interfaces

- **Server Requirements**: The system will run on **cloud-based infrastructure**
- **User Devices**: The platform supports **desktop and mobile browsers** but does not require dedicated hardware.
- **Smart Contract Execution**: Runs on Hyperledger Fabric nodes.

2.1.4 Software Interfaces

- **Hyperledger Fabric**
  - i) **Mnemonic:** HLF
  - ii) **Specification Number:** N/A
  - iii) **Version Number:** Latest Stable Release
  - iv) **Source:** Hyperledger Foundation
  - v) **Purpose:** Used for executing smart contracts and storing immutable auction data.
  - vi) **Interface Definition:** Interaction occurs through smart contract APIs and blockchain query functions.
- **Wallets**
  - i) **Mnemonic:** Web3 Wallets
  - ii) **Specification Number:** N/A
  - iii) **Version Number:** Latest Stable Release
  - iv) **Source:** N/A

   v) **Purpose:** Provides authentication and transaction signing for users.
   vi) **Interface Definition:** N/A

- **External Payment Libraries**
   i) **Mnemonic:** CryptoPay
   ii) **Specification Number:** N/A
   iii) **Version Number:** Latest Stable Release
   iv) **Source:** Open-source or third-party providers
   v) **Purpose:** Handles cryptocurrency transactions securely.
   vi) **Interface Definition:** Uses RESTful APIs and Web3 smart contract calls for payment processing.
- **Oracles for Timestamping**
   i) **Mnemonic:** Oracles
   ii) **Specification Number:** N/A
   iii) **Version Number:** Latest Stable Release
   iv) **Source:** Chainlink or similar providers
   v) **Purpose:** Provides accurate timestamps for bid ordering.
   vi) **Interface Definition:** Uses API calls to fetch external time data and push to the blockchain.

2.1.5 Communications Interfaces

- **WebSockets**: Enables real-time bid updates.
- **gRPC APIs**: Used for secure communication between the frontend and blockchain services.
- **HTTPS APIs**: For general user interactions, such as auction creation and user authentication.

2.1.6 Memory Constraints

There are no memory constraints

2.1.7 Operations

- **Normal Operations**: Auctions run continuously with **real-time updates**.
- **Scheduled Downtime**: Maintenance performed during off-peak hours.
- **Backup & Recovery**: Data is replicated across blockchain nodes.

2.1.8 Site Adaptation Requirements

N/A

2.2 Application Software Functions

Provide a summary of the major functions that the software will perform.

The BitAuction platform performs the following key functions:

1. **User Authentication & Access Control** - Users authenticate using Web3 wallets, and access control mechanisms ensure appropriate permissions.
2. **Auction Management** - Sellers create and manage auctions, including setting parameters such as starting bid, reserve price, and duration.
3. **Bidding Process** - Buyers participate in open outcry auctions with real-time bidding updates.
4. **Transaction Settlement** - Smart contracts handle automatic fund transfers and refunds for unsuccessful bidders.
5. **Timestamp Synchronization** - The system integrates with NTP-based oracles to ensure accurate bid ordering.
6. **Data Storage & Logging** - Auction transactions are stored immutably using Hyperledger Fabric.
7. **Real-Time Notifications** - Users receive updates on auction completion, and payment confirmation.

3. Specific Requirements

3.1 External interfaces

Inputs and Outputs

1. **User Authentication & Access Control**
    - **Name:** User Login Request
    - **Purpose:** Authenticate users via Web3 wallets
    - **Source:** Web3 wallet
    - **Valid Range:** Must be a valid blockchain wallet address
    - **Timing:** Occurs when users log in or perform secure actions
    - **Relationships:** Required for all user interactions
    - **Format:** JSON request containing wallet signature
    - **Response:** Success/failure message, user session token
2. **Auction Management**
    - **Name:** Create Auction Request
    - **Purpose:** Allows **sellers** to create new auctions
    - **Source:** Seller's input through the web UI
    - **Valid Range:** Auction parameters must be within system constraints
    - **Timing:** On seller submission
    - **Relationships:** Linked to bidding and transaction settlement
    - **Format:** JSON request with auction details (item, starting bid, duration)
    - **Response:** Success/failure message with auction ID
3. **Bidding Process (Two-Stage Bidding)**
    - **Create Bid**
        - **Purpose:** Allows **buyers** to initiate a bid without publicly revealing bid details.
        - **Source:** Buyer's request via the bidding interface.
        - **Valid Range:** Bid must comply with auction rules and minimum increment.
        - **Timing:** Anytime during an active auction.
        - **Relationships:** Links to submit bid, timestamp synchronization, and fraud detection.
        - **Format:** JSON request with bid price and encrypted bid ID.
        - **Processing:**
            - Executed by peers who are **not** the auction owners.
            - Sets the **offered price** and creates an encrypted bid ID.
            - Data privacy is ensured; bid details are **not visible** to other peers.
            - A **POST request** is sent to an external API for **trusted timestamp creation**.
            - This transaction is endorsed by **a single party** and does not cause R/W conflicts.
        - **Response:** Success/failure message with encrypted bid ID.
    - **Submit Bid**
        - **Purpose:** Confirms a bid, making it publicly registered in the auction.
        - **Source:** The peer who initially created the bid.
        - **Valid Range:** Must reference a **previously created bid ID**.
        - **Timing:** Anytime before auction ends, after creating a bid.
        - **Relationships:** Retrieves timestamps and finalizes bid entry.
        - **Format:** JSON request referencing encrypted bid ID.

        - **Processing:**
            - **GET request** is sent to an external API to retrieve **trusted timestamps**.
            - Generates a **deterministic integer hash value** based on the **transaction ID (txID)**.
            - The hash is used to **shuffle timestamps** deterministically and select one for final bid ordering.
            - Requires endorsement by **multiple interacting peers**.
        - **Response:** Success/failure message with timestamp verification.
    - **Withdraw Bid**
        - **Purpose:** Withdraw the user's bids.
        - **Source:** Any peer who already submitted a bid and is not the winner.
        - **Valid Range:** Must reference a **previously created bid ID**.
        - **Timing:** Anytime after creating a bid.
        - **Relationships:** Retrieves the submitted money back.
        - **Format:** JSON request referencing encrypted bid ID.
        - **Response:** Success/failure message with money transferred back.

4. **Transaction Settlement**
    - **Name:** Auction Completion Event
    - **Purpose:** Finalizes the auction and transfers assets
    - **Source:** Smart contract execution
    - **Valid Range:** Only triggers if auction meets all conditions
    - **Timing:** When auction ends
    - **Relationships:** Interacts with escrow, notifications, and refund mechanisms
    - **Format:** Blockchain event notification
    - **Response:** Success message confirming asset transfer
5. **Security & Fraud Prevention**
    - **Name:** Fraud Detection Alert
    - **Purpose:** Flags suspicious activities
    - **Source:** Internal fraud detection module
    - **Valid Range:** Detects unusual bid patterns or unauthorized access
    - **Timing:** Real-time during auctions
    - **Relationships:** May result in user suspension or bid rejection
    - **Format:** JSON event log entry
    - **Response:** Administrator notification
6. **Timestamp Synchronization**
    - **Name:** NTP Timestamp Verification
    - **Purpose:** Ensures accurate bid ordering
    - **Source:** External oracles fetching NTP timestamps
    - **Valid Range:** Must match the latest blockchain state
    - **Timing:** Every transaction requiring timestamp validation
    - **Relationships:** Ensures fairness in bid sequencing
    - **Format:** JSON request containing timestamp data
    - **Response:** Confirmed timestamp stored on blockchain
7. **Data Storage & Logging**
    - **Name:** Immutable Transaction Record
    - **Purpose:** Logs all transactions securely
    - **Source:** Hyperledger Fabric ledger
    - **Valid Range:** Permanent, cannot be modified
    - **Timing:** After each auction or bid event
    - **Relationships:** Required for audit and dispute resolution
    - **Format:** Blockchain ledger entry
    - **Response:** Readable history query output
8. **Real-Time Notifications**
    - **Name:** User Notification Event

  - **Purpose:** Alerts users about bid status and auction results
  - **Source:** System notification service
  - **Valid Range:** Applies to relevant user actions
  - **Timing:** Immediately after key events (bid success, auction end)
  - **Relationships:** Enhances user engagement
  - **Format:** WebSocket message or push notification
  - **Response:** Message displayed to user

3.2 Functional Requirements

Bidding Process (Two-Stage Bidding)

- The system shall implement a **two-stage bidding process** consisting of **Create Bid** and **Submit Bid**.
- The system shall support only non-physical entities for auction items, like NFTs.
- The **Create Bid** transaction shall be executed by peers who are **not** the owners of the auction and will store bid data privately.
- The **Create Bid** transaction shall send a **POST request** to a **time oracle** to establish a **trusted timestamp**.
- The **Submit Bid** transaction shall reference a **previously created bid ID** and will be executed by the **same peer** who created the bid.
- The **Submit Bid** transaction shall send a **GET request** to an external API to retrieve **trusted timestamps**.
- The system shall generate an **integer hash value** from the transaction ID (txID) to **shuffle timestamps deterministically**.
- The system shall require **multiple endorsements** from interacting peers for the **Submit Bid**

User Authentication & Access Control

- The system shall allow users to log in using Web3 wallets.
- The system shall enforce role-based access control for different user types, which are buyers, and sellers.

Auction Management

- The system shall allow sellers to create auctions with predefined parameters.
- The system shall enforce open outcry auction rules.
- The system shall prevent modifications to auctions after creation.

Bidding Process

- The system shall validate bids before submission.
- The system shall reject bids that do not exceed the current highest bid.
- The system shall support real-time bid updates via WebSockets.

Transaction Settlement

- The system shall automatically transfer assets upon auction completion.

Ordering Service Affairs (Timestamping & Ordering)

- The system shall integrate with oracles to fetch timestamps from NTP servers.

3.3 Performance Requirements

- Bid updates shall reflect within 1 second. (experimental)
- Smart contract execution time shall not exceed 1 second. (experimental)
- System shall support up to 1000 auctions at the same time.
- System shall support up to 1000 bids per auction.

3.4 Logical Database Requirements

This section specifies the logical requirements for any information that is to be placed into a database. This may include:
• Types of information used by various functions
• Frequency of use
• Accessing capabilities
• Data entities and their relationships
• Integrity constraints
• Data retention requirements
If the client provided you with data models, those can be presented here. ER diagrams (or static class diagrams) can be useful here to show complex data relationships.

3.5 Design Constraints

The BitAuction platform operates within the constraints of the **blockchain trilemma**, which states that decentralization, security, and scalability cannot all be maximized simultaneously. Due to this limitation:

- The system prioritizes **scalability and security** over full decentralization.
- **Lower decentralization** is achieved by using **Hyperledger Fabric**, a permissioned blockchain, rather than a fully decentralized public blockchain. And by using an external timestamp service, rather than relying on a Proof-of-Work puzzle.

3.5.1 Standards Compliance

N/A

3.6 Software System Attributes

3.6.1 Reliability

N/A

3.6.2 Availability

N/A

3.6.3 Security

- The system shall implement **end-to-end encryption** for bid transactions and user authentication.
- Smart contracts shall be **audited** before deployment to prevent vulnerabilities.
- Role-based access control shall be enforced for **sellers, and bidders**.

3.6.4 Maintainability

- The system shall use a **modular architecture** to facilitate updates and bug fixes.

3.6.5 Portability

N/A

4. Change Management Process

- All changes must be formally reviewed by the team
- In case of a dispute against a decision, a majority vote shall be taken.
- Changes to code shall be logged using a version control system like GitHub
- Changes to documents shall be logged in a document management system like Confluence.