Software Requirements Specification

for eBidX: Online Auction System

Sreejith P V Aswin Raju Alen T L Nikhil Sreekumar
Sanju M B

2026-01-05

Revision History

Name	Date	Reason for Changes	Version
Alen T L	2026-01-05	Initial draft created	1.0

Introduction

Purpose

The purpose of this document is to specify the Software Requirements Specification (SRS) for **eBidX**, an online auction system. The system facilitates real-time bidding on products, allowing users to list items for sale and place bids in a competitive environment. This SRS outlines the system's features, interfaces, and constraints.

Document Conventions

- Headings are numbered automatically.
- Key terms such as *Bidder* and *Seller* are capitalized when first introduced.
- Functional requirements are labeled as **REQ-1**, **REQ-2**, etc.
- Placeholder values are marked as **TBD**.

Intended Audience and Reading Suggestions

This document is intended for developers, project managers, and testers.

- **Developers:** Focus on Section 3 and Section 5.
- **Testers:** Focus on functional requirements (REQ tags).

Product Scope

eBidX is a web-based platform designed to support dynamic pricing through competitive bidding. It provides a secure environment for transactions and real-time bid updates.

Definitions, Acronyms, and Abbreviations

- **SRS:** Software Requirements Specification
- **SPA:** Single Page Application
- **JWT:** JSON Web Token
- **DFD:** Data Flow Diagram
- **ERD:** Entity Relationship Diagram
- **API:** Application Programming Interface
- **HTTPS:** HyperText Transfer Protocol Secure

References

- IEEE 29148:2018 — Systems and Software Engineering — Life Cycle Processes — Requirements Engineering
- IEEE Software Requirements Specification (SRS) Template
- Django Documentation
- React / Vite Documentation

Overview

This document provides a complete description of the functional and nonfunctional requirements of the eBidX Online Auction System. It serves as a reference for system design, development, testing, and future enhancement activities.

Overall Description

Product Perspective

eBidX is a self-contained web application that replaces traditional offline auction processes.

- **Frontend:** React.js (SPA)
- **Backend:** Django REST Framework
- **Database:** PostgreSQL

Product Functions

The major system functions include:

1. **User Authentication:** Secure signup and login using JWT
2. **Auction Management:** Item listing with images and base price
3. **Real-Time Bidding:** Instant updates of highest bid
4. **Admin Controls:** User moderation and listing removal

User Classes and Characteristics

- **Guest:** View-only access
- **Registered User:**
  - *Seller:* Lists auction items
  - *Bidder:* Places bids frequently
- **Administrator:** Manages system and users

Operating Environment

- Client: Modern web browsers
- Server: Ubuntu Linux (20.04 or higher)
- Network: Stable internet connection

Design and Implementation Constraints

- Frontend must use React.js
- Backend must use Django
- PostgreSQL is mandatory

User Documentation

- Online help via “How It Works” page
- Developer documentation via repository README

Assumptions and Dependencies

- System clock synchronization via NTP

External Interface Requirements

This section describes how the eBidX system interacts with users, hardware components, external software systems, and communication protocols.

User Interfaces

- Clean, modern responsive UI
- Persistent navigation bar

Hardware Interfaces

- Standard HTTP over TCP/IP

Software Interfaces

- RESTful APIs
- PostgreSQL database

Communications Interfaces

- HTTPS protocol
- JSON data format

System Features

Auction Listing (Seller)

Description

Registered users can create auction listings by providing product details.

Stimulus / Response Sequences

- **Stimulus:** User submits auction form
- **Response:** System validates data and creates listing

Functional Requirements

- **REQ-1:** Base price must be a positive integer
- **REQ-2:** End time must be in the future
- **REQ-3:** Image uploads must support JPG and PNG

Bidding Engine (Bidder)

Description

Users compete by placing bids on active auctions.

Functional Requirements

- **REQ-4:** Bid must exceed current highest bid plus increment
- **REQ-5:** Bids after end time are rejected
- **REQ-6:** Highest bid updates instantly

Other Nonfunctional Requirements

This section defines quality attributes and constraints that determine how the system performs rather than what it does.

Performance Requirements

Performance requirements define the expected responsiveness and capacity of the system under normal and peak operating conditions. These requirements are critical for maintaining a fair and real-time auction experience.

- Bid update latency <= 1 second
- Support at least 50 concurrent users

Safety Requirements

- No bids allowed on closed auctions

Security Requirements

Security requirements ensure the protection of user data, financial integrity of bids, and resistance against common web vulnerabilities such as unauthorized access, data leakage, and session hijacking.

- Password hashing via PBKDF2
- JWT-protected APIs
- Enforced HTTPS

Software Quality Attributes

- **Reliability:** 99% uptime during auctions
- **Maintainability:** Modular frontend and backend

System Architecture

The eBidX system follows a three-tier client-server architecture:

- **Presentation Layer:** React.js frontend responsible for UI and user interaction
- **Application Layer:** Django REST API handling business logic
- **Data Layer:** PostgreSQL database for persistent storage

This architecture improves scalability, maintainability, and security.

Use Case Descriptions

This section provides detailed textual use cases that complement the graphical use case diagram. Each use case describes actor interactions, preconditions, normal flow, and post-conditions, aiding developers and evaluators in understanding system behavior.

Use Case: Place Bid

- **Actor:** Registered Bidder
- **Preconditions:** User is authenticated; auction is active
- **Postconditions:** Bid is recorded and highest bid is updated

Main Flow:

1. Bidder selects an active auction
2. System displays current highest bid
3. Bidder enters bid amount
4. System validates bid amount
5. System updates highest bid and notifies users

Alternate Flow:

- If bid amount is invalid, system rejects the bid and displays an error message

Use Case: Create Auction

- **Actor:** Seller
- **Preconditions:** Seller is authenticated
- **Postconditions:** Auction is created and visible to bidders

Main Flow:

1. Seller selects “Create Auction”

2. Seller enters product details and images
3. Seller sets base price and end time
4. System validates input
5. System creates auction listing

Alternate Flow:

- If validation fails, system prompts seller to correct input

Future Enhancements

- Real-time bidding using WebSockets
- Payment gateway integration
- Auction analytics dashboard
- Automated fraud detection

Additional Requirements

Data Requirements

- User profiles
- Auction listings
- Bid history
- Audit logs

Validation Rules

- Mandatory fields must not be empty
- Bid values must follow increment rules
- Uploaded images must meet format and size constraints

System Constraints

- Internet connectivity required
- System dependent on server availability
- Time synchronization required for auctions

Conclusion

eBidX provides a scalable and secure platform for online auctions. The system requirements defined in this document form a strong foundation for implementation and future enhancements.

Other Requirements

Requirement Traceability Matrix (RTM)

The Requirement Traceability Matrix ensures that all functional requirements are traceable to corresponding system features, use cases, and test considerations. This matrix helps verify coverage and supports validation during testing and evaluation phases.

Requirement ID	Description	Related Use Case	System Module
REQ-1	Validate base price	Create Auction	Auction Management
REQ-2	End time must be future	Create Auction	Auction Management
REQ-3	Image upload support	Create Auction	Media Service
REQ-4	Enforce bid increment	Place Bid	Bidding Engine
REQ-5	Reject late bids	Place Bid	Bidding Engine
REQ-6	Real-time bid update	Place Bid	Notification Service

Detailed Functional Requirements

This section elaborates each functional requirement in a structured format for clarity and implementation reference.

FR-1 User Registration

- **Description:** System shall allow new users to register using email and password.
- **Inputs:** Email, Password
- **Outputs:** User account created
- **Priority:** High

FR-2 User Login

- **Description:** System shall authenticate users and issue JWT tokens.
- **Inputs:** Credentials
- **Outputs:** Access token
- **Priority:** High

FR-3 Create Auction

- **Description:** Seller can create auction listings with product details.
- **Inputs:** Product info, images, base price
- **Outputs:** Auction listing
- **Priority:** High

FR-4 Place Bid

- **Description:** Bidder places a bid higher than current highest bid.

- **Inputs:** Bid amount
- **Outputs:** Updated highest bid
- **Priority:** High

FR-5 Close Auction

- **Description:** System automatically closes auction at end time.
- **Priority:** Medium

Nonfunctional Requirements (Detailed)

Performance

- System shall support up to 100 concurrent users during peak load.
- Bid processing latency shall not exceed 1 second.

Scalability

- System shall support horizontal scaling of backend services.

Availability

- System shall achieve 99% uptime during auction periods.

Security

- Sensitive data shall be encrypted at rest and in transit.
- Protection against common web attacks (SQL Injection, XSS).

Maintainability

- Code shall follow modular architecture and documentation standards.

Risk Analysis

This section identifies potential risks and mitigation strategies.

Risk	Impact	Mitigation
Server overload during live auctions	High	Load balancing, caching
Time synchronization issues	Medium	NTP synchronization
Fraudulent bidding	High	Bid validation and monitoring
Network latency	Medium	Optimized APIs

Future Enhancements

- Integration with payment gateways
- WebSocket-based real-time bidding
- Mobile application support

Appendix A: Glossary

- **Bid Increment:** Minimum amount above previous bid
- **Sniper:** User bidding at last seconds

Appendix B: Analysis Models

This appendix contains the analysis diagrams used to model the behavior, data flow, and structure of the eBidX system.

B.1 Use Case Diagram

The use case diagram illustrates interactions between users and the system.

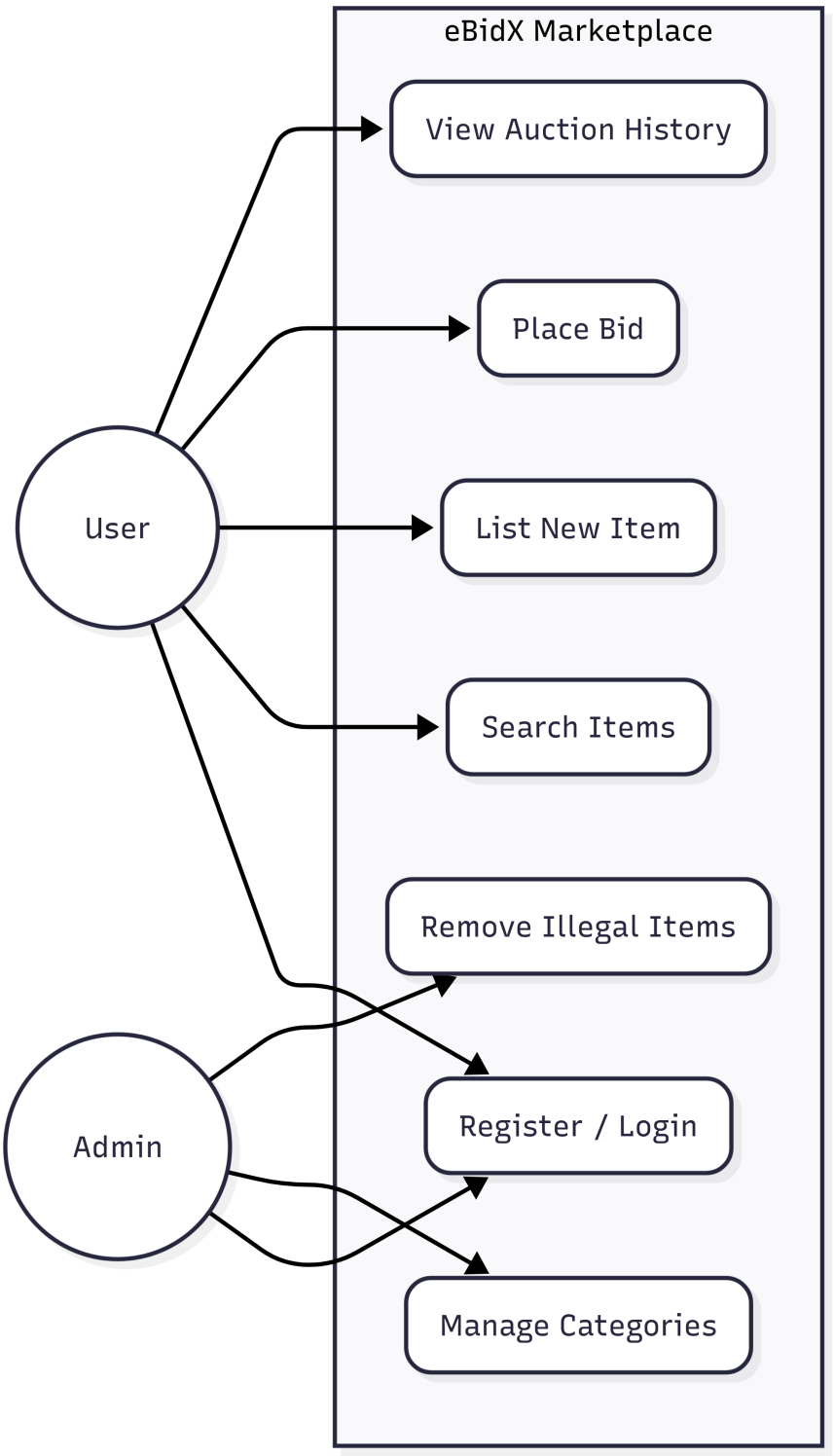

Figure 1: Figure B.1: Use Case Diagram for eBidX

B.2 Data Flow Diagrams

Level 0: Context Diagram This context-level DFD represents the entire eBidX system as a single process and illustrates data exchange between external entities and the system.

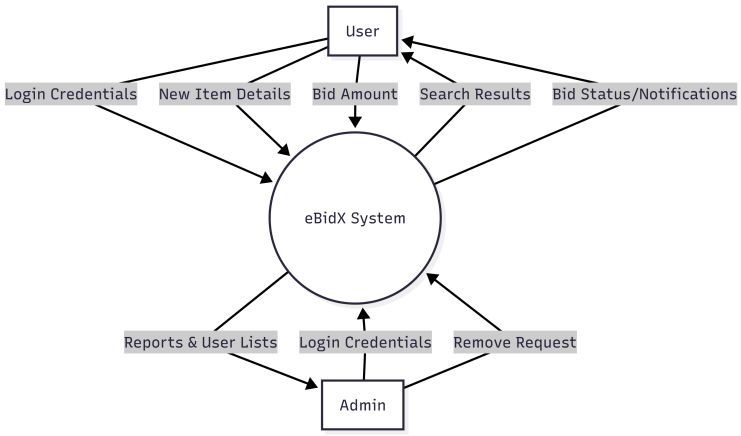

Figure 2: Level 0 Data Flow Diagram (Context Diagram)

Level 1: Process Decomposition Diagram This diagram decomposes the eBidX system into major internal processes such as authentication, auction management, bidding engine, and administration.

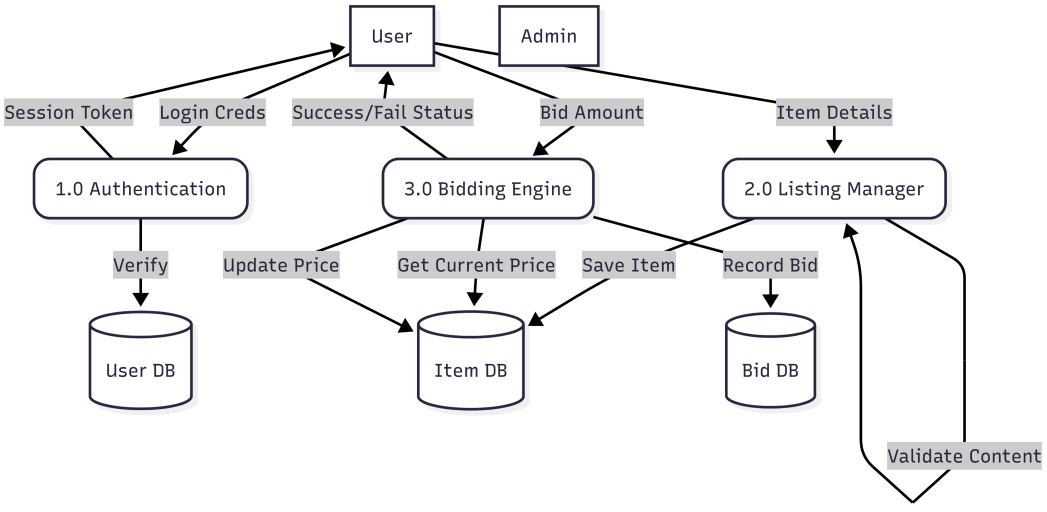

Figure 3: Level 1 Data Flow Diagram (Process Decomposition)

B.3 Activity Diagram

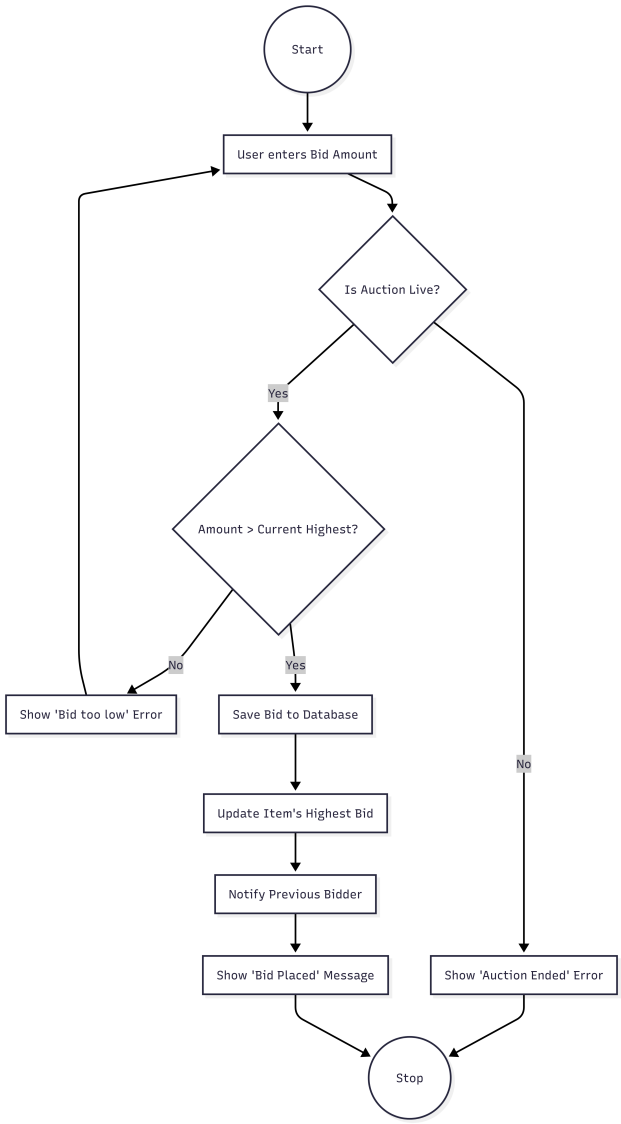

Figure 4: Figure B.4: Activity Diagram for Bidding Process

B.4 Entity Relationship Diagram

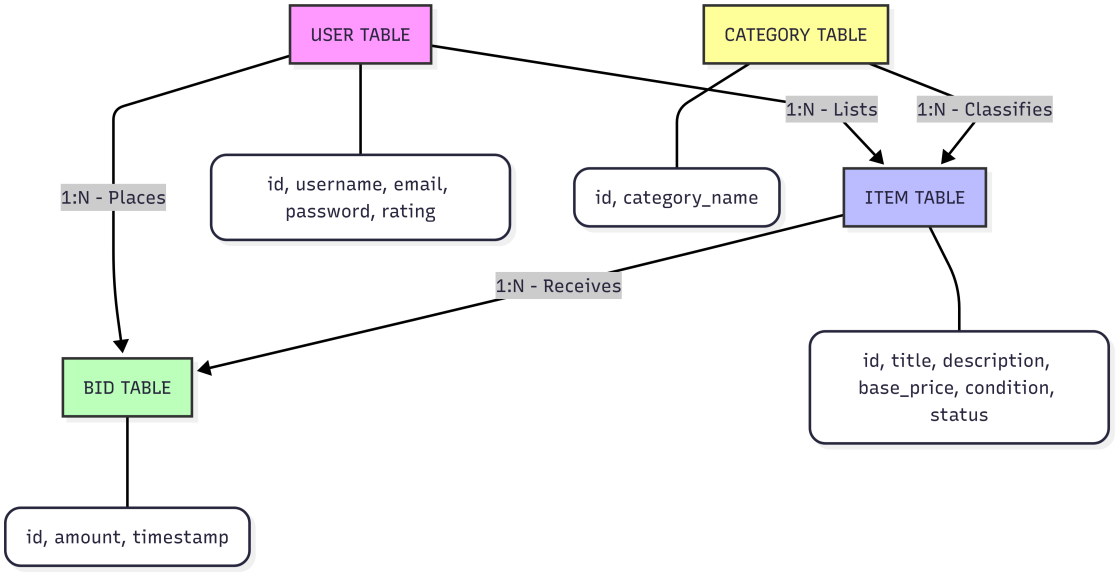

Figure 5: Figure B.5: Entity Relationship Diagram for eBidX Database

Appendix C: To Be Determined

This section lists features and design decisions that are planned for future versions of the system but are outside the scope of the current release.

- **TBD:** Payment gateway integration
- **TBD:** WebSocket-based live updates