Software Requirements Specification

for

eBidX: Online Auction System

Version 1.0 Approved

Prepared by:

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

Department of Computer Science

2026-01-13

Contents

Revision History

Name	Date	Reason for Changes	Version

1 Introduction

1.1 Purpose

This document specifies 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.

1.2 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**.

1.3 Intended Audience and Reading Suggestions

This document is for developers, project managers, testers, marketing teams, and end users to understand the system's functionalities, constraints, and potential use cases.

For better understanding, it is recommended to read in the following order:

A. Introduction - Overview of the project.
B. Overall Description - System components and features.
C. External Interface Requirements - Hardware and software dependencies.
D. System Features - Detailed breakdown of functionalities.
E. Other Nonfunctional Requirements - Performance, security, and quality attributes.

1.4 Product Scope

eBidX is a feature-rich online auction marketplace designed to democratize the buying and selling process through real-time competitive bidding. The platform connects sellers looking to liquidate assets with buyers seeking value, facilitating transactions in a secure and transparent digital environment.

The system's core functionality revolves around a dynamic bidding engine that supports real-time price updates via WebSockets, ensuring that all participants see the latest highest bid instantly. Beyond basic bidding, eBidX includes advanced features such as proxy bidding, watchlist management, and automated outbid notifications to keep users engaged.

For administrators, the platform provides comprehensive oversight tools, including user management, listing moderation, and transaction analytics. The goal of eBidX is to create a trustworthy and efficient community-driven marketplace that overcomes the geographical and temporal limitations of traditional physical auctions.

1.5 References

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

2 Overall Description

2.1 Product Perspective

eBidX is a robust, self-contained web application designed to facilitate secure online auctions, replacing traditional offline bidding processes with a digital real-time alternative. Unlike standard e-commerce platforms with fixed pricing, eBidX creates a dynamic marketplace where item values fluctuate based on user demand and competitive bidding. This fosters a transparent environment for fair market value discovery.

The system operates as a standalone platform but is architected to scale and integrate with external services such as payment gateways and email notification systems. It acts as a comprehensive solution for managing the entire auction lifecycle, from user registration and item listing to live bidding and final transaction closure, ensuring a seamless experience for both buyers and sellers.

The core components include:

A. A Frontend Application (built with React.js) for browsing auctions and placing bids.

B. A Backend Server (built with Django REST Framework) for managing logic and authentication.
C. A Database (PostgreSQL) for storing user profiles, auction items, and bid logs.

2.2 Product Functions

The eBidX system will perform the following major functions:

I. User Authentication & Security: The system will facilitate secure registration and login using JWT tokens, managing roles such as Guest, Bidder, Seller, and Administrator to ensure appropriate access levels.
II. Auction Management: Sellers will be provided with tools to create comprehensive item listings, upload high-quality images, set base prices, and define auction durations to effectively manage their inventory.
III. Real-Time Bidding Engine: The system will process bids in real-time, enforcing minimum increment rules and instantly broadcasting the highest bid to all connected users via WebSockets to ensure a fair competitive environment.
IV. Search and Discovery: Users will be able to easily locate items through an advanced search interface that supports keyword queries, category filtering, and sorting by price or time remaining.
V. Notification System: The platform will automatically send alerts for critical events, such as outbid notifications, auction wins, and payment confirmations, to maintain high user engagement.
VI. Admin Controls: Administrators will have access to a centralized dashboard to moderate user activity, remove inappropriate listings, and oversee platform analytics for maintenance and compliance.

2.3 User Classes and Characteristics

eBidX will cater to the following user classes:

I. Administrators
   A. Frequency of Use: Frequent (daily or as needed).
   B. Functions Used: Managing user accounts, moderating listings, resolving disputes, and monitoring system analytics.
   C. Characteristics: Technical or operational staff responsible for the health and compliance of the marketplace. They ensure fair play and system stability by overseeing all activities on the platform.

II. Sellers
   - A. Frequency of Use: Occasional (when listing new items or checking auction status).
   - B. Functions Used: Creating auction listings, uploading images, setting base prices, and monitoring incoming bids.
   - C. Characteristics: Individuals or commercial entities wishing to liquidate assets. They seek a secure and efficient platform to maximize the value of their items through competitive bidding.

III. Bidders
   - A. Frequency of Use: Frequent (during active auctions or while searching).
   - B. Functions Used: Searching for items, placing real-time bids, managing watchlists, and processing payments for won items.
   - C. Characteristics: Users looking for value or specific items. They rely on the real-time nature of the platform to place timely bids and track their winning status.

IV. Guests
   - A. Frequency of Use: Occasional (browsing).
   - B. Functions Used: Viewing active auctions and searching for items (read-only access).
   - C. Characteristics: Unregistered visitors exploring the platform. They cannot place bids or list items until they register for an account.

2.4 Operating Environment

A. Hardware Platform:
   - a. Client: Any desktop, laptop, or mobile device with a modern web browser.
   - b. Server: Virtual Private Server (VPS) or Cloud Instance.
     - i. CPU: 2 vCPUs minimum (for handling concurrent WebSocket connections).
     - ii. RAM: 4 GB minimum.
     - iii. Storage: 25 GB SSD or higher.

B. Operating System:
   - a. Client: Platform Independent (Windows, macOS, Linux, Android, iOS).
   - b. Server: Linux (Ubuntu 20.04 LTS or later recommended).

C. Software Components:
   - a. Client: Modern Web Browser (Chrome, Firefox, Safari, Edge) with JavaScript enabled.
   - b. Server:
     - i. Python 3.9+ (Runtime environment).
     - ii. PostgreSQL 13+ (Relational Database).
     - iii. Redis 6+ (For WebSocket channel layers).

        iv. Nginx or Apache (Web Server/Reverse Proxy).

D. External Interfaces:
   a. Stable broadband internet connection required for real-time bidding updates.
   b. HTTPS (Port 443) accessible for secure transactions.

2.5 Design and Implementation Constraints

I. Hardware Limitations:
   A. The real-time nature of the bidding system places a constraint on the server hardware, which must be capable of handling high concurrency and maintaining thousands of active WebSocket connections simultaneously without latency.
   B. Clients are assumed to have a stable broadband internet connection. Users on high-latency networks may experience delays in bid updates, which is a constraint of the real-time architecture.

II. Software and Libraries:
   A. The frontend implementation is strictly constrained to the React.js library. This dependency requires developers to adhere to a component-based architecture and manage state effectively for real-time updates.
   B. The backend logic must be implemented using the Django REST Framework (Python). This constrains the architectural pattern to Django's MVT model and requires adherence to Python's PEP-8 coding standards.
   C. PostgreSQL is mandated as the database management system to ensure ACID compliance, which is non-negotiable for financial transaction integrity.

III. Data Privacy and Security:
   A. The system must not store raw credit card details. Implementation is constrained to using token-based authentication with third-party payment processors to meet PCI-DSS standards.
   B. All user passwords must be hashed using robust algorithms like PBKDF2 or Argon2 before storage, preventing plain-text retrieval.

IV. Compliance with Regulations:
   A. The implementation must support GDPR requirements, specifically the ability for users to export their data or request account deletion, which constrains the database schema design (cascading deletes vs. soft deletes).

V. User Interface Design:
   A. The design must prioritize the visibility of dynamic elements like the countdown timer and current bid. The interface is constrained to update these elements via DOM manipulation without requiring full page reloads.

2.6 User Documentation

A. User Manual:
   a. Comprehensive guide covering the complete eBidX experience, including account registration, browsing auctions, placing proxy bids, and managing seller inventory.
   b. Format: PDF, available for download from the platform's footer.

B. Online Help (Help Center):
   a. A dedicated, searchable web section providing immediate answers to specific platform functionalities. It includes context-sensitive tooltips within the seller dashboard to assist with item listing.
   b. Format: Integrated web pages accessible via the navigation bar.

C. Tutorials:
   a. Step-by-step video guides and interactive walkthroughs demonstrating key actions such as "How to win your first auction" and "Best practices for listing items."
   b. Format: Embedded video player and rich-text articles.

D. System Administration Guide:
   a. Technical documentation for platform administrators detailing server maintenance, database backup procedures, user moderation tools, and resolving payment disputes.
   b. Format: Secure PDF accessible only to users with Admin privileges.

E. FAQs:
   a. A frequently updated list of common questions regarding payment security, shipping policies, bid retractions, and troubleshooting login issues.
   b. Format: Collapsible web interface for easy browsing.

2.7 Assumptions and Dependencies

I. Assumptions:
   A. It is assumed that all end users have access to a stable internet connection with sufficient bandwidth to receive real-time WebSocket updates.
   B. Users are assumed to possess valid payment methods (Credit Card, Debit Card, or digital wallet) for processing transactions upon winning an auction.
   C. The system clock on the hosting server is assumed to be synchronized precisely via NTP to ensure auction start and end times are strictly enforced.

II. Dependencies:
   A. Payment Processing: The system depends entirely on third-party payment gateways (e.g., Stripe, PayPal) for handling financial transactions. Any downtime in these services will directly impact the ability to close auctions.

B. Email Service: The notification system relies on external SMTP providers (e.g., SendGrid, AWS SES) for delivering account verification emails and outbid alerts.
C. Real-Time Infrastructure: The live bidding feature is dependent on the availability and performance of the Redis backing store to manage WebSocket channel layers.
D. Third-Party Libraries: The project depends on the continued maintenance and compatibility of key open-source libraries (React, Django REST Framework) with current browser and server standards.

3 External Interface Requirements

3.1 User Interfaces

A. Logical Characteristics:
   a. The user interface shall be intuitive and navigable, requiring no formal training for standard users (bidders/sellers).
   b. Critical real-time elements, such as the ”Current Bid” and ”Time Remaining,” must be prominently displayed and updated dynamically without user intervention.
   c. The layout must follow a responsive design philosophy, adapting seamlessly to desktop, tablet, and mobile screen resolutions.
   d. Navigation should be persistent and clear, providing easy access to ”My Bids,” ”Watchlist,” and ”Profile” sections from any page.
   e. Interactive elements (buttons, forms) must provide immediate visual feedback (e.g., loading spinners, success toasts) upon user interaction.

B. GUI Design Standards:
   a. Font: Use modern sans-serif fonts (e.g., Inter, Roboto) to ensure maximum readability across devices.
   b. Colors: Employ a high-contrast color palette for call-to-action buttons (e.g., Green for ”Bid Now”, Red for ”End Soon”) to guide user behavior effectively.
   c. Consistency: All forms, modal windows, and dashboards must adhere to a unified design language (Material Design or similar) to maintain visual consistency.

3.2 Hardware Interfaces

I. Supported Device Types:
   A. Client Devices: The system allows access via any standard personal computer, laptop, tablet, or smartphone capable of running a modern web browser.

   B. Server Hardware: The backend operates on standard virtualized server hardware (VPS) with network interfaces capable of handling high-throughput TCP/IP traffic.

II. Data and Control Interactions:
   A. Input Devices: The system interacts with standard input hardware (keyboard, mouse, touchscreens) to receive user commands such as bid entry and navigation.
   B. Network Interfaces: The system utilizes standard network cards (NICs) to transmit and receive data packets over the internet, adhering to standard Ethernet/Wi-Fi protocols.

3.3 Software Interfaces

A. Databases:
   a. The system interfaces with a PostgreSQL database to persistently store and retrieve structured data, including user credentials, auction listings, and transaction history.
   b. It also interfaces with Redis, an in-memory data structure store, to handle ephemeral data for real-time WebSocket communication and session caching.

B. Libraries and Tools:
   a. React.js: Used for rendering the dynamic user interface in the client's browser.
   b. Django REST Framework: Used to expose API endpoints and handle business logic on the server.
   c. Axios/Fetch API: Used for handling HTTP requests between the client and server.

C. Communication between Components:
   a. The frontend and backend communicate via a RESTful API for standard CRUD operations (Create, Read, Update, Delete).
   b. Real-time updates (e.g., new bids) are pushed from the server to connected clients via the WebSocket protocol using Django Channels.

3.4 Communications Interfaces

I. Web Interface:
   A. The system utilizes the HTTPS protocol (Hypertext Transfer Protocol Secure) for all client-server communication to ensure data integrity and privacy.
   B. Data exchange is primarily conducted using the JSON (JavaScript Object Notation) format, which is lightweight and easily parsed by web applications.

II. Data Security:
   A. Transport Layer Security (TLS/SSL) encryption is mandatory for all data in transit to protect against eavesdropping and man-in-the-middle attacks.

B. Sensitive payment data is handled via tokenization, ensuring that raw credit card numbers never traverse the primary application server.

III. Real-time Data Sync:

A. The system employs the Secure WebSocket (WSS) protocol to maintain persistent, full-duplex communication channels between the server and clients for instant bid updates.

B. Low-latency messaging is prioritized to ensure fair play, minimizing the delay between a bid submission and its reflection on other users' screens.

4 System Features

4.1 Auction Listing (Seller)

4.1.1 Description and Priority

- Description: This feature enables registered Sellers to list items for auction. The Seller allows the input of item details including a descriptive title, detailed description, starting base price, and auction duration. The system validates these inputs to ensure the auction is fair and legitimate before publishing it to the marketplace.
- Priority: High (Essential for inventory creation and platform utility).

4.1.2 Stimulus / Response Sequences

- Stimulus 1: User clicks ”Create Auction” and submits the form with item details and images.
- Response 1: System verifies that the base price is positive and the end date is in the future.
- Stimulus 2: User confirms the preview of the listing.
- Response 2: The system saves the listing to the database and redirects the user to the item's detail page.
- Stimulus 3: User attempts to submit a form with a date in the past.
- Response 3: The system displays specific error messages on the form.

4.1.3 Functional Requirements

- REQ-1: Base price must be a positive integer greater than zero.
- REQ-2: Auction end time must be at least 1 hour in the future from the creation time.

- REQ-3: Image uploads must support JPG, PNG, and WEBP formats, with a maximum size of 5MB per image.

4.2 Bidding Engine (Bidder)

4.2.1 Description and Priority

- Description: The Bidding Engine is the core component that handles real-time interactions between users and active auctions. It ensures that all bids are valid, higher than the current price, and placed within the auction's time window. It manages concurrency to prevent race conditions when multiple users bid simultaneously.
- Priority: High (Critical core functionality).

4.2.2 Stimulus / Response Sequences

- Stimulus 1: User enters a bid amount and clicks ”Place Bid”.
- Response 1: System validates the bid amount against the current high bid + increment.
- Stimulus 2: System accepts the valid bid.
- Response 2: System updates the database and broadcasts the new price to all connected clients via WebSocket.
- Stimulus 3: User places a bid after the auction has closed.
- Response 3: System rejects the bid and displays a ”Bidding Closed” error.

4.2.3 Functional Requirements

- REQ-4: A new bid must exceed the current highest bid by at least the minimum bid increment (e.g., $1.00 or 5% of current price).
- REQ-5: Bids placed after the auction end time must be instantly rejected with a ”Bidding Closed” error.
- REQ-6: The system shall update the highest bid display for all connected users via WebSocket within 1 second.

4.3 Watchlist Management

4.3.1 Description and Priority

- Description: This feature allows users to ”watch” specific auctions they are interested in without placing a bid. This helps users track items and receive updates as the auction nears its conclusion.
- Priority: Medium (Enhances user engagement).

4.3.2 Stimulus / Response Sequences

- Stimulus 1: User clicks the ”Watch” icon on an item card.
- Response 1: System adds the item to the user’s watchlist database table.
- Stimulus 2: User views their profile dashboard.
- Response 2: System displays a list of watched items with current status.

4.3.3 Functional Requirements

- REQ-7: Users shall be able to add or remove active auctions from their personal watchlist.
- REQ-8: The system shall provide a ”My Watchlist” dashboard displaying current prices and time remaining for watched items.

4.4 Search and Filtering

4.4.1 Description and Priority

- Description: To enhance discoverability, the system provides a robust search engine allowing users to find items based on keywords and specific criteria.
- Priority: High (Essential for navigation).

4.4.2 Stimulus / Response Sequences

- Stimulus 1: User types ”Laptop” into the search bar and presses Enter.
- Response 1: System queries the database for items with ”Laptop” in the title or description.
- Stimulus 2: User applies a ”Price: Low to High” filter.

- Response 2: System re-renders the list sorted by current price.

4.4.3 Functional Requirements

- REQ-9: Users can search for items by entering keywords matching the item title or description.
- REQ-10: Search results shall be filterable by category (e.g., Electronics, Art), price range, and item condition.
- REQ-11: Users can sort results by ”Ending Soonest”, ”Newly Listed”, or ”Price: Low to High”.

4.5 User Notifications

4.5.1 Description and Priority

- Description: The system automatically notifies users of critical events to keep them engaged and informed about the status of their bids and listings.
- Priority: Medium (Important for user retention).

4.5.2 Stimulus / Response Sequences

- Stimulus 1: A user is outbid by another participant.
- Response 1: System triggers an email and in-app notification to the previous high bidder.
- Stimulus 2: An auction ends successfully.
- Response 2: System sends a ”You Won” email to the winner and a ”Item Sold” email to the seller.

4.5.3 Functional Requirements

- REQ-12: A bidder shall receive an immediate notification (email/in-app) if they are outbid.
- REQ-13: The seller shall be notified whenever a new bid is placed on their item.
- REQ-14: Both the winner and the seller shall receive a comprehensive summary email upon the successful closure of an auction.

4.6 Rating and Review System

4.6.1 Description and Priority

- Description: This feature builds trust within the platform by allowing buyers and sellers to rate their transaction experience.
- Priority: Low (Trust-building feature).

4.6.2 Stimulus / Response Sequences

- Stimulus 1: Transaction is marked as ”Completed”.
- Response 1: System enables the review form for both parties.
- Stimulus 2: User submits a 5-star rating and comment.
- Response 2: System updates the target user’s public profile with the new rating average.

4.6.3 Functional Requirements

- REQ-15: Winners can leave a star rating (1-5) and a text review for the seller after the auction is closed and paid for.
- REQ-16: The system shall calculate and display an average reputation score on every user’s public profile.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

I. Response Time:
   A. WebSocket latency for bid updates must remain below 500 milliseconds for 95% of users to ensure fair real-time competition.
   B. General page loads should complete within 2 seconds on standard broadband connections.

II. Concurrency:
   A. The system must support at least 1,000 concurrent active users during peak auction hours without performance degradation.

III. Throughput:

A. The database and backend must be optimized to handle a throughput of at least 50 bid transactions per second.

5.2 Safety Requirements

A. Transaction Integrity: The system must enforce ACID properties on all financial transactions to prevent partial updates (e.g., deducting money without awarding the item).
B. Data Redundancy: Automated full database backups must be performed daily, with point-in-time recovery options available for the last 24 hours.
C. Failover Mechanisms: In the event of a primary server failure, traffic should automatically reroute to a standby server to minimize downtime.

5.3 Security Requirements

I. Authentication:
   A. Multi-Factor Authentication (MFA) should be available for all user accounts and mandatory for administrator accounts.
   B. Session tokens (JWT) must have a short expiration time and be securely refreshed.
II. Encryption:
   A. All sensitive data at rest (e.g., user PII, database backups) must be encrypted using AES-256 standards.
III. Audit Logs:
   A. Comprehensive logs of all administrative actions and financial transactions must be maintained for at least 12 months for compliance auditing.

5.4 Software Quality Attributes

A. Reliability: The system targets an availability of 99.9% during scheduled auction periods.
B. Usability: The interface should be designed such that a new user can place a bid within 3 clicks of landing on the homepage.
C. Maintainability: The codebase must adhere to PEP-8 (Python) and Airbnb (React) style guides to ensure ease of future updates.
D. Scalability: The architecture should support horizontal scaling (adding more application servers) to handle increased load during high-traffic events.

5.5 Business Rules

I. Eligibility:
   A. Only users aged 18 and above are legally permitted to register and participate in auctions.

II. Bidding Restrictions:
   A. Sellers are strictly prohibited from placing bids on their own listings (shill bidding) to artificially inflate prices.

III. Payment Obligation:
   A. Winning bidders must complete the payment process within 24 hours of the auction closing, otherwise the item may be offered to the next highest bidder.

6 Other Requirements

A. Database Requirements:
   a. The system requires a relational database management system (PostgreSQL recommended) to handle complex relationships between users, items, and bids.
   b. The database schema must be normalized to at least the Third Normal Form (3NF) to reduce data redundancy.
   c. Indexing strategies must be applied to frequently queried fields (e.g., item category, end time) to optimize search performance.

B. Internationalization Requirements:
   a. While the initial release will be in English, the system architecture must support future localization (i18n).
   b. Currency symbols and date formats should be dynamically rendered based on the user's detected locale settings.

C. Legal Requirements:
   a. The platform must comply with the General Data Protection Regulation (GDPR) for European users, including ”Right to be Forgotten” features.
   b. All payment handling must be compliant with the Payment Card Industry Data Security Standard (PCI-DSS).
   c. Users must explicitly agree to the Terms of Service and Privacy Policy during registration.

D. Reuse Objectives:
   a. The authentication module (Signup/Login/Reset Password) should be developed as a standalone component for potential reuse in other company projects.
   b. The real-time notification service should be decoupled to allow integration with

different frontend clients (e.g., mobile apps) in the future.

Appendix A: Glossary

- Bid Increment: The minimum amount by which a new bid must exceed the previous high bid.
- Sniper: A user who places a bid in the very last seconds of an auction.
- Reserve Price: The minimum price a seller is willing to accept.
- Proxy Bidding: Automated bidding where the system bids on behalf of the user up to a max limit.
- Soft Close: Extending the auction time if a bid is placed in the final moments to prevent sniping.
- JWT: JSON Web Token, used for secure transmission of information between parties.

Appendix B: Analysis Models

B.1 Use Case Diagram

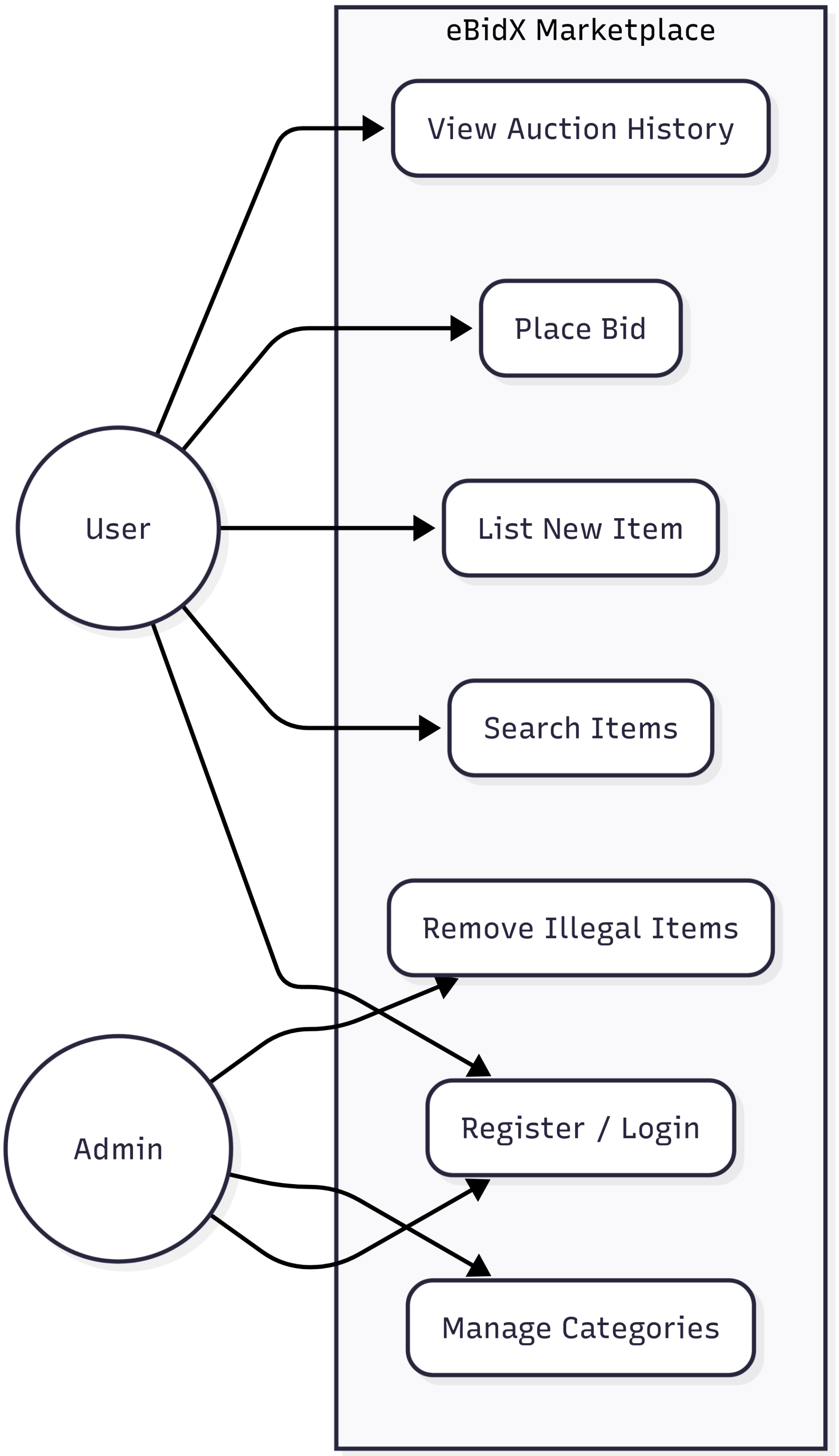

Figure 1: Use Case Diagram for eBidX

B.2 Data Flow Diagrams

Level 0: Context Diagram

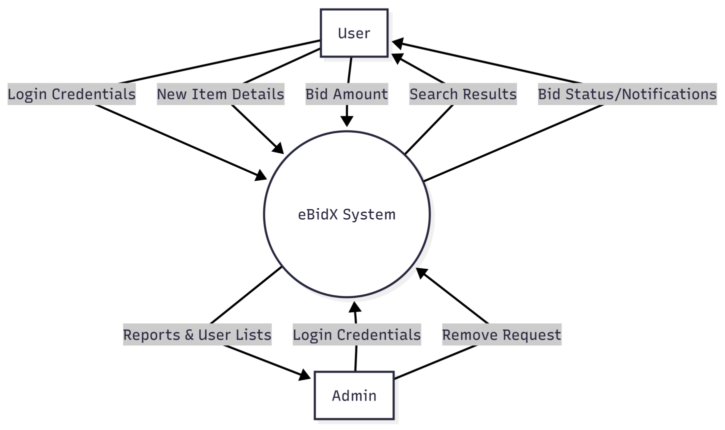

Figure 2: Level 0 Data Flow Diagram

Level 1: Process Decomposition

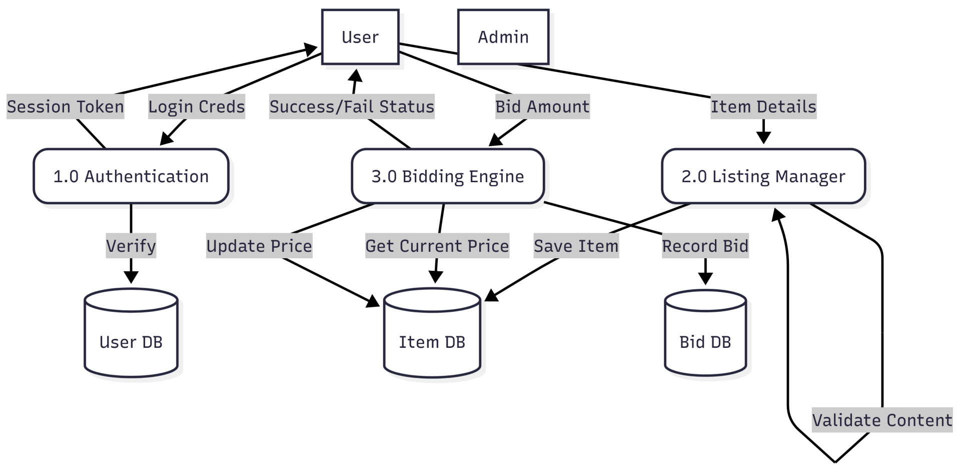

Figure 3: Level 1 Data Flow Diagram

B.3 Activity Diagram

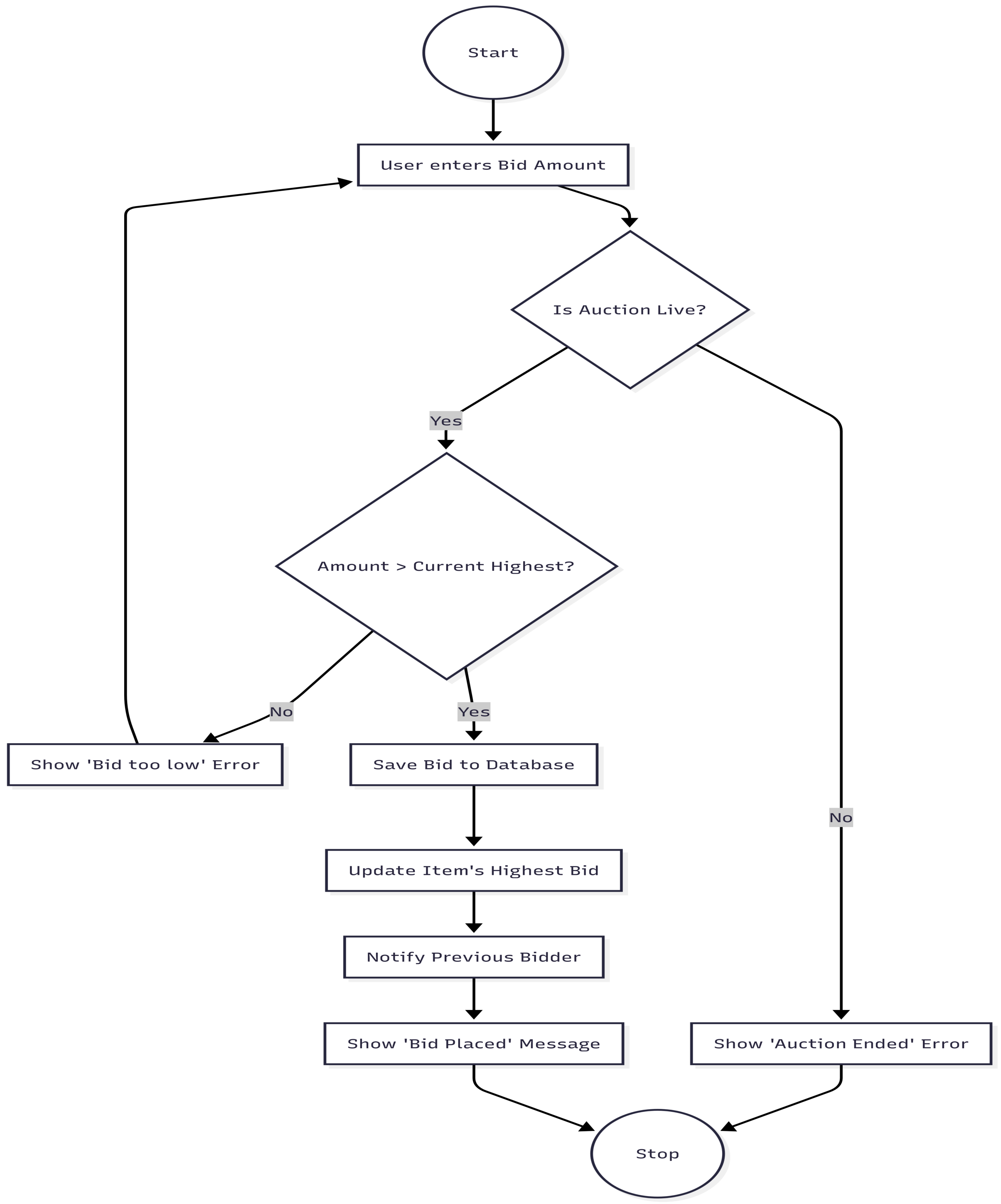

Figure 4: Activity Diagram for Bidding Process

B.4 Entity Relationship Diagram

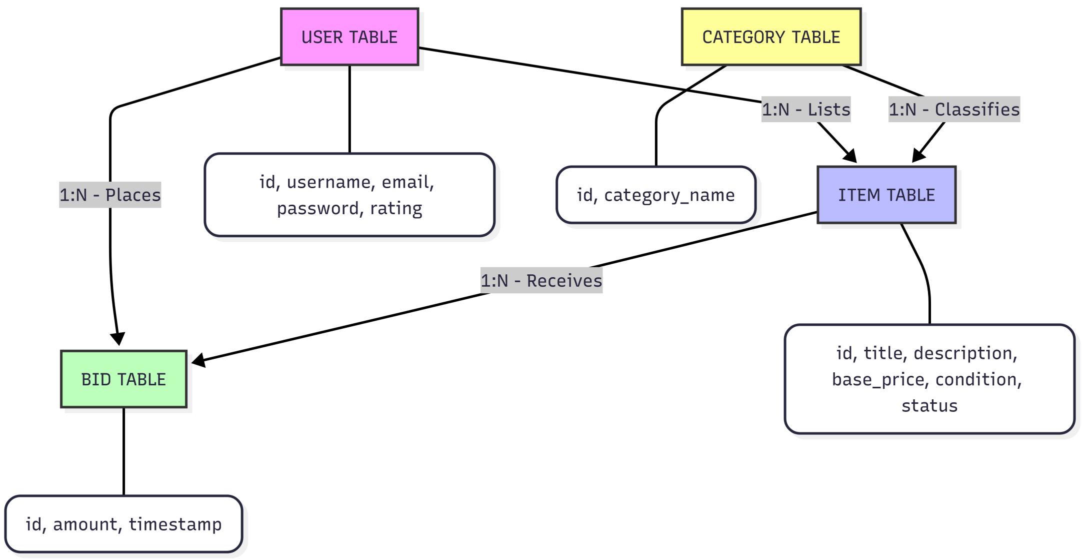

Figure 5: Entity Relationship Diagram for eBidX Database

B.5 Sequence Diagram

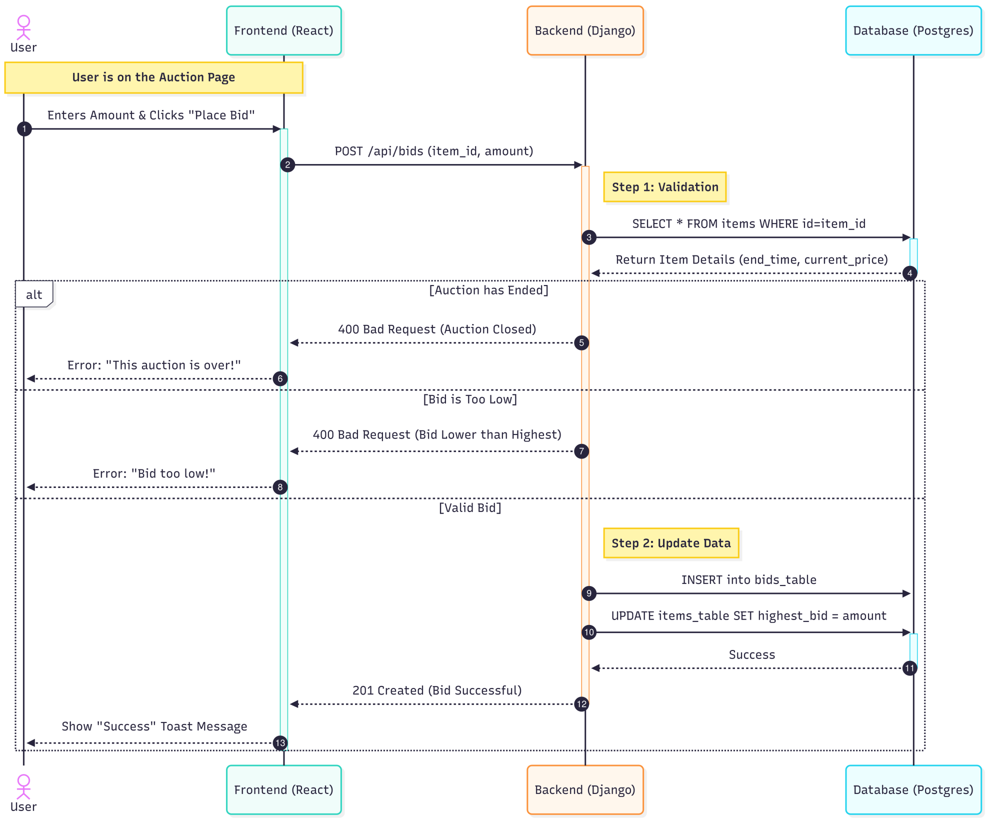
User
Frontend (React)
Backend (Django)
Database (Postgres)
User is on the Auction Page
Enters Amount & Clicks "Place Bid"
POST /api/bids (item_id, amount)
Step 1: Validation
SELECT * FROM items WHERE id=item_id
Return Item Details (end_time, current_price)
alt
[Auction has Ended]
400 Bad Request (Auction Closed)
Error: "This auction is over!"
[Bid is Too Low]
400 Bad Request (Bid Lower than Highest)
Error: "Bid too low!"
[Valid Bid]
Step 2: Update Data
INSERT into bids_table
UPDATE items_table SET highest_bid = amount
Success
201 Created (Bid Successful)
Show "Success" Toast Message

Figure 6: Sequence Diagram for eBidX Database

Appendix C: To Be Determined

- **A. Payment Gateway:** Finalizing the choice between Stripe Connect and PayPal Marketplace for handling multi-party payments.
- **B. Dispute Resolution:** Defining the exact protocol for handling item disputes (e.g., item not as described).
- **C. Mobile App:** Deciding whether to build a native mobile app (React Native) or stick to a PWA (Progressive Web App).