Product Requirement Document (PRD)

Product Name: Decentralized USDT Escrow System
Version: 1.0
Status: Initial Specification
Target Teams: Backend Engineering, Blockchain Engineering

---

1. Overview

1.1 Product Description

The product is a **web-based decentralized escrow system** built on a **private blockchain**, enabling secure peer-to-peer (P2P) transactions between two users using **USDT** on either the **Ethereum** or **Binance Smart Chain (BSC)** network.

The system uses **automated smart contracts and a bot-driven workflow** inside a private chat room to manage transaction agreements, fund custody, verification, and settlement without requiring a trusted third party.

1.2 Goals

- Enable trust-minimized P2P transactions using USDT
- Remove centralized custody of user funds
- Provide a guided, bot-driven escrow experience
- Support Ethereum and BSC networks
- Ensure transparency and transaction traceability

1.3 Out of Scope (v1)

- Full dispute resolution logic on-chain

- Arbitration smart contracts
- Multi-token or cross-chain bridging
- Fiat on/off ramp
- Advanced KYC / AML

---

2. User Roles

Role	Description
User	Authenticated platform user
Sender	User who sends USDT into escrow
Receiver	User who receives USDT from escrow
Bot	Automated system agent handling logic
Admin (Future)	Staff reviewing disputes (mock only in v1)

---

3. Supported Networks & Assets

- **Asset:** USDT
- **Networks:**
    - Ethereum (ERC-20)
    - Binance Smart Chain (BEP-20)
- Each room is locked to **one network** at creation.

---

4. High-Level System Flow

1. User authentication
2. Room creation / joining
3. Role selection (Sender / Receiver)
4. Deal amount agreement
5. Fee configuration (1%)
6. Escrow address generation
7. Deposit detection
8. Release or cancellation
9. Settlement
10. Transaction recording

---

5. Detailed User Flow & Functional Requirements

5.1 Authentication

Backend Requirements

- User login system
- Persistent user ID, username
- Session or token-based authentication

5.2 Room Creation & Joining

Backend

- Create room with:

    - Unique room code
    - Selected network (ETH or BSC)
    - Status: `OPEN`
- Allow another user to join via room code
- Limit room to **exactly 2 users + 1 bot**

Constraints

- Room network is immutable
- No third user allowed

5.3 Chat Room & Bot Presence

Backend

- Real-time or pseudo-real-time chat (polling acceptable for v1)
- Bot messages generated by system logic
- Store chat history

5.4 Role Selection

Bot Flow

- Bot presents:
    - Sender
    - Receiver
    - Reset selection

- Both users must confirm roles
- Roles must be mutually exclusive

Backend

- Persist role mapping
- Allow reset before next stage

5.5 Deal Amount Agreement

Bot Flow

- Bot asks for deal amount (USDT)
- Sender proposes amount
- Receiver must confirm
- Both confirmations required to proceed

Validation

- Amount > 0
- Amount precision follows USDT decimals

5.6 Fee Configuration (1%)

Bot Flow

- Bot calculates:
    1. Deal amount
    2. 1% fee

- Bot presents fee payment options:
    1. Sender pays full fee
    2. Receiver pays full fee
    3. Split fee (50/50)

Backend

- Store fee payer configuration
- Calculate required deposit amount dynamically

5.7 Escrow Address Generation

Blockchain

- Deploy a new escrow smart contract instance
- Contract must:
    - Accept USDT
    - Store sender & receiver addresses
    - Enforce fee logic
    - Allow release or refund

Backend

- Trigger deployment
- Store:
    - Contract address
    - Network

    - Expected deposit amount

5.8 Deposit Detection

Bot Behavior

- Display:
    - Escrow address
    - Exact amount to send

Blockchain / Backend

- Monitor blockchain for incoming USDT
- Detect the closest matching amount
- Wait for required confirmations

After Confirmation

- Bot announces funds received
- Present options:
    - Release
    - Cancel

5.9 Release Flow

Sender Action

- Sender selects **Release**
- Double confirmation required

Bot Flow

- Bot requests receiver wallet address
- Validate address format
- Trigger smart contract release

Blockchain

- Transfer USDT to receiver
- Transfer fee to platform wallet

5.10 Cancellation Flow

Sender Action

- Sender selects **Cancel**
- Receiver must confirm cancellation

Bot Flow

- Request sender wallet address
- Refund funds via smart contract

5.11 Completion & History

Backend

- Mark deal as `COMPLETED` or `CANCELLED`
- Close room (read-only)
- Record transaction:

    - Users involved
    - Network
    - Amount
    - Fee
    - Tx hash

Public Access

- Transaction history viewable publicly
- No private chat content exposed

---

6. Dispute Handling (Mock – v1)

Trigger

- User clicks **Report** button

Bot Flow

- Ask for:
    - Explanation
    - Proof (text only or file reference)

Backend

- Store dispute data
- Flag room for admin review
- No automated resolution required

---

7. Timeout Rules

7.1 Pre-Funding Timeout

- If no interaction **15 minutes** before deposit stage:
    - Room auto-closes
    - Status: `EXPIRED`

7.2 Funding Timeout

- After escrow address shown:
    - If no deposit within **30 minutes**
    - Bot asks:
        - Continue waiting
        - Cancel deal

---

8. Backend Requirements Summary

- Authentication & user management
- Room lifecycle management
- Bot logic engine
- Blockchain interaction (ethers.js / viem)
- Timeout scheduler
- Transaction history storage

- Dispute data storage

---

9. Blockchain Requirements Summary

- USDT escrow smart contract
- Support ETH & BSC
- Fee logic enforcement
- Release & refund functions
- Event emission for monitoring
- One contract per deal

---

10. Non-Functional Requirements

- Backend never holds user funds
- Smart contracts immutable after deployment
- Private keys securely stored
- All blockchain actions logged
- Clear separation of backend & blockchain layers

---

11. Success Metrics

- Successful escrow completion rate
- Zero backend fund custody

- Accurate fee calculation
- No unauthorized releases
- Transparent transaction history