Product Requirements Document (PRD)

AI Wallet Assistant Frontend

Version 2.0 | Optimized for Intuitive User Experience

---

1. Introduction

The **AI Wallet Assistant** aims to provide a **seamless, AI-driven interface** for managing cryptocurrency transactions. While traditional interfaces like **Discord** cater to community-based interactions, this product focuses on **efficiency, clarity, and simplicity** in handling wallet-related commands.

Why Not Discord?

While Discord's UI is designed for community interactions, the AI Wallet Assistant requires a **focused, transactional experience**. Below is a comparison:

Discord UI	AI Wallet Assistant UI
Built for group chats and channels.	**1:1 conversational focus** (user ↔ AI).
Complex menus for roles, servers, and bots.	**Zero distractions**—only commands and transaction confirmations.
Notification overload.	**Progressive disclosure**: Only critical alerts (e.g., low balance).

Winning UI Inspiration

- **ChatGPT's minimalism + MetaMask's branding**
- **Mobile banking apps** (clean transaction flows)
- **Voice assistants** (e.g., Google Assistant) for guided actions

---

2. Core UI Requirements

Key Principles

1. **Zero Learning Curve** – Users should immediately know how to interact.
2. **Glanceable Security** – Transaction details must be clear and unambiguous.
3. **MetaMask Integration** – Should feel like a natural extension of the wallet.

UI Components

1. **Command Input Bar**

    - Placeholder text: *"What would you like to do? Try 'Send 0.1 ETH to…"*
    - Autocomplete for common commands: *"Swap," "Balance," "Transaction History"*
2. **Chat History Panel**
    - User messages on the right (purple bubbles).
    - AI responses on the left (gray bubbles, Markdown-supported).
    - Interactive buttons for quick actions (e.g., "Confirm" or "Edit").
3. **Transaction Summary Modal**
    - Displays recipient address, amount, network, and gas fee.
    - **"Confirm in MetaMask"** button triggers wallet popup.
4. **Tutorial Overlay**
    - First-time user walkthrough: *"Type a command like 'Show my balance' to start!"*
    - Persistent help button with an *"Examples"* dropdown.

---

3. Wireframes

Desktop & Mobile Views

(Include Figma links or images in final document.)

1. **Connected Wallet State**
    - Top bar: Network indicator, ETH balance, account avatar.
    - Left sidebar (collapsible): Command history and saved templates.
2. **Transaction Flow**
    - User types *"Send 0.1 ETH to 0x123…"*
    - AI parses command and displays summary:
    - ✅ `**Parsed Command**`
    - `- **Action**: Send ETH`
    - `- **Amount**: 0.1 ETH (~$180)`
    - `- **Recipient**: 0x123...789 (ENS: **alice.eth**)`
    - `- **Gas Fee**: $1.20`
    -
    - `[Confirm in MetaMask] [Edit]`
    - Clicking "Confirm" opens MetaMask for signing.

---

4. Design System

Visual Identity

- **Primary Color**: MetaMask Orange (`#F6851B`)
- **Secondary Colors**: Slate (`#64748B`), Green (`#22C55E` for success).
- **Typography**:
    - **Inter** (modern, readable).
    - **Roboto Mono** (for wallet addresses/JSON).

Animations & Responsiveness

- Smooth modal transitions.
- Loading spinner featuring Ethereum logo.
- **Mobile-first approach** with larger tap targets.

---

5. User Flows

Flow 1: First-Time User Onboarding

1. User connects MetaMask.
2. Tutorial overlay provides example commands.
3. Empty chat state suggests: *"Try 'What's my balance?'"*

Flow 2: Transaction Error Handling

1. User types invalid command (e.g., sending more ETH than available).
2. AI responds:

```
3. ✕ **Error**
4. - Your balance is **1.0 ETH**.
5. - Adjust the amount or [Add Funds].
```

---

6. Technical Implementation

Frontend Stack

- **Framework**: Next.js + TypeScript
- **Styling**: Tailwind CSS + Radix UI for accessibility
- **MetaMask SDK Integration**
    - Auto-reconnect on reload.
    - Listen for `disconnect` events to reset UI.

AI Parsing & Performance

- Store common commands in `localStorage` to **reduce API calls**.
- **Lazy-load chat history** for active users.
- Optimize images with `next/image`.

---

7. PRD Summary

Why This UI Wins

1. **Familiar & Intuitive** – Mimics chat interfaces (WhatsApp, ChatGPT).

2. **Trustworthy** – Aligns with MetaMask's branding for user confidence.
3. **Hackathon Ready** – The value proposition is clear within **10 seconds**.

Key Differentiators

- **Command Autocomplete** reduces friction.
- **ENS Integration** (displays names instead of raw addresses).
- **Progressive Disclosure** (hides advanced options by default).

---

8. Final Deliverables

- **GitHub Repository** (MIT License).
- **2-Minute Loom Demo Video** showcasing:
    1. Wallet connection.
    2. Command → Transaction flow.
    3. Error handling.
- **Live Demo URL** (Vercel/Netlify).

This document presents a structured approach for developing a highly intuitive AI Wallet Assistant frontend. **By blending AI-powered command parsing with MetaMask's transactional capabilities, we create an efficient, frictionless experience for users.**