PoC Plan: Intent-Based Swap Chatbot (1-click SDK Version)

Objective: To validate the feasibility and superior user experience of an intent-based transaction system by using the **one-click TypeScript SDK** to power a conversational UI. This is the fastest path to a high-fidelity demo.

Key Success Metrics: (Unchanged)

1. **Task Completion Rate:** >90% of users can successfully initiate and sign a swap transaction via the chat UI.
2. **Time to Transact:** The time from stating the intent to the Privy wallet prompt appearing is under 5 seconds.
3. **Qualitative Feedback:** A clear "wow" factor from test users regarding the simplicity of the process.

Core Tech Stack:

- **Frontend:** Next.js Boilerplate
- **Wallet:** Privy
- **UI/Chat:** Simple React chat component
- **Backend:** Next.js API Routes
- **Protocol SDK: @defuse-protocol/one-click-sdk-typescript**

Final High-Level Architecture

By using the SDK, our backend becomes a lightweight, secure wrapper. All the complex logic of communicating with the NEAR Intents system is handled by the SDK.

```
[ User ]
   |
   v
[ Next.js Frontend ] -- (Privy SDK for Auth & Signing)
```

```
   |   - Chat UI
   |
   v  (User intent as API call: e.g., { from: 'NEAR', to: 'USDC', amount: 1 })
   |
[ Next.js API Backend ]
   |   - Initializes the 1-click SDK
   |   - Calls the SDK's `getQuote()` method with the user's intent
   |
   v
[ @defuse-protocol/one-click-sdk-typescript ]
   |   - Handles all communication with the NEAR Intents API
   |   - Returns a quote and a pre-constructed transaction object
   |
   v
[ Next.js API Backend ]
   |   - Receives the quote and transaction from the SDK
   |   - Sends the simple transaction back to the frontend
   |
   v
[ Next.js Frontend ]
   |   - Receives the simple transfer transaction
   |   - Passes it to Privy SDK to prompt user for signature
   |
   v  (Signed transaction)
   |
[ NEAR Testnet ] --> (Funds sent to depositAddress, intent is executed by solvers)
```

Final User Flow

The flow is now maximally efficient from a development standpoint.

1. **Onboarding & Auth:** (Unchanged) User logs in with Privy.
2. **Stating Intent:** (Unchanged) User types swap 1 NEAR for wUSDC.
3. **Agent Resolution (SDK-Powered):**
   - The backend initializes the 1-click SDK and calls its getQuote() method, passing in the parsed user intent.
   - The SDK handles the API request and returns a full quote object, which includes the

human-readable details *and* the necessary transaction for the deposit.

4. **Confirmation:** The chatbot uses the quote data to display the summary: "I can swap 1 NEAR for ~1.5 wUSDC. Ready to proceed?".
5. **Signature (Simplified):**
    - The user clicks "Confirm."
    - The frontend passes the simple **transfer transaction** (received directly from our backend via the SDK) to Privy.
    - The Privy modal pops up with a clear, human-readable action: **"Send 1 NEAR to [deposit_address]?"**.
6. **Execution & Feedback:**
    - Once the user signs, the transfer is sent.
    - Our chatbot can use the SDK's status-checking methods to poll for updates and inform the user of the progress in real-time.

Final 1-Week Sprint Plan

The SDK simplifies the backend work to a single day, allowing us to deliver a more polished and robust PoC within the week.

Step 1 Setup & Frontend Foundation

- **Task:** Initialize Next.js project, integrate Privy SDK, and build the login flow.
- **Task:** Build the basic chat UI.
- **Goal:** A user can log in with Privy and chat with a "dumb" bot.

Step 2: Backend SDK Integration

- **Task:** Create the /api/intent-handler endpoint in Next.js.
- **Task:** In this endpoint, npm install and initialize the @defuse-protocol/one-click-sdk-typescript.
- **Task:** Implement the logic: parse the request from the chat UI, call the SDK's getQuote() function, and return the resulting transaction object.
- **Goal:** The API endpoint can receive an intent and, using the SDK, return a valid, simple transfer transaction object. **This should be a very fast day.**

Step 3: End-to-End Integration & Status Polling

- **Task:** Connect the frontend chat input to our new /api/intent-handler backend.
- **Task:** When the transaction object is received, pass it to the Privy wallet for signing.
- **Task:** Implement the status polling feature using the SDK's methods to provide live

updates in the chat UI.
- **Goal:** The full, interactive user flow is complete.

Step 4: Polish, Error Handling & Demo Prep

- **Task:** Implement robust error handling (e.g., what if the SDK can't find a quote?).
- **Task:** Refine the chat UI with loading states, clear success messages (with explorer links), and helpful error messages.
- **Task:** Clean up the code and prepare the final demo script.
- **Goal:** A polished, demonstrable, and resilient PoC.

This is the most direct and professional way to execute. It minimizes custom code, relies on the official SDK, and focuses our efforts on the user experience. I'll get the project boilerplate initialized with the SDK as a dependency. Ready when you are.