XPulse: The Tribunal — Production Blueprint (CliqTrix Edition)

Target: Zoho CliqTrix Hackathon (Finance & Collaboration Category)

Objective: Build a Web3 "Forensic Auditor" that lives inside Zoho Cliq.

1. Goal (The Winning Pitch)

Deliver a production-ready Zoho Cliq extension named XPulse that transforms invoice approval into a multi-signature blockchain event.

Instead of a passive log, XPulse acts as an active "Tribunal Bot" inside Cliq. It connects Zoho Books to Polygon, intercepts "Invoice Paid" events, triggers a collaborative approval card in a Cliq Channel, and—only upon consensus—mints an immutable "Proof of Payment" receipt.

The Hook: "Zoho Books for accounting. XPulse for forensic truth."

2. Architecture (High-Level)

Flow:

1. **Zoho Books:** User marks Invoice #INV-1024 as Paid.
2. **XPulse Backend:** Receives Webhook → Verifies Data → Pushes **Adaptive Card** to Cliq Channel (#Finance-Audit).
3. **The Tribunal (Cliq):** The bot demands **2 unique approvers** (e.g., Accountant & Manager) to click [Mint Proof] on the card.
4. **Consensus Engine:** FastAPI counts votes. If votes >= 2, it triggers the Blockchain Service.
5. **Blockchain:** Node.js signs transaction → Writes hash(invoice) to Polygon.
6. **Closure:** Bot updates the original Cliq message with a green "Verified on-chain" badge and Transaction Hash.

3. Tech Stack (CliqTrix Specialized)

Component	Technology	Why Chosen for CliqTrix
User Interface	**Zoho Cliq (ZML + Adaptive Cards)**	**Critical.** Keeps users inside the chat. No external websites.
Admin UI	**Zoho Books Web Tab**	Simple HTML settings page embedded in Books (just for setup).
Backend API	**FastAPI (Python)**	Async handling of high-concurrency webhooks from Zoho.
Task Queue	**Redis + RQ**	Prevents bot timeouts during blockchain operations.
Database	**PostgreSQL**	Stores "Pending Votes" and mapping of Cliq User IDs to Wallets.
Blockchain	**Polygon (Amoy Testnet)**	Fast, cheap EVM chain.
Smart Contract	**Solidity (Tribunal.sol)**	Custom contract handling logReceipt with approver_ids.
Signer	**Node.js + ethers.js**	Securely manages the "Relayer" private key.
Hosting	**Render / Railway**	One-click deploy for Python + Node microservices.

4. Schema & Data Model

•

invoices_pending: (id, invoice_id, amount, payer_name, status, required_votes, current_votes)

- votes: (id, invoice_id, cliq_user_id, timestamp)
- users: (cliq_user_id, wallet_address, role)
- receipts: (id, invoice_id, tx_hash, block_number, metadata_uri)

5. API Endpoints (Core)

- POST /webhooks/zoho-books: Ingests "Invoice Paid" event.
- POST /cliq/interaction: The main handler. Receives **Button Clicks** from the chat.
- GET /widget/history: Returns ZML (Zoho Markup Language) for the sidebar widget.
-

POST /admin/link-wallet: Maps a Cliq User ID to a Wallet Address (for the admin tab).

6. The "Tribunal" Logic (The Unique Feature)

- **Trigger:** When Invoice Paid > $1,000.
- **Logic:**
  - Bot posts Card with UUID session_123.
  - User A clicks Approve. Backend checks: "Has User A voted on session_123?" No -> Record Vote.
  - User A clicks again. Backend checks: "Has User A voted?" Yes -> Ignore (Prevent double voting).
  - User B clicks Approve. Total Votes = 2. **Trigger Blockchain Mint.**

7. Smart Contract: Tribunal.sol

Solidity

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract Tribunal {
event ProofMinted(string indexed invoiceId, string[] approvers, uint256 timestamp);
function mintProof(string memory invoiceId, string[] memory approverIds, bytes32 dataHash) external {
// Only the Relayer (XPulse Server) can call this
emit ProofMinted(invoiceId, approverIds, block.timestamp);
}
}
```

8. In-Zoho UI (Deliverables)

1.

The Bot (XPulse Bot):

- **Icon:** Use a shield or radar pulse logo.
- **Command:** /xpulse status [invoice_id] (Returns current blockchain status).

2.

The Widget (XPulse History):

- A right-sidebar widget in Cliq.
- **Tab 1:** "Recent Proofs" (List of last 10 minted receipts).
- **Tab 2:** "My Approvals" (Gamification: How many invoices did I verify?).

9. Security Checklist

- **Relayer Pattern:** Do NOT ask users to sign with MetaMask in chat. The Server signs the transaction (Gasless for users).
- **Vote Debouncing:** Ensure Redis locks the voting process so one user cannot spam-click to fake a consensus.
- **Secret Management:** Private Keys stored in Railway/Render Environment Variables.

10. Deployment Timeline (Hackathon Mode)

- **Day 1:** FastAPI Setup + Zoho Books Webhook connection.
-

Day 2: Build the "Adaptive Card" JSON and get the Bot posting to a channel.

- **Day 3:** Implement the "Button Click" handler in Python (The Vote Logic).
- **Day 4:** Write & Deploy Tribunal.sol to Polygon Amoy.
- **Day 5:** Connect the two: Button Consensus -> Triggers Blockchain Transaction.
- **Day 6:** Build the ZML Widget (Sidebar history).
- **Day 7:** Demo Video Recording & Pitch Deck.

11. Demo Script (For the Video)

1. **Intro:** "B2B Fraud is real. Approvals are messy. XPulse fixes both using Zoho Cliq + Polygon."
2.

The Action:

- Screen Split: Zoho Books on Left, Cliq on Right.
- User marks Invoice as Paid in Books.
- Boom: XPulse Card appears in Cliq immediately.

3.

The Collaboration:

- "I am the Manager." (Click Approve). Bot says: "1/2 Votes".
- "I am the Auditor." (Click Approve). Bot says: "2/2 Votes. Minting..."

4.

The Payoff:

- The Card turns **Green.** A link to PolygonScan appears.
-

Click the link -> Show the transaction on-chain.

5.

The Widget: Open the sidebar. Show the new receipt listed there.

12. Deliverables to Antigravity

- **Source Code:** Monorepo with backend/ and smart-contracts/.
- **Zoho Bundle:** The plugin.json for the extension.
- **Bot Script:** The Deluge/JSON code used for the Cliq handler.
- **Pitch Deck:** 10 Slides focusing on "Chat-First Finance".
- **Video:** 2 Minutes max. Fast cuts. High energy.