

# Project Design Document: MicroTerm

## *The Unbundled, Micro-Payment Financial Terminal*

### 1. Executive Summary

The Problem: Institutional terminals (Bloomberg, PitchBook) cost \$25,000/year because they bundle every data point into one license. Most users only need specific insights (e.g., one cap table, one whale alert).

The Solution: MicroTerm is a modular intelligence platform. It aggregates high-value financial data but hides it behind a "Micro-Paywall".

The Mechanism: Using the x402 Protocol (HTTP 402 Payment Required), users pay small amounts (\$0.25 - \$1.00) in USDC on the Base network to unlock specific data points instantly.

---

### 2. System Architecture

The system consists of three distinct layers:

1. **The Data Factory (Python Backend):** A set of 24/7 scripts that ingest, clean, and store data.
  2. **The API & Ledger (Node/Next.js):** Manages the paywall, verifies blockchain transactions, and serves unblurred data.
  3. **The Terminal (React Frontend):** The user interface that displays public data for free and handles the wallet interactions for premium data.
- 

### 3. The "x402" Monetization Protocol

This is our core business logic. We do not use monthly subscriptions. We use **Atomic Unlocks**.

#### The Workflow

1. **Request:** User clicks "Unlock Deal Details." Frontend requests GET /api/deal/123.
2. **Denial:** Server checks DB. If paid == false, return **HTTP 402 (Payment Required)**.
  - *Payload:* { "cost": 0.50, "currency": "USDC", "chain": "Base", "recipient": "0xOurWallet" }
3. **Action:** Frontend triggers OnchainKit (Coinbase) to prompt the user's wallet.
4. **Payment:** User signs transaction. 0.50 USDC moves to our Treasury Wallet.
5. **Verification:** Frontend sends tx\_hash to Server. Server verifies hash on-chain.
6. **Delivery:** Server records the purchase and returns the un-blurred JSON.

---

## 4. The Data Factory (The Backend Engine)

*Stack: Python, SQLite (MVP) or PostgreSQL (Production).*

This engine runs four parallel workers to generate our "Inventory."

### Module A: Private Market Intelligence (SEC)

*Goal:* Detect startup fundraising before the news reports it.

- **Source:** SEC EDGAR RSS Feed (Form D).
- **Logic:**
  1. Poll RSS feed every 10 minutes.
  2. Parse XML for Form D (Notice of Exempt Offering).
  3. Extract: <issuerName>, <offeringAmount>, <industryGroup>.
  4. **Filtering:** Ignore deals under \$1M. Flag deals > \$50M as "High Signal."
- **The Product:** "Stealth Equity Alert."
  - *Free:* "New AI Company raising capital."
  - *Paid (\$0.50):* "Anthropic just filed for a \$450M raise. Link to filing."

### Module B: On-Chain Intelligence (The "Decoder")

*Goal:* Translate raw blockchain noise into readable trading signals.

- **Source:** Alchemy/Infura RPC Node (Base Mainnet).
- **Sub-Module 1: The Whale Watcher**
  - Monitor Transfer events for USDC/ETH.
  - If Value > \$1,000,000: Trigger Alert.
- **Sub-Module 2: The Calldata Decoder**
  - Capture pending transactions in the mempool or latest block.
  - Use web3.py + Uniswap ABIs to decode input data.
  - *Identify:* Is it a Swap? Add Liquidity? Remove Liquidity?
- **Sub-Module 3: The Address Labeler**
  - Maintain a local table: known\_addresses (address, label, category).
  - *Lookup:* When a tx comes in, check if sender/receiver is in the table.
  - *Value Add:* Replace 0x7a2... with "**Wintermute Trading**".

### Module C: Global News Aggregator

*Goal:* Clean, deduplicated financial news.

- **Source:** RSS Feeds (Reuters, CoinDesk, PR Newswire).
- **Enhancement:**
  - Use a lightweight LLM (GPT-4o Mini) to generate a **one-sentence summary**.
  - *Sentiment Analysis:* Tag news as "Bullish" or "Bearish."

- **The Product:**
  - *Free:* Headlines.
  - *Paid (\$0.10):* "Impact Analysis: Why this news affects ETH price."

## Module D: Public Market Data

*Goal:* The "Hook" to keep users on the site.

- **Source:** Yahoo Finance (yfinance) or CoinGecko API.
  - **Data:** Live prices for BTC, ETH, SOL, NVDA, COIN.
  - **Status: Always Free.** This is the "Loss Leader" to demonstrate the terminal is live.
- 

## 5. The Alerts System

Users don't stare at screens all day. They pay for proactive notifications.

The Trigger Logic:

We run a "Rule Engine" against incoming data.

- IF (SEC\_Filing > \$10M) AND (Sector == 'AI') -> PUSH\_ALERT
- IF (Wallet == 'Wintermute') AND (Action == 'Buy') -> PUSH\_ALERT

The Delivery:

- **Free:** In-App Notification (Unlocks required to view details).
  - **Paid:** "Pro Mode" allows auto-unlocking of alerts (Pre-paid credits).
- 

## 6. Frontend & User Experience

*Stack:* Next.js, Tailwind CSS, Lucide Icons, Wagmi/OnchainKit.

### Visual Language

- **Theme:** "Terminal Dark" (Black background, monospaced fonts, high contrast).
- **The Blur:** High-value fields (amounts, names) are blurred with CSS backdrop-filter: blur(4px).

### The Dashboard Layout

1. **Ticker Tape (Top):** Live prices (Free).
2. **Left Panel (The Feed):** A mixed stream of SEC filings, News, and Whale Alerts.
  - *UX:* Cards with "Unlock" buttons overlaying the blurred text.
3. **Right Panel (Deep Dives):**
  - *Cap Table Analyzer:* Visual breakdown of investors (Premium).
  - *Whale Map:* Visualization of money flow (Premium).

---

## 7. Implementation Roadmap

### Phase 1: The "Skeleton" (Days 1-2)

- Set up Next.js repo.
- Integrate OnchainKit (Connect Wallet).
- Build the "Mock Data" UI (Hardcoded blurred items).
- **Goal:** Prove the "Click -> Pay -> Unlock" loop works on Base.

### Phase 2: The "Factory" (Days 3-5)

- Set up the Python script.
- Write `fetch_sec_filings()` and `fetch_public_markets()`.
- Store data in `financial_data.db`.
- Create an API route to serve this DB to the frontend.

### Phase 3: The "Decoder" (Days 6-8)

- Connect Python script to Alchemy RPC.
- Implement `web3.py` decoding for Uniswap swaps.
- Build a small list of "Known Addresses" (Exchanges, Top VCs).

### Phase 4: Launch (Day 10)

- Deploy Frontend to Vercel.
- Deploy Backend (Python) to a VPS (e.g., Railway/DigitalOcean).
- Seed the DB with 24 hours of data.

---

## 8. Database Schema (SQL Reference)

Table: `private_deals`

SQL

```
id INTEGER PRIMARY KEY,  
company_name TEXT,  
amount_raised REAL, -- The Filter (e.g., >$1M)  
filing_url TEXT,    -- Source of Truth  
sector TEXT,        -- e.g., "Technology"  
is_premium BOOLEAN -- True
```

### Table: whale\_alerts

SQL

```
id INTEGER PRIMARY KEY,  
tx_hash TEXT,  
sender_address TEXT,  
sender_label TEXT, -- e.g., "Binance Hot Wallet"  
receiver_address TEXT,  
token_symbol TEXT, -- e.g., "USDC"  
amount REAL,  
timestamp DATETIME,  
is_premium BOOLEAN -- True
```

### Table: user\_unlocks

SQL

```
user_wallet TEXT,  
item_id INTEGER,  
tx_hash TEXT, -- Proof of Payment  
unlocked_at DATETIME
```