Execution Plan: Automated Escrow & Group Ownership Transfer Bot

---

Phase 0 – Foundation (Week 1–2)

Before building, you need to clarify the foundations.

Tasks

1. Finalize requirements (document we enhanced).
2. Choose **crypto(s)** for escrow (USDT (TRC20/ETH), BTC, or stablecoin).
3. Decide hosting stack (e.g., AWS / DigitalOcean / Heroku).
4. Decide if escrow will be **smart contract-based** or **custodial wallet-based**.

Tech to Study / Prep

- **Telegram Bot API** basics.
- **Crypto payment APIs** (Binance Pay API, Coinbase Commerce, NOWPayments, or direct Web3 smart contracts).
- **Databases** (PostgreSQL recommended).
- **Basic backend frameworks** (Django/DRF or FastAPI).

Phase 1 – Escrow Core (Month 1)

Build the escrow/payment backbone.

Tasks

1. Integrate crypto payments:
    - Option A: Custodial → APIs like Binance Pay / Coinbase Commerce.
    - Option B: Non-custodial → Smart contract escrow (Ethereum Solidity, Polygon, or Tron).
2. Escrow logic:
    - Deposit → Hold → Conditional Release → Refund.
    - Store transactions in DB.
3. Build admin dashboard (web panel):
    - Track escrow states.
    - Manual override for disputes.

Tech to Study

- **Smart contracts (Solidity + Web3.py)** if going blockchain route.
- **Crypto payment APIs** if centralized approach.
- Django or FastAPI for backend.

Deliverable

✅ Working escrow wallet service (can accept payments, hold funds, and release/refund manually).

Phase 2 – Telegram Bot Core (Month 2)

Build the bot and connect it with escrow system.

Tasks

1. Bot features:

    - /start → register user.
    - Buyer: browse listings → pay escrow.
    - Seller: create listing → validate bot is admin in group.
2. Group monitoring:
    - Use `getChatAdministrators` to check admin list.
    - Detect ownership changes.
    - Snapshot group metadata.
3. Transaction state machine inside bot:
    - Funded → Awaiting transfer → Ownership verified → Escrow release/refund.

Tech to Study

- **python-telegram-bot** or **aiogram**.
- Telegram Bot API methods:
    - `getChat`, `getChatAdministrators`, `getChatMember`, `exportChatInviteLink`.
- Webhooks vs polling (webhooks preferred).

Deliverable

✅ Bot can manage escrow lifecycle and detect group ownership transfers.

Phase 3 – Verification & Disputes (Month 3)

Add robustness and safety.

Tasks

1. Automated verification of metadata:
    - Title, username, description, pinned message.
2. Dispute system:
    - Bot logs all actions.
    - Human admin panel to review disputes.
    - Arbitration flow: approve seller, refund buyer.
3. Logging & audit trail.

Tech to Study

- Django admin (for manual review).
- Logging frameworks (ELK stack optional).

Deliverable

✅ Disputes can be reviewed and resolved; bot generates reliable logs.

Phase 4 – Marketplace (Month 4–5)

Expand to a listing/discovery system.

Tasks

1. Sellers can list groups in the bot.
2. Bot validates group via admin status.
3. Listings posted in a channel or marketplace UI.
4. Buyers can browse → pay → engage escrow.

Tech to Study

- Marketplace logic (listings DB).
- Pricing conversions (USD ↔ crypto via API like CoinGecko).

Deliverable

✅ Marketplace live where groups can be discovered and bought.

Phase 5 – Security & Scaling (Month 6 onward)

Tasks

1. Multi-sig escrow or audited smart contract.
2. Scaling bot for thousands of users.
3. Add KYC-lite (optional, only if regulations require).
4. Expand to support **NFT-like "digital ownership certificates."**

Tech to Study

- Smart contract audits.
- Security best practices (SQL injection, escrow fraud prevention).
- Deployment automation (Docker, Kubernetes optional).

Timeline Overview

Phase	Duration	Deliverable
0. Foundation	2 weeks	Tech prep & decisions

1. Escrow Core	1 month	Escrow wallet + admin dashboard
2. Bot Core	1 month	Bot integrated with escrow
3. Verification & Disputes	1 month	Verification + dispute system
4. Marketplace	1–2 months	Listing/discovery marketplace
5. Security & Scaling	Ongoing	Audit, KYC-lite, scaling infra

Total: **~5–6 months MVP** → then ongoing scaling.

Risks & Restrictions

- **Telegram API restriction:** Bot cannot *force* transfer ownership. Seller must cooperate.
- **Crypto regulation:** Some countries may restrict crypto escrow usage.
- **Fraud attempts:** Need strong dispute mechanism.
- **Custodial risk:** If you hold funds yourself, you become a financial middleman → legal exposure.

Recommended Stack

- **Backend**: Django + Django REST Framework (you already know Django).
- **Database**: PostgreSQL.
- **Bot**: `aiogram` (async, scalable).
- **Payments**:
    - Short term: Coinbase Commerce / NOWPayments.

    - Long term: Solidity smart contracts.

- **Deployment**: Docker + AWS EC2 / DigitalOcean.
- **Disputes panel**: Django Admin + custom UI.

👉 My suggestion: Build **custodial MVP first** (simpler, fast to market), then upgrade to **smart contract escrow** once you validate demand.