# 🧾 BlockReceipt.ai – Autonomous NFT Purchase Bot (Under $0.10)

## 🎯 Purpose

Automatically purchase and transfer an NFT (under $0.10 in cost) to a user's wallet after they upload a purchase receipt. This flow must not charge the user any gas or transaction fees. Instead, the app operates a backend-funded wallet to execute NFT procurement and transfer based on marketplace API data.

## 🧠 Key Components

- Receipt upload trigger  
- NFT market scan (OpenSea API)  
- Purchase under $0.10 using backend wallet  
- Gasless transfer to user's wallet (ERC-721/ERC-1155)  
- Fraud/spam protection (1 NFT/day per wallet)  
- Logging and fallback mechanism

## 🔌 Marketplace API Integration

Use OpenSea API to search for affordable NFTs:  
Example endpoint:  
```  
GET https://api.opensea.io/api/v2/listings/collection/{slug}?limit=50&price\_max=0.1  
```  
- Use keyword filtering (e.g., from parsed receipt data like 'candy', 'electronics')  
- Pick 5–10 candidates randomly, then select one to buy

## ⚙️ Backend Wallet Flow (Node.js with ethers.js)

```ts  
import { ethers } from 'ethers';  
import ERC721ABI from './abis/ERC721.json';  
  
const provider = new ethers.JsonRpcProvider('https://polygon-rpc.com');  
const wallet = new ethers.Wallet(process.env.PRIVATE\_KEY, provider);  
  
async function buyAndSendNFT(nft, userWallet) {  
 const nftContract = new ethers.Contract(nft.contractAddress, ERC721ABI, wallet);  
  
 // Assuming direct marketplace purchase (adjust for marketplace protocol)  
 const tx = await nftContract.transferFrom(wallet.address, userWallet, nft.tokenId);  
 await tx.wait();  
  
 return { success: true, txHash: tx.hash };  
}  
```

## 🛡 Security & Rate Limiting

- Enforce: 1 NFT per wallet every 24 hours  
- Log all actions: wallet, IP, timestamp, receipt tier  
- Use captchas or auth if open to public  
- Store receipts to prevent abuse and replay attacks

## 💸 Budgeting & Cost Controls

- Use \*\*Polygon or Zora L2\*\* for low gas fees  
- Limit NFT price to $0.10 max (including gas)  
- Auto-rotate through a few collections or fallback art pool if market is dry  
- If gas exceeds threshold, cancel purchase gracefully

## 🧾 Trigger from Receipt Upload

Add this logic after receipt parsing:  
```ts  
if (receipt.total > 5.00) {  
 const userWallet = req.body.walletAddress;  
 const nft = await fetchRandomNFT();  
 const result = await buyAndSendNFT(nft, userWallet);  
 return res.status(200).json({ ...receiptData, nftResult: result });  
} else {  
 return res.status(200).json({ ...receiptData, note: 'No NFT for small purchases' });  
}  
```

## 🔁 Optional Fallback (If No Cheap NFT Found)

- Fallback to minting a custom pixel art NFT from internal collection  
- Use Replicate or local model to generate pixel art  
- Store art in IPFS or Replit local `/public/nft-images/` and mint to user  
- Keeps UX fun even if market is dry

## 📦 Replit Prompt

Build a backend function that triggers after a receipt is uploaded. Query OpenSea or a Polygon-based NFT indexer to find NFTs under $0.10. Automatically purchase one using a backend wallet and transfer it to the user's connected wallet. Add logic to prevent multiple daily claims from the same wallet. Use Polygon mainnet or Zora L2. Fallback to in-house NFT minting if no market item is available.