# BlockReceipt.ai – Comprehensive v28 Plan

## 1. v28 Codebase Audit Summary

- \*\*ERC-1155 Contract\*\*: `Receipt1155\_for\_remix.sol` supports dynamic URIs and receiptHash mapping.  
- \*\*Backend Flow\*\*: Single `/upload-and-mint` route covering upload → OCR → TACo encryption → IPFS pin → on-chain mint → metadata storage.  
- \*\*Frontend\*\*: `ReceiptUpload`, `NFTSelection`, `ReceiptGallery` components with `useWalletConnect` hook.  
- \*\*Tools & Scripts\*\*: Hardhat for deploy/verify, Vite for React, modular service files.  
- \*\*Pain Points\*\*: Repo clutter, duplicated scripts, regex-based OCR, missing on-chain events, ID collisions, catch-all error handling, limited UX feedback.

## 2. Off‑App Business Tasks

1. Register \*\*blockreceipt.ai\*\* and key TLDs (.com, .io).  
2. Secure social handles (Twitter, Discord, Instagram).  
3. Form legal entity (LLC), open business bank account.  
4. Draft Terms of Service & Privacy Policy.  
5. Create Stripe & Pinata/NFT.Storage accounts.  
6. Obtain Google Cloud Vision credentials.  
7. Fund dev hot-wallet with MATIC and secure keys in a vault.  
8. Build a simple landing page and one‑pager deck for early adopters/investors.

## 3. In‑App Prioritized Sprint Plan

\*\*Sprint 1: Clean & Consolidate (1–2 days)\*\*

- Restructure repo into monorepo: `/contracts`, `/server`, `/frontend`, `/scripts`, `/data`, `/test`.  
- Remove legacy/duplicate scripts, mock servers, and orphaned folders.  
- Ensure single `hardhat.config.js` and one deploy script.

\*\*Sprint 2: Contract Hardening & Deployment (2 days)\*\*

- Add events: `ReceiptMinted(address,uint256,string)` and `EncryptedData(uint256,bytes,bytes)`.  
- Switch to on-chain counter for token IDs.  
- Deploy to Polygon testnet, verify contract, update `.env`.

\*\*Sprint 3: OCR & Metadata Reliability (2 days)\*\*

- Integrate Google Cloud Vision or `receipt-parser` in `ocrService` with unit tests for sample receipts.  
- Enhance `ipfsService` with retry logic and CID validation; pin existing metadata and update URIs.

\*\*Sprint 4: Backend Refactor & Error Handling (1–2 days)\*\*

- Refactor `uploadAndMint` into discrete steps with individual try/catch blocks.  
- Return user‑friendly error messages per stage.  
- Log granular errors via `winston` or `pino`.

\*\*Sprint 5: Frontend UX Polish & Feedback (1–2 days)\*\*

- Disable upload until wallet & public key connection.  
- Add toasts/spinners for each stage: Uploading, Parsing, Encrypting, Pinning, Minting, Success.  
- Implement `ErrorBoundary` and responsive layouts.

\*\*Sprint 6: QA & Launch Prep (1 day)\*\*

- Perform end-to-end tests on Polygon testnet with diverse receipts.  
- Conduct UAT with 2–3 users; collect feedback.  
- Deploy backend to production (Replit Deployments or Vercel Functions).  
- Point `blockreceipt.ai` to live application.

## 4. Technical Recommendations

- Replace regex OCR with managed service (Google Cloud Vision).  
- Emit smart contract events and store receiptHash on-chain.  
- Move ID generation on-chain to avoid collisions.  
- Harden IPFS pinning with retries.  
- Adopt ESLint, Prettier, TypeScript strict mode.  
- Integrate Sentry for frontend/backend error monitoring.  
- Use GitHub + Replit Git integration for code synchronization.

## 5. Mindset & Next Steps

Stay focused on core user value: secure, private NFT receipts. Measure traction before scaling. Build in public to attract early adopters. Use this sprint plan to systematically polish and prepare for launch.