# 🚀 BlockReceipt.ai – Final Integration Plan to Go Live

## 📦 Overview

This document outlines the final development work required to bring BlockReceipt to a production-ready state. The backend logic, encryption, wallet generation, and NFT procurement logic are complete. The remaining work is focused on improving UX flow, task tracking, wallet onboarding, and frontend UI polish to ensure user clarity and engagement.

## 👛 1. Wallet Linkage to User Sessions

- After signup/login (email or Web3), auto-load the user’s wallet address.  
- Use it to:  
 - Pre-fill wallet address in receipt uploads  
 - Show NFTs tied to their address  
 - Retrieve encrypted metadata for that wallet

Backend Code Sample:  
--------------------  
const wallet = await db.userWallets.findOne({ userId });  
req.session.walletAddress = wallet.address;

## 🖼 2. User NFT Gallery + Receipt Viewer

Create a new React component to display NFTs tied to the user’s wallet:

<Route path="/gallery" element={<NFTGallery wallet={userWalletAddress} />} />  
  
Component Flow:  
- Fetch: /api/gallery/:wallet  
- Show cards with metadata (tier, source, tokenId)  
- Show lock status if encrypted  
- Link to receipt image + metadata page

## 🔁 3. Task Polling After Upload

After user uploads a receipt, poll `/api/task/:taskId/status` to detect NFT mint completion.  
  
Frontend Code Example:  
-----------------------  
useEffect(() => {  
 if (!taskId) return;  
 const interval = setInterval(async () => {  
 const res = await fetch(`/api/task/${taskId}/status`);  
 const json = await res.json();  
 if (json.data?.status === 'completed') {  
 setNftReady(true);  
 clearInterval(interval);  
 }  
 }, 5000);  
 return () => clearInterval(interval);  
}, [taskId]);

## 🔐 4. Wallet Creation at Signup (UI)

Update signup form to ask:  
✅ “Would you like us to generate a free Polygon wallet for you?”  
  
- Generate wallet via ethers.js  
- Encrypt private key with TACo  
- Show backup modal:  
 - wallet address  
 - ⚠️ “Please store this key securely”

## 🌐 5. Polygon Minting (Switch from Mainnet)

Update blockchainService to use Polygon RPC provider:

const provider = new ethers.providers.JsonRpcProvider(process.env.POLYGON\_RPC\_URL);  
const wallet = new ethers.Wallet(process.env.PRIVATE\_KEY, provider);

Ensure minting contracts and metadata support Polygon-specific chains.

## 🎨 6. UX Clarity Enhancements

- ✅ Show user’s wallet address in profile  
- ✅ Show upload success message: “NFT is being minted, this may take 30s…”  
- ✅ Label NFTs clearly as “Polygon-based”  
- ✅ Add tooltip: “Unlock metadata by verifying wallet ownership”  
- ✅ Receipt view: thumbnail preview + download button

## 📂 Files To Update or Create

- `client/pages/gallery.tsx`  
- `client/components/ReceiptUpload.jsx`  
- `client/components/WalletCreateModal.jsx`  
- `server/services/blockchainService.ts`  
- `server/routes/wallet.ts`  
- `.env`: switch to POLYGON RPC + contract addresses

## ✅ Summary Instruction

This document completes the MVP integration path. Implement these frontend and session-layer improvements, connect gallery views, and finalize the real-time NFT feedback loop. Once done, the product will be demo-ready and user-onboardable at scale.