# 📋 BlockReceipt.ai – Replit Feedback (Round 3)

## ✅ Progress Overview

The latest implementation represents a major leap forward. OCR parsing, wallet validation, tier logic, and full TACo encryption of parsed metadata are all functional. The NFT purchase bot is also correctly triggered via `createNFTPurchaseTask()` when a receipt qualifies.

## 🎉 What’s Working Well

- [x] `uploadReceipt.ts` now includes full validation, OCR, and metadata enrichment  
- [x] `encryptLineItems()` is called with wallet address and passed to the NFT bot  
- [x] NFT task is initiated for eligible receipts with correct payload  
- [x] Well-structured response object includes `tier`, `encryptedMetadata`, and `nftGift` status

## ❌ Still Missing or Incomplete

- [ ] NFT Gallery component is still not implemented → users cannot view what they’ve received  
- [ ] No `/gallery/:wallet` API route to fetch NFTs with lock status or metadata references  
- [ ] Encrypted metadata (capsule, ciphertext, policyPublicKey) is not stored in DB or persistent storage  
- [ ] No decryption utility (backend route or frontend component) for unlocking metadata via wallet  
- [ ] Frontend does not consume `nftGift` or `encryptedMetadata` → users get no reward feedback  
- [ ] Task queue logic does not yet return confirmed tokenId for completed NFTs

## 🔁 Deliverables for Final MVP Closure

1. 🖼 \*\*NFT Gallery\*\*  
- Add `/gallery/:wallet` backend route  
- Return tokenId, image, encryption lock status  
- Build `NFTGallery.jsx` or equivalent to render the NFT cards and status

2. 🔐 \*\*Decryption Route or Frontend Flow\*\*  
- Add `/decrypt/:tokenId` route (or local decrypt in React if user owns NFT)  
- Use stored `capsule + ciphertext` to decrypt if wallet matches  
- UI should reflect locked/unlocked receipt state

3. 💾 \*\*Persistent Metadata Storage\*\*  
- Store encrypted metadata in a DB or as part of a token-linked metadata record  
- Ensure metadata can be fetched for gallery views and revalidation

4. 🧪 \*\*Token Confirmation\*\*  
- Confirm that `createNFTPurchaseTask()` resolves with actual tokenId  
- Store and return tokenId to the frontend for NFT gallery population

## 📦 Suggested Files to Add or Modify

- `server/routes/gallery.ts`  
- `server/routes/decrypt.ts` (optional)  
- `client/components/NFTGallery.jsx`  
- `client/components/UnlockMetadataButton.jsx`  
- `db/storeEncryptedMetadata.ts` or equivalent

## 🧠 Final Recommendation

The backend is finally doing what it was designed to do. The next step is bringing that to life visually — with persistent metadata, gallery viewing, and unlockable proof of purchase receipts. Once those are in place, the MVP will be truly live-ready.