# 📋 BlockReceipt.ai – Replit Feedback: Iteration for MVP Completion

## 🧾 Summary of Current Progress

The backend architecture continues to mature, and the newly added `gallery.ts` route confirms NFT visibility integration. TACo-encrypted metadata, NFT minting logic, and service abstractions are all coming together well.

## ✅ Recently Delivered

- [x] `uploadReceipt.ts` validates input and triggers encryption & NFT task  
- [x] `encryptLineItems()` is properly executed  
- [x] `createNFTPurchaseTask()` is conditionally called  
- [x] `/gallery/:walletAddress` route returns user-owned NFTs + lock status  
- [x] Service-oriented architecture for gallery, metadata, blockchain, taco

## ❌ Remaining High-Priority Gaps

- [ ] Encrypted metadata (`capsule`, `ciphertext`, `policyPublicKey`) must be stored and retrievable via tokenId  
- [ ] `/gallery/:walletAddress` must include locked vs. unlocked metadata state  
- [ ] No frontend NFT gallery UI component exists yet (e.g. `NFTGallery.jsx`)  
- [ ] No method to decrypt metadata (either `/decrypt/:tokenId` route or client-side unlock button)  
- [ ] NFT task confirmation should ensure the minted tokenId is returned + associated with encrypted metadata

## 🛠 MVP Iteration Deliverables

1. 🖼 \*\*Gallery Frontend Component\*\*  
- Build `NFTGallery.jsx` or similar React component  
- Connect to `/gallery/:wallet` route and show cards per NFT  
- Add locked/unlocked status overlay to metadata sections

2. 🔐 \*\*Encrypted Metadata Storage\*\*  
- Persist `capsule`, `ciphertext`, `policyPublicKey` from `encryptLineItems()` to DB  
- Use tokenId or wallet as key  
- Add `metadataService.getEncryptedMetadata(tokenId)` method

3. 🔓 \*\*Unlock Flow\*\*  
- Option 1: Add `/decrypt/:tokenId` route if backend manages key  
- Option 2: Add frontend decryption logic using Threshold SDK  
- Validate user owns token before allowing decryption (either on-chain or passed-in signature)

4. 🎁 \*\*NFT Task Completion Validation\*\*  
- Confirm `createNFTPurchaseTask()` writes minted tokenId back to response or DB  
- Connect tokenId to metadata for user gallery experience

## 📦 Suggested File Additions or Edits

- `server/routes/decrypt.ts` (if backend handles unlock)  
- `server/services/metadataService.ts` (with persistent metadata store)  
- `client/components/NFTGallery.jsx`  
- `client/components/UnlockMetadataButton.jsx`  
- Update `uploadReceipt.ts` to log tokenId after mint

## 🧠 Guidance for Ongoing Iteration

This app is entering a usable MVP state. Focus now on enabling the user to:  
- Upload receipt  
- See a visible, ownable NFT  
- Unlock its metadata if they own it  
- Experience the full encrypted-UX journey