# 📋 BlockReceipt.ai – Replit Feedback: Iteration Round 5

## ✅ What You’ve Successfully Delivered

- [x] `/gallery/:walletAddress` returns NFT list by wallet  
- [x] `/gallery/metadata/:tokenId` returns encrypted receipt data + preview + owner  
- [x] `/gallery/unlock/:tokenId` accepts wallet address and tokenId for decryption request  
- [x] Zod-based validation for inputs is consistent and secure  
- [x] Codebase is now modular, readable, and aligned to production-grade service architecture

## ❌ Still Missing or Unconfirmed

- [ ] Unlock handler logic (`/unlock/:tokenId`) is incomplete or not visible – unclear if real decryption happens  
- [ ] Metadata persistence (ciphertext, capsule, policyKey) must be stored at receipt upload time if not already  
- [ ] Frontend UI still missing (`NFTGallery.jsx`, unlock button, and wallet-driven view)  
- [ ] No frontend check to validate token ownership prior to calling unlock route  
- [ ] Upload ➝ Mint ➝ Encrypt ➝ Store ➝ View ➝ Unlock lifecycle needs end-to-end testing

## 🛠 Deliverables for This Iteration

1. 🔓 \*\*Unlock Logic Completion\*\*  
- Complete the `/gallery/unlock/:tokenId` route  
- Ensure `tacoService` decrypts with wallet key or generates a re-encryption token  
- Respond with decrypted payload or proof of access

2. 💾 \*\*Persistent Storage Validation\*\*  
- Ensure `encryptLineItems()` result is stored in DB with tokenId link  
- Implement `metadataService.storeEncryptedMetadata(tokenId, metadata)`  
- Confirm data is retrievable in `getEncryptedMetadata()`

3. 🖼 \*\*Frontend Gallery Integration\*\*  
- Add React gallery component (grid or card view)  
- Connect to `/gallery/:wallet` API and list NFTs  
- Show locked/unlocked badge  
- Include unlock button if metadata is locked

4. 🧠 \*\*Token Ownership Check for Unlock\*\*  
- Add logic to check that `req.body.walletAddress` matches NFT owner from chain or DB  
- Or, implement client-side `eth\_signMessage()` verification before calling unlock

## 📦 Files to Update or Create

- `server/routes/gallery.ts`  
- `server/services/metadataService.ts`  
- `server/services/tacoService.ts`  
- `client/components/NFTGallery.jsx`  
- `client/components/UnlockMetadataButton.jsx`

## 📣 Recommendation

Backend logic is nearly complete. Focus now on exposing this flow to the user:  
- Upload ➝ NFT ➝ Lock ➝ View ➝ Unlock. Once users can experience that flow in a single session, the MVP will be demo-ready and unlock viral potential.