# 🛠 BlockReceipt.ai – Follow-Up Corrections After Task Implementation

## 🧾 Review Summary

Thank you for the updates. The backend route `uploadReceipt.ts` shows excellent progress with file upload handling, wallet validation, and OCR extraction. However, the core functionality of issuing an NFT reward and encrypting the metadata with Threshold TACo is not yet complete.

## ✅ What Was Done Well

- [x] Multer upload middleware is correctly configured and stable  
- [x] Wallet address is validated before file processing  
- [x] Receipt is parsed using OCR  
- [x] Error conditions are gracefully handled (empty items, bad uploads)  
- [x] Good use of logs for debugging receipt uploads

## ❌ What Is Still Missing

- [ ] `encryptLineItems()` is imported but never called → no metadata is encrypted  
- [ ] `nftPurchaseBot()` or `createNFTPurchaseTask()` is not executed → no NFT is minted or transferred  
- [ ] No tokenId is linked to the user or returned to frontend  
- [ ] No storage of `{ciphertext, capsule, policyPublicKey}` for future unlocks  
- [ ] No NFT gallery frontend or `/gallery/:wallet` backend route exists

## 🔁 What Needs to Be Added Next

1. 🔐 \*\*Metadata Encryption\*\*  
- After parsing the receipt successfully, call `encryptLineItems(userPublicKey, receiptData)`  
- Store the `ciphertext`, `capsule`, and `policyPublicKey` in your backend DB, linked to wallet + tokenId

2. 🎁 \*\*Trigger NFT Purchase\*\*  
- After validating the receipt, call `nftPurchaseBot(walletAddress, tier)`  
- On success, return the NFT tokenId in the API response  
- Store tokenId with user and encrypted metadata

3. 🖼 \*\*Build or Connect NFT Gallery\*\*  
- Create `/gallery/:wallet` route to return user NFTs + lock status  
- Build frontend to fetch and display NFTs (basic card layout)  
- Add locked/unlocked visual indicator, unlock button if wallet matches

## ✅ Deliverables Required to Complete MVP

- [ ] Call `encryptLineItems()` post-OCR  
- [ ] Call `nftPurchaseBot()` and return the NFT to the user  
- [ ] Store tokenId with encrypted metadata and user wallet  
- [ ] Create `/gallery/:wallet` endpoint  
- [ ] Display NFTs in frontend (locked/unlocked view)

## 🧠 Recommendation

All infrastructure and foundational logic is strong. Now, complete the flow by calling the functions you've already imported. Once the user uploads a receipt, they should immediately receive:  
- A confirmed NFT sent to their wallet  
- The ability to view that NFT in a gallery  
- A locked view of receipt data unless they own the token