# 🔄 BlockReceipt.ai – Audit & Fix: Wallet Integration & NFT Gallery

## 1. Overview of Issues

After reviewing build v22, the following critical issues were identified:  
1. Multiple “Connect Wallet” buttons remain visible; no unified auth state.  
2. Upload succeeds but mint never fires (500 Internal Server Error).  
3. NFT Gallery component is not rendered on the main page.  
4. TACo encryption is a toggle; needs to be enforced and guaranteed.

## 2. Code Fixes & Snippets

### 2.1 Enhanced Wallet Hook with Unified Context

File: `src/hooks/useWalletConnect.ts`  
  
```ts  
import { createContext, useContext, useState, useEffect } from 'react';  
import { ethers } from 'ethers';  
  
const WalletContext = createContext({  
 provider: null,  
 signer: null,  
 address: null as string|null,  
 connect: async () => {}  
});  
  
export function WalletProvider({ children }) {  
 const [provider, setProvider] = useState<ethers.providers.Provider|null>(null);  
 const [signer, setSigner] = useState<ethers.Signer|null>(null);  
 const [address, setAddress] = useState<string|null>(null);  
  
 const connect = async () => {  
 if (window.ethereum) {  
 const web3 = new ethers.providers.Web3Provider(window.ethereum);  
 await web3.send('eth\_requestAccounts', []);  
 const s = web3.getSigner();  
 setProvider(web3);  
 setSigner(s);  
 setAddress(await s.getAddress());  
 } else {  
 alert('Please install MetaMask');  
 }  
 };  
  
 useEffect(() => {  
 if (window.ethereum) {  
 window.ethereum.request({ method: 'eth\_accounts' })  
 .then(accounts => accounts[0] && connect());  
 }  
 }, []);  
  
 return (  
 <WalletContext.Provider value={{ provider, signer, address, connect }}>  
 {children}  
 </WalletContext.Provider>  
 );  
}  
  
export function useWallet() {  
 return useContext(WalletContext);  
}  
```

### 2.2 Block Upload & Mint Until Wallet Present

File: `server/routes/uploadReceipt.ts`  
  
```ts  
if (!req.body.walletAddress) {  
 return res.status(400).json({  
 success: false,  
 message: 'Connect your wallet before uploading a receipt.'  
 });  
}  
// ... continue with OCR, TACo encryption, then mint flow  
```

### 2.3 Enforce TACo Encryption Before Mint

Update UI to show a padlock instead of toggle:  
  
```tsx  
// ReceiptUpload.jsx snippet  
<LockClosedIcon className="h-5 w-5 text-green-600" />  
<span>All metadata encrypted by Threshold TACo</span>  
<input type="file" onChange={handleUpload} />  
```  
  
Backend always calls:  
  
```ts  
const encrypted = await tacoService.encryptLineItems(  
 req.body.walletAddress, receiptData.items  
);  
```

### 2.4 Real ERC-1155 Mint Call

File: `server/routes/nfts.ts`  
  
```ts  
import abi from '../abi/BlockReceiptCollection.json';  
import { ethers } from 'ethers';  
  
router.post('/mint', async (req, res) => {  
 const { walletAddress, nftId } = req.body;  
 if (!walletAddress) {  
 return res.status(400).json({ success: false, msg: 'Connect wallet first' });  
 }  
 const provider = new ethers.providers.JsonRpcProvider(process.env.POLYGON\_RPC\_URL);  
 const devWallet = new ethers.Wallet(process.env.PRIVATE\_KEY, provider);  
 const contract = new ethers.Contract(process.env.CONTRACT\_ADDRESS, abi, devWallet);  
 try {  
 const tx = await contract.mint(walletAddress, nftId);  
 const receipt = await tx.wait();  
 return res.json({ success: true, tokenId: nftId, txHash: receipt.transactionHash });  
 } catch (err) {  
 console.error('Mint error:', err);  
 return res.status(500).json({ success: false, message: 'Mint failed', error: err.message });  
 }  
});  
```

### 2.5 Restore NFT Gallery on Main Page

File: `src/components/ReceiptGallery.tsx`  
  
```tsx  
import React, { useEffect, useState } from 'react';  
import { useWallet } from '../hooks/useWallet';  
  
export default function ReceiptGallery() {  
 const { address } = useWallet();  
 const [nfts, setNfts] = useState<any[]>([]);  
  
 useEffect(() => {  
 if (!address) return;  
 fetch(`/api/gallery/${address}`)  
 .then(res => res.json())  
 .then(data => setNfts(data.nfts || []));  
 }, [address]);  
  
 if (!address) return null;  
  
 return (  
 <section>  
 <h2>Your MintVault</h2>  
 <div className="grid grid-cols-2 md:grid-cols-3 gap-4">  
 {nfts.map(n => (  
 <div key={n.tokenId} className="border p-2 rounded">  
 <img src={n.image} alt={`NFT ${n.tokenId}`} className="object-cover w-full h-32"/>  
 <p>Token ID: {n.tokenId}</p>  
 <p>Tier: {n.tier}</p>  
 </div>  
 ))}  
 </div>  
 </section>  
 );  
}  
```  
  
Place in your main layout under upload & selection.

## 3. Next Iteration for Replit

1. Merge hook & route changes; restart server.  
2. Deploy ERC-1155 on Polygon testnet, update `.env`.  
3. Test upload → mint → gallery end-to-end.  
4. Add success/failure toasts.  
5. QA both MetaMask and dev-key fallback paths.

## 4. Final Score & Progress

| Category | % Complete |  
|----------------------------------|------------|  
| Wallet/Auth UX | 85% |  
| Receipt → OCR → TACo Encryption | 95% |  
| NFT Pool & Selection | 90% |  
| On-Chain Minting | 75% |  
| Frontend Gallery & Feedback | 80% |  
| IPFS Metadata Pinning | 0% |  
| \*\*Overall MVP Completion\*\* | \*\*75%\*\* |