# 🔗 BlockReceipt.ai – Dual Metadata & Promotion Integration

## 1. Overview

We’re extending NFT Gallery receipts with two distinct metadata layers:  
A. \*\*User-Controlled Metadata\*\* – Private purchase details extracted via OCR and encrypted via Threshold PRE, only decryptable by the user.  
B. \*\*Vendor-Controlled Promotion Metadata\*\* – Time-limited promo codes or offers encrypted via Threshold PRE with a vendor-defined TTL, decryptable by the user until expiration.  
  
Both layers live in the same NFT metadata JSON on IPFS, but are separately encrypted and managed.

## 2. Metadata JSON Schema

```json  
{  
 "name": "Receipt NFT #<tokenId>",  
 "description": "Your digital NFT receipt",  
 "userData": {  
 "capsule": "<base64>",  
 "ciphertext": "<base64>",  
 "policyId": "<tpre-policy-user>"  
 },  
 "promoData": {  
 "capsule": "<base64>",  
 "ciphertext": "<base64>",  
 "policyId": "<tpre-policy-promo>",  
 "expiresAt": 1717180800  
 },  
 "receiptImage": "ipfs://<CID>/receipt.png"  
}  
```

## 3. Backend Integration

### 3.1 Extend Metadata Service

In `server/services/metadataService.ts`, store both userData and promoData:

```ts  
export interface EncryptedField {  
 capsule: string;  
 ciphertext: string;  
 policyId: string;  
 expiresAt?: number; // only for promoData  
}  
  
export async function storeNFTMetadata(  
 tokenId: string,  
 walletAddress: string,  
 userData: EncryptedField,  
 promoData: EncryptedField  
) {  
 await db.nftMetadata.insertOne({  
 tokenId,  
 walletAddress,  
 userData,  
 promoData,  
 createdAt: new Date()  
 });  
}  
```

### 3.2 Vendor Promotion Route

Add `/api/promotions/create` for vendors to define offers:

```ts  
import express from 'express';  
import { thresholdClient } from '../services/tpreService';  
import { pinJSONToIPFS } from '../services/ipfsService';  
const router = express.Router();  
  
router.post('/promotions/create', async (req, res) => {  
 const { userPubKey, promoPayload, validUntil } = req.body;  
 const { capsule, ciphertext, policyId } = await thresholdClient.encrypt({  
 recipientPublicKey: userPubKey,  
 data: Buffer.from(promoPayload),  
 expiresAt: validUntil  
 });  
 const promoJson = { capsule, ciphertext, policyId, expiresAt: validUntil };  
 const { url: promoUri } = await pinJSONToIPFS(promoJson);  
 res.json({ success: true, promoUri });  
});  
export default router;  
```

### 3.3 Upload-and-Mint Enhancement

Modify `uploadAndMint.ts` to accept `promoUri` and include in metadata:

```ts  
// After OCR userData encryption  
const userField = await encryptUserData(req.file.buffer);  
// Receive promoUri from vendor or UI  
const promoUri = req.body.promoUri;  
const promoJson = await fetchJson(promoUri);  
  
// Combine metadata and pin full JSON  
const metadataUri = await pinJSONToIPFS({  
 name: 'Receipt NFT',  
 userData: userField,  
 promoData: promoJson,  
 receiptImage: receiptIpfsUrl  
});  
  
await contract.mintNewReceipt(  
 req.body.walletAddress,  
 metadataUri  
);  
await storeNFTMetadata(tokenId, req.body.walletAddress, userField, promoJson);  
```

## 4. Frontend Integration

### 4.1 NFT Gallery Display

In `ReceiptGallery.tsx`, show reveal UI for both metadata layers:

jsx  
// For each NFT in gallery  
<div className="nft-card">  
 <h3>{nft.name}</h3>  
 <button onClick={() => revealField(nft.metadata.userData)}>  
 Reveal Receipt Details  
 </button>  
 {Date.now() < nft.metadata.promoData.expiresAt ? (  
 <button onClick={() => revealField(nft.metadata.promoData)}>  
 Reveal Promotion  
 </button>  
 ) : (  
 <span className="text-gray-500">Promotion Expired</span>  
 )}  
</div>  
```

### 4.2 Decryption Helper

Use a shared helper to decrypt any EncryptedField:

ts  
async function revealField(field: EncryptedField): Promise<string> {  
 const decrypted = await thresholdClient.decrypt({  
 capsule: field.capsule,  
 ciphertext: field.ciphertext,  
 policyId: field.policyId  
 });  
 return new TextDecoder().decode(decrypted);  
}  
```

## 5. Replit Integration Instructions

1. Update `server/services/metadataService.ts` to use `storeNFTMetadata` with dual fields.  
2. Add the vendor promotions route in `server/routes/promotions.ts` and register it in your Express app.  
3. Modify `uploadAndMint.ts` to accept `promoUri` and include `promoData` in the pinned metadata.  
4. Pin combined metadata JSON to IPFS and ensure `Receipt1155` contract supports metadata URI.  
5. Enhance `ReceiptGallery.tsx` to display reveal buttons for both userData and promoData, with expiration logic.