# 🔔 BlockReceipt.ai – Time-Limited Promo Codes with TACo PRE

## Why This Matters

Just like the classic $3-off coupon printed on a paper receipt, our digital NFT receipts can carry exclusive, time-bound offers. By leveraging Threshold Network's Proxy Re-Encryption (TACo PRE), we ensure that only the rightful owner of the NFT can decrypt and redeem the offer—and only within a limited validity period. This feature:  
- Drives repeat engagement and loyalty  
- Preserves user privacy and data security  
- Differentiates BlockReceipt.ai with real-world utility

## Feature Overview

At mint time:  
1. A unique promo code is generated and encrypted under the user's public key with a 2-week TTL.  
2. Encrypted coupon data is embedded in the NFT's metadata, pinned to IPFS.  
3. The NFT is minted to the user's wallet on Polygon Amoy.  
  
During the validity window:  
- Users click “Reveal Coupon” on the receipt's dedicated Offers page.  
- TACo PRE nodes decrypt the coupon only if the policy is still valid.  
  
After expiration:  
- Decryption fails, preventing unauthorized or late redemptions.

## Implementation Details

### 1. Coupon Service

File: `server/services/couponService.ts`

```ts  
import { thresholdClient } from './tpreService';  
import { pinJSONToIPFS } from './ipfsService';  
  
export async function createTimedCoupon(  
 userPubKey: string,  
 couponCode: string,  
 validUntil: number  
) {  
 const { capsule, ciphertext, policyId } = await thresholdClient.encrypt({  
 recipientPublicKey: userPubKey,  
 data: Buffer.from(couponCode),  
 expiresAt: validUntil  
 });  
 const metadataUri = await pinJSONToIPFS({  
 name: 'Time-Limited Coupon',  
 description: `Valid until ${new Date(validUntil).toLocaleDateString()}`,  
 coupon: { capsule, ciphertext, policyId },  
 expiresAt: validUntil  
 });  
 return { metadataUri };  
}  
```

### 2. Mint Route Integration

File: `server/routes/uploadAndMint.ts` (excerpt)

```ts  
// Generate and embed a 2-week coupon  
const twoWeeksMs = 14 \* 24 \* 60 \* 60 \* 1000;  
const validUntil = Date.now() + twoWeeksMs;  
const couponCode = `OFF${Math.random().toString(36).slice(-6).toUpperCase()}`;  
const { metadataUri: couponUri } = await createTimedCoupon(  
 req.body.recipientPublicKey,  
 couponCode,  
 validUntil  
);  
// Mint NFT with coupon metadata URI  
const tx = await contract.mintNewReceipt(  
 req.body.walletAddress,  
 couponUri  
);  
await tx.wait();  
```

### 3. Frontend Redeem Component

File: `frontend/components/CouponCard.jsx`

```jsx  
import React, { useState } from 'react';  
import { thresholdClient } from '../services/tpreClient';  
  
export default function CouponCard({ nft }) {  
 const [coupon, setCoupon] = useState(null);  
 const now = Date.now();  
  
 const revealCoupon = async () => {  
 try {  
 const { capsule, ciphertext, policyId } = nft.metadata.coupon;  
 const decrypted = await thresholdClient.decrypt({ capsule, ciphertext, policyId });  
 setCoupon(new TextDecoder().decode(decrypted));  
 } catch {  
 setCoupon('❌ Coupon expired or unavailable');  
 }  
 };  
  
 return (  
 <div className="p-4 border rounded">  
 <h3 className="font-bold">{nft.name}</h3>  
 {now < nft.metadata.expiresAt ? (  
 coupon ? (  
 <p className="text-green-600">Your code: {coupon}</p>  
 ) : (  
 <button onClick={revealCoupon} className="btn-primary">  
 Reveal Coupon  
 </button>  
 )  
 ) : (  
 <p className="text-gray-500">Coupon expired</p>  
 )}  
 </div>  
);  
}  
```

## Dedicated Offers Page

Create a new page and header link titled \*\*“Offers & Coupons”\*\* in your main navigation. Suggested file: `frontend/pages/OffersPage.jsx`

Example skeleton:

```jsx  
import React from 'react';  
import CouponCard from '../components/CouponCard';  
  
export default function OffersPage({ nfts }) {  
 return (  
 <div className="container mx-auto p-4">  
 <h1 className="text-2xl font-bold mb-4">Exclusive Offers</h1>  
 <div className="grid grid-cols-1 md:grid-cols-2 gap-6">  
 {nfts.map(nft => (  
 <CouponCard key={nft.tokenId} nft={nft} />  
 ))}  
 </div>  
 </div>  
);  
}  
```

## Replit Integration Instructions

1. Add `couponService.ts` under `server/services`.  
2. Update `/upload-and-mint` route to call `createTimedCoupon` before minting.  
3. Deploy updated ERC-1155 contract with `mintNewReceipt(address, string)` signature.  
4. Create `CouponCard.jsx` and `OffersPage.jsx` in `frontend/components` and `frontend/pages`.  
5. Add nav link “Offers & Coupons” in your main layout (e.g., `Header.jsx`).  
6. Test full flow on Polygon Amoy: upload → mint with coupon → navigate to Offers → reveal code within 2 weeks.