# 🌍 BlockReceipt.ai – Scaling the Passport Stamp Concept

## 1. The Challenge

A city-based passport stamp theme is engaging, but there are far fewer cities than potential receipts. To maintain uniqueness and scalability, we need to combine a fixed 'city emblem' with dynamic, generative overlays.

## 2. Scalable Stamp Architecture

Each NFT stamp will consist of:  
A. \*\*Base City Seal\*\* – a static graphic representing the city or region (e.g., Eiffel Tower for Paris).  
B. \*\*Dynamic Overlay\*\* – procedurally generated patterns, color schemes, and icons derived from the receipt data hash and purchase metadata.  
C. \*\*Promo Badge\*\* – vendor promo indicator (e.g., a colored border or icon) that changes or fades when expired.

## 3. Technical Implementation

### 3.1 Generative Stamp Service

File: `server/services/stampService.ts`

```ts  
import { createCanvas } from 'canvas';  
import fetch from 'node-fetch';  
import { pinFileToIPFS } from './ipfsService';  
  
interface StampOptions {  
 cityCode: string;  
 receiptHash: string;  
 promoActive: boolean;  
}  
  
export async function generatePassportStamp(options: StampOptions): Promise<string> {  
 const { cityCode, receiptHash, promoActive } = options;  
 const canvas = createCanvas(512, 512);  
 const ctx = canvas.getContext('2d');  
  
 // 1. Draw base city seal  
 const baseImage = await loadImage(`assets/cities/${cityCode}.png`);  
 ctx.drawImage(baseImage, 0, 0, 512, 512);  
  
 // 2. Compute overlay seed from hash  
 const seed = parseInt(receiptHash.slice(0, 8), 16);  
 const rand = mulberry32(seed);  
  
 // 3. Draw dynamic circles  
 for (let i = 0; i < 5; i++) {  
 ctx.strokeStyle = `hsl(${rand()\*360}, 70%, 50%)`;  
 ctx.lineWidth = 5 + rand()\*10;  
 ctx.beginPath();  
 ctx.arc(256, 256, 50 + i\*30, 0, Math.PI \* 2 \* rand());  
 ctx.stroke();  
 }  
  
 // 4. Draw promo badge if active  
 if (promoActive) {  
 ctx.fillStyle = 'rgba(255, 215, 0, 0.6)';  
 ctx.font = 'bold 48px Sans';  
 ctx.fillText('PROMO', 350, 100);  
 }  
  
 // 5. Convert to buffer and pin  
 const buffer = canvas.toBuffer('image/png');  
 const { cid } = await pinFileToIPFS(buffer, `stamp-${options.receiptHash}.png`);  
 return `ipfs://${cid}`;  
}  
  
// deterministic PRNG  
function mulberry32(a: number): () => number {  
 return () => {  
 let t = a += 0x6D2B79F5;  
 t = Math.imul(t ^ t >>> 15, t | 1);  
 t ^= t + Math.imul(t ^ t >>> 7, t | 61);  
 return ((t ^ t >>> 14) >>> 0) / 4294967296;  
 };  
}  
```

### 3.2 Mint Route Update

In `server/routes/uploadAndMint.ts`, generate and include stampUri:

```ts  
import { generatePassportStamp } from '../services/stampService';  
// ... inside upload-and-mint handler  
const cityCode = receiptData.cityCode; // extracted via geocoding  
const stampUri = await generatePassportStamp({  
 cityCode,  
 receiptHash: txHash, // or hashed metadata  
 promoActive: true  
});  
const metadata = {  
 name: `Receipt #${tokenId}`,  
 description: `Minted in ${cityCode}`,  
 stamp: stampUri,  
 userData,  
 promoData  
};  
const metadataUri = await pinJSONToIPFS(metadata);  
await contract.mintNewReceipt(walletAddress, metadataUri);  
```

### 3.3 Frontend Gallery Integration

Update `ReceiptGallery.tsx` to display the stamp image:

jsx  
<div className="nft-card">  
 <img src={nft.metadata.stamp} alt="Passport Stamp" className="w-full" />  
 {/\* Existing reveal buttons \*/}  
</div>  
```

## 4. Assets & Scaling

- \*\*City Assets:\*\* Curate a set of ~200–300 major city seal PNGs (136×136). Stored under `assets/cities/`.  
- \*\*Geocoding:\*\* Use store address from OCR to map receipts to the nearest city via a simple lookup.  
- \*\*Infinite Variations:\*\* Dynamic overlays ensure each stamp is unique even if city repeats.  
- \*\*Edge Cases:\*\* Fallback to 'Global' seal for unrecognized locations.

## 5. Replit Integration Instructions

1. Install dependencies: `npm install canvas node-fetch`.  
2. Add city PNG assets to `server/assets/cities/`.  
3. Create `stampService.ts` as above.  
4. Update `uploadAndMint.ts` to call `generatePassportStamp` and include `stampUri` in metadata.  
5. Pin city assets on IPFS or include in static hosting.  
6. Modify `ReceiptGallery` to render `nft.metadata.stamp` image.