# 🚀 BlockReceipt.ai – Final Un‑Block Sprint for Live MVP

## 🎯 Goals for This Sprint

1. Deploy an ERC‑1155 collection on Polygon (Amoy → Mainnet).  
2. Wire backend mint call after user selects an NFT.  
3. Enforce wallet connection before any receipt upload.  
4. Pin catalog images + metadata to IPFS and update nft\_pool.  
5. Auto‑refresh gallery once the NFT task completes.

## 🛠 1. Smart‑Contract Deployment

`contracts/BlockReceiptCollection.sol`

// SPDX-License-Identifier: MIT  
pragma solidity ^0.8.24;  
  
import "@openzeppelin/contracts/token/ERC1155/ERC1155.sol";  
import "@openzeppelin/contracts/access/Ownable.sol";  
  
contract BlockReceiptCollection is ERC1155, Ownable {  
 constructor(string memory baseURI) ERC1155(baseURI) {}  
 function mint(address to, uint256 id) external onlyOwner {  
 \_mint(to, id, 1, "");  
 }  
}

Deploy with Hardhat:  
`npx hardhat run scripts/deploy.js --network amoy`  
Save CONTRACT\_ADDRESS in `.env`.

## 🔗 2. Backend Mint Endpoint Update

`server/routes/nfts.ts` – replace mock transfer with real mint:

import ERC1155\_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: 'wallet required' });  
  
 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, ERC1155\_ABI, devWallet);  
  
 const tx = await contract.mint(walletAddress, nftId);  
 const receipt = await tx.wait();  
 return res.json({ success: true, tokenId: nftId, txHash: receipt.transactionHash });  
});

## 🔒 3. Enforce Wallet on Upload

In `server/routes/uploadReceipt.ts`, add at top:  
if (!req.body.walletAddress) {  
 return res.status(400).json({ success: false, msg: 'Connect wallet first' });  
}

## 📦 4. Pin Catalog Assets to IPFS

- Upload PNGs and JSON for each NFT in `/client/assets/nft/` to Pinata or NFT.Storage.  
- Update `data/nft\_pool.json` entries:  
 { "id": "nft\_001", "image": "https://gateway.pinata.cloud/ipfs/<CID>/1.png", "metadataUri": "ipfs://<CID>/1.json", ... }

## 🔁 5. Gallery Auto‑Refresh

After POST /mint returns success, trigger gallery reload:  
setTimeout(() => fetchGallery(), 10000);  
// Or poll `/api/task/${taskId}/status` until 'completed'

## 📂 Files to Create/Update

- contracts/BlockReceiptCollection.sol  
- scripts/deploy.js  
- server/routes/nfts.ts  
- server/routes/uploadReceipt.ts  
- data/nft\_pool.json  
- client/hooks/useGalleryPoll.js  
- client/pages/gallery.jsx

## ✅ Deliverables

- Deployed contract address in .env  
- Backend mint endpoint tested on Polygon Amoy  
- Upload blocked w/o wallet  
- Gallery shows minted NFT within 15s