# 🔍 BlockReceipt.ai – OCR Enhancement with Google Cloud Vision

## 📣 Objective

Improve the reliability and accuracy of receipt data extraction by integrating Google Cloud Vision's Document Text Detection API. This replaces the current OCR bottleneck with a robust, managed service that can handle diverse receipt formats.

## 🛠 Dependencies & Setup

- \*\*Install the Node.js client library\*\*:  
 ```bash  
 npm install @google-cloud/vision  
 ```  
- \*\*Service Account Credentials\*\*:  
 1. In Google Cloud Console, create a service account with Vision API access.  
 2. Download the JSON key file and add it to Replit's Secrets as:  
 - Key name: `GOOGLE\_APPLICATION\_CREDENTIALS`  
 - Value: (entire JSON contents)  
 3. Ensure the environment variable is set:  
 ```bash  
 export GOOGLE\_APPLICATION\_CREDENTIALS=/path/to/key.json  
 ```

## 📄 New OCR Service: `ocrService.ts`

Create a dedicated service to call Google Vision and parse structured receipt data:

```ts  
// server/services/ocrService.ts  
import vision from '@google-cloud/vision';  
  
const client = new vision.ImageAnnotatorClient();  
  
export interface ReceiptItem {  
 description: string;  
 quantity: number;  
 price: number;  
}  
  
export interface ReceiptData {  
 merchant: string;  
 date: string;  
 total: number;  
 items: ReceiptItem[];  
}  
  
export async function extractReceiptData(imageBuffer: Buffer): Promise<ReceiptData> {  
 // Call Vision API for document text detection  
 const [result] = await client.documentTextDetection({ image: { content: imageBuffer } });  
 const fullText = result.fullTextAnnotation?.text || '';  
   
 // TODO: Implement regex-based parsing or line-by-line logic  
 // Example: split by lines, find date, total, and items patterns  
 const lines = fullText.split('\n');  
 const items: ReceiptItem[] = [];  
 let merchant = lines[0] || '';  
 let date = '';  
 let total = 0;  
  
 lines.forEach(line => {  
 // Simple date match (MM/DD/YYYY)  
 const dateMatch = line.match(/\d{1,2}\/\d{1,2}\/\d{2,4}/);  
 if (dateMatch) date = dateMatch[0];  
  
 // Simple total match  
 const totalMatch = line.match(/Total\s\*\$?([0-9]+\.?[0-9]\*)/i);  
 if (totalMatch) total = parseFloat(totalMatch[1]);  
  
 // Item line match: e.g., "ItemName 2 $5.00"  
 const itemMatch = line.match(/([a-zA-Z ]+)\s+(\d+)\s+\$?([0-9]+\.?[0-9]\*)/);  
 if (itemMatch) {  
 items.push({  
 description: itemMatch[1].trim(),  
 quantity: parseInt(itemMatch[2], 10),  
 price: parseFloat(itemMatch[3])  
 });  
 }  
 });  
  
 return { merchant, date, total, items };  
}  
```

## 🔗 Integrate with Upload Route

In `server/routes/uploadReceipt.ts`, replace the current OCR call with the new service:

```ts  
import { extractReceiptData } from '../services/ocrService';  
  
// In your upload handler:  
const imageBuffer = fs.readFileSync(savedFilePath);  
const receiptData = await extractReceiptData(imageBuffer);  
if (!receiptData.items.length) {  
 throw new Error('No items detected on receipt');  
}  
// Proceed with categorization, encryption, and NFT flow  
```

## 🧪 Testing & Validation

- \*\*Unit Test\*\*: Write tests for `extractReceiptData` with sample images saved in `/test/receipts/`.  
- \*\*Integration Test\*\*: Upload known receipts via Postman or frontend; verify JSON response contains correct `merchant`, `date`, `total`, and `items`.  
- \*\*Error Handling\*\*: Ensure meaningful errors when Vision API fails or parsing finds no lines.

## 📂 Files to Create/Update

- `server/services/ocrService.ts`  
- `server/routes/uploadReceipt.ts` (replace OCR logic)  
- Add sample receipt images in `/test/receipts/` for dev use  
- Update `package.json` dependencies

## ✅ Summary Instruction to Replit

1. Install `@google-cloud/vision` in the backend.  
2. Configure service account credentials via Replit Secrets.  
3. Implement `ocrService.ts` and integrate in upload route.  
4. Write tests and validate parsing accuracy across various receipt formats.