# 🧾 BlockReceipt.ai – Professional File Upload Rewrite & Audit Fix

## 🔍 Audit Summary

This document provides a professional rewrite of the `uploadReceipt.ts` route along with detailed recommendations based on an audit of your current code. The original upload logic had structural and middleware issues that could result in failed uploads, lack of error feedback, and brittle error handling.

## ⚠️ Issues Found:

- Inline use of multer middleware caused inconsistent `req.file` parsing  
- Missing `await` for asynchronous receipt extraction logic  
- No front-end friendly error messages  
- Excessive debug logs left in production path  
- No verification of file presence or content safety beyond basic filters

## ✅ Professional Rewrite – `uploadReceipt.ts`

Use this version of your route for stable, secure file upload with receipt data extraction.

import express from 'express';  
import multer from 'multer';  
import path from 'path';  
import fs from 'fs';  
import { extractReceiptData, determineReceiptTier } from '../../shared/utils/receiptLogic';  
  
const router = express.Router();  
  
const uploadDir = path.join(\_\_dirname, '../../uploads');  
if (!fs.existsSync(uploadDir)) {  
 fs.mkdirSync(uploadDir, { recursive: true });  
}  
  
const storage = multer.diskStorage({  
 destination: (\_req, \_file, cb) => cb(null, uploadDir),  
 filename: (\_req, file, cb) => {  
 const uniqueSuffix = Date.now() + '-' + Math.round(Math.random() \* 1e9);  
 const ext = path.extname(file.originalname);  
 cb(null, 'receipt-' + uniqueSuffix + ext);  
 }  
});  
  
const upload = multer({  
 storage,  
 limits: { fileSize: 10 \* 1024 \* 1024 },  
 fileFilter: (\_req, file, cb) => {  
 const allowedTypes = /jpeg|jpg|png|gif|pdf/;  
 const isValid = allowedTypes.test(file.mimetype) &&  
 allowedTypes.test(path.extname(file.originalname).toLowerCase());  
 cb(null, isValid);  
 }  
});  
  
router.post('/upload-receipt', upload.single('receipt'), async (req, res) => {  
 try {  
 if (!req.file) {  
 return res.status(400).json({ success: false, message: 'No file uploaded' });  
 }  
  
 const filePath = req.file.path;  
 const data = await extractReceiptData(filePath);  
 if (!data || !data.total) {  
 return res.status(400).json({ success: false, message: 'Could not extract data from receipt' });  
 }  
  
 const tier = determineReceiptTier(data.total);  
  
 return res.status(200).json({  
 success: true,  
 parsed: data,  
 tier  
 });  
  
 } catch (error) {  
 console.error('[Upload Error]', error.message);  
 return res.status(500).json({  
 success: false,  
 message: 'Server error while processing receipt',  
 details: error.message  
 });  
 }  
});  
  
export default router;

## 📦 File Organization

- Save this file as `server/routes/uploadReceipt.ts`  
- Ensure `/uploads` directory is writable by Replit  
- Add route to your Express app in `server/index.ts`:  
  
 `import uploadReceipt from './routes/uploadReceipt';`  
 `app.use('/api', uploadReceipt);`

## 🎯 Recommendations Moving Forward

- Implement `sharp` or `magic-bytes.js` to further verify uploaded files  
- Add frontend status messages (e.g. 'Upload failed, try again')  
- Add retry support for failed uploads  
- Clean up old files periodically to save disk space