Awesome — this is where the JG-Tech pipeline really flexes its automation muscles, Jim. 🧠📁 Tier 3 brings high-level sorting, integrity checks, metadata embedding, and escalation protocols — it’s your system’s brain + spine.

Let’s lay out the section cleanly so it prints beautifully and stores with confidence in your hard-copy backup binder.

# **🔴 Tier 3: Auto-Sorting + Validation + Tier Escalation**

### **🧩 Summary:**

Tier 3 automatically categorizes files by type (image, video, audio, etc.), verifies integrity via checksum, embeds watermarks, and escalates any failures to manual review. Ideal for high-volume workflows and client reliability.

### **📋 Workflow Logic:**

python

def tier3\_intake(file, user\_id, project\_id):

file\_type = mimetypes.guess\_type(file.name)[0] or "unknown"

base\_path = f"/intake/{project\_id}/{file\_type}/{date.today()}/"

os.makedirs(base\_path, exist\_ok=True)

# Save original

original\_file = f"{base\_path}/{file.name}"

shutil.copy(file.name, original\_file)

# Compute checksum

checksum = hashlib.sha256(open(file.name, "rb").read()).hexdigest()

# Create metadata

metadata = {

"user": user\_id,

"project": project\_id,

"filename": file.name,

"file\_type": file\_type,

"checksum": checksum,

"received": datetime.now().isoformat(),

"status": "validated"

}

if not validate\_checksum(checksum):

metadata["status"] = "validation\_failed"

escalate\_to\_manual\_review(file.name)

with open(f"{base\_path}/audit.json", "w") as f:

json.dump(metadata, f)

### **🖼️ Watermark Logic (Visible + Invisible)**

#### **✅ Visible (for JPEG):**

python

def apply\_visible\_watermark(image\_path, watermark\_path, output\_path):

base = Image.open(image\_path).convert("RGBA")

mark = Image.open(watermark\_path).convert("RGBA")

mark = mark.resize((200, 100))

base.paste(mark, (base.width - mark.width - 20, base.height - mark.height - 20), mark)

base.save(output\_path)

#### **✅ Invisible (embedded metadata):**

python

def embed\_invisible\_watermark(image\_path, metadata\_dict):

from PIL import PngImagePlugin

im = Image.open(image\_path)

meta = PngImagePlugin.PngInfo()

for key, val in metadata\_dict.items():

meta.add\_text(key, str(val))

im.save(image\_path.replace(".jpeg", "\_embed.jpeg"), "PNG", pnginfo=meta)

### **📝 Notes:**

* Uses folder structure: /intake/{project}/{file\_type}/{date}/
* Marks file copies using suffixes like \_wm.jpeg or \_embed.jpeg
* Failsafe trigger sends flagged assets back to Tier 1 for manual review
* All activity logged in audit.json for rollback or client proof