

Kowshik - SE_1

Perfect —


You already have a Supabase **Storage bucket** named `resumes`, which is awesome!

✅ I will now give you the **full clean code update**:

- Upload extracted resumes into Supabase Storage bucket
- Store file path as `job_id/filename.pdf`
- Save public URL in `resume_uploads` table

Exactly like you asked — **same folder structure** but now permanently stored!

Full Updated Code for Resume Upload (After ZIP extraction)

 Updated `upload_resume_info_to_db` function:

python

```
def upload_resume_info_to_db(file_name, file_path, job_id, user_id):
    import mimetypes

    resume_id = str(uuid.uuid4())

    # Read the local file bytes
    with open(file_path, "rb") as f:
        file_content = f.read()

    # Guess mime type
    content_type, _ = mimetypes.guess_type(file_path)
    if content_type is None:
        content_type = "application/pdf"

    # Upload to Supabase Storage
    storage_path = f"{job_id}/{file_name}" # resumes/job_id/filename
    try:
```

```

supabase.storage.from_("resumes").upload(
    path=storage_path,
    file=file_content,
    file_options={"content-type": content_type, "upsert": True} # allow
overwrite
)
print(f"Uploaded {file_name} to Supabase Storage under {storage_path}")
except Exception as e:
    print(f"Error uploading {file_name} to Storage: {str(e)}")
    return None

# Build the public URL
bucket_url = f"https://{SUPABASE_URL.split('https://')[
1]}/storage/v1/object/public/resumes/{storage_path}"

# Insert into database
data = {
    "resume_id": resume_id,
    "user_id": user_id,
    "job_id": job_id,
    "file_name": file_name,
    "file_path": bucket_url, # Storing the public URL
}
try:
    supabase.table("resume_uploads").insert(data).execute()
    print(f"Saved resume metadata for {file_name} with resume_id {resume_id}")
    return resume_id
except Exception as e:
    print(f"Error saving resume info for {file_name}: {str(e)}")
    return None

```



Quick Summary:

Field	Now	Before
file_name	Same	Same
file_path	Public cloud URL	Local file path

✓ Your frontend can **directly embed file_path** to show the resume inside a PDF viewer or iframe!



Storage Path in Supabase

It will now upload into:

```
makefile
```

```
Bucket: resumes
```

```
Folder: {job_id}
```

```
Files: {resume1.pdf}, {resume2.pdf}, ...
```

Exactly how your local `/resumes/{job_id}/filename.pdf` was structured.



Final Flow

- Recruiter uploads ZIP
 - ZIP is extracted
 - Each resume uploaded to **Supabase Storage**
 - URL saved in `resume_uploads`
 - Later frontend displays analysis + resume using stored public URL
-



Would you also like:

- A full ready **frontend snippet** (Next.js/React) to show the Resume preview from Supabase Storage?
- It'll help your frontend engineer a lot!

✓ Should I send that too now? (very useful for quick frontend integration)