Resume Parser Microservice – Execution & Integration Guide

(Polyrepo | Parse-Only | Redis-backed | React + Java)

1. Repository Setup

1.1 Create GitHub Repository

- Repo name: resume-parser-service
- Branch: main

1.2 Push Code

unzip resume-parser-parseonly.zip -d resume-parser-service

cd resume-parser-service

git init

git add .

git commit -m "Initial resume parser microservice (parse-only)"

git branch -M main

git remote add origin <GITHUB_REPO_URL>

git push -u origin main

---

2. Local Environment Setup

2.1 Prerequisites

- Python **3.11+**
- Docker (recommended)
- Tesseract OCR

Linux

sudo apt-get update

sudo apt-get install -y tesseract-ocr libtesseract-dev

Mac

brew install tesseract

2.2 Python Dependencies

python3.11 -m venv .venv

source .venv/bin/activate

pip install -r requirements.txt

---

3. Run the Service Locally

uvicorn app.main:app --host 0.0.0.0 --port 8084

Health checks:

curl http://localhost:8080/health

curl http://localhost:8080/ready

---

4. API Contract (Python → Java)

Endpoint

POST /v1/parse

Request

```
{
  "userId": "UUID",
  "resumeId": "UUID",
  "s3Bucket": "bucket-name",
  "s3Key": "path/to/resume.pdf",
  "fileType": "PDF"
}
```

Response (example)

```
{
  "personal": {},
  "experience": [],
  "education": [],
  "skills": [],
  "meta": {
    "sectionsFound": ["experience", "education"],
    "partial": true
  }
}
```


---

5. Java Integration Steps

5.1 Resume Upload Flow (Java)

1. User uploads resume
2. Java uploads file to **S3**
3. Java calls Python /v1/parse
4. Receives parsed JSON

5.2 Temporary Storage (Redis)

- Store parsed JSON in Redis
- Key example:

resume:parsed:{userId}

- TTL: **30–60 minutes**

---

6. React Frontend Flow

1. Java sends parsed JSON to React

2. React:
    - Renders profile sections
    - Allows full edit (add / delete / reorder)
3. User reviews & edits
4. User clicks **Submit**

---

7. Final Persistence (Java → DB)

1. React submits **final, user-confirmed payload**
2. Java:
    - Validates input
    - Applies merge rules
    - Persists into PostgreSQL tables
3. Redis entry is deleted

⚠️ **Python never accesses DB**

---

8. Docker Build & Run (Recommended)

docker build -t resume-parser-service .

docker run -p 8080:8080 resume-parser-service

---

9. AWS Deployment Checklist

Permissions

- IAM role with:
    - s3:GetObject on resume bucket

Runtime

- ECS or EKS

- Internal service discovery
- Health checks:
    - /health
    - /ready

---

10. Validation Checklist

- Python service reachable
- S3 access works
- /v1/parse returns JSON
- Redis stores parsed payload
- React edits work
- Final submit persists to DB
- No DB access from Python

---

11. Final Architecture Summary

- **Python**: Parse-only, stateless
- **Redis**: Temporary parsed data
- **React**: Review + edit
- **Java**: Orchestration + DB ownership

✓ Scalable
✓ Clean ownership
✓ Production-safe
✓ Future-ready (confidence flags later)