Smart Resume Compliance Checker

A serverless web application that allows users to upload resumes and job descriptions, then evaluates how well the resume matches the job description using AWS-powered analysis.

🚀 Live Demo

Frontend URL: Smart Resume Checker

Demo Video: Watch on YouTube

Project Structure

```
your-project-root/
├── lambda/
│   ├── smart_compliance/         # Lambda 1: Triggered by S3 upload
│   │   └── handler.py
│   ├── delete_resumes/           # Lambda 2: Invoked via API Gateway
│       └── handler.py
├── flask_api/                    # Zappa-deployed Flask API (Lambda 3)
│   └── app.py
├── frontend/                     # React frontend deployed to S3 + CloudFront
│   └── ...                       # Build output from React app
├── template.yml (Optional)       # AWS SAM/CloudFormation template (not used)
```

Key Features

- Login and Signup functionality
- Upload Resume + Job Description (PDF/TXT)
- Analyze compatibility using ATS-like score
- View matched vs missing keywords
- Secure storage of uploaded data in user-specific folders
- Resume management dashboard with selection and deletion

AWS Services Used

Service	Purpose
Lambda	Resume analyzer, delete handler, Flask callback logic
S3	Stores resumes, JDs, frontend static files
Textract	Extracts text from uploaded PDFs
API Gateway	Exposes REST endpoints for Lambda invocation
Zappa	Deploys Flask API as Lambda
CloudFront	HTTPS + CDN for frontend deployment

 AWS Lambda Usage

1. SmartCompliance Lambda

- **Trigger**: S3 (ObjectCreated)
- **Input**: Uploaded resume
- **Function**: Extracts resume text using Textract, loads matching JD from S3, computes ATS score
- **Output**: Sends score and keyword info to Flask API or stores fallback in S3

2. DeleteResumes Lambda

- **Trigger**: API Gateway (POST /delete-resumes)
- **Function**: Deletes selected resume files based on email and filenames sent from frontend

3. Flask Callback Lambda

- **Deployed using**: Zappa
- **Function**: Acts as the main API Gateway for all endpoints: login/signup, resume/JD uploads, ATS score retrieval, and result storage

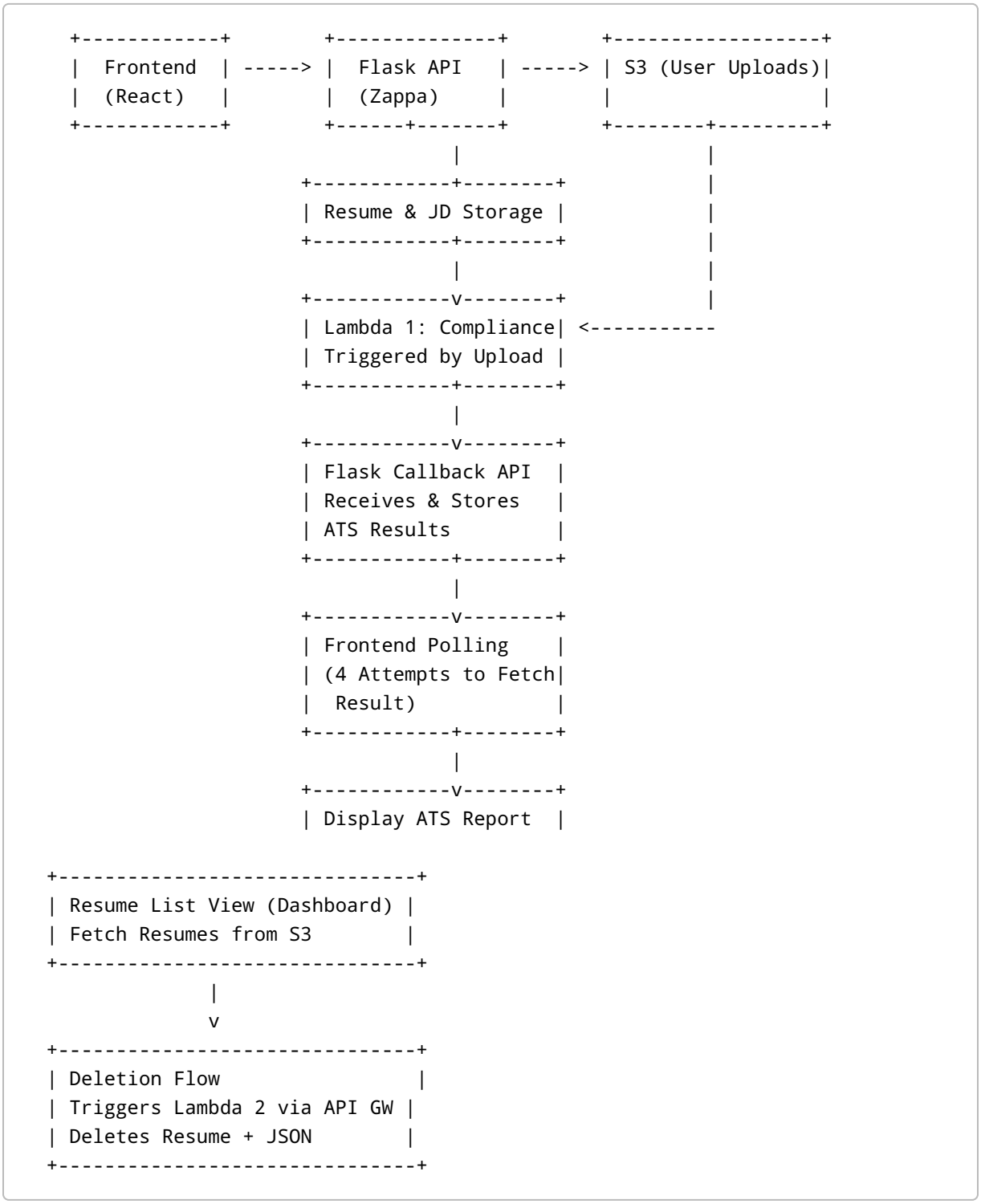
---

 Frontend

- **Built with**: React
- **Hosted on**: S3 static website + CloudFront distribution
- **Features**: Upload form, resume list, ATS result viewer, delete option, real-time polling

---

🚡 Architecture & Flow Diagram



🚡 Data Flows

Flow 1: Login & Signup

- Managed on frontend and passed to Flask API (authentication backend)

Flow 2: ATS Score Evaluation

- Resume and JD uploaded via Flask API Gateway (Zappa Lambda)
- Flask stores them to:
- `email/uploads/filename.pdf`
- `email/job_descriptions/filename.txt`
- S3 upload event triggers **SmartCompliance Lambda**
- Lambda extracts text, calculates ATS score, sends result **back to Flask callback endpoint**
- Frontend polls `/get-ats-score?filename=...&email=...` up to 4 times to fetch and show result

Flow 3: Resume Management

- Frontend fetches uploaded resumes from `files/email/uploads/`
- User selects files → `delete-resumes` POST to Flask
- Flask invokes **Delete Lambda** via API Gateway
- Lambda deletes resume and `files/email/results/filename.json`

Input:

- **JD** (Job Description text or extracted from PDF)
- **Resume** (PDF format)

Output:

- ATS Score (0–100%)
- Rating (1 to 5 stars)
- Suggested Keywords
- Matched Keywords

🕍 Video Demo

🎥Click here to watch the demo video on YouTube

> Demonstrates full functionality including login, resume upload, ATS score generation, and deletion with AWS Lambda flows.

Deployment Notes

- All Lambdas were deployed directly via AWS Console / Zappa
- Buckets created manually (4 total: uploads, JD, frontend, config/DB)
- CloudFormation `template.yml` was not used to avoid affecting deployed resources
- Frontend built and deployed via S3 → CloudFront
- Flask API handles all uploads, deletions, and score fetching

---

Submission Checklist

- **Public GitHub Repository**\ ➤ Contains all code for Lambda functions, Flask API, and frontend\ ➤ Includes detailed README with architecture, data flow, and deployment details
- **README.md Requirements**\ Explanation of how AWS Lambda is used\ Architecture + flow diagrams\ AWS services used\ Inputs/outputs and complete data flow
- **Demo Video** \ ➤ Approximately 3 minutes\ ➤ Demonstrates full app functionality (login, upload, ATS score, delete)\ ➤ Explains Lambda integration\ ➤ Uploaded to YouTube and marked public
- **AWS Lambda Explanation**\ Lambda 1: ATS Scoring (S3 Triggered)\ Lambda 2: Delete resumes (API Triggered)\ Lambda 3: Flask callback + main API (Zappa-deployed Lambda)
- **List of AWS Tools Used**\ Lambda, S3, Textract, API Gateway, CloudFront, Zappa

---

🚎Contact / Credits

Built with using AWS for the AWS Lambda Hackathon 2025.

Contributors:

- Saisharan Bhonagiri