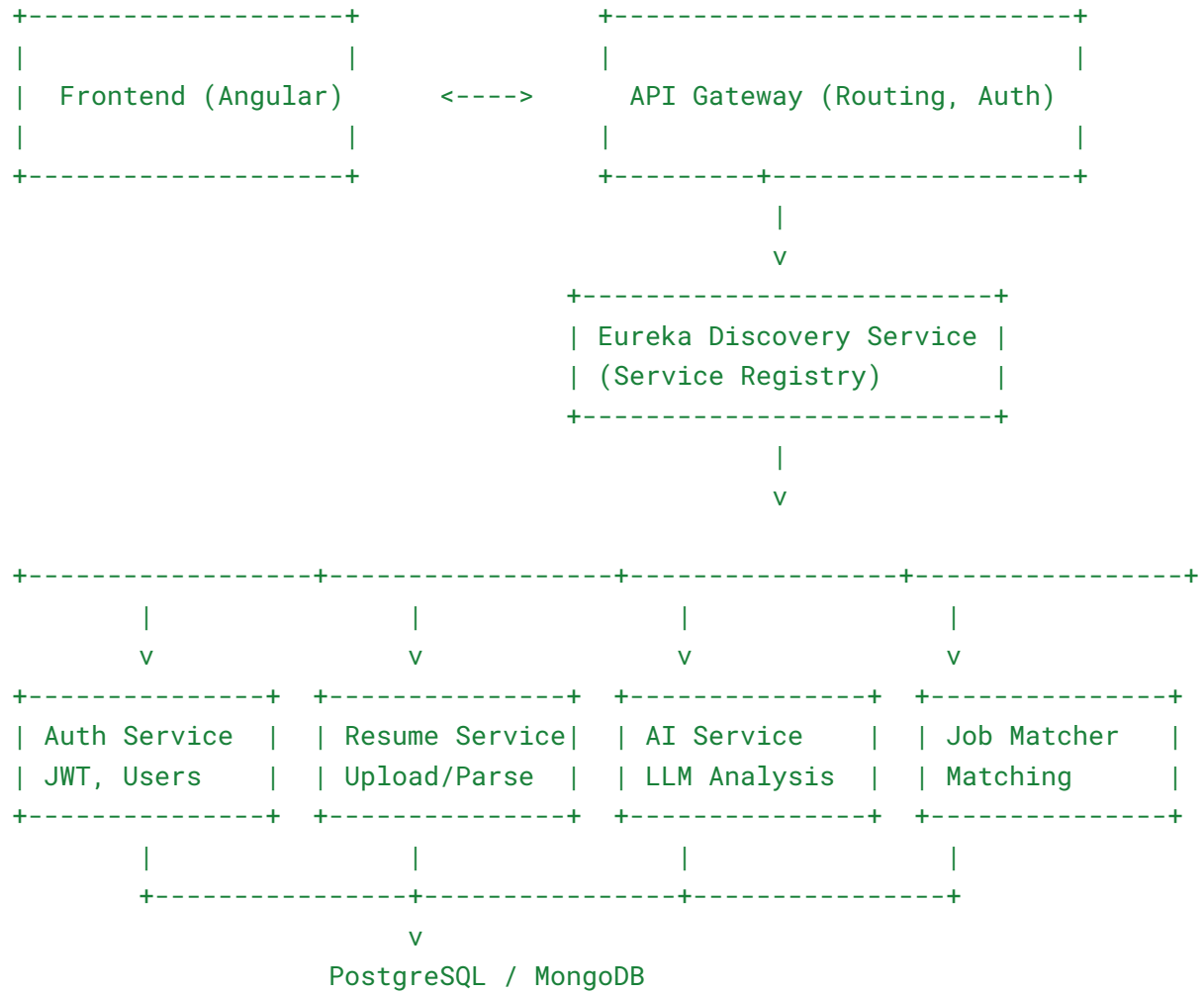


Architecture Components



Microservice Responsibilities

Service	Role	Database	Key Function
Discovery Service	Registers all services	None	Enables dynamic service discovery
API Gateway	Single entry point	None	Routes requests, handles JWT verification, rate limiting
Auth Service	User registration & login	PostgreSQL	Provides JWT tokens for secure communication
Resume Service	Upload, parse resumes	MongoDB	Extracts text from resumes (PDF/DOCX)
AI Service	Analyze resume & job description	MongoDB	Sends resume/job data to LLM, generates ATS score
Job Matcher Service	Match resume with jobs	PostgreSQL	Stores jobs, computes match score, returns results
Config Service (Optional)	Centralized config	None	Manages <code>application.yml</code> for all services

3 Control Flow

Here's a **step-by-step request flow** from frontend to microservices:

1. User Authentication

- Frontend → API Gateway → Auth Service
- Auth Service validates credentials, returns **JWT**
- JWT cached in frontend for subsequent requests

2. Resume Upload

- Frontend → API Gateway → Resume Service
- Resume Service parses the resume (PDF/DOCX → JSON)
- Parsed resume saved in **MongoDB**

3. AI Analysis

- Resume Service → AI Service (REST call)
- AI Service sends parsed resume and job description to **OpenAI / HuggingFace LLM**
- Receives ATS/fit score → saves in MongoDB
- Returns score to Resume Service → forwarded to frontend

4. Job Matching

- Frontend → API Gateway → Job Matcher Service
- Job Matcher fetches job postings from PostgreSQL
- Compares resume data + AI analysis
- Returns ranked job matches

5. Service Discovery & Routing

- All services register themselves to **Eureka**
- API Gateway discovers services dynamically (**lb://service-name**)
- Microservices communicate internally using service names

4 Sequence Diagram (Simplified)

Frontend -> API Gateway: Login (credentials)

API Gateway -> Auth Service: Authenticate

Auth Service -> API Gateway: JWT Token

API Gateway -> Frontend: JWT

Frontend -> API Gateway: Upload Resume

API Gateway -> Resume Service: Resume File

Resume Service -> MongoDB: Save Parsed Resume

Resume Service -> AI Service: Send Resume + Job Desc

AI Service -> LLM: Analyze & Score

LLM -> AI Service: Return Score

AI Service -> Resume Service: Score

Resume Service -> API Gateway: Return Score

API Gateway -> Frontend: Display Score

Frontend -> API Gateway: Get Job Matches

API Gateway -> Job Matcher: Request Matches

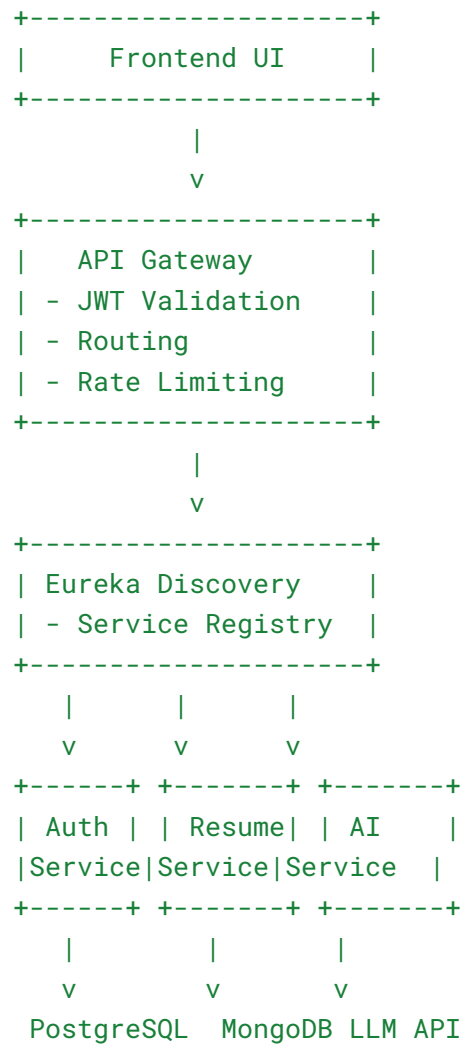
Job Matcher -> PostgreSQL: Fetch Jobs

Job Matcher -> AI Service (optional): Get AI Score

Job Matcher -> API Gateway: Return Job Matches

API Gateway -> Frontend: Display Jobs







5 Component Diagram (Optional for Docs)






6 Key Points for Documentation

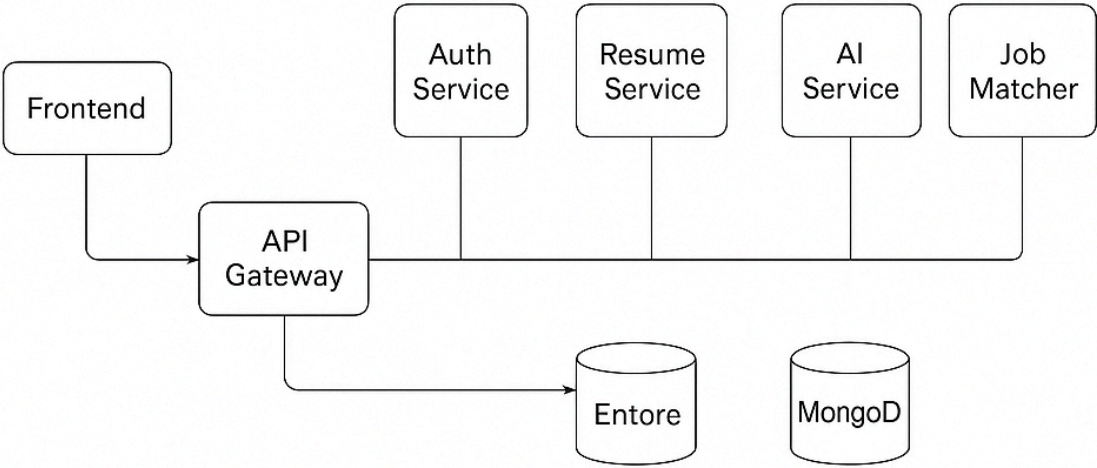
- **Why Microservices?**
 - Scalability: Each service can scale independently
 - Maintainability: Teams can work on separate services
 - Tech Flexibility: AI Service can integrate with LLM APIs without affecting others
- **Why Eureka + Gateway?**
 - Service discovery avoids hardcoded URLs
 - API Gateway handles routing, authentication, and rate-limiting centrally
- **Why Hybrid DB?**
 - PostgreSQL for structured transactional data (users, jobs)
 - MongoDB for flexible AI/resume storage
- **AI Integration**
 - Uses Spring AI → connects to OpenAI/HuggingFace LLMs
 - Generates ATS score, skill matching, and resume analysis

Dependencies for Each Microservice (Updated Full List)

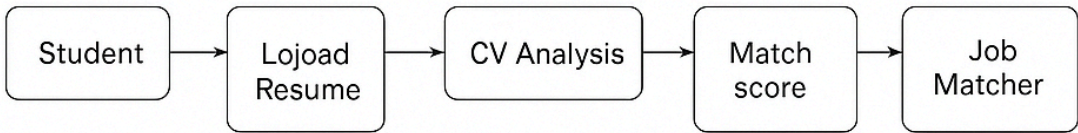
Service	Key Dependencies
 Discovery Server	<code>spring-boot-starter-web,</code> <code>spring-cloud-starter-netflix-eureka-server</code>
 API Gateway	<code>spring-cloud-starter-gateway,</code> <code>spring-cloud-starter-netflix-eureka-client,</code> <code>spring-boot-starter-security,</code> <code>spring-boot-starter-oauth2-resource-server</code> (if JWT validation here)
 Auth Service	<code>spring-boot-starter-web,</code> <code>spring-boot-starter-security,</code> <code>spring-boot-starter-data-jpa,</code> <code>spring-cloud-starter-netflix-eureka-client,</code> <code>spring-boot-starter-validation,</code> <code>postgresql</code>
 Resume Service	<code>spring-boot-starter-web,</code> <code>spring-boot-starter-data-mongodb,</code> <code>spring-cloud-starter-netflix-eureka-client,</code> <code>spring-boot-starter-validation,</code> <code>spring-boot-starter-security</code> (optional JWT)
 AI Analyzer Service	<code>spring-boot-starter-web,</code> <code>spring-boot-starter-data-mongodb</code> (optional), <code>spring-cloud-starter-netflix-eureka-client,</code> <code>spring-ai-openai-spring-boot-starter,</code> <code>spring-boot-starter-validation</code>
 Job Matcher Service	<code>spring-boot-starter-web,</code> <code>spring-boot-starter-data-jpa,</code> <code>spring-cloud-starter-netflix-eureka-client,</code> <code>spring-boot-starter-validation,</code> <code>postgresql</code>

 Profile Service	<code>spring-boot-starter-web, spring-boot-starter-data-jpa, spring-cloud-starter-netflix-eureka-client, postgresql</code>
 Config Server (optional)	<code>spring-cloud-config-server</code>
 Monitoring (optional)	<code>spring-boot-starter-actuator, micrometer-registry-prometheus</code>

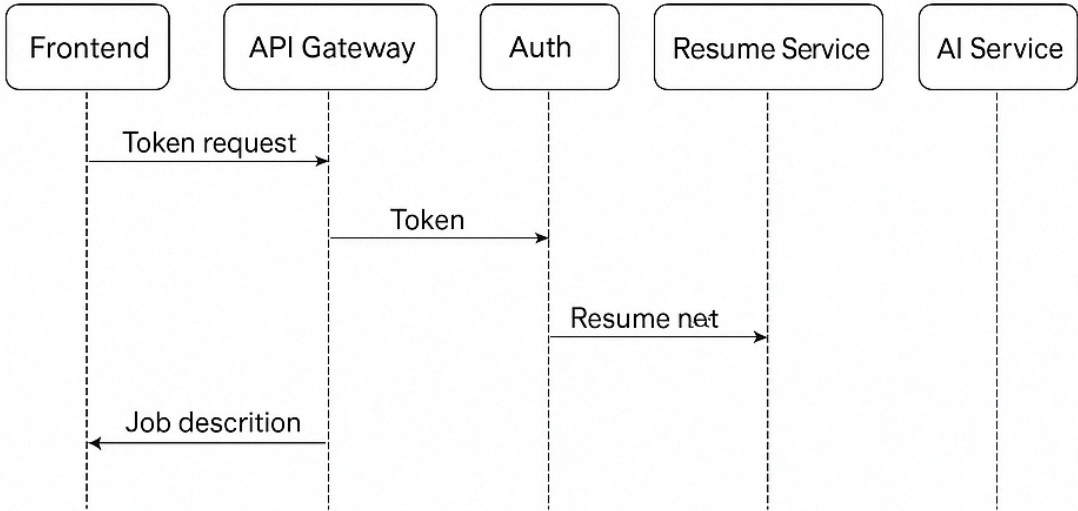
AI Resume Scanner & Job Matcher



Control Flow



Sequence Diagram



Component Diagram

