Agentic AI Hiring Platform - API Endpoints Documentation

Base URL

```
https://agentic-v2-0.onrender.com
```

Interactive Documentation

- **Swagger UI**: https://agentic-v2-0.onrender.com/docs
- **ReDoc**: https://agentic-v2-0.onrender.com/redoc

1. Root & Health Endpoints

1.1 Welcome Endpoint

Method: `GET`

Endpoint: `/`

Description: API information and available endpoints

Input: None

Output:

```
{
  "message": "Welcome to Agentic AI Hiring Platform API",
  "version": "2.0",
  "features": ["AI-powered resume evaluation", "..."],
  "endpoints": {
    "documentation": "/docs",
    "health": "/health",
    "companies": "/company",
    "jobs": "/job",
    "applications": "/apply",
    "candidates": "/candidate",
    "analytics": "/analytics"
  }
}
```

1.2 Health Check

Method: `GET`

Endpoint: `/health`

Description: Service health status

Input: None

Output:

```
{
  "status": "healthy",
  "service": "Agentic AI Hiring Platform",
  "version": "2.0.0"
}
```

2. Company Endpoints

2.1 Create Company

Method: `POST`

Endpoint: `/company/`

Description: Register a new company

Input (Form Data):

```
name: string (required)
description: string (required)
```

Output:

```
{
  "id": 1,
  "name": "TechCorp",
  "description": "Leading technology company",
  "created_at": "2026-02-12T10:00:00"
}
```

3. Job Endpoints

3.1 Create Job with Company (Merged Endpoint) ■ RECOMMENDED

Method: POST

Endpoint: /job/create-with-company

Description: Create company and job posting in a single API call. If company exists, reuses it.

Input (Multipart Form):

```
# Company Details
company_name: string (required)
company_description: string (required)

# Job Details
role: string (required)
location: string (optional, default: "")
salary: string (optional, default: "")
employment_type: string (optional, default: "Full-time")
required_experience: integer (required)
jd_pdf: file (required, .pdf only)
```

Output:

```
{
  "company": {
    "id": 1,
    "name": "TechCorp",
    "description": "Leading technology company",
    "created_at": "2026-02-12T10:00:00",
    "status": "created"
  },
  "job": {
    "id": 1,
    "company_id": 1,
    "role": "Python Developer",
    "location": "Remote",
    "salary": "$80k-$120k",
    "employment_type": "Full-time",
    "required_experience": 0,
    "created_at": "2026-02-12T10:00:00",
    "jd_text_preview": "We are looking for...",
    "embedding_dimensions": 768,
    "skills_stored": 15
  },
  "message": "Company created and job created successfully",
  "pages_parsed": 2,
  "skills_extracted": ["python", "flask", "rest api", "..."],
  "technical_skills": ["python", "flask", "sql", "git", "docker"],
  "soft_skills": ["communication", "teamwork", "problem solving"]
}
```

Python Example:

```
import requests

with open("job_description.pdf", "rb") as f:
    response = requests.post(
        "https://agentic-v2-0.onrender.com/job/create-with-company",
        data={
            "company_name": "TechCorp",
            "company_description": "Leading technology company",
            "role": "Python Developer",
            "location": "Remote",
            "salary": "$80k-$120k",
```

```
            "employment_type": "Full-time",
            "required_experience": 0
        },
        files={"jd_pdf": f}
    )
print(response.json())
```

3.2 Create Job Posting (Separate)

Method: POST

Endpoint: /job/

Description: Create a new job by uploading JD PDF (requires existing company_id)

Input (Multipart Form):

```
company_id: integer (required)
role: string (required)
location: string (optional, default: "")
salary: string (optional, default: "")
employment_type: string (optional, default: "Full-time")
required_experience: integer (required)
jd_pdf: file (required, .pdf only)
```

Output:

```
{
  "job": {
    "id": 1,
    "company_id": 1,
    "role": "Python Developer",
    "location": "Remote",
    "salary": "$80k-$120k",
    "employment_type": "Full-time",
    "required_experience": 0,
    "created_at": "2026-02-12T10:00:00",
    "jd_text_preview": "We are looking for...",
    "embedding_dimensions": 768,
    "skills_stored": 15
  },
  "message": "Job created and stored in database successfully",
  "pages_parsed": 2,
  "skills_extracted": ["python", "flask", "rest api", "..."],
  "technical_skills": ["python", "flask", "sql", "git", "docker"],
  "soft_skills": ["communication", "teamwork", "problem solving"]
}
```

3.3 Get Job Details

Method: GET

Endpoint: /job/{job_id}

Description: Get detailed job information

Input (Path Parameter):

```
job_id: integer (required)
```

Output:

```
{
  "id": 1,
```

```
  "company_id": 1,
  "company_name": "TechCorp",
  "company_description": "Leading technology company",
  "role": "Python Developer",
  "location": "Remote",
  "salary": "$80k-$120k",
  "employment_type": "Full-time",
  "required_experience": 0,
  "jd_text": "Full job description text...",
  "created_at": "2026-02-12T10:00:00",
  "application_count": 25,
  "skills_required": ["python", "flask", "rest api", "..."],
  "technical_skills": ["python", "flask", "sql"],
  "soft_skills": ["communication", "teamwork"]
}
```

3.4 List Jobs

Method: `GET`

Endpoint: `/job/`

Description: Browse available jobs with filters

Input (Query Parameters):

```
company_id: integer (optional)
location: string (optional)
employment_type: string (optional)
skip: integer (optional, default: 0)
limit: integer (optional, default: 100)
```

Output:

```
{
  "total": 50,
  "showing": 10,
  "skip": 0,
  "limit": 100,
  "jobs": [
    {
      "id": 1,
      "role": "Python Developer",
      "company_id": 1,
      "company_name": "TechCorp",
      "location": "Remote",
      "salary": "$80k-$120k",
      "employment_type": "Full-time",
      "required_experience": 0,
      "created_at": "2026-02-12T10:00:00",
      "application_count": 25,
      "skills_preview": ["python", "flask", "sql", "git", "docker"]
    }
  ]
}
```

3.5 Get Company Applications

Method: `GET`

Endpoint: `/job/{company_id}/applications`

Description: Get all applications across all jobs for a specific company

Input:

```
Path: company_id: integer (required)
Query: status_filter: string (optional)
```

Output:

```
{
  "company": {
    "id": 1,
    "name": "TechCorp"
  },
  "total_applications": 25,
  "total_jobs": 3,
  "statistics": {
    "fast_track": 5,
    "selected": 8,
    "hire_pooled": 7,
    "rejected": 4,
    "review_required": 1
  },
  "applications": [...]
}
```

4. Application Endpoints

4.1 Submit Application

Method: POST

Endpoint: /apply/{company_id}

Description: Submit job application with resume PDF - automatically finds the job for the specified company

Path Parameters:

- company_id: integer (required) - The ID of the company you're applying to

Multipart Form Parameters:

- name: string (required) - Candidate's full name
- email: string (required) - Candidate's email address
- mobile: string (required) - Candidate's mobile number
- linkedin: string (optional) - LinkedIn profile URL
- github: string (optional) - GitHub profile URL
- experience: integer (required) - Years of experience
- resume_pdf: file (required, .pdf only) - Resume in PDF format

Example Request:

```
import requests

with open("resume.pdf", "rb") as f:
    response = requests.post(
        "https://agentic-v2-0.onrender.com/apply/15",  # company_id = 15
        data={
            "name": "Arjun Malhotra",
            "email": "arjun@example.com",
            "mobile": "1234567890",
            "experience": 2
        },
        files={"resume_pdf": f}
    )
print(response.json())
```

Output:


```
{
  "application_id": 101,
  "candidate_id": 42,
  "job_id": 10,
  "company_id": 15,
  "decision": "Selected",
  "composite_score": 0.78,
  "explanation": {
    "basic_explanation": {...},
    "xai_explanation": {...},
    "skill_gap_analysis": {...},
    "skill_evidence_graph": {...}
  },
  "message": "Application evaluated successfully",
  "pages_parsed": 2,
  "skills_detected": 12
}
```


How It Works:

- If job_id is NOT in form data → Uses old format (path param is job_id)
- If job_id IS in form data → Uses new format (path param is company_id with validation)

4.2 Get Application Details

Method: GET

Endpoint: /apply/{application_id}

Description: Get complete application details

Input (Path Parameter):

```
application_id: integer (required)
```

Output:

```
{
  "application": {
    "id": 101,
    "job_id": 1,
    "candidate_id": 42,
    "rfs": 0.85,
    "dcs": 0.75,
    "elc": 1.0,
    "composite_score": 0.78,
    "rank": 3,
    "decision": "Selected",
    "fraud_flag": false,
    "created_at": "2026-02-12T10:00:00"
  },
  "candidate": {
    "id": 42,
    "name": "Arjun Malhotra",
    "email": "arjun@example.com",
    "mobile": "1234567890",
    "experience": 2
  },
  "job": {
    "id": 1,
    "role": "Python Developer",
    "company_name": "TechCorp"
  },
  "explanation": {...}
}
```

4.3 Get Application History

Method: GET

Endpoint: /apply/{application_id}/history

Description: Get audit trail for application

Input (Path Parameter):

```
application_id: integer (required)
```

Output:

```
{
  "application_id": 101,
  "total_events": 5,
  "history": [
```

```
    {
      "timestamp": "2026-02-12T10:00:00",
      "event_type": "application_created",
      "details": {...}
    }
  ]
}
```

4.4 List Applications

Method: `GET`

Endpoint: `/apply/`

Description: List all applications with filters

Input (Query Parameters):

```
decision: string (optional)
fraud_flag: boolean (optional)
skip: integer (optional, default: 0)
limit: integer (optional, default: 100)
```

Output:

```
{
  "total": 150,
  "showing": 100,
  "skip": 0,
  "limit": 100,
  "applications": [...]
}
```

5. Candidate Endpoints

5.1 Get Candidate Details

Method: `GET`

Endpoint: `/candidate/{candidate_id}`

Description: Get candidate information

Input (Path Parameter):

```
candidate_id: integer (required)
```

Output:

```
{
  "id": 42,
  "name": "Arjun Malhotra",
  "email": "arjun@example.com",
  "mobile": "1234567890",
  "linkedin": "linkedin.com/in/arjun",
  "github": "github.com/arjun",
  "experience": 2,
  "created_at": "2026-02-12T10:00:00",
  "skills_extracted": {
    "all_skills": ["python", "flask", "sql", "..."],
    "technical_skills": ["python", "flask", "sql"],
    "soft_skills": ["communication", "teamwork"]
  }
}
```

5.2 Get Candidate Applications

Method: `GET`

Endpoint: `/candidate/{candidate_id}/applications`

Description: Get all applications by candidate

Input (Path Parameter):

```
candidate_id: integer (required)
```

Output:

```
{
  "candidate": {
    "id": 42,
    "name": "Arjun Malhotra",
    "email": "arjun@example.com"
  },
  "total_applications": 3,
  "applications": [...]
}
```

5.3 Get Candidate History

Method: GET

Endpoint: /candidate/{candidate_id}/history

Description: Get audit trail for candidate

Input (Path Parameter):

```
candidate_id: integer (required)
```

Output:

```
{
  "candidate_id": 42,
  "total_events": 8,
  "history": [...]
}
```

5.4 Search Candidate by Email

Method: GET

Endpoint: /candidate/search/by-email

Description: Find candidate by email address

Input (Query Parameter):

```
email: string (required)
```

Output: Same as Get Candidate Details

5.5 List All Candidates

Method: GET

Endpoint: /candidate/

Description: List all candidates with pagination

Input (Query Parameters):

```
skip: integer (optional, default: 0)
limit: integer (optional, default: 100)
```

Output:

```
{
  "total": 200,
  "showing": 100,
  "skip": 0,
  "limit": 100,
  "candidates": [...]
}
```

5.6 Get Candidate Master Details ■

Method: GET

Endpoint: /candidate/{candidate_id}/master

Description: Comprehensive master endpoint returning complete candidate profile with all applications, scores, decisions, and detailed analysis

Input (Path Parameter):

```
candidate_id: integer (required)
```

Output:


```
{
  "candidate_profile": {
    "candidate_id": 42,
    "name": "Arjun Malhotra",
    "email": "arjun@example.com",
    "mobile": "+91-9876543210",
    "linkedin": "linkedin.com/in/arjun",
    "github": "github.com/arjun",
    "years_of_experience": 2,
    "skills": ["Python", "Flask", "SQL", "Docker", "REST APIs"],
    "resume_text": "Full resume text content...",
    "profile_created_at": "2026-02-01T10:00:00"
  },
  "application_summary": {
    "total_applications": 10,
    "selected": 3,
    "rejected": 5,
    "pending": 2,
    "average_composite_score": 78.5,
    "best_application": {
      "application_id": 101,
      "job_role": "Senior Python Developer",
      "composite_score": 92.3,
      "decision": "Selected",
      "rank": 1
    }
  },
  "applications": [
    {
      "application_id": 101,
      "applied_at": "2026-02-10T14:30:00",
      "status": "evaluated",
      "job_details": {
        "job_id": 25,
        "role": "Senior Python Developer",
        "location": "Bangalore, India",
        "salary": "■15-20 LPA",
        "employment_type": "Full-time",
        "required_experience": 3,
        "job_description": "We are seeking a talented Python developer...",
        "required_skills": ["Python", "Django", "PostgreSQL", "Docker", "AWS"]
      },
      "company_details": {
        "company_id": 5,
        "company_name": "Tech Innovations Ltd",
        "company_description": "Leading software development company..."
      },
      "scores": {
        "role_fit_score": 88.5,
        "domain_competency_score": 92.0,
        "experience_level_compatibility": 95.0,
        "composite_score": 92.3,
        "rank": 1,
        "rank_description": "Ranked #1"
      },
      "decision": {
        "status": "Selected",
        "reason": "Excellent technical skills with strong Python expertise and relevant experience",
        "detailed_explanation": {
          "strengths": [
            "Strong Python and Flask experience",
            "Good understanding of databases",
```

```
            "Experience with REST APIs"
          ],
          "concerns": [],
          "recommendations": "Proceed with interview"
        }
      },
      "fraud_detection": {
        "fraud_flag": false,
        "similarity_index": 0.15,
        "fraud_details": null
      },
      "skill_analysis": {
        "skill_match": {
          "matched_skills": ["Python", "SQL"],
          "missing_skills": ["Django", "AWS"],
          "match_percentage": 60.0
        },
        "experience_details": {
          "candidate_years": 2,
          "required_years": 3,
          "compatibility": "good"
        }
      }
    }
  ]
}
```


Features:

- Complete candidate profile with all personal details
- Skills extracted from resume
- Summary statistics (total applications, selection rate, average scores)
- Best performing application highlight
- Detailed breakdown of each application including:
- Full job and company information
- All scoring metrics (RFS, DCS, ELC, Composite)
- Ranking position
- Selection/rejection decision with reasoning
- Fraud detection results
- Skill match analysis

Use Cases:

- HR dashboard for comprehensive candidate review
- Candidate performance tracking across multiple applications
- Export candidate data for reporting and analytics
- Audit trail and compliance documentation
- Application history visualization

5.7 Get All Candidates Master Details ■

Method: `GET`

Endpoint: `/candidate/master/all`

Description: Comprehensive master endpoint returning complete details for ALL candidates with pagination. Each candidate includes full profile, applications, scores, and decisions.

Input (Query Parameters):

```
skip: integer (optional, default: 0) - Number of candidates to skip
limit: integer (optional, default: 100, max: 500) - Maximum candidates to return
```

Output:


```
{
  "total_candidates": 250,
  "showing": 100,
  "skip": 0,
  "limit": 100,
  "global_statistics": {
    "total_applications": 523,
    "total_selected": 145,
    "total_rejected": 298,
    "total_pending": 80
  },
  "candidates": [
    {
      "candidate_profile": {
        "candidate_id": 1,
        "name": "John Doe",
        "email": "john@example.com",
        "mobile": "+1234567890",
        "linkedin": "linkedin.com/in/johndoe",
        "github": "github.com/johndoe",
        "years_of_experience": 5,
        "skills": ["Python", "JavaScript", "React"],
        "resume_text": "Full resume...",
        "profile_created_at": "2026-01-15T10:00:00"
      },
      "application_summary": {
        "total_applications": 8,
        "selected": 2,
        "rejected": 4,
        "pending": 2,
        "average_composite_score": 75.5,
        "best_application": {
          "application_id": 25,
          "job_role": "Senior Developer",
          "composite_score": 89.5,
          "decision": "Selected",
          "rank": 1
        }
      },
      "applications": [
        {
          "application_id": 25,
          "applied_at": "2026-02-10T14:30:00",
          "status": "evaluated",
          "job_details": { },
          "company_details": { },
          "scores": { },
          "decision": { },
          "fraud_detection": { },
          "skill_analysis": { }
        }
      ]
    },
    {
      "candidate_profile": { },
      "application_summary": { },
      "applications": [ ]
    }
  ]
}
```


Features:

- Returns complete data for all candidates in the system
- Pagination support to handle large datasets (max 500 per request)
- Global statistics across all candidates shown
- Each candidate includes:
- Complete profile information
- All applications with full details

- Summary statistics
- Best application highlight
- Same detailed application data as single candidate endpoint

Use Cases:

- Admin dashboard showing all candidates
- Bulk data export for HR analytics
- System-wide reporting and statistics
- Candidate comparison and benchmarking
- Mass data processing and analysis
- Complete database backup/export

Pagination Example:

```
# Get first 50 candidates
GET /candidate/master/all?limit=50

# Get next 50 candidates
GET /candidate/master/all?skip=50&limit=50

# Get all candidates (up to 500)
GET /candidate/master/all?limit=500
```

6. Analytics Endpoints

6.1 Get XAI Explanation

Method: GET

Endpoint: /analytics/application/{application_id}/xai

Description: Get explainable AI explanation for application

Input (Path Parameter):

```
application_id: integer (required)
```

Output:

```
{
  "application_id": 101,
  "candidate": {
    "name": "Arjun Malhotra",
    "experience": 2
  },
  "job": {
    "role": "Python Developer",
    "required_experience": 0
  },
  "xai_explanation": {
    "decision": "Selected",
    "confidence_level": {
      "level": "high",
      "score": 0.85,
      "explanation": "Decision confidence based on score clarity"
    },
    "key_factors": [
      {
        "factor": "Skill Match",
        "impact": "High",
        "value": "75%",
        "contribution": "+25 points"
      }
    ],
    "strengths": ["Strong Python skills", "Flask experience", "..."],
    "areas_for_improvement": ["Learn Docker", "AWS certification", "..."],
    "score_breakdown": {
      "rfs": 0.85,
      "dcs": 0.75,
      "elc": 1.0,
      "composite": 0.78
    },
    "decision_rationale": "Candidate shows strong...",
    "recommendations": ["Consider for interview", "..."]
  }
}
```

6.2 Get Skill Gap Analysis

Method: GET

Endpoint: /analytics/application/{application_id}/skill-gap

Description: Get detailed skill gap analysis

Input (Path Parameter):

```
application_id: integer (required)
```

Output:


```
{
  "application_id": 101,
  "candidate": {
    "name": "Arjun Malhotra",
    "email": "arjun@example.com"
  },
  "job": {
    "role": "Python Developer"
  },
  "skill_gap_analysis": {
    "summary": {
      "total_required_skills": 20,
      "skills_matched": 12,
      "skills_missing": 8,
      "skills_missing_required": 2,
      "skills_missing_nice_to_have": 6,
      "gap_percentage": 40.0,
      "gap_percentage_required": 14.29,
      "severity": "Medium - Moderate skills gap",
      "is_closeable": true,
      "focus_on_required": true
    },
    "gap_breakdown": {
      "critical_missing": {
        "count": 1,
        "skills": ["kubernetes"],
        "impact": "High - Essential for role performance (Required Skills)"
      },
      "important_missing": {
        "count": 1,
        "skills": ["docker"],
        "impact": "Medium - Important but can be learned on job (Required Skills)"
      },
      "nice_to_have_missing": {
        "count": 6,
        "skills": ["aws", "azure", "terraform", "..."],
        "impact": "Low - Beneficial but not critical (Nice-to-Have Skills)"
      }
    },
    "transferable_skills": {
      "count": 2,
      "skills": [
        {
          "from_skill": "python",
          "to_skill": "java",
          "transfer_ease": "Easy - Related technology"
        }
      ]
    },
    "learning_roadmap": [
      {
        "priority": 1,
        "skill": "kubernetes",
        "difficulty": "intermediate",
        "estimated_weeks": 8,
        "learning_resources": [
          "Online courses for kubernetes",
          "Official kubernetes documentation"
        ]
      }
    ],
    "estimated_closure_time": {
      "total_weeks": 34,
      "total_months": 8.4,
      "skills_count": 8
    },
    "recommendations": [
      "PRIORITY: Focus on learning critical skills: kubernetes",
      "Next, work on important skills: docker",
      "Optional: Add value with: aws, azure"
```

```
    ]
  }
}
```

6.3 Get Skill Evidence Graph

Method: GET

Endpoint: /analytics/application/{application_id}/skill-graph

Description: Get graph data for skill visualization

Input (Path Parameter):

```
application_id: integer (required)
```

Output:

```
{
  "application_id": 101,
  "skill_evidence_graph": {
    "graph": {
      "nodes": [
        {
          "id": "job",
          "label": "Job Requirements",
          "type": "job",
          "size": 30,
          "color": "#3498db"
        },
        {
          "id": "matched_1",
          "label": "python",
          "type": "matched",
          "size": 20,
          "color": "#27ae60"
        }
      ],
      "edges": [
        {
          "source": "job",
          "target": "matched_1",
          "type": "required",
          "strength": 1.0,
          "color": "#27ae60",
          "label": "Matched"
        }
      ]
    },
    "statistics": {
      "total_nodes": 25,
      "total_edges": 35,
      "matched_skills_count": 12,
      "missing_skills_count": 8,
      "extra_skills_count": 5,
      "match_rate": 60.0
    },
    "legend": {
      "green": "Skills that match job requirements",
      "red": "Skills missing from candidate profile",
      "purple": "Additional skills candidate possesses"
    },
    "visualization_config": {
      "layout": "force-directed",
      "show_labels": true,
      "interactive": true,
      "zoom_enabled": true
    }
  }
}
```

6.4 Get Job Application Rankings

Method: `GET`

Endpoint: `/analytics/job/{job_id}/rankings`

Description: Get ranked list of all applications

Input:

```
Path: job_id: integer (required)
Query: limit: integer (optional, default: 100)
```

Output:

```
{
  "job": {
    "id": 1,
    "role": "Python Developer",
    "company_id": 1
  },
  "total_applications": 25,
  "rankings": [
    {
      "rank": 1,
      "application_id": 101,
      "candidate": {
        "id": 42,
        "name": "Arjun Malhotra",
        "email": "arjun@example.com",
        "experience": 2
      },
      "scores": {
        "composite_score": 0.89,
        "rfs": 0.90,
        "dcs": 0.85,
        "elc": 1.0
      },
      "decision": "Fast-Track Selected",
      "fraud_flag": false,
      "created_at": "2026-02-12T10:00:00"
    }
  ]
}
```

6.5 Get Top N Candidates

Method: `GET`

Endpoint: `/analytics/job/{job_id}/top-candidates`

Description: Get top candidates for recruiter dashboard

Input:

```
Path: job_id: integer (required)
Query: top_n: integer (optional, default: 10)
```

Output:

```
{
  "job": {
    "id": 1,
```

```
    "role": "Python Developer",
    "company_id": 1
  },
  "top_n": 10,
  "top_candidates": [
    {
      "rank": 1,
      "application_id": 101,
      "candidate": {
        "id": 42,
        "name": "Arjun Malhotra",
        "email": "arjun@example.com",
        "mobile": "1234567890",
        "experience": 2,
        "linkedin": "linkedin.com/in/arjun",
        "github": "github.com/arjun"
      },
      "scores": {
        "composite_score": 0.89,
        "match_percentage": "89.0%"
      },
      "skill_match_summary": {
        "matched_skills_count": 15,
        "missing_skills_count": 3,
        "match_score": 0.85
      },
      "decision": "Fast-Track Selected",
      "created_at": "2026-02-12T10:00:00"
    }
  ]
}
```


6.6 Get Job Statistics

Method: GET

Endpoint: /analytics/job/{job_id}/statistics

Description: Get comprehensive job application statistics

Input (Path Parameter):

```
job_id: integer (required)
```

Output:


```
{
  "job": {
    "id": 1,
    "role": "Python Developer",
    "company_id": 1
  },
  "total_applications": 25,
  "decision_breakdown": {
    "Fast-Track Selected": 5,
    "Selected": 8,
    "Hire-Pooled": 7,
    "Rejected": 4,
    "Review Required": 1
  },
  "average_scores": {
    "composite": 0.6543,
    "rfs": 0.7021,
    "dcs": 0.6234,
    "elc": 0.7891
  },
  "fraud_statistics": {
    "total_fraud": 2,
    "fraud_percentage": 8.0
  },
  "top_candidate": {
```

```
    "name": "Arjun Malhotra",
    "rank": 1,
    "score": 0.89,
    "decision": "Fast-Track Selected"
  }
}
```

6.7 Get Candidate Applications (Analytics)

Method: GET

Endpoint: /analytics/candidate/{candidate_id}/applications

Description: Get all applications by candidate (for candidate portal)

Input (Path Parameter):

```
candidate_id: integer (required)
```

Output:

```
{
  "candidate": {
    "id": 42,
    "name": "Arjun Malhotra",
    "email": "arjun@example.com"
  },
  "total_applications": 3,
  "applications": [
    {
      "application_id": 101,
      "job": {
        "id": 1,
        "role": "Python Developer",
        "company_name": "TechCorp"
      },
      "rank": 1,
      "scores": {
        "composite_score": 0.89,
        "percentage": "89.0%"
      },
      "decision": "Fast-Track Selected",
      "status": "evaluated",
      "applied_at": "2026-02-12T10:00:00"
    }
  ]
}
```

6.8 Get Candidates Dashboard

Method: GET

Endpoint: /analytics/candidates/dashboard

Description: Comprehensive candidates dashboard with all details, statistics, and tier classification

Input (Query Parameters):

```
tier_filter: string (optional) - Filter by tier: Excellent/Good/Average/Poor
status_filter: string (optional) - Filter by status: Selected/Rejected/Pending
skip: integer (optional, default: 0)
limit: integer (optional, default: 100)
```

Output:

```
{
  "statistics": {
    "total_candidates": 50,
    "by_status": {
      "selected": 20,
      "rejected": 15,
      "pending": 15
    },
    "by_tier": {
      "excellent": 8,
      "good": 18,
      "average": 14,
      "poor": 10
    },
    "by_decision": {
      "fast_track": 5,
      "selected": 15,
      "hire_pooled": 15,
      "rejected": 15,
      "review_required": 0
    },
    "average_score": 0.68
  },
  "filters_applied": {
    "tier_filter": null,
    "status_filter": null
  },
  "pagination": {
    "total": 50,
    "showing": 50,
    "skip": 0,
    "limit": 100
  },
  "candidates": [
    {
      "candidate_id": 42,
      "candidate_name": "Arjun Malhotra",
      "email": "arjun@example.com",
      "mobile": "1234567890",
      "experience": 3,
      "total_applications": 2,
      "best_application": {
        "application_id": 101,
        "job_role": "Python Developer",
        "company_name": "TechCorp",
        "company_id": 1,
        "applied_date": "2026-02-12T10:00:00"
      },
      "scores": {
        "composite_score": 0.89,
        "percentage": "89.0%",
        "rfs": 0.90,
        "dcs": 0.85,
        "elc": 1.0
      },
      "decision": "Fast-Track Selected",
      "status": "Selected",
      "tier": "Excellent",
      "rank": 1,
      "fraud_flag": false
    }
  ]
}
```


Tier Classification:

- **Excellent**: Score ≥ 0.85 (85%)

- **Good**: Score ≥ 0.70 (70%)

- **Average**: Score ≥ 0.50 (50%)

- **Poor**: Score < 0.50 (50%)

Python Example:

```
import requests

# Get all candidates
response = requests.get(
    "https://agentic-v2-0.onrender.com/analytics/candidates/dashboard"
)

# Filter by tier
response = requests.get(
    "https://agentic-v2-0.onrender.com/analytics/candidates/dashboard",
    params={"tier_filter": "Excellent"}
)

# Filter by status
response = requests.get(
    "https://agentic-v2-0.onrender.com/analytics/candidates/dashboard",
    params={"status_filter": "Selected", "limit": 20}
)

print(response.json())
```

Summary by Category

Total Endpoints: 30

| Job | 5 | `POST /job/create-with-company` ■, `POST /job/`, `GET /job/{id}`, `GET /job/`, `GET

| Application | 4 | `POST /apply/{company_id}`, `GET /apply/{id}`, `GET /apply/{id}/history`,

| Candidate | 5 | `GET /candidate/{id}`, `GET /candidate/{id}/applications`, `GET /candidate/{id}/history`, `GET /candidate/search/by-email`, `GET /candidate/` |

| Analytics | 8 | XAI, Skill Gap, Skill Graph, Rankings, Top Candidates, Statistics, Candidate Applications, **Candidates Dashboard** ■ |

Common Response Codes

500	Server Error	Internal error

Key Features

1. **Resume Parsing**: Automatic PDF parsing and text extraction
2. **Skill Extraction**: NLP-based technical and soft skill detection
3. **AI Evaluation**: Multi-score evaluation (RFS, DCS, ELC, Composite)
4. **Fraud Detection**: Duplicate resume and email detection
5. **XAI Explainability**: Transparent decision explanations
6. **Skill Gap Analysis**: Learning roadmaps and recommendations
7. **Ranking System**: Automatic candidate ranking per job
8. **Analytics Dashboard**: Comprehensive statistics and insights
9. **Audit Trail**: Complete history logging
10. **Filtering**: Advanced filtering on jobs and applications

Authentication

Currently: **None** (Open API)

Note: In production, add authentication middleware (JWT, OAuth2, etc.)

Rate Limiting

Currently: **None**

Recommended: Implement rate limiting for production (e.g., 100 requests/minute)

Testing the API

1. **Using Swagger UI**: Visit https://agentic-v2-0.onrender.com/docs
2. **Using cURL**: See example commands in each endpoint section
3. **Using Postman**: Import OpenAPI spec from `/openapi.json`
4. **Using Python requests**:

```
import requests

# Create company
response = requests.post(
    "https://agentic-v2-0.onrender.com/company/",
    params={"name": "TechCorp", "description": "Tech company"}
)
print(response.json())

# Create job
with open("job_description.pdf", "rb") as f:
    response = requests.post(
        "https://agentic-v2-0.onrender.com/job/",
        data={
            "company_id": 1,
            "role": "Python Developer",
            "location": "Remote",
            "salary": "$80k-$120k",
            "employment_type": "Full-time",
            "required_experience": 0
        },
        files={"jd_pdf": f}
    )
print(response.json())

# Apply for job
with open("resume.pdf", "rb") as f:
    response = requests.post(
        "https://agentic-v2-0.onrender.com/apply/1",
        data={
            "name": "Arjun Malhotra",
            "email": "arjun@example.com",
            "mobile": "1234567890",
            "experience": 2
        },
        files={"resume_pdf": f}
    )
print(response.json())
```

Contact & Support

For issues or questions, refer to the project README or contact the development team.