API Documentation - Match Endpoints

Overview

The Match API allows users to manage matches between CVs and job descriptions. The system supports matching CVs against job postings and vice versa.

Base URL: `/api/matches`

Endpoints

Get a Specific Match

Retrieves details for a specific match by its ID.

URL: `GET /api/matches/{id}`

URL Parameters:

- `id` (required): The ID of the match to retrieve

Authentication Required: Yes

Response Codes:

- `200 OK`: Match found and returned successfully
- `404 Not Found`: Match not found

Response Body Example (200 OK):

```json
{
  "id": 1,
  "cvId": 123,
  "jdId": 456,
  "candidateName": "John Doe",
  "position": "Java Developer",
  "score": 8.5,
  "matchDate": "2025-04-27T14:30:00",
  "userId": 789,
  "userEmail": "user@example.com",
  "missingSkills": ["Kafka", "GraphQL"],
  "matchingSkills": ["Java", "Spring Boot", "REST API"],
  "reasoning": "The candidate has strong Java skills but lacks experience with Kafka a
}
```

Get All Matches for Current User

Retrieves all matches associated with the currently authenticated user.

URL: `GET /api/matches`

Authentication Required: Yes

Response Codes:

- `200 OK`: Matches retrieved successfully
- `404 Not Found`: No matches found or error occurred

Response Body Example (200 OK):

```json
[
  {
    "id": 1,
    "cvId": 123,
    "jdId": 456,
    "candidateName": "John Doe",
    "position": "Java Developer",
    "score": 0.85,
    "matchDate": "2025-04-27T14:30:00",
    "userId": 789,
    "userEmail": "user@example.com",
    "missingSkills": ["Kafka", "GraphQL"],
    "matchingSkills": ["Java", "Spring Boot", "REST API"],
    "reasoning": "The candidate has strong Java skills but lacks experience with Kafka
  },
  {
    "id": 2,
    "cvId": 124,
    "jdId": 457,
    "candidateName": "Jane Smith",
    "position": "Data Scientist",
    "score": 9.2,
    "matchDate": "2025-04-27T15:45:00",
    "userId": 789,
    "userEmail": "user@example.com",
    "missingSkills": ["PyTorch", "Computer Vision"],
    "matchingSkills": ["Python", "Data Analysis", "Machine Learning", "TensorFlow"],
    "reasoning": "The candidate has excellent machine learning skills but lacks experi
  }
]
```

Match CV Against Jobs

Finds job matches for a specific CV with weighted skills input.

URL: `POST /api/matches/cvs/{cvId}`

URL Parameters:

- `cvId` (required): The ID of the CV to match

Authentication Required: Yes

Request Body:

```
json
```

```
[
  {
    "skill": "Java",
    "weight": 0.5
  },
  {
    "skill": "Spring Boot",
    "weight": 0.3
  },
  {
    "skill": "SQL",
    "weight": 0.2
  }
]
```

Important Note: The sum of all weights must equal 1.0

Response Codes:

- `200 OK`: Matching process completed successfully
- `400 Bad Request`: Invalid input (e.g., weights don't sum to 1.0)
- `500 Internal Server Error`: Server-side error during matching process

Response Body Example (200 OK):

```
"Match done for cv with id: 123"
```

Response Body Example (400 Bad Request):

```
"Sum of weights must be equal to 1"
```

Match Job Description Against CVs

Finds CV matches for a specific job description.

URL: `POST /api/matches/jds/{jdId}`

URL Parameters:

- `jdId` (required): The ID of the job description to match

Authentication Required: Yes

Response Codes:

- `200 OK`: Matching process completed successfully
- `404 Not Found`: Job description not found or error occurred

Response Body Example (200 OK):

```
"Match done for jd with id: 456"
```

Delete All Matches

Deletes all matches associated with the currently authenticated user.

URL: `DELETE /api/matches`

Authentication Required: Yes

Response Codes:

- `200 OK`: All matches deleted successfully
- `403 Forbidden`: User not authorized to delete matches
- `500 Internal Server Error`: Server error during deletion process

Response Body Example (200 OK):

```
"All matches are deleted successfully!"
```

Data Models

InputDTO

Used for providing weighted skills for CV matching.

```
java
```

```
{
  "skill": "String",  // Skill name
  "weight": Double    // Weight value (between 0 and 1)
}
```

MatchDTO

Represents a match between a CV and a job description.

java

```
{
  "id": Long,                    // Match ID
  "cvId": Long,                  // CV ID
  "jdId": Long,                  // Job Description ID
  "candidateName": "String",     // Name of the candidate
  "position": "String",          // Job position
  "score": Double,               // Match score (between 0 and 1)
  "matchDate": "DateTime",       // Date and time when the match was created
  "userId": Long,                // User ID
  "userEmail": "String",         // User email
  "missingSkills": ["String"],   // Array of missing skills
  "matchingSkills": ["String"],  // Array of matching skills
  "reasoning": "String"          // Text explaining the match reasoning
}
```

Error Handling

The API uses standard HTTP status codes to indicate success or failure:

- `200 OK`: Request succeeded
- `400 Bad Request`: Invalid input data
- `403 Forbidden`: Insufficient permissions
- `404 Not Found`: Resource not found
- `500 Internal Server Error`: Server error

Error responses include a message explaining the error.