Complete Matching Flow:

Skill Extraction (already covered)

- Resume and JD text → 4 parallel LLM calls → Extracted skills, education, certifications

Before matching, all skill names are normalized using SkillMatcher.normalize_skill_name():

Normalization Rules:
- Lowercase everything: "Python" → "python"
- Remove .js/.jsx/.ts/.tsx suffixes: "React.js" → "react"
- Replace hyphens/underscores with spaces: "C++" → "c++"
- Remove version numbers: "Python 3.9" → "python"
- Remove parenthetical content: "Python (3.9)" → "python"
- Remove common prefixes: "Proficient in Python" → "python"
- Remove common suffixes: "Python experience" → "python"

Example: "React.js 18.2.0" → "react"

Skill Matching:

The system uses a cascading matching strategy (tries each in order):

Match Type 1: Exact Match (Priority: 5)

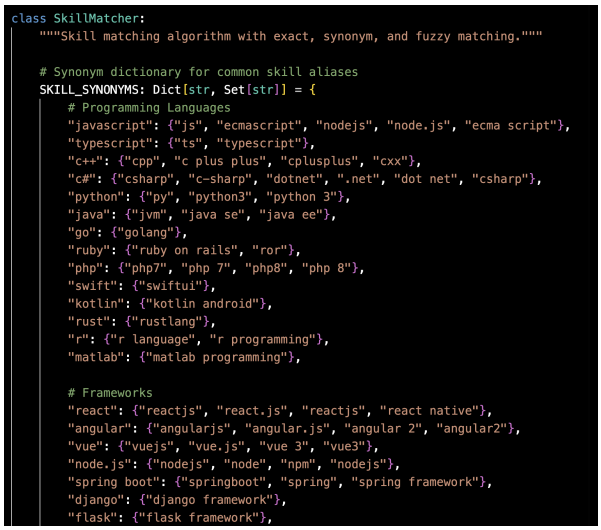
- Normalised names are identical
- Example: "Python" matches "python"

Purpose: handles typos, variations, and abbreviations

Match Type 2: Synonym Match (Priority: 4)

- Uses a synonym dictionary with 90+ entries
- Examples:
  - "JavaScript" ↔ "JS" ↔ "Node.js"
  - "AWS" ↔ "Amazon Web Services"
  - "PostgreSQL" ↔ "Postgres"

- "CI/CD" ↔ "Continuous Integration"



```
class SkillMatcher:
    """Skill matching algorithm with exact, synonym, and fuzzy matching."""

    # Synonym dictionary for common skill aliases
    SKILL_SYNONYMS: Dict[str, Set[str]] = {
        # Programming Languages
        "javascript": {"js", "ecmascript", "nodejs", "node.js", "ecma script"},
        "typescript": {"ts", "typescript"},
        "c++": {"cpp", "c plus plus", "cplusplus", "cxx"},
        "c#": {"csharp", "c-sharp", "dotnet", ".net", "dot net", "csharp"},
        "python": {"py", "python3", "python 3"},
        "java": {"jvm", "java se", "java ee"},
        "go": {"golang"},
        "ruby": {"ruby on rails", "ror"},
        "php": {"php7", "php 7", "php8", "php 8"},
        "swift": {"swiftui"},
        "kotlin": {"kotlin android"},
        "rust": {"rustlang"},
        "r": {"r language", "r programming"},
        "matlab": {"matlab programming"},

        # Frameworks
        "react": {"reactjs", "react.js", "reactjs", "react native"},
        "angular": {"angularjs", "angular.js", "angular 2", "angular2"},
        "vue": {"vuejs", "vue.js", "vue 3", "vue3"},
        "node.js": {"nodejs", "node", "npm", "nodejs"},
        "spring boot": {"springboot", "spring", "spring framework"},
        "django": {"django framework"},
        "flask": {"flask framework"},
```

```
    # Frameworks
    "react": {"reactjs", "react.js", "reactjs", "react native"},
    "angular": {"angularjs", "angular.js", "angular 2", "angular2"},
    "vue": {"vuejs", "vue.js", "vue 3", "vue3"},
    "node.js": {"nodejs", "node", "npm", "nodejs"},
    "spring boot": {"springboot", "spring", "spring framework"},
    "django": {"django framework"},
    "flask": {"flask framework"},
    "express": {"express.js", "expressjs"},
    "next.js": {"nextjs", "next"},
    "nuxt": {"nuxt.js", "nuxtjs"},
    "laravel": {"laravel framework"},
    "symfony": {"symfony framework"},

    # Tools & Platforms
    "aws": {"amazon web services", "amazon aws", "aws cloud"},
    "azure": {"microsoft azure", "azure cloud"},
    "gcp": {"google cloud", "google cloud platform", "gcp cloud"},
    "kubernetes": {"k8s", "kubernetes cluster"},
    "docker": {"docker container", "docker compose"},
    "git": {"git version control", "git scm"},
    "jenkins": {"jenkins ci", "jenkins pipeline"},
    "terraform": {"terraform iac"},
    "ansible": {"ansible automation"},
    "ci/cd": {"cicd", "continuous integration", "continuous deployment", "ci cd
    "github": {"github actions", "github ci"},
    "gitlab": {"gitlab ci", "gitlab pipeline"},
```

Match Type 3: Fuzzy Match (Priority: 3) - Typos

compute string similarity
Compares normalised skill names character-by-character
Returns a similarity score between 0.0 and 1.0

- Uses Levenshtein distance (string similarity)
- Threshold: 75% similarity
- Example:

"react" vs "reakt": ~80% similarity → matches

"python" vs "pythn": ~83% similarity → matches

Match Type 4: Cross-Category Match (Priority: 2)

Handles cases where the LLM categorised the same skill differently

- Same normalized name but different categories
- Handles cases where the LLM categorized the same skill differently
- Example: "Docker" in tools_platforms matches "Docker" in devops

When it's used

- LLM categorised "Docker" as tools_platforms in resume but devops in the JD
- Prevents false negatives when the skill name is the same

Match Type 5: Category Match (Priority: 1)

- Same category + ≥60% similarity
- Example: "PostgreSQL" and "MySQL" both in databases with similarity ≥60%