✅Updated Matching Logic – Focused on Technologies Only

---

1. Extracted Texts

We already get:

- `resume_text` (from resume file)
- `job_text` (from job description file)

Example:

```
resume_text = "Python developer with experience in Flask and SQL databases"
job_text = "We need a Python Developer with knowledge of Flask, APIs, and SQL"
```

---

2. Focus on Predefined Technology List

Instead of checking every word, we only care about **technologies/skills**.

✔️Create a **predefined technology keywords list** (can be expanded):

```
TECH_KEYWORDS = [
    'python', 'java', 'c++', 'flask', 'django', 'react', 'nodejs', 'sql',
'mongodb',
    'apis', 'machine learning', 'deep learning', 'html', 'css', 'javascript',
'aws', 'docker'
]
```

---

3. Extract Only Relevant Technologies

✔️Convert text to lowercase. ✔️Check only for words that are in `TECH_KEYWORDS`.

```
resume_text = resume_text.lower()
job_text = job_text.lower()

resume_tech = {tech for tech in TECH_KEYWORDS if tech in resume_text}
job_tech = {tech for tech in TECH_KEYWORDS if tech in job_text}
```

Example Output:

```
resume_tech → {'python', 'flask', 'sql'}
job_tech → {'python', 'flask', 'apis', 'sql'}
```

---

4. Find Matched & Missing Technologies

✔️Simple set operations:

```
matched_tech = resume_tech & job_tech
missing_tech = job_tech - resume_tech
```

Output:

```
matched_tech → {'python', 'flask', 'sql'}
missing_tech → {'apis'}
```

---

5. Calculate Technology Match Percentage

✔️Based only on technologies:

```
match_percent = (len(matched_tech) / len(job_tech)) * 100 if job_tech else 0
match_percent = round(match_percent, 2)
```

Example:

```
Match % → 75.0 %
```

---

6. Flask Integration

- Replace the old matching function with this new logic.
- Show only `missing_tech` and `match_percent` on `result.html`.

✅ **Advantages:** ✔️ Ignores useless words (like "strong" or "problem-solving"). ✔️ Focuses on relevant technical skills. ✔️ Better representation for hiring/recommendation system.

Do you want me to now **rewrite your Flask app.py** with this new technology-based matching logic? Or should I first give you a **standalone test script** to verify it?