Algorithm Breakdown

The algorithm is divided into five main steps:

1. Data Input and Preprocessing

- **User Input Data**:
    - Each user fills in their profile with:
        - Demographic details: Age, gender, location, etc.
        - Interests: Hobbies, favorite activities, or topics.
        - Club memberships: Professional groups, sports clubs, or affiliations.
        - Behavior data: Likes/dislikes captured during app usage.
- **Data Processing**:
    - Convert text-based inputs (e.g., bio, interests) into machine-readable formats using **natural language processing (NLP)** (e.g., TF-IDF vectorization).
    - Structure all data into a vector format to enable similarity computation.

---

2. Finding Similar Profiles

This step identifies the three profiles most similar to Person A.

- **Similarity Computation**:
    - Use a weighted combination of similarity metrics to calculate closeness between Person A and other users.
        1. **Interests Similarity**:
            - Use a **cosine similarity** score between user vectors representing their interests.
        2. **Demographic Similarity**:
            - Match based on overlapping demographic categories like age group, location, etc.
        3. **Club/Organization Affiliation**:
            - Check for shared affiliations or memberships.
- **Randomization**:
    - From the top 10 similar users, randomly shuffle to select three profiles.
    - Ensure the three selected profiles:

        - Include at least one highly similar user.
        - Exclude users already matched or interacted with.
- **Output**:
    - Return a list of three users similar to Person A.

---

3. Gamified Profile Display

This step presents the profiles to Person B in a way that maintains mystery.

- **Profile Display Rules**:
    - Show Person B **four profiles**:
        - Person A (the one who liked them).
        - Three other similar users.
    - Randomize the display order to prevent Person B from guessing who liked them.
- **User Interaction**:
    - Person B interacts with each profile by choosing to "like" or "dislike."
    - Store these interactions for processing in the next step.

---

4. Match Confirmation

This step determines whether a mutual match occurs.

- **Logic**:
    1. If Person B "likes" Person A during the gamified interaction:
        - A mutual match is confirmed.
        - Notify both users of the match.
    2. If Person B doesn't like Person A:
        - No match is made, and Person A remains unaware.
    3. The process continues with Person A and others until a match is found.

---

5. Continuous Personalization

To improve recommendations over time, the algorithm learns from user interactions.

- **Machine Learning Personalization**:

    - Track patterns in Person B's likes/dislikes.
    - Update Person B's profile vector dynamically based on:
        - Common traits of liked profiles.
        - Traits of profiles they interact with the most.
- **Adaptation Over Time**:
    - As the user interacts with the app, the recommendation system becomes more accurate, increasing the chances of successful matches.

App.py

```
from flask import Flask, request, jsonify, render_template
import random
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

app = Flask(__name__)

# Step 1: Demo Profiles
profiles = [
    {"id": 1, "name": "Alice", "age": 25, "interests": [1, 0, 1, 1, 0], "affiliations": [1, 0, 0]},
    {"id": 2, "name": "Bob", "age": 27, "interests": [0, 1, 1, 0, 1], "affiliations": [0, 1, 0]},
    {"id": 3, "name": "Carol", "age": 23, "interests": [1, 1, 0, 0, 1], "affiliations": [0, 0, 1]},
    {"id": 4, "name": "David", "age": 26, "interests": [1, 0, 1, 1, 1], "affiliations": [1, 0, 0]},
```

{"id": 5, "name": "Eve", "age": 22, "interests": [0, 1, 0, 1, 0], "affiliations": [0, 1, 1]},
{"id": 6, "name": "Frank", "age": 28, "interests": [1, 1, 0, 1, 0], "affiliations": [1, 0, 1]},
{"id": 7, "name": "Grace", "age": 24, "interests": [0, 0, 1, 1, 1], "affiliations": [0, 1, 0]},
{"id": 8, "name": "Hank", "age": 25, "interests": [1, 1, 1, 0, 0], "affiliations": [1, 0, 0]},
{"id": 9, "name": "Ivy", "age": 23, "interests": [0, 1, 1, 1, 0], "affiliations": [0, 1, 1]},
{"id": 10, "name": "Jack", "age": 27, "interests": [1, 0, 1, 0, 1], "affiliations": [1, 0, 0]},
{"id": 11, "name": "Liam", "age": 30, "interests": [1, 1, 0, 0, 1], "affiliations": [0, 0, 0]},
{"id": 12, "name": "Mia", "age": 22, "interests": [0, 1, 1, 1, 0], "affiliations": [0, 1, 1]},
{"id": 13, "name": "Nathan", "age": 25, "interests": [1, 0, 1, 1, 1], "affiliations": [1, 0, 0]},
{"id": 14, "name": "Olivia", "age": 27, "interests": [0, 1, 0, 1, 0], "affiliations": [1, 1, 0]},
{"id": 15, "name": "Paul", "age": 29, "interests": [1, 0, 1, 0, 1], "affiliations": [0, 1, 0]},
{"id": 16, "name": "Quinn", "age": 23, "interests": [0, 1, 1, 0, 0], "affiliations": [0, 0, 1]},
{"id": 17, "name": "Rachel", "age": 24, "interests": [1, 0, 1, 1, 0], "affiliations": [1, 1, 0]},
{"id": 18, "name": "Sam", "age": 26, "interests": [0, 0, 0, 1, 1], "affiliations": [1, 0, 1]},
{"id": 19, "name": "Tina", "age": 22, "interests": [1, 1, 0, 1, 0], "affiliations": [0, 1, 1]},
{"id": 20, "name": "Ursula", "age": 28, "interests": [0, 0, 1, 1, 1], "affiliations": [1, 0, 0]},
{"id": 21, "name": "Vince", "age": 29, "interests": [1, 1, 0, 0, 1], "affiliations": [0, 1, 0]},
{"id": 22, "name": "Wendy", "age": 26, "interests": [1, 1, 1, 0, 0], "affiliations": [0, 0, 1]},
{"id": 23, "name": "Xander", "age": 31, "interests": [1, 0, 1, 1, 0], "affiliations": [1, 1, 0]},
{"id": 24, "name": "Yara", "age": 27, "interests": [0, 1, 1, 0, 0], "affiliations": [0, 1, 1]},
{"id": 25, "name": "Zane", "age": 23, "interests": [1, 0, 1, 0, 1], "affiliations": [1, 0, 1]},
{"id": 26, "name": "Aiden", "age": 24, "interests": [0, 1, 0, 1, 0], "affiliations": [0, 1, 1]},
{"id": 27, "name": "Bella", "age": 28, "interests": [1, 0, 1, 1, 1], "affiliations": [0, 1, 0]},
{"id": 28, "name": "Caden", "age": 23, "interests": [0, 0, 0, 1, 1], "affiliations": [1, 0, 0]},
{"id": 29, "name": "Dylan", "age": 26, "interests": [1, 0, 0, 1, 0], "affiliations": [1, 1, 1]},
{"id": 30, "name": "Ethan", "age": 29, "interests": [0, 1, 1, 1, 0], "affiliations": [1, 0, 0]},
{"id": 31, "name": "Fiona", "age": 24, "interests": [1, 1, 1, 0, 1], "affiliations": [0, 1, 0]},
{"id": 32, "name": "Gabe", "age": 27, "interests": [0, 0, 1, 1, 1], "affiliations": [0, 1, 1]},
{"id": 33, "name": "Holly", "age": 28, "interests": [1, 1, 0, 0, 0], "affiliations": [0, 1, 0]},

```
    {"id": 34, "name": "Iris", "age": 23, "interests": [1, 0, 1, 1, 0], "affiliations": [1, 0, 1]},
    {"id": 35, "name": "Jake", "age": 25, "interests": [0, 1, 0, 1, 1], "affiliations": [0, 0, 1]},
    {"id": 36, "name": "Kara", "age": 22, "interests": [1, 0, 0, 0, 1], "affiliations": [1, 0, 0]},
    {"id": 37, "name": "Luca", "age": 27, "interests": [0, 1, 1, 1, 0], "affiliations": [0, 1, 0]},
    {"id": 38, "name": "Maya", "age": 24, "interests": [1, 1, 0, 1, 0], "affiliations": [0, 1, 1]},
    {"id": 39, "name": "Noah", "age": 28, "interests": [0, 0, 1, 1, 1], "affiliations": [1, 0, 1]},
    {"id": 40, "name": "Olga", "age": 30, "interests": [1, 1, 0, 0, 0], "affiliations": [1, 0, 0]}
]

# Step 2: Function to calculate similarity scores
def calculate_similarity(profile1, profile2):
    # Cosine similarity of interests and affiliations
    interest_similarity = cosine_similarity([profile1['interests']], [profile2['interests']])[0][0]
    affiliation_similarity = cosine_similarity([profile1['affiliations']], [profile2['affiliations']])[0][0]

    # Combine similarities (You can adjust the weight of each if desired)
    total_similarity = (interest_similarity + affiliation_similarity) / 2
    return total_similarity

# Step 3: Function to find the best match for a given user
def find_best_match(user_profile):
    best_match = None
    best_score = -1

    for profile in profiles:
        if profile['id'] != user_profile['id']:  # Don't compare with itself
            score = calculate_similarity(user_profile, profile)
            if score > best_score:
                best_score = score
```

```
            best_match = profile

    return best_match, best_score

# Step 4: API endpoint to match profiles
@app.route('/match', methods=['POST'])
def match_profiles():
    user_data = request.get_json()
    user_profile = {
        "id": random.randint(1000, 9999),  # Random ID for the user
        "name": user_data['name'],
        "age": user_data['age'],
        "interests": user_data['interests'],
        "affiliations": user_data['affiliations']
    }

    best_match, best_score = find_best_match(user_profile)

    if best_match:
        return jsonify({
            'message': 'Best match found!',
            'user': user_profile,
            'best_match': best_match,
            'similarity_score': best_score
        })
    else:
        return jsonify({'message': 'No match found!'})

# Step 5: Home Route
```

```
@app.route('/')
def home():
    return render_template('index.html')


if __name__ == "__main__":
    app.run(debug=True, port=8080)
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Dating App</title>
    <link rel="stylesheet" href="/static/styles.css">
</head>
<body>
    <h1>Welcome to the Dating App</h1>
    <div id="profiles"></div>
    <script>
        fetch('/api/get_profiles')
            .then(response => response.json())
            .then(data => {
                const profilesDiv = document.getElementById('profiles');
                data.slice(0, 4).forEach(profile => {
                    const profileCard = document.createElement('div');
                    profileCard.innerHTML = `
                        <h3>${profile.name}, ${profile.age}</h3>
                        <button onclick="likeProfile(${profile.id})">Like</button>
                    `;
                    profilesDiv.appendChild(profileCard);
                });
            });

        function likeProfile(id) {
            fetch('/api/get_similar_profiles', {
                method: 'POST',
```

```
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ liked_profile_id: id })
        })
        .then(response => response.json())
        .then(data => {
            window.location.href = '/similar';
            localStorage.setItem('similarProfiles', JSON.stringify(data));
        });
    }
  </script>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Similar Profiles</title>
    <link rel="stylesheet" href="/static/styles.css">
</head>
<body>
    <h1>Similar Profiles</h1>
    <div id="similarProfiles"></div>
    <script>
        const similarProfiles = JSON.parse(localStorage.getItem('similarProfiles'));
        const similarProfilesDiv = document.getElementById('similarProfiles');

        similarProfiles.forEach(profile => {
            const profileCard = document.createElement('div');
            profileCard.innerHTML = `<h3>${profile.name}, ${profile.age}</h3>`;
            similarProfilesDiv.appendChild(profileCard);
        });
    </script>
</body>
</html>
```

```
body {
    font-family: Arial, sans-serif;
    text-align: center;
}
div {
    margin: 10px auto;
    border: 1px solid #ccc;
    padding: 10px;
    width: 300px;
}
button {
    padding: 5px 10px;
    background-color: #007bff;
    color: white;
    border: none;
    cursor: pointer;
}
```