Mahdi Mirzahosseinian

Email: mahdi.mirzahosseinian@gmail.com

Approach

1. Sign-Up Process:
    - When a user signs up, they provide a list of attributes (e.g., profession, location, skills).
    - These attributes are serialized (sorted and joined into a string) to make it easier to compare them with other users' attributes.
    - The system checks the existing groups (in the `groups` table) to see if there is any group that shares any subset of attributes with the new user's attributes.
    - If a matching group is found (using simple matching logic where any shared attribute between the user's attributes and the group's attributes is considered a match), the user is assigned to that group.
    - If no matching group is found, a new group is created, and the user is added to it.
2. Sign-In Process:
    - The user provides their user ID for sign-in.
    - The system authenticates the user by checking if the user ID exists in the `users` table.
    - If the user is found, a success message and the `user_id` are returned.
3. Group Retrieval:
    - The user can see which group they belong to by providing their `user_id`.
    - The system queries the `users` table to fetch the user's group ID.
    - Using the `group_id`, it retrieves the full group of users from the `groups` table and returns all members in the same group.

Explanation of the Solution

1. **Database Structure:**
    - `users` **Table**:
        - Stores each user's ID, their attributes (as a serialized string), and the ID of the group they belong to.
    - `groups` **Table**:
        - Stores each group's unique ID and the **attributes** that define the group (stored as a serialized string of attributes).
        - A group is defined by a set of attributes that have a match with users' attributes.
2. **Grouping Logic**:
    - The core challenge here is identifying users with **matching attributes**. To achieve this, the attributes are serialized into a sorted string and checked against existing groups. If any attribute in the new user's set matches a group's attributes (even a subset), the user is considered part of that group.
3. **Scalability**:
    - The system is designed to handle a large number of users and groups. However, the current logic for matching attributes is simple and may not scale efficiently for millions of users. Optimizations such as **caching**, **queueing**, or more advanced matching algorithms (e.g., set theory, machine learning-based similarity) could be introduced to improve performance.

User Stories

User Story 1: Sign-Up

As a new user
I want to provide my attributes during sign-up
So that I can be assigned to a group with similar users.

Acceptance Criteria:

- The user should be able to sign up by providing their attributes.
- The user should receive a unique `user_id` and `group_id` upon successful sign-up.

Testing Scenarios:

- **Scenario 1**: Sign up with valid attributes.
  - **Input**: `["Software", "Engineer", "Brussels", "Senior"]`
  - **Expected Output**: `{"group_id": 1, "user_id": 1}`
- **Scenario 2**: Attributes matching an existing group.
  - **Input**: `["Software", "Engineer", "Brussels", "Senior"]`
  - **Expected Output**: `{"group_id": 1, "user_id": 2}`
- **Scenario 3**: Sign up with invalid data (e.g., empty attributes).
  - **Input**: `[]`
  - **Expected Output**: `400 Error: Attributes are required.`

User Story 2: Sign-In

As a returning user
I want to log in using my `user_id`
So that I can access my group information.

Acceptance Criteria:

- The system should verify the `user_id` during sign-in.
- If the `user_id` is valid, the user is successfully signed in.

Testing Scenarios:

- **Scenario 1**: Successful sign-in with a valid `user_id`.
  - **Input**: `{ "user_id": 1 }`
  - **Expected Output**: `{ "message": "Sign-in successful", "user_id": 1 }`
- **Scenario 2**: Sign-in with an invalid `user_id`.
  - **Input**: `{ "user_id": 999 }`

- **Expected Output**: `404 Error: User not found.`

User Story 3: View Group Members

As a user
I want to view all members of my group
So that I can connect with other users who share similar attributes.

Acceptance Criteria:

- The system should return a list of users in the same group as the logged-in user.
- If no group is found, the system should notify the user.

Testing Scenarios:

- **Scenario 1**: View group members for a valid user.
  - **Input**: `user_id = 1`
  - **Expected Output**: `{ "group_id": 1, "members": [{"id": 1, "attributes": ["Software", "Engineer", "Brussels", "Senior"]}, {"id": 2, "attributes": ["Software", "Engineer", "Brussels", "Senior"]}] }`
- **Scenario 2**: User with no group members.
  - **Input**: `user_id = 3`
  - **Expected Output**: `{ "group_id": 2, "members": [{"id": 3, "attributes": ["Researcher", "Gamer", "Imec"]}] }`
- **Scenario 3**: Invalid user ID.
  - **Input**: `user_id = 999`
  - **Expected Output**: `404 Error: User not found.`

Testing Results

User Story 1: Sign-Up

- **Scenario 1**

```
$ curl -X POST http://127.0.0.1:5000/signup \
> -H "Content-Type: application/json" \
> -d '{"attributes": ["Software", "Engineer", "Brussels"]}'
{"group_id":1,"user_id":1}
$
```

- **Scenario 2**

```
$ curl -X POST http://127.0.0.1:5000/signup -H "Content-Type: application/json" -d '{"attributes": ["S
oftware", "Engineer", "Brussels"]}'
{"group_id":1,"user_id":2}
$
```

- **Scenario 3**

```
$ curl -X POST http://127.0.0.1:5000/signup -H "Content-Type: application/json" -d '{"attributes": []}
'
{"error":"Attributes are required"}
$
```

User Story 2: Sign-In

- **Scenario 1**

```
$ curl -X POST http://127.0.0.1:5000/signin \
> -H "Content-Type: application/json" \
> -d '{"user_id": 1}'
{"message":"Sign-in successful","user_id":1}
$
```

- **Scenario 2**

```
$ curl -X POST http://127.0.0.1:5000/signin -H "Content-Type: application/json" -d '{"user_id": 999}'
{"error":"User not found"}
$
```

User Story 3: View Group Members

- **Scenario 1**

```
$ curl http://127.0.0.1:5000/group/1
{"group_id":1,"members":[{"attributes":"Brussels,Engineer,Software","id":1},{"attributes":"Brussels,En
gineer,Software","id":2}]}
$
```

- **Scenario 2**

```
$ curl http://127.0.0.1:5000/group/3
{"group_id":2,"members":[{"attributes":"Gamer,Imec,Researcher","id":3}]}
$
```

- **Scenario 3**

```
$ curl http://127.0.0.1:5000/group/999
{"error":"User not found"}
$
```