Online Dating Application

Problem Definition:

You have to create an online dating application. Every active user account will have location, age and gender information. The application should show users their potential matches in order of relevance. The ordering of relevance will be following:

1. Gender : Opposite gender to be given higher priority.
2. Proximity: Nearer matches should be given more priority. Use euclidean distance for computing distance between two locations (***see appendix for euclidean distance**).
3. Age: Less the age difference should be given more priority.

Operations:

A user can perform these operations in this application:

1. Create Account: A person can create an account with interest and profile details.
2. Potential Match: Provides all the potential match of a user in relevance order.
3. Like: User can like a potential match user.
4. Show Matches: Showing the users which match against a user. A match happens when both the users have liked each other.
5. Show All Matches: Showing system view by displaying all the matches in the system.
6. Ignore: User can ignore a potential match user.
7. Delete Account: If a user deletes account, then all matches and likes will be removed.

Use case:

1. If a user **A** likes user **B**, the data should be stored for further processing.
2. All the matches(case where 2 users have liked each other) in the system should be shown.

Expectations:

1. **Code should be demo-able**. Either use a main driver program on command line or test cases.
2. **Code should be functionally correct and complete.**
3. Code should be readable, modular, testable and use proper naming conventions. It should be easy to add/remove functionality without rewriting entire codebase.
4. Code should cover all the edge cases possible and work for them or fail gracefully for errors.

Guidelines:

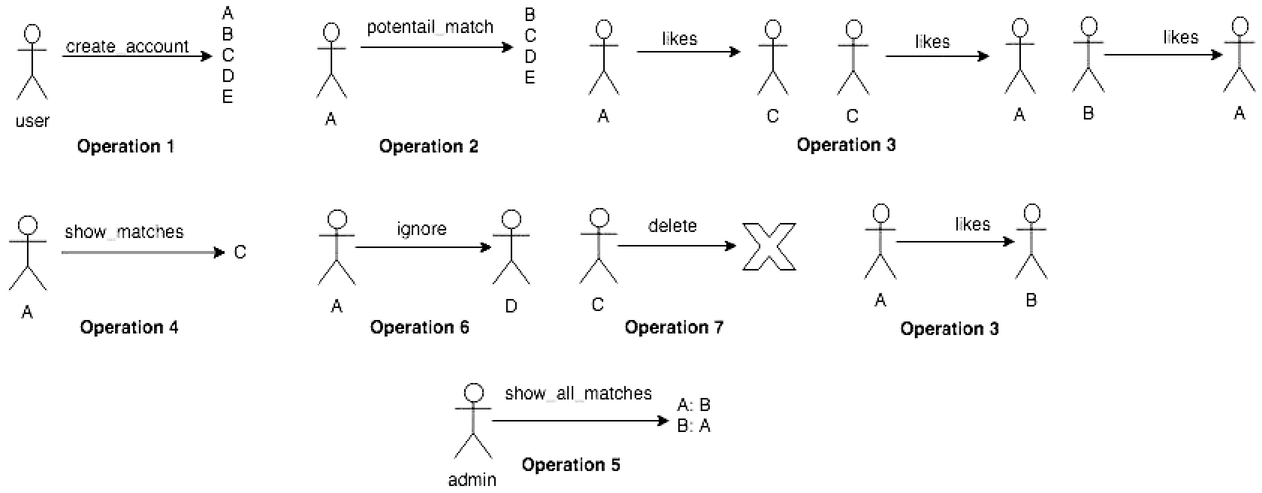
1. Use language of your choice.
2. Output can be written to a file or STDOUT.
3. Feel free to store all interim/output data in-memory.
4. Restrict internet usage to looking up syntax.
5. Please discuss any doubts you have with an interviewer.

Appendix:

The distance between two points in the plane with coordinates **(x, y)** and **(a, b)** is given by:

$$dist((x, y), (a, b)) = sqrt((x - a)^2 + (y - b)^2)$$

Sample User Data:



1. **Input format:**
    a. create_account(user_name, x_coordinate, y_coordinate, age, gender)
    b. delete_account(user_name)
    c. potential_match(user_name)
    d. like(user_name, user_name)
    e. ignore(user_name, user_name)
    f. show_matches(user_name)
    g. show_all_matches()

2. **Sample test data:**

input	output
create_account(user_A, 2, 3, 23, Male)	**true**
create_account(user_B, 3, 4, 45, Female)	**true**
create_account(user_C, 4, 6, 32, Female)	**true**
create_account(user_D, 7, 8, 23, Male)	**true**
create_account(user_E, 8, 12, 23, Female)	**true**
potential_match(user_A)	**user_B,user_C,user_E,user_D**
like(user_A, user_B)	**true**
like(user_A, user_C)	**true**
potential_match(user_B)	**user_A,user_D,user_C,user_E**

like(user_B, user_A)	**true**
show_matches(user_A)	**user_B**
potential_match(user_E)	**user_D,user_A,user_C,user_B**
like(user_E, user_D)	**true**
potential_match(user_D)	**user_E,user_A,user_C,user_B**
like(user_D, user_E)	**true**
show_all_matches()	**user_A: user_B** **user_B: user_A** **user_D: user_E** **user_E: user_D**
potential_match(user_C)	**user_A,user_D,user_B,user_E**
like(user_C, user_A)	**true**
show_all_matches()	**user_A: user_B,user_C** **user_B: user_A** **user_C: user_A** **user_D: user_E** **user_E: user_D**
delete_account(user_A)	**true**
show_all_matches()	**user_D: user_E** **user_E: user_D**