Online Aashqui Application [Low-Level Design]:

Problem Statement:

Create an online dating application. Every Active user account will have location, age and gender info. The App should show users their potential matches in order of relevance. The ordering of relevance will be following:

1. Gender : Opposite gender -> high priority.
2. Proximity: Nearer matches should be given more priority. Use following formula for computing distance between two locations (***dist((ax, y), (a, b)) = sqrt($(x - a)^2$ + $(y - b)^2$)**).
3. Age: Less age difference -> high 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.

Guidelines:

- Input can be read from a file or STDIN or coded in a driver method.
- Output can be written to a file or STDOUT.
- Feel free to store all interim/output data in-memory.
- Restrict internet usage to looking up syntax
- You are free to use the language of your choice.
- Save your code/project by your name and email it. Your program will be executed on another machine. So, explicitly specify dependencies, if any, in your email.

Expectations:

- Code should be demo-able (very important). Code should be functionally correct and complete.
  - At the end of this interview round, an interviewer will provide multiple inputs to your program for which it is expected to work
- Code should handle edge cases properly and fail gracefully. Add suitable exception handling, wherever applicable.
- Code should have good object-oriented design.
- Code should be readable, modular, testable and extensible. Use intuitive names for your variables, methods and classes.
  - It should be easy to add/remove functionality without rewriting a lot of code.
  - Do not write monolithic code.
- Don’t use any databases.

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()