Project Specifications: Matching App

Overview: The goal of this project is to implement a basic (tinder-like) matching app using Python. Students will create a command-line application that allows users to create profiles, browse other profiles, like/dislike users, and find matches. The project will cover essential programming concepts including data structures, object-oriented programming (OOP), vectorization techniques, and database integration. Bonus points will be given for creative graphical UI and exceptional matching algorithms. **The grades for projects will be relative among the groups and a ChatGPT solution will serve as a poor baseline reference point.**

Project Requirements (Functionalities)

1. User Profile Management

- **Attributes:** Each user profile should include the following attributes:
    - user_id (unique identifier)
    - name
    - age
    - gender
    - location
    - interests (list of interests)
- **Actions:**
    - Create a new user profile
    - View existing user profiles
    - Edit user profile details
    - Delete user profile

2. User Interaction

- **Attributes:** Track user interactions using the following attributes:
    - liked_users (list of user IDs that the user has liked)
    - disliked_users (list of user IDs that the user has disliked)
    - matches (list of user IDs that are mutual likes – sophisticated matching algorithm will be required)
- **Actions:**
    - Like a user profile
    - Dislike a user profile

- View matches (based on search**, should remember likes and dislikes**)

3. Data Storage and Management

- **Data Structures:**
  - Use Python lists, tuples, and dictionaries to manage user profiles and interactions.
- **Database Integration:**
  - Implement a SQLite database to store user profiles and interactions.
  - Perform CRUD operations (Create, Read, Update, Delete) using SQL queries.
  - Use SQLite with Python's sqlite3 library for database interactions.

4. Vectorization and Data Processing

- **Using Pandas and NumPy:**
  - Implement vectorized functions (via NumPy or Pandas) to process user data and find potential matches based on interests and location.

5. Command-Line Interface (CLI)

- **User Commands:**
  - create_user: Create a new user profile
  - view_profiles: View all user profiles
  - edit_profile: Edit an existing user profile
  - delete_profile: Delete a user profile
  - like_user: Like a user profile
  - dislike_user: Dislike a user profile
  - view_matches: View matched user profiles

Project Milestones

Meeting 1: User Profile Management and Basic Data Structures

- **Objective:** Introduce essential data structures and set the groundwork for the project.
- **Tasks (start in class as Ex1 and continue after class):**
  - Implement user profile creation, viewing, editing, and deletion.
  - Use lists, tuples, and dictionaries to manage user data.

Meeting 2: Advanced Data Structures, OOP, and User Interaction

- **Tasks (start in class as Ex2 and continue after class):**

- Define a User class with necessary attributes and methods (extension of Ex1).
- Implement user actions (like, dislike, view matches) within the User class.

Meeting 3: Vectorization, Data Management, and Database Integration

- **Objective:** Focus on managing user data, implementing vectorization techniques.
- **Tasks (start in class as Ex3 and continue after class):**
  - Use Pandas DataFrames and NumPy for efficient data processing.
  - Implement vectorized function(s) for finding matches.
  - Integrate SQLite for persistent data storage.
  - Create a UI (Graphical gets you bonus).

Deliverables

1. **Source Code:** A fully functional command-line application that meets the project requirements. A creative GUI will yield a bonus of 5%.
2. **Documentation:** Clear and concise documentation explaining how to set up, run, and use the application.
3. **Presentation:** A brief presentation (10 minutes) showcasing the application, its features, and the implementation process.

Grading Criteria

1. **Functionality (50%)**: The app meets all specified requirements and performs all intended actions correctly.
2. **Code Quality (30%)**: Code is well-organized, follows best practices, and is adequately commented.
3. **Documentation (10%)**: Documentation is clear, comprehensive, and helpful.
4. **Presentation (10%)**: Presentation is clear, concise, and effectively demonstrates the application's features and implementation.

Final code submission deadline: **Friday, August 23, 2024**

Opt-out project specification:

An extended version of the basic matching app that would include the following.

Core Requirements:

- User authentication and profile creation (as in basic project).
- Dashboards and visualizations of past usage.
- Advanced matching algorithm (e.g., collaborative filtering).
- Responsive and interactive user interface (GUI) with animations.

Documentation:

- Detailed inline comments and a comprehensive README file.
- Additional documentation outlining the system architecture, design decisions, and user guide.

Testing:

- Unit tests covering core functionalities.
- Integration tests to ensure seamless operation of different components.

Presentation:

- Presentation with a live demo on Friday, August 16, 2024 (or earlier on Zoom).
- User feedback and improvements based on user testing.

Final code submission deadline: **Friday, August 23, 2024**