Marriage Matchmaking App

Brief Description

The Marriage Matchmaking App is a simple backend application designed to help users find potential matches based on their profile information. The app allows users to create, read, update, and delete profiles with details such as name, age, gender, email, city, and interests.

What is Provided?

This project provides a basic skeleton for a FastAPI-based backend application. The project can be found at this github repo: https://github.com/abhishek-UM/UrbanMatch-PythonTask/tree/master

The provided code includes:

Basic Project Structure:

- **main.py** : The main application file with basic CRUD operations for user profiles.
- **models.py**: SQLAlchemy models defining the User schema.
- **database.py**: Database configuration and setup.
- **schemas.py**: Pydantic schemas for data validation and serialization.

Functionality:

- Create User Endpoint: Create a new user profile.
- Read Users Endpoint: Retrieve a list of user profiles.
- Read User by ID Endpoint: Retrieve a user profile by ID.
- SQLite Database: The application uses SQLite as the database to store user profiles.

What is Required?

Tasks:

1. Add User Update Endpoint:
   - Implement an endpoint to update user details by ID in the main.py file.
2. Add User Deletion Endpoint:
   - Implement an endpoint to delete a user profile by ID.
3. Find Matches for a User:
   - Implement an endpoint to find potential matches for a user based on their profile information.
4. Add Email Validation:
   - Add validation to ensure the email field in user profiles contains valid email addresses.

Instructions:

Implement the required endpoints and email validation:

1. Add the necessary code for the update, delete, match and validation endpoints
2. Test Your Implementation:
    1. Verify that users can be updated and deleted correctly.
    2. Check that matches are correctly retrieved for a given user.
    3. Ensure email validation is working as expected.

Submit Your Work:

Provide the updated code files (main.py, models.py, database.py, and schemas.py). Include a brief report explaining your approach and any assumptions made.

Prerequisites

- Python 3.7+
- FastAPI
- SQLAlchemy
- SQLite