Design and Implementation of UDINDER

Nicolas Avendano Barajas - 20231020113
Distrital University Francisco Jose de Caldas
navendanob@udistrital.edu.co

Juan Sebastian Vega Diaz - 20231020087
Distrital University Francisco Jose de Caldas
jusvega@udistrital.edu.co

***Abstract*—**

I. Introduction

UDinder is a cutting-edge dating platform that leverages the power of object-oriented programming (OOP) to create a seamless and intuitive user experience. In UDinder, each user profile is represented as an object, encapsulating a wealth of information including personal interests, photos, and preferences. By utilizing OOP principles such as encapsulation, inheritance, and polymorphism, UDinder ensures efficient data management and facilitates smooth user interactions.

Encapsulation in UDinder ensures that user data is securely stored within profile objects, allowing for controlled access and manipulation. This promotes privacy and data integrity, crucial aspects of any modern dating platform. Additionally, inheritance enables the creation of specialized profile types, catering to diverse user preferences and demographics. Whether it's for casual hookups, meaningful relationships, or platonic connections, UDinder offers tailored experiences to suit every user's needs.

Polymorphism in UDinder allows for dynamic and adaptable interactions between users. By treating different profile types as instances of a common superclass, UDinder fosters a flexible matchmaking environment where compatibility is key. Whether it's through algorithmic matching or manual browsing, users can discover potential matches with ease, confident in the platform's robust architecture and intelligent algorithms.

UDinder is not just another dating app; it's a testament to the power of object-oriented programming in revolutionizing the way we connect and form relationships in the digital age. With UDinder, finding love has never been more intuitive, efficient, and fun.

II. METHODS AND MATERIALS

This project aims to develop a functional online dating platform. For this purpose, the following technical decisions will be made. First, Python version 3.12.1 will be used for the development of the logical part of the software, and the FastApi framework will also be used for communication between the different layers of the platform. For the data layer, we will use an ORM tool, in this way the SQLalchemy library is selected which will facilitate the connection of the database from the backend of the platform and thus be able to manage the database with a syntax similar to that of the python programming language. The softwares Html, CSS, and Javascript will be used for the development of the frontend, in the same way, the Apache server software will be used to deploy the application as localhost and the Django framework will be used to obtain a better graphical interface. Regarding the development process,it was decided to use the GitHub hosting software to facilitate the cooperative development and the management of its versions.

A. Class diagram decisions:

- It was decided to use a User class, because it contains the necessary attributes to identify the user, define their type of access (User or administrator) and the methods sing up and sing in.

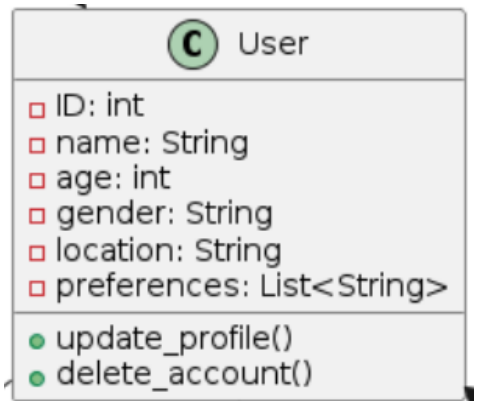

Fig. 1. User Class

Users can create and customize their profiles, including photos, bio, and preferences such as age range and location.

- InteractionProfiles:This component is responsible for suggesting user profiles for a given user. It uses the "showsSuggestedProfiles" function, which takes a user as input and displays the suggested profiles for that user. It also has a list of existing matches.

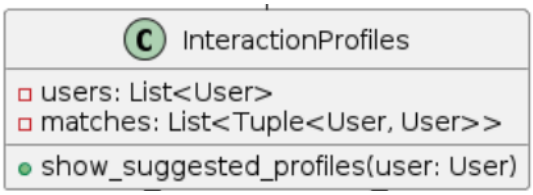

Fig. 2. InteractionProfiles Class

- MatchManager:This component is responsible for creating matches between two users. It uses the ”createMatch” function, which takes two users as input and generates a match between them. It also has a list of registered users.

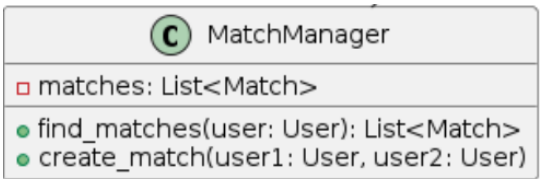

Fig. 3. MatchManager Class

- MessageManager: This component is responsible for sending messages between users. It uses the ”sendMessage” function, which takes the sender, recipient, and message as input and sends the message to the recipient. It also has the ”blockUser” function to block a user and the ”unblockUser” function to unblock a user.

Fig. 4. MessageManager Class

- Administrator: This component is responsible for managing the system. It has the ”viewExistingUsers” function that displays a list of existing users in the database. It can also send messages to other users using the ”sendMessage” function.

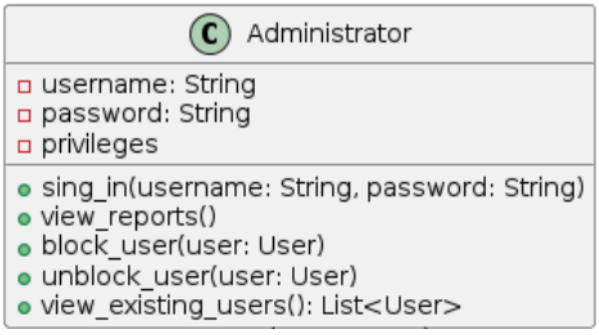

Fig. 5. Admin Class

- Profile: This component represents a user’s profile. It has properties such as photos and description. Users can complete their profile using the ”completeProfile” function.

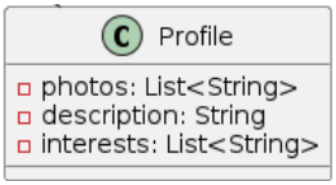

Fig. 6. Profile Class

- Database: This component stores the information of the users in the system. It has a list of registered users and provides functions to add users and retrieve users by username.

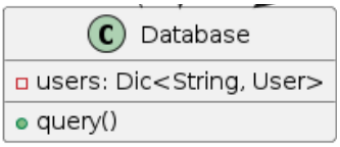

Fig. 7. DB Class

III. Testing and Evaluation

Preliminary testing of UDINDER was conducted with a group of beta testers to assess usability, performance, and user satisfaction. Feedback was collected through surveys and interviews, and initial results indicate positive reception of the app’s features and functionality.

IV. Conclusions

In conclusion, UDINDER represents a successful implementation of a dating application, offering users a platform for connecting with potential romantic partners in their vicinity. Future work will focus on refining the app’s features, optimizing performance, and expanding user base through marketing and outreach efforts.

References

[1] Tinder, ”Tinder: The World’s Most Popular Dating App”, [Online]. Available: https://tinder.com/
[2] Firebase, ”Firebase Realtime Database”, [Online]. Available: https://firebase.google.com/docs/database
[3] MongoDB, ”MongoDB: The Modern, General Purpose Database”, [Online]. Available: https://www.mongodb.com/