Wellesley College Computer Science Department
CS304, Spring 2021

Winterest: A Matching Platform

Julie Lely, Melissa Guzman, Shreya Kochar

April 2021

Abstract

Winterest serves as a means to link students together – in romantic, friendly, or professional ways. The platform matches a given user with several other people that have common interests. The matching algorithm calculates a “match score” for each person compared to all other people in the database and generates matches accordingly. This project used PyMySQL, SQL, Python Flask, HTML/CSS, and Javascript to implement a user-friendly application. Inspiration was drawn from modern social media platforms, such as Tinder, Bumble, and Linkedin.

Keywords: matching, Winterest, SQL, Python

1 Introduction

Winterest is a social media platform for Wellesley students and alumni to network. It is a mixture of Tinder and Linkedin, where users are matched on the basis of what’s important to them - interests, values, personalities, and demographic traits like being first gen, low income, or of a minority ethnic group. Our aim is to create a way for the Wellesley community to create and maintain meaningful relationships in order to intentionally overcome the isolation produced by digital-first communication and continue to provide valuable connections across existing groups going into the future.

On a global level, the coronavirus pandemic has rendered many of the ways we have been accustomed to finding and developing relationships impossible. Isolation has always been a problem at Wellesley because of the heavily academic culture and the difficulty of scheduling time with friends. For many students coming from underrepresented backgrounds, the ins and outs of networking and building relationships with the community of alumni are bewildering and difficult to navigate. Due to the pandemic, there is a pressing need for alternative methods of organic social interaction and the building of effective support networks for students.

Our project is potentially a solution to these problems. It enables students to rebuild some of the sense of community that has been lost. It enables freshmen to form communities amongst themselves and with upperclassmen, from whom they are largely separated in Wellesley's physical space. It enables connection to alumni mentors, helping students to develop their networks and learn about what life after Wellesley could look like, and providing alumni with a fulfilling way to stay connected to the school.

2 Materials and Methods

- PyMySQL – used to execute SQL code using Python; backend created through this
- Python Flask – used for routing pages and connecting the UI to the backend
- HTML/CSS/Javascript – used to build the UI
- Figma - used for initial design layout

3 Design

The entity relation diagram in Figure 1 sums up the Winterest backend design. The Winterest application has several different regions that will be discussed in depth in our paper. There are 3 different aspects of the application to consider:

- Onboarding: Logging in and signing up for Winterest
- Matching algorithm: How people are matched, the efficiency of the algorithm, and one-sided vs. two-sided matches
- Features to make matching simple: Scheduling meet-ups, matching/unmatching with a user, searching for a user, updating a profile, and scrolling/swiping through users

Data Storage

User information is primarily stored in the userAccount table. After this (initial design) entity relation diagram was built, while building the tables, special features were implemented; this includes the pronouns attribute and separation of first and last names. The following code snippet shows the table for user information:

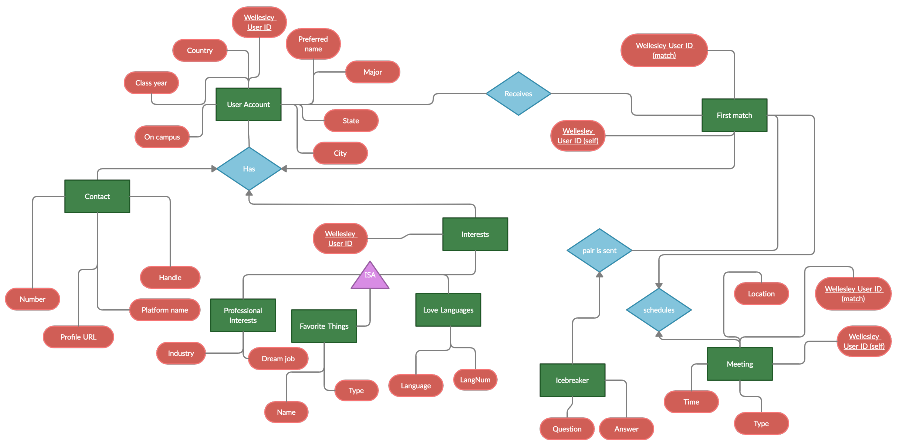


Figure 1: The entity relation diagram for Winterest. Each module of the application can be traced accordingly here.

```
create table userAccount (
    wemail varchar(20) not null primary key,
    fname varchar(30),
    lname varchar(30),
    major varchar(50),
    year int unsigned,
    country varchar(50),
    state varchar(2),
    city varchar(50),
    onCampus enum("yes", "no"),
    pronouns varchar(10)
)

ENGINE = InnoDB;
```

A user's Wellesley email (referred to as "wemail" in the table above) is used as a foreign key throughout the data storage process. This means that any user's information can be accessed and combined with information in other tables storing their data simply by inputting their email. Besides this table, another important table that will be referenced throughout the paper is the matches_scored table. This stores the score that is generated between two users based off of their similarities and differences. It is the main storage table for the matching process.

3.1 Onboarding

The onboarding process is split into two screens: basic information, and interests/contact. While the signup page displays two different screens, both sections live on one page. We decided to use Javascript to hide one section at a time in the case a user wishes to stop mid-signup. Thus, information will not be inserted into the database until the user completes both sections. The user is asked for basic information, such as email, name, location, and major in the first section. Additionally, the user can set a password, which will be protected and encoded via bcrypt. On the interests page, users can enter in three interests, as well as a bio. Users can also upload a photo, which will be saved in an uploads folder in our app.

3.2 Matching Algorithm

Scoring

The matching algorithm has been designed to incorporate every part of a user's input and check similarity between said user and every other person in the database. Here is a snippet of the Python code:

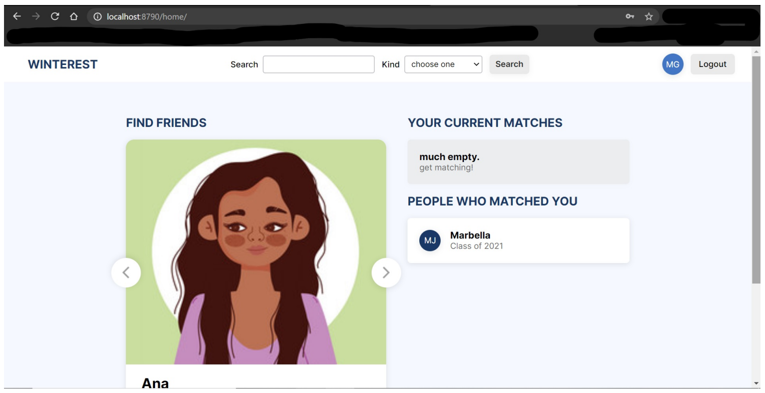
```
score = 0

if secondPerson.get('major', -1) == firstPerson.get('major', -2):
    score += 1
if secondPerson.get('city', -1) == firstPerson.get('city', -2):
    score += 1
if secondPerson.get('state', -1) == firstPerson.get('state', -2):
    score += 1
if secondPerson.get('country', -1) == firstPerson.get('country', -2):
    score += 1
if secondPerson.get('onCampus', -1) == firstPerson.get('onCampus', -2):
    score += 1

# add in score from love languages
score += matchScore_LL(conn, wemail, wemail2)

# add in score from favorites
score += matchScore_favorites(conn, wemail, wemail2)
```

The algorithm takes two users (one being the current logged in user and the other being any user from the database) and scores them as per their matching features. For instance, if the users are from the same country, one point is added to their total match score. The more the similarities between the two users, the higher their match score will be. This score will be calculated for every pair of users.

Every time a new user joins Winterest, their match score (based on their initial inputs) will be calculated compared to every user in the system. When a user updates their information, their match scores with every user in the system will be recalibrated. Because every user's profile is used for data extraction, the algorithm has a time complexity of O(n^2).

One Sided vs. Two Sided Matches

When a user has logged into their Winterest account, they are able to click through a carousel of people. If they wish to match with someone, they can click the match button below the person's photo and information. However, the user will not automatically be matched with the person. The user just requested to match with the person hence a one sided matching. When the person that the user clicked to match with logs into their account, the user's request will be displayed under “People Who Matched You” in the person's home page, as shown in Figure 2.

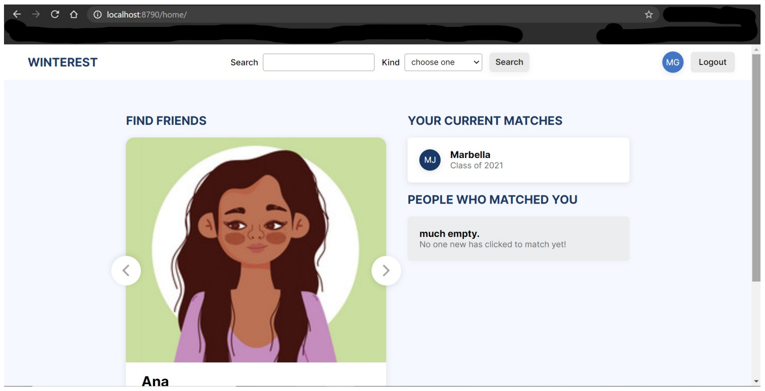


much empty.
get matching!

Marbella
Class of 2021

Figure 2: This figure shows the UI for an incoming one-sided match.

If the person clicks on the user, they will be redirected to the user's match page where they can click to match. If the person clicks to match then the matching becomes two sided and the user will then be moved to be displayed under "Your Current Matches" on the person's home page, as shown in Figure 3.



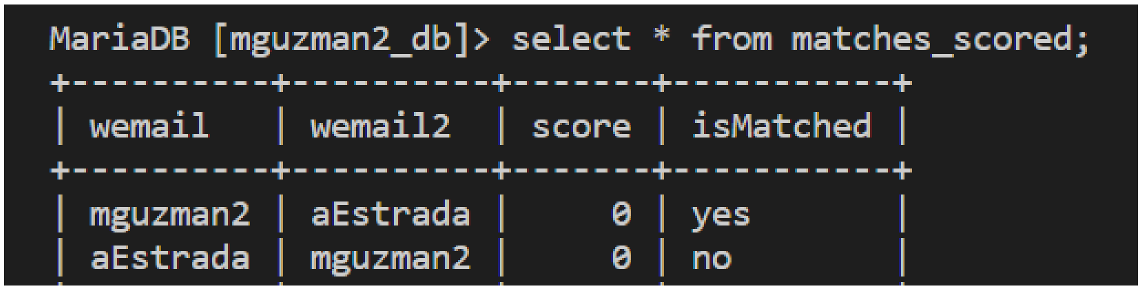
Marbella
Class of 2021

much empty.
No one new has clicked to match yet!

Figure 3: This figure shows the UI for the case where two users have matched with one another.

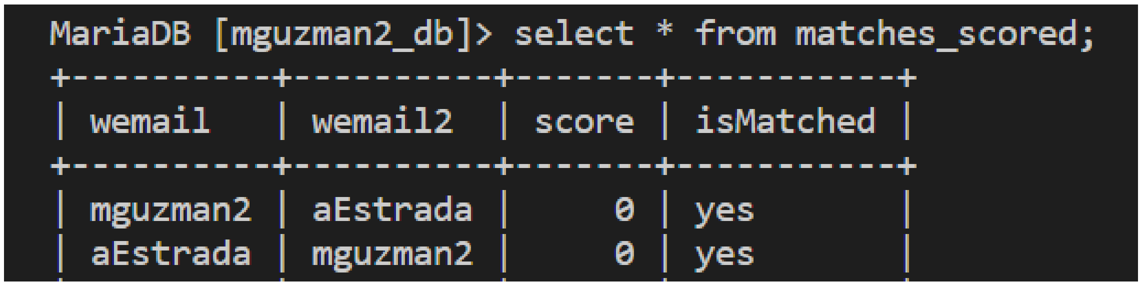
The person can then click to schedule a meeting with the user. As for the user's account, the person that the user officially matched with will be displayed under "Your Current Matches" on the user's home page. For example, if Melissa (mguzman2) clicked to match with Ana (aEstrada) then the one sided matching will appear like the image in Figure 4 in the matches_scored table.

If Ana decides to click to match with Melissa then this matching will become symmetric (two-sided) in the matches_scored table as shown in Figure 5. To collect all of a user's two sided matches and display them under "Your Current Matches," we first check if a match exists between the user (mguzman2) and a person (aEstrada) and if a matching exists between the person (aEstrada) and the user (mguzman2) in the matches_scored table. A match will only exist if a person clicked to match with the user or vice versa. If a match exists both ways (symmetric) then we append the person to a list of people that the user



```
MariaDB [mguzman2_db]> select * from matches_scored;
+----------+----------+-------+-----------+
| wemail   | wemail2  | score | isMatched |
+----------+----------+-------+-----------+
| mguzman2 | aEstrada |     0 | yes       |
| aEstrada | mguzman2 |     0 | no        |
```

Figure 4: The image above shows how the database interprets one-sided matches.



```
MariaDB [mguzman2_db]> select * from matches_scored;
+----------+----------+-------+-----------+
| wemail   | wemail2  | score | isMatched |
+----------+----------+-------+-----------+
| mguzman2 | aEstrada |     0 | yes       |
| aEstrada | mguzman2 |     0 | yes       |
```

Figure 5: The image above shows how the database interprets two-sided matches.

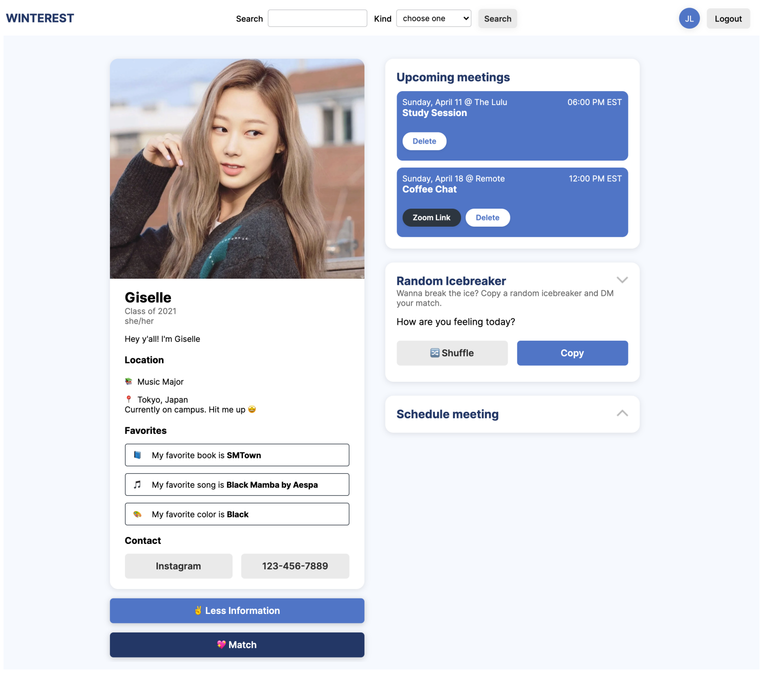
has a two sided match with and later return the list to be used in templating. To collect all of a user's one sided matches and display them under "People Who Matched You," we use SQL to collect all of the people that clicked to match with the user in a list. We also call the function that collects all of the people that have two sided matches with the user. We do this because there may be some overlap between the two lists and we need to guarantee that the list of all people that clicked to match with the user does not contain people that have two sided matches with the user. If there does exist two sided matches with the user, then we check each person from our list of possible one sided matches and only append to our resulting list if the person does not exist in the list of two sided matches and return the list later. If two-sided matches with the user does not exist then it is safe for us to return the list of all of the people that clicked to match with the user that we collected from our SQL code.

3.3 Features

Contact and Scheduling

While Winterest allows students to connect with each other, we outsourced communication to third-party applications. Thus, if a student matches with someone, the student can use their match's linked contact information (Facebook, Instagram, or Text) to communicate with them. The contact information is inserted during onboarding and can be updated when the user chooses to edit their profile.

The matches page has a random icebreaker generator to help jumpstart conversation. After a pair communicates through a third-party application, they can choose to schedule a meeting through Winterest. Once they agree upon a time, they can go on the matching



Giselle
Class of 2021
she/her

Hey y'all! I'm Giselle

Location

Music Major

Tokyo, Japan
Currently on campus. Hit me up

Favorites

My favorite book is **SMTown**

My favorite song is **Black Mamba by Aespa**

My favorite color is **Black**

Contact

Instagram | 123-456-7889

Less Information

Match

Upcoming meetings

Sunday, April 11 @ The Lulu — 06:00 PM EST
Study Session
Delete

Sunday, April 18 @ Remote — 12:00 PM EST
Coffee Chat
Zoom Link | Delete

Random Icebreaker
Wanna break the ice? Copy a random icebreaker and DM your match.

How are you feeling today?

Shuffle | Copy

Schedule meeting

Figure 6: UI seen while viewing a matched user's profile

screen and schedule a meeting. The scheduler includes features such as remote/in-person meetings, and the event card visualization differs depending on what type of it meeting is. Using templating, if the meeting is remote, a "Zoom Link" button appears on the card. If the meeting is in-person, the location will appear on the top left corner of the card. Users can delete scheduled meetings if necessary.

After a pair communicates through a third-party application, they can choose to schedule a meeting through Winterest. Once they agree upon a time, they can go on the matching screen and schedule a meeting. The scheduler includes features such as remote/in-person meetings, and the event card visualization differs depending on what type of it meeting is. Using templating, if the meeting is remote, a "Zoom Link" button appears on the card. If the meeting is in-person, the location will appear on the top left corner of the card. Users can delete scheduled meetings if necessary.

Viewing and Editing Profile

Users can view their profile by clicking a customized icon on the top right corner of the screen. On the left column of the profile page, they can view their Winterest Card, which includes the user's photo, personal info, interests, and contact information. Similar to the home and matching page, the user can expand and collapse their card. On the right column of the page, users can see their upcoming meetings and edit their profile.

If the user chooses to edit their profile, they will be directed to an edit page which resembles the signup page. The user's information will be autofilled on the edit page, so they can edit their information accordingly. Users can edit basic information such as name, location, interests, profile picture, and contact information.

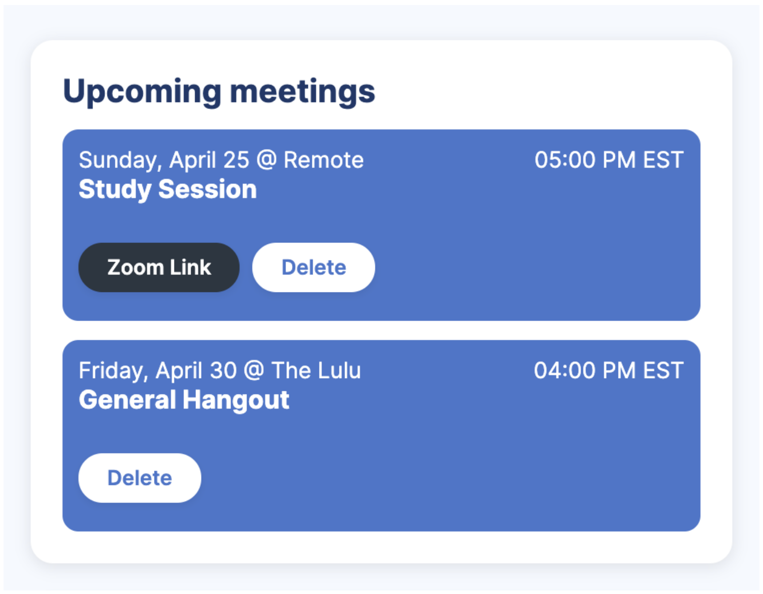


Figure 7: Setting up an appointment to meet with a match

Match and Unmatch

Earlier in the paper under the "Matching Algorithm" section, we discussed how users can match with someone. As for unmatching, if a user wishes to unmatch from a two-sided matching then they are able to unmatch three different ways. The quickest method would be to click on the person's name found under the user's section "Your Current Matches" in the user's homepage. The user could also use the search bar to find the person. These two methods will redirect the user to the matching page of the person where they can click the button "Unmatch" which can be found below the person's image and information. A third method to unmatch would be for the user to click through the carousel to find the person and click the "Unmatch" button found below their picture and information.

As for unmatching in a one-sided match, the person (aEstrada) who received a request from a user (mguzman2) to match can only click to match or ignore since a request to match is not an official matching until a person (aEstrada) consents and matches back. However, the user (mguzman2) who sent the request to match does have the option to unmatch. This means they withdraw their request to match and will no longer appear under the section "People Who Matched You" in the person's (aEstrada) homepage.

Unmatching using whichever method will update the "isMatched" column in the matches_scored table in our database to be "No" for both sides. Thus, the row where wemail = user email and wemail2 = person's email and the row where wemail = person's email and wemail2 = user email will both have their "isMatched" columns reverted back to "No." In the user's home page, the person's name will be removed from the section "Your Current Matches." The same will be true for the originally matched person's home page.

Searching

Users are also able to search for a person using either their name or their Wellesley email without the “@wellesley.edu.”

Searching by name

If a user searches a person by their name and there is exactly 1 person found then the user will be redirected to the person’s profile. If multiple similar people are found then a template that displays a list of the people will be rendered. This is shown in the image below. The user can then click on the suggested people which will redirect them to that person’s profile. If no people were found, then a message is flashed, notifying the user that the searched person does not have a Winterest account yet.

Searching by email

If a user searched a person by their email, then it is assumed that the user spelt the email correctly. Since wellesley emails are assumed to be unique then there are only two cases handled here: 1 person found and no person found. If 1 person is found then the user will be redirected to the person’s profile. If the person is not found, then a message is flashed, notifying the user that the searched person does not have a Winterest account yet.

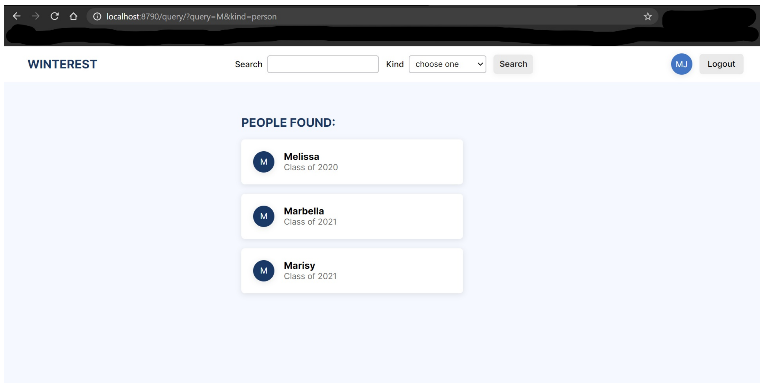

[redacted]

Figure 8: This figure depicts searching by name where the searched value is common amongst multiple people.

Sessions

To create a seamless experience, we stored the user’s email address as a session variable to keep the user logged in. To implement the swiping feature, we stored an “index” session variable that kept track of the current potential match the user was looking at. Thus, each time a user pressed back or next, the index session variable would be incremented or decremented. We chose to use sessions for both index and the user’s email to enforce privacy.

4 Application Safety

4.1 Locking and Threading

Since Flask apps handle concurrent requests, we need our Winterest application to achieve ACID properties. We mainly implemented strict two-phase locking. We locked full tables whenever we needed to make multiple operations, like updating or inserting data, on a table. To ensure thread safety, we made sure that every request uses a separate connection, avoided using global variables, and used locks to delegate exclusive access to shared data structures.

4.2 Password Protection

To increase the security of our web application, we utilized the bcrypt module to encode a user's password. Thus, we call "gensalt" on the user's password to increase variability, hashed it, and then stored it in the database. Even those who have access to the database cannot see the password.

5 Conclusion and Future Plans

The majority of features that were planned were implemented in Winterest. However, in the future, the algorithm should be optimized to increase efficiency and reduce the time complexity. Furthermore, we aim to allow users to leave fields blank and still be able to submit their profiles. We will also make the UI able to mold to these changes.

Overall, we believe that Winterest will be a very helpful tool for students to interact. Hopefully, in the future, we will be able to implement additional features to make the application as user-friendly as possible.