Project Description: Ever felt lonely and a bit too warm? If so, our dating site will solve your problems. After creating a profile you will be able to match with other overheated, lonely individuals. You are matched based on two criteria: your taste in movies, and your taste in air conditioners. When you create a profile you will be able to list your favorite movies. Additionally, you will then be presented with images of air conditioners, click on the ones that appear the most attractive to you! You will then be able to match with people who share similar tastes, and message them!

Directory Structure:

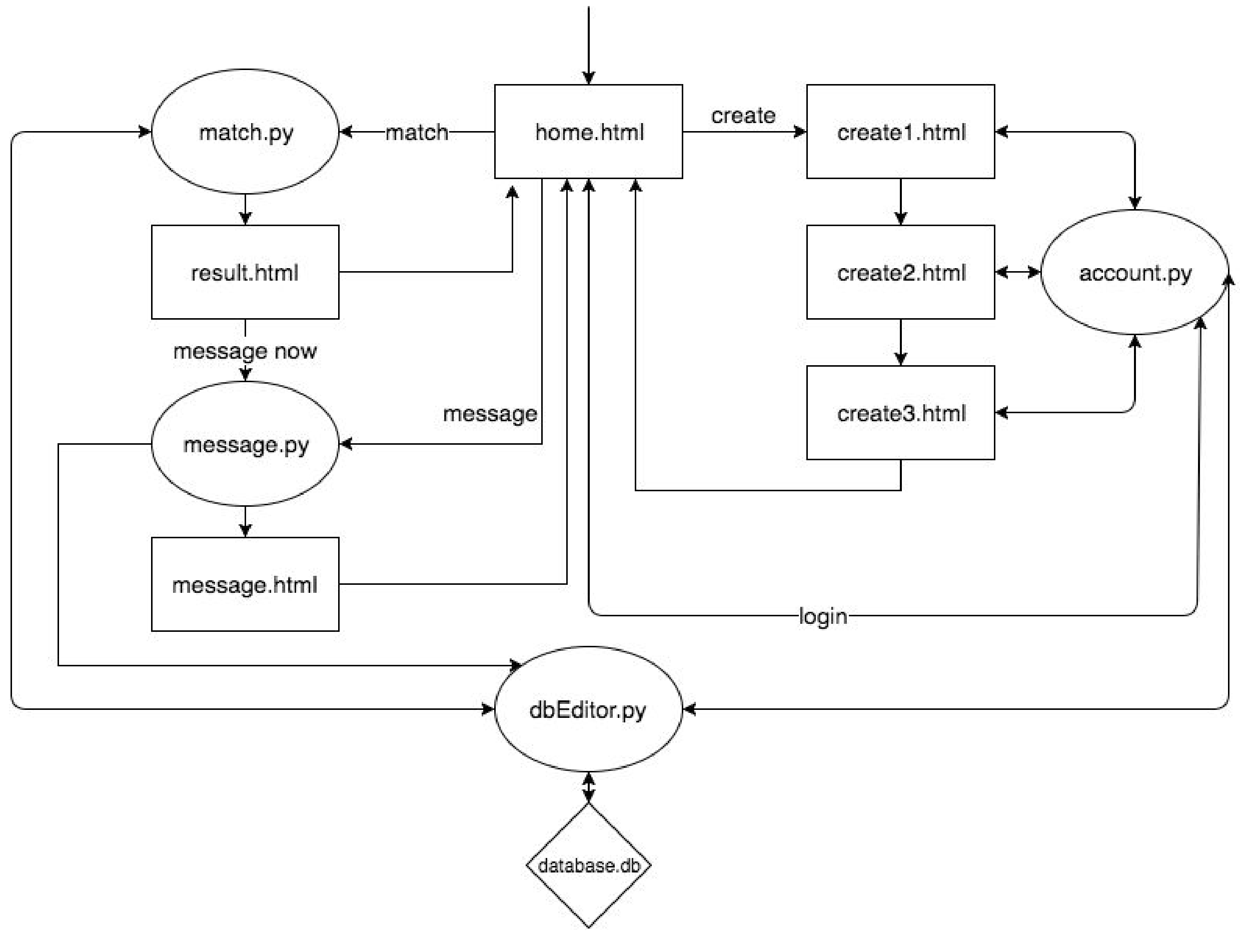
- app.py
- utils/
  - account.py
  - dbEditor.py
  - match.py
  - message.py
- data/
  - users.db
- static/
  - image files for users
- templates/
  - home.html
  - create1.html
  - create2.html
  - create3.html
  - results.html
  - message.html

Project components:

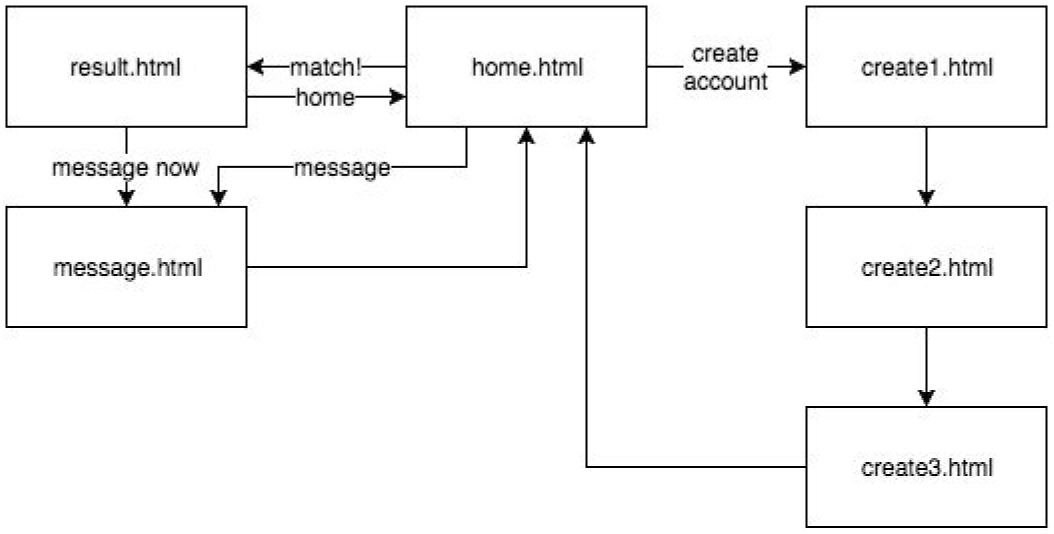
- **app.py:** This is the central app file. It deals with rerouting.
- **account.py:**
  - This file deals with creating the user's account, there are three main steps:
    1) Creating a username, password (hashed client side), bio, and uploading a profile image.
    2) Selecting your favorite movies. This is then run through the <INSERT API> api, and keywords, titles, and genres are stored with the user in the database.
    3) Presents images of air conditioners from google images, based on certain keywords. You select your favorite air conditioners, and then it stores the airconditioner keywords.
    4) Finally, you are redirected to home. This file also deals with logging in functions.
- **dbEditor.py:** This file deals with the database. It contains all the functions for adding and searching for data. See database.db for how the information is stored.

- **match.py:** This file deals with matching the users. When a user clicks the match button, they will be matched with another user based on the keywords for movies and air conditioners. Note, because all the data is present, this will not need to deal with APIs, and just be going through the database.
- **message.py:** This file deals with interuser communication. It stores the message history in a database.
- **home.html:** If not logged in a window will popup with a window to create an account or login. If the user clicks create account then they will be redirected to create1.html. If not, their homepage will be displayed. The homepage will show your sent and received messages. You will be able to message your friends, or make a new match. To message a friend click on them.
- **create1.html:** This is similar to a standard create account page. You enter your name, password, password again, bio, and profile picture. Additionally the password will be hashed client side for more security. It then redirects to create2.html.
- **create2.html:** This is the movie search page. You enter the names of your favorite movies. When you click submit it registers these, and redirects to create3.html.
- **create3.html:** Displays a few air conditioners. You click on the air conditioners you like, and then are directed back to the homepage where you can now log in.
- **results.html:** After you click the match button this page displays with your result. You can choose to either match, or try again. Try again refers back to the same page. Match returns to home.html where you can message you matches, or message.html to message the person now.
- **message.html:** This page loads your previous message history with a user. You can send a new message, or return to home page.

Component Map:



Site map:



Database Schema:

There are two types of tables.

The first is the "users" table. This table contains all the data on each user. A record is created when a user creates an account. This data is referenced when searching for matches. When a match is made, the user is added to matches, and if the person clicks "try again" the name is added to rejects. Example:

users:

Username	Password	Bio	movie keywords (csv)	ac keywords (csv)	matches (csv)	rejects (csv)
esandine	*******	I cool	action, comedy,	roof, big, fan	browndyn amite	loser1, loser2
browndyn amite	******	I coolish	romance, horror	energy, thin, floor	esandine	loser2, loser3

The second type of table is the one for messaging. There is a table for each user, the user's name is the table name. The table contains other users and message histories. To get the conversation for one user you search for the username in the table. You then get the message history and replace "user" and "other" accordingly.

<username>:

other	messagehistory (csv)
browndynamite	'user:hey dynamite', 'other:hey'

To-Do:

1. Account creation and uploading pictured
2. Format database
3. Initial pages
4. Matching algorithm and implementation
5. Messaging between users
6. Get database ready for presentation

Task Distribution

- Jonathan: Backend ; Account Creation, Messaging
- Nala: Backend ; Netflix API, Google Images API
- Dhiraj: Front End (styling, routing HTML pages) ; Project Manager
- Ely: Front End (styling, routing HTML pages)