# Tech Basics II: Project Report – Thies Petersen

## Introduction:

This second projects builds upon the idea I had for the first Tech Basics Project; thus, I will refer simply to that text when it comes to the general idea of the app This second iteration had two primary goals. Firstly, it had to implement the concept presented within the first iteration into a new library (streamlit). Secondly, it had to have all relevant functions for the idea of this app implemented. Additionally, I got feedback from three testers, which will be covered in a third part of this report.

## Streamlit:

Since we already spent a lot of time in class dealing with streamlit, it's widgets, functions and limitations, this was not the most challenging part of the development process. In the beginning I was quite attached to the general visual identity the UI of my last prototype had, which seemed impossible to properly implement it streamlit. I explored a lot of additional streamlit components one can install and tried to use the markdown/html functions it has, but with no real success. In the end I decided to favour making a better user experience and properly implementing the necessary functions of this app, over trying my hardest to recreate my previous UI design. Thus, the UI I ended up with is rather basic, however I believe it communicates it's functions to the user efficiently.

Another difficulty when dealing with streamlit were the session state variables. I ended up having to rely on them heavily in order to implement my functions, since I need to regularly update and draw from my online database, which requires my code to be rerun. If I wouldn't have stored the relevant variables in the session state, they values would have been lost, making user-input in that case much more difficult.

Beyond these elements, integrating streamlit with the necessary libraries we had been introduced to in class, which I needed to make my app work (primarily MongoDB) went very smoothly. The UI also ended up being fairly user friendly after a few suggestions from my testers.

## Functions:

Most of the proper functions of this app were initially not implemented in the last iteration of this app, which meant that my goal was to figure out how to properly make them work. The most central piece here was definitely MongoDB, a mostly free online database service, to which we were introduced to in class. Thankfully, the instructions provided in class made the barrier of entry for working with this library very low and allowed me to pretty much immediately during planning for my app, to know it's functions and means.

The user account was conceptualized to be very basic. An entry saved on the MongoDB Databank which attributes all relevant information to that user based upon the username. I would later end up storing all user relevant Info (friends, preferences, description) in this collection. I ended up neglecting proper encryption for the user data, even the saved passwords. This was primarily due to time constraints and my knowledge about encryption being very basic and also this knowledge having been acquired about 5 years ago. Beyond that, during implementation I struggled with re-familiarizing myself with pandas to properly read out an compare the data. Working with that took quite some time but ultimately functioned well. MongoDB also allows for quick and simple updates to the online collections. This however required me to properly familiarize myself with the different parameters and syntax the search and alteration queries for MongoDB required, which were not similar to the database focused programming language I know (SQL). Still, they thankfully had a great online documentation. The user system works well so far and since account creation is supposed to yield unique usernames, the sign-up form checks for that as well. In a future iteration I would like to add the ability for users to upload profile pictures, but due to my uncertainty with how to store them, I ended up not attempting to implement that function.

The finder function was planned to be very simple. I defined very basic preference parameters that could possibly lead to matching. I went with a restrained approach here as I wanted to keep overview of all the parameters and would thus not add very many. These were saved in the user-database. When a user would then request to be recommended new possible friends, I would compare the active users' preferences to all saved users' preferences and calculate a compatibility score. The other users with the highest score would be presented to the active user, who would then get the choice on how they would like to handle the recommendation. This system ended up being rather easy to implement, beyond a few roadblocks when it came to working with the streamlit logic. While the system is not super complex and is still lacking some features (for example the physical proximity is currently now taken into consideration) it is still a functioning implementation of the concept.

The chat function was the biggest unknown during programming and the last function I added. The approach I used was generating a  generate a simple key for each user pair consisting of both of their user names, to which a dictionary was attach which stored every message written, by attributing them to a key consisting of a username and the current date-time the message was written. That way, unless a user manages to write two messages in the same second, each key would be unique and would be clearly attributable to one of the two chat participants. Unfortunately, the chat does not automatically reload when you receive a message, this requires manual button presses. Implementing the possibility for the chat to automatically refresh would be my biggest priority in a following iteration.

## Feedback:

As mentioned prior, I also got feedback from three testers, who helped me improve in terms of design and utility. Specifically, I was told that some functions lacked clear descriptions and were interpreted as ambiguous, which I would implement by adding descriptions to them. This is e.g. why the chat and finder function have explanations attached to them on the homepage. Some other feedback went towards account creation. Initially the description was only editable after an account was created and not during account creation, which I changed. Some other feedback here, like letting users pick multiple identities I didn't manage to add in time anymore, but would definitely be added in future updates. The chat function and finder function were viewed positively and as functional, although the manual refreshing of the chat was lamented. Other small additions were added due to tester feedback, like a return button when viewing a friends' profile and other's little elements that improved the user experience. Unfortunately, my app is not very accessible and this was not something the testers focused on. Generally, the were satisfied with the functions implemented and viewed the flow of the website as self-evident and its functions mostly well explained. The feedback and suggestions they did mention I tried to implement if I still had the time. Overall, the impression of my testers that the app is easy to use and that the functions were well implemented left me satisfied. The UI was referred to by one tester as "standard" which, while implying a certain genericism, also meant that it was well navigable as a website.

## Conclusion:

Ultimately, I'm proud of the app I managed to create and the functions it already possesses. While it is still a pretty barebones app, I think the basic functions needed for what is essentially a cross between a social media- and a dating app are there and functionable. The UI while simple and generic, was perceived as pleasing and easy to navigate (once a few additions were made) making the user experience straightforward. The additions I mentioned previously would definitely be the next step in improving the app, but I'm satisfied with this iteration and believe I'm justified in calling it a MVP (minimally viable Product).