WeThinkCode_ Project Report: Matcha

Sameh Mohammed AbuRadi (student username: SAbuRadi)

Content:

This report contains a brief overview of the Matcha project while providing a brief review of the project's presentation and demonstration.

Introduction

Matcha is a web project that challenges you to create a dating website / web app. The application allows two potential partners to meet by allowing a user to register, connect, fill their profile, search and look into the profile of other users, like them and, chat with those who "like them back" from the registration to the final encounter. The application supports the following features:

- User Registration: the site enables users to register an account requiring a username, a nickname, a password that has to have at least one capital character, one small character, one number and be 8 characters long at least. It's also required to provide a valid email, the gender of the user, and their sexual orientation in order to complete the account registration. Upon registration, the user needs to confirm their registration by clicking on a link that's sent to their email once submitting a sign up form.

- User Authentication: once verified, the user can log into their account to complete their profile (if signing in for the first time). While signed in, the user has access to a sign out button no matter which page they're on. They can disconnect their profile by simply clicking on it.

- User Profile Modification: a user can add, remove and modify information on their own profile while they're signed in. They can also delete their account if they wanted to. That being said, the user can perform all the CRUD operations on their profile accordingly :
  - Create: a user can create a profile to access the dashboard of the site.
  - Read: a user can sign in and access their profile information on the site.
  - Update: a user can modify their profile and interact with other users.
  - Delete: a user can delete their account if they wanted to.

- User Connection: a user can **only** connect with other users once they've uploaded a photo to their profile. They'll be enabled to access the dashboard afterwards. They can "like" the other users, they also can "block" other users for any reason. In case they do, the blocked user no longer appears in the current user's dashboard, they are added to the list of "people you don't like" however, and the user can unblock them from there.

- The Profiles of Other: a user can see more information about the users listed in the dashboard by simply hovering the mouse over their "tiles / cards", including the location they logged in from the last time they did. That location is auto filled using a simple **geolocation** functionality.

- Popularity and Fame: the more "likes" a users gets the higher their fame rate would be, in our use case, the fame rate is equal to the number of the "likes" a user has.

- Notifications: once a user likes another, the other user gets a notification informing them of that event. They also can like them back if they wanted to.

- Chat and Messages: if a two users like each other, a chat button appears on their "tiles / cards" so they can talk to each other. Once any of them clicks it, they move to a conversation page where they can send messages or read the previous messages in that conversation.

- Search and Filter: a user can type in certain information in the search bar to find people they might be interested in. That input can be plain normal input, and it will be compared with the names and the interests of the other users. Or it can be a "filterTag" where a specific information is given to look for in the users' data bool. In our use case, two "filterTag" are currently implemented: **famerate:n**, where n is the fame rate you'd be interested in as a user, and **gender:fe/male** to look for a specific gender in the users' data bool.

Presentation

The project presentation covers the entire design process for the development of the application including documentation and testing plans. Research for this phase is conducted using a computer to access the information websites on the internet such as w3Schools and python.org documentation.

Researched was done for the web technologies such as Python, Flask, Php, Lumen, Redis, Mongo DB and Relational Database Management Systems (such as MySql) for the back end implementation. HTML, CSS and Javascript were looked at for the front end implementation. The applications data needs such as the need for accessible and persistent data were taken into consideration and researched too.

As a result, Redis was considered and researched for the applications data needs. However, as it was clarified later, an SQL database had to be used based on the advantages it provides for a web app of this nature. Based on that the code was refactored to make use of MySql. In depth research was conducted in order to choose tools and products that make up the stack with which the app is to be built. The chosen stack was Python / Flask for the back end (integrating SendGrid for the email trafficking), and plain HTML, CSS for the front end. I used github to maintain version control, and I used sublime text to write and edit my code. The UI was sketched on a notebook as I took a very minimalistic approach while realising it.

Demonstration

The project demonstration showcases the source code and functionality of the web application. The demonstration provides clear instructions on how to download the source code, setup and configure the database and web server and how to run the application itself. A code breakdown is provided in the form of all the different files present in the source code. Documentation is provided in the form of a README.md file that contains instructions on how to set up and run the application. A clear testing plan is outlined and the expected test results are explained for future reference.

Conclusion

The end result is a web app that is a proof of concept combined of different use cases of different web technologies. I came to find that Flask is the right tool for such a project due to how easy it is to set up and start coding. Documentation is clearly outlined in a format that will allow users and developers alike to understand the project from their own-experience based perspectives. Flask proofed to be really useful when quickly prototyping a product (at the MVP level of its creation), therefore I recommend it for building prototypes and projects at small to medium scale. I actually worked on this project and finished it in October last year. I had to modify it during the last couple of days (beginning of June, 2020) to satisfy the SETA qualification requirements. The modifications include integrating mySQL and re-approaching the problems this project solves from a different angle. As the project was originally written in Python 2.7.X and then I had to change the syntax slightly to run with a Python 3.6.X interpreter. The tests to be performed are to confirm that the previously mentioned features (pages 2 and 3) are completely functional.