**Facial Keypoint Recognition**

This documentation corresponds to the project of the *Machine Learning* subject at the FH Technikum Wien in the *Software Engineering* master(3rd Semester).

**Project description:**

While browsing [kaggle](https://www.kaggle.com) a promising dataset was found for facial keypoint recognition. Although there are already existing solutions for the problem, the point of the project was to build up as much from ground up as possible and learn through the process. The time constraints of this subject were taken in account. The end product of the project was planned to be a web application with a GUI to access the webcam, take a picture and send it to the backend to apply a pre-trained machine learning model and send back the picture with the found facial keypoints on the picture. The project is open source and can be found on Github [here](https://github.com/SoMoMaS/MachineLearning).

**Training dataset:**

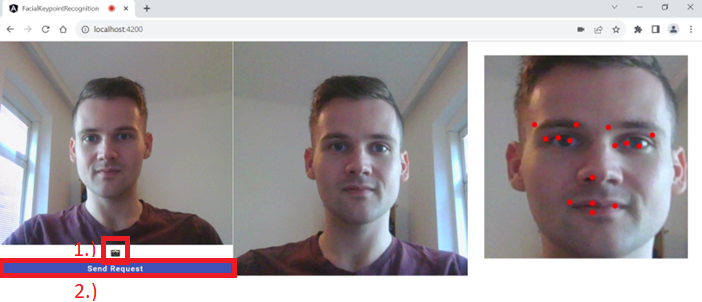
As mentioned before the training dataset for the project was found [here](https://www.kaggle.com/c/facial-keypoints-detection) on Kaggle. A script for cleaning the data creating more training data was also found here. The machine learning model was trained on more than 7000 images with 15 facial keypoints on each image. The keypoints on the images were represented by x and y coordinates so to each image corresponds 30 attributes. The training of a ML model was done with the help of tensorflow. Different settings were applied for each trained ML model. The number of epochs were changed from 10 to 50 until 100. The increasing number of epochs resulted in a more accurate prediction of the facial keypoints.

**Application:**

As mentioned before the project can be found on Github [here](https://github.com/SoMoMaS/MachineLearning). In the *readme* file you’ll find a description how to start the backend and the frontend.

The frontend is written in typescript and built on the [Angular](https://angular.io/) framework. The workflow is the following:

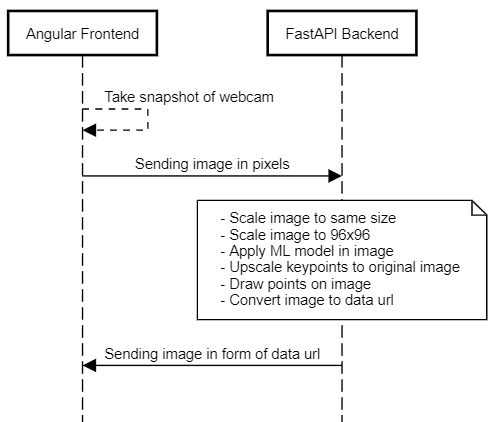
1. Take a picture with the picture emoji
2. Send the taken picture to the backend by pressing the *Send request* button



The taken picture will appear next to the webcam. After sending the request the backend will process the image and send the result back to the frontend.

The backend will further process the image as following:

1. The image will first be scaled to the same side sizes.
2. Scale the image to 96x96 (This is important because the ML model was trained on this sizes of images)
3. Applying ML model on the downsized image
4. The found facial keypoints will be upscaled to the original picture with same side sizes
5. Drawing points on image
6. Sending back the resulted image



After this the resulted image will be displayed on the right side in the browser.