GROUP A5 DOCUMENTATION

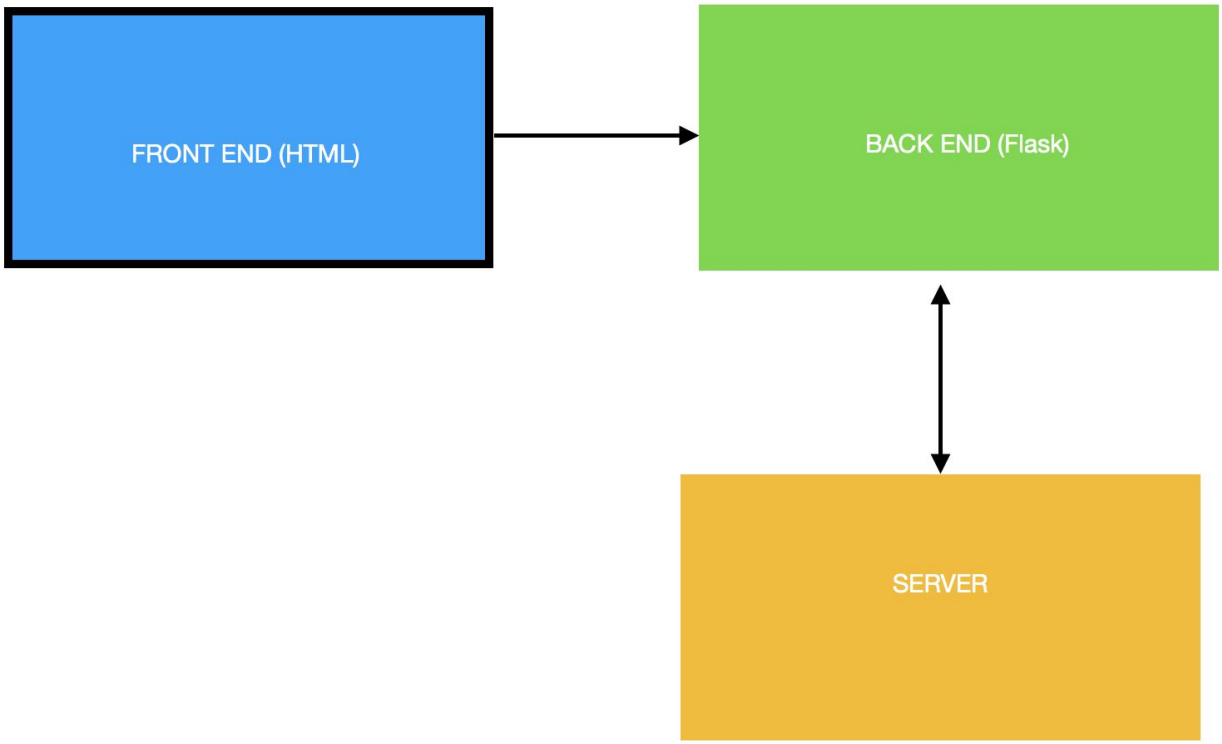
Our groups idea was to create a web application that would take calorie data from a fitness tracker and output a recipe based on what the user wants to eat. For example, if a user burns 700 calories and they want to each a protein-heavy meal, they could type in chicken and the recipes outputted would be below 700 calories.

We originally began this project using Node.js and Express because we were given the most instruction with that, but we realized a bit too late that because none of us were proficient in Javascript, Node was probably not the best direction to go in.

After the prototype assignment, we decided to do a 360 and change our plans. Because we all had experience with python, it made more sense to switch over to Flask. For the front-end specifically, we used HTML which we are connecting the front end to the back-end servers which connect to a router in another system. This router then reconnects back to the back-end server which will then display the front-end system. We specifically used the render function to incorporate the code in the back-end to the HTML in the front end.

In terms of connecting the API's, we had a couple of issues. Having had no back-end experience, we were learning as we went. We were able to create a get-request using Postman as a foundation and after the request was created and modified to work in our files, we were able to create a food input template and also a recipe display page. When working with Postman, we were able to After this, we would have chosen to tackle the fitness API, but all fitness API's require some sort of an authentication. Because Google OAuth was not compatible with the FitBit API, we chose to use Google Fitness expecting more documentation on that. However, we were not able to find much and all the options we were trying were throwing errors for us. We still tried to use Google Fitness packages from the Google Fit website and API's that were written but there were not many from Flask, and so we wrote a backend one that was from scratch that counted calories and calculated the number of steps walked.

We also had to connect mongoDB to make sure that all the data that we had to process from the Edamam API had to be stored in some sort of database. Alongside the inputs for what the user chooses to eat, the username retrieved from the authentication was also stored into the database. The data from the api was connected to the database which was then queried in order to be displayed on the app.


FRONT END (HTML)
BACK END (Flask)
SERVER