

Project Documentation

Team 2: Eating out healthy with Menual

Andrei Vlad

Christopher Harth-Kitzerow

Shayan Siddiqui



July 2018

Outline

1	Problem definition	3
1.1	Literature review	3
1.2	Survey	4
1.3	Combining literature review and survey results, identifying three key tasks	7
2	Methodology	8
2.1	Solution 1: Identify healthy dishes and provide quickly understandable ratings for each dish	10
	• The Evaluation Algorithm	12
2.2	Solution 2: Identify nearby, healthy restaurants and give users recommendations automatically	19
2.3	Solution 3: Provide more detailed, optional information and education for each dish	20
2.4	Detecting dishes in a menu	23
3	Functionalities	28
4	Architecture	33
5	Key responsibilities	33
6	Result of Evaluations	34
7	Limitation	35
8	Outlook on future development	37

1 Problem definition

At the beginning of the practical course, we agreed on one mission: Helping people to eat healthier in their daily life. The goal was to target the mass market to have as much impact as possible. When brainstorming and doing literature research on unhealthy eating habits, we found several problems that are associated with eating out. We decided to develop an app that helps solving main problems of eating out nowadays. Certainly, it is a problem of mass market and also one where we could not find any popular app on the appstore that already attends this problem.

1.1 Literature Review

When looking for unhealthy eating habits and problems in the literature, the first article that got our attention was a study done by the University of Illinois in cooperation with the American Cancer society. After evaluating multiple restaurants in the US their result was, “the majority of dishes served in restaurants were associated with a higher intake of saturated fat, salt, and sugar when compared to a home cooked meal” (The American Cancer society, 2003). According to another article of the American Cancer Society, high intakes saturated fats may be correlated to cancer and other health consequences (The American Cancer Society, 1996). While we were aware that eating out can be unhealthy, we were surprised that 1. Even the majority of dishes was found to be unhealthy and more importantly 2. the statement implied that known healthy dishes served in a restaurant may become unhealthy due to mentioned added salt, fat and sugar. As the article did not prove this implication explicitly, we decided to do more research in that area.

Indeed, there are multiple concerns in the literature that indicate that a variety of originally healthy dishes were found to be modified to an extent by restaurants, that original health benefits get outweighed by added unhealthy ingredients. The European Journal of Clinical Nutrition (2015) even stated in an article that eating out in a full-service restaurant was correlated with more added sodium and cholesterol than compared to a fast food restaurant. While eating at traditional restaurants added roughly 58 mg of cholesterol and 412 mg of sodium to a person’s average daily intake, switching from home-cooked to fast food meals added just 10 mg of cholesterol and 287 mg of sodium (data derived in the US). The article identified a misconception between how healthy restaurant dishes are perceived by customers and their actual health consequences. An article in the American Journal of Health that criticized missing nutritional information of restaurant menus (Maalouf J, Cogswell ME, Gunn JP, et al., 2015) supports this observation. It is further stated that while there is detailed information on packaged food in supermarkets, there is a lack of information on restaurant dishes. This makes it complicated for customers to accurately assess the health of restaurant dishes.

Another study not only detected that most restaurants use add too much sodium and saturated fats to their meals but also lower amount of fruits, vegetables, fiber and several vitamins and minerals (Kant A, Graubard B, 2018).

Both findings about originally healthy dishes that are made unhealthy and traditional restaurants being unhealthier than fast food restaurant in certain points was against intuition. Clearly, there is a misconception between which dishes are healthy when prepared at home and which can still assumed healthy when ordered in a restaurant.

This made clear to us: There is a lack of information and education of processed food. We need a framework that evaluates the actual healthiness of dishes ordered in a restaurant and not takes the same dishes cooked at home as the evaluation standard.

1.2 Survey

After doing research in the literature, we conducted a survey to get more information on people's eating habits, preferences and expectations for an app targeting problems about healthy eating. We surveyed 19 people, mostly students and in the age group of 18-30. In the following, one can find the survey results and reasons why we asked each question.

Quality of the survey

The surveyed age group of 18-30 is most likely the most relevant age group for using an app for a healthy diet. Certainly, the quality of the survey could be increased by having more surveyed and different occupations among surveyed. Nevertheless, results were clear and mostly reflected our expectations.

Survey part 1: Why do people eat unhealthy?

In the first part of our survey we focused on people's eating habits, preferences and interest in a healthy diet to get a fundamental idea what key target areas our app should focus on.

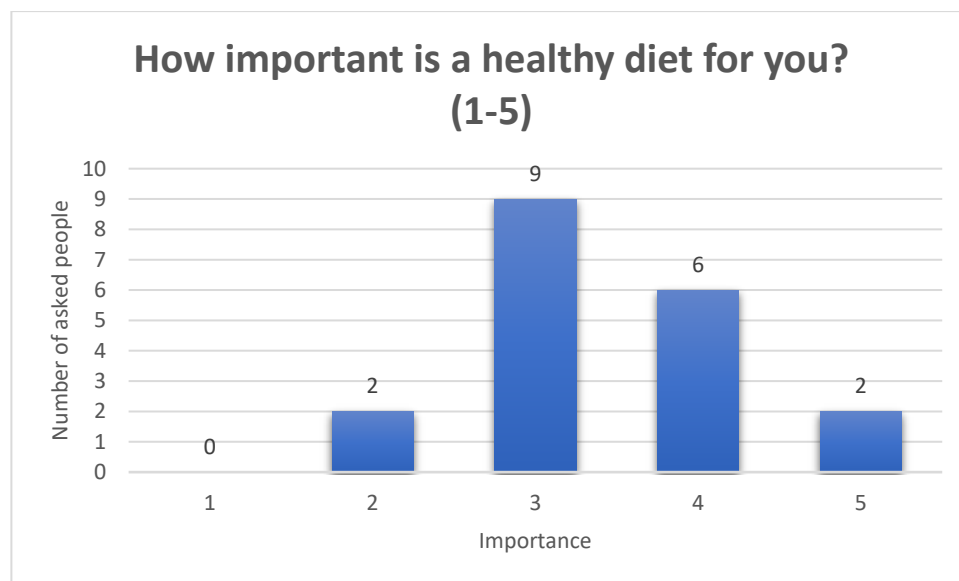


Figure 1: How important is a healthy diet for you?

This first question was important to find out whether people are interested in a healthy diet and if there is even a chance and market opportunity to dedicate an app to it. After all, people have to download and use it voluntarily. Luckily, most of the people described themselves as highly interested in a healthy diet.

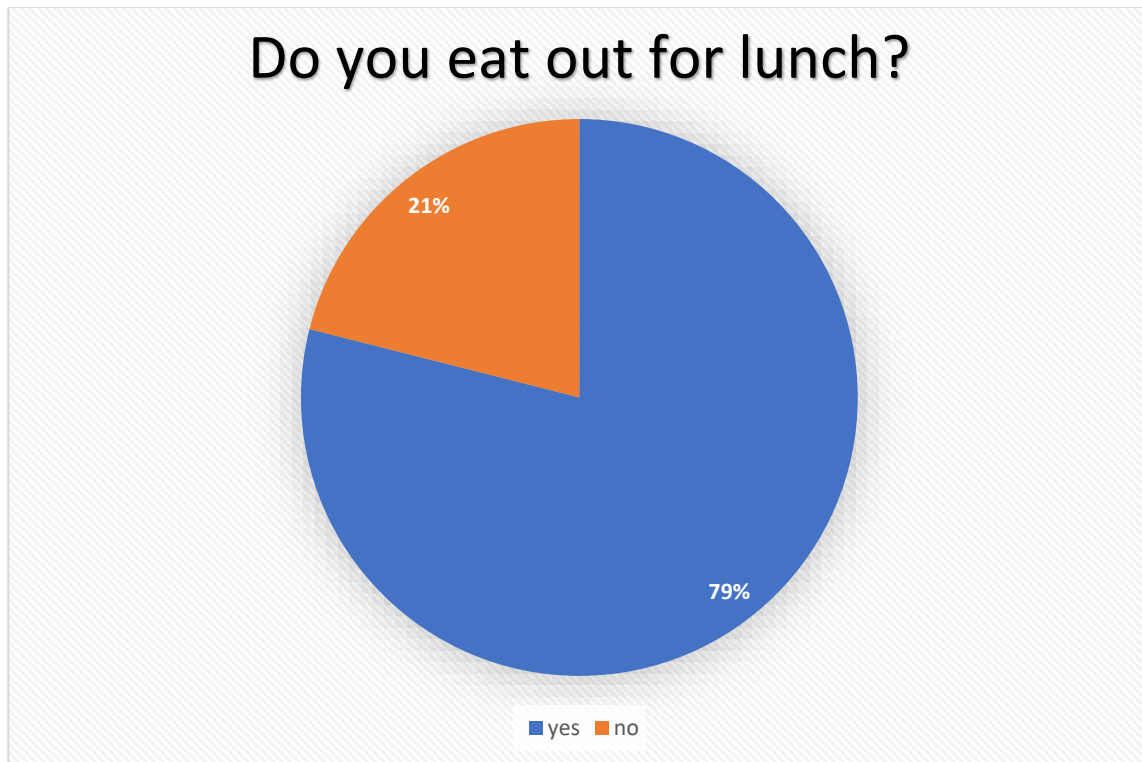


Figure 2: Do you eat out for lunch?

To identify whether eating out and associated problems are indeed a mass market phenomena we proceeded to ask if people regularly eat out for lunch. As the vast majority of people does indeed eat out for lunch, in combination with former identified problems associated with eating out, we knew that we can focus our app around this problem.

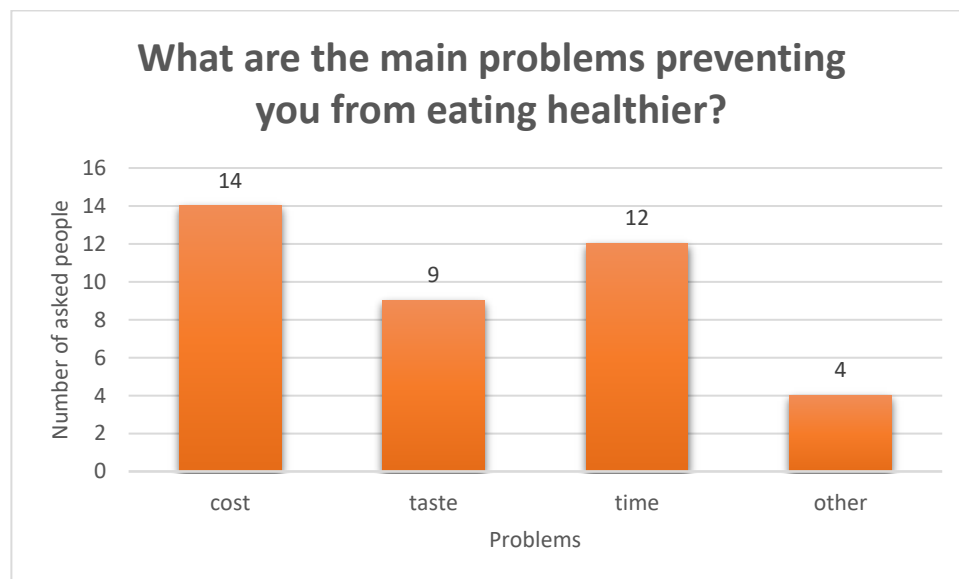


Figure 3: What are the main problems preventing you from eating healthier?

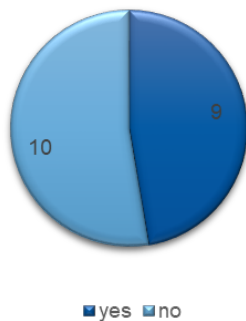
We asked further what main problems are that prevented them from eating healthier (multiple selections possible). We decided that we do not have the ability to offer people cheaper healthy food with an app, especially not as student group without connections to restaurants. In addition, interviewing mostly students may be a reason for such a high focus on cost. For this reason, we targeted the second most mentioned

problem, which is additional time it takes to implement a healthy lifestyle. Interviewed people were not willing to spend additional time on their lunch by preparing it at home or traveling large distances to find a healthy restaurant. Thus, the only way we can make people eat healthier for lunch is to encourage them to pick healthier restaurants that have to be near to their current location.

Survey part 2: Identifying market opportunities

The next part evolved around identifying expectations in a healthy diet app and also preventing key problems of current apps in the market.

Did you ever try out an app that assisted your diet?



Why did you stop using it?

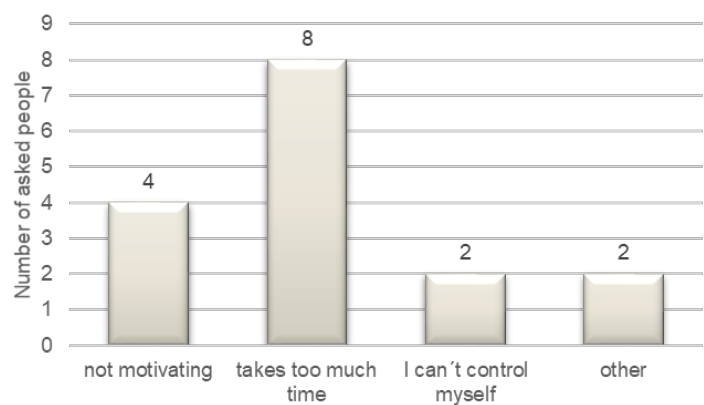


Figure 4: Problems with current dietary assistants

Surprisingly, about 50% of surveyed individuals already used a healthy diet app in the past, however mostly less than a week. When proceeding to reasons why people stopped, most did not mention motivation but actually, the time it takes to use the app. Too much manual obligations and the need to spend more than a few minutes daily to effectively use the app, apparently was reason enough to make used apps fail. Thus, our UI needs to provide information on a very simple level with focus on navigating through the app as fast as possible.

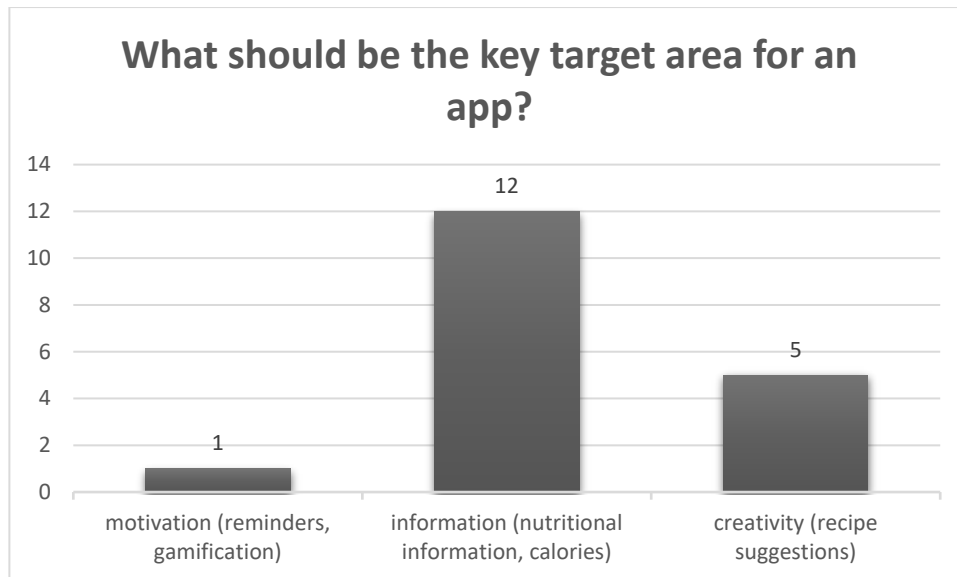


Figure 5: What should be the key target area for an app?

When asked for a key target area for the app, a vast majority wants to receive informational and educational contents about dishes. This matched the survey results from before where most people stated that they are already interested in a healthy diet, thus not needing further motivation. This was a clear indication that our app should be as informational as possible and should not provide any distraction from the content that users need to see.

Survey results

Surveyed people are highly interested in a healthy diet yet do not have enough time to always make healthy eating choices and eat out regularly. They expect an app to provide nutritional information in as fast as possible and do not want to spend a lot of time in an app.

Thus, the content of our app should be to improve people's selection process for restaurants and dishes. The user interaction should be fast and should only expect manual interactions from the user if necessary.

1.3 Combining literature review and survey results, identifying 3 key tasks

In summary, we identified 3 key problems about eating out that are supported by the literature:

- 1) Most of Restaurant dishes are unhealthy.
- 2) A lot of Restaurant offer unhealthy dish types to begin with.
- 3) Even typical healthy dishes can be unhealthy in a restaurant. There is a misconception between what is healthy and if it is still healthy in a restaurant.

This means on a broader level, we have 1) a supply problem of healthy restaurants and healthy dishes offered in restaurants and 2) an educational and informational problem about healthy dishes in restaurants.

Also, we had the following key findings in our survey:

- 1) An app should be quick, fundamental and provide nutritional information.
- 2) One of the main problems is time. We cannot change that people will eat out.
- 3) People are highly interested in a healthy diet.

By matching key literature problems with key survey findings, we can derive following tasks that an ideal app for eating out should solve.

Task 1: Identify healthy dishes and provide quickly understandable ratings for each dish

Literature problem 1 + Survey problem 1. Most of restaurant dishes are unhealthy. That means we have to have a strict framework in our evaluation algorithm. First priority should be to identify healthy dishes and only recommend those to the user, which are healthy without a doubt. Reaching a high level of confidence in our dish recommendations is acceptable even at the expense of rating only partially healthy dishes more strictly and therefore lower.

Task 2: Identify nearby, healthy restaurants and give users recommendations automatically

Survey Result 2 + Literature Problem 2. According to Survey result 2, we cannot change that people will eat out and opt for the fastest available option. Furthermore, according to literature problem 2, a large proportion of restaurants offers mainly unhealthy dish types to begin with. This means it is crucial to lead the user to a restaurant that offers healthy dishes and is nearby. Helping the user to select a healthier restaurant can already increase the chances of him/her also ordering a healthy dish. As we cannot expect users to manually look for recommendations on our app and open it prior to every restaurant selection process, we need to send users recommendations automatically at the right time. This way, before the user makes a decision he/she can be convinced to switch to a healthy nearby restaurant. Therefore, timing and content of this notification is crucial to succeed. For content, we can additionally acquire information like user's typical meal time, diet preferences and preferred kitchen type into consideration to offer the perfect individual recommendation for each user.

Task 3: Provide more detailed, optional information and education on each dish

Literature problem 3 + Survey problem 3. According to the third literature problem, there is a misconception between what is healthy and if it is still healthy when offered by a restaurant. This means it is highly necessary to evaluate dishes by our app and do not let the user decide which dish may be healthy or unhealthy. The only way to do this is to provide information to the user before he/she orders so we can provide quickly understandable recommendations.

Although, according to the third survey finding, people are highly interested in a healthy diet. This means there should be also a manually accessible statistic page on each dish that provides detailed insights and explanation why a healthy perceived dish may not be healthy when eaten in a restaurant or even not healthy at all. This can solve the addressed misconception and also satisfy the users interest for additional information. Important is, that this information is dedicated and not required to look at in order to not create a conflict with people's expectation to not spend too much time in an app. This means we need quick information as well as the more detailed statistic page to be accessible to the user.

2 Methodology

For the development process, it was now clear that the main features of our app have to solve each of the three identified key problems we derived from combining the literature review with our survey findings. Due to the complex nature of the problems, we explain each solution concept in this chapter before showing the visual result in the Functionalities chapter.

The following image (Figure 6) gives a quick overview to our three solutions to our three identified key problems.

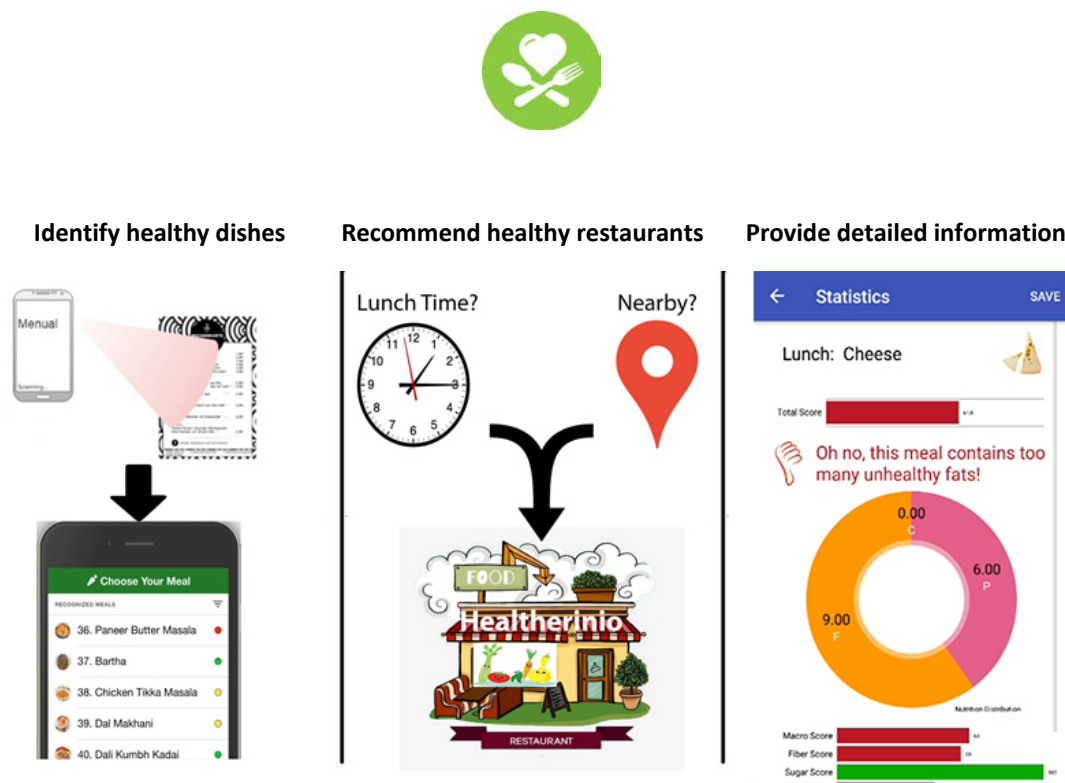


Figure 6: The three solutions of Manual to address identified key problems

Given our key features we gave our app the name Manual. A combination of the words “manual” and “menu” represent the guidance function our app has when it comes to choosing the right dish on the menu.

2.1 Solution 1: Identify healthy dishes and provide quickly understandable ratings for each dish

This task is the most crucial and most complex one of our App. We have only the menu with several dish names as input and have to present ratings understandable and at a glance to the user. The process of how we achieved to derive only relevant dish names from an optical image of a menu is described in the dish detection section. Here we want to focus on how we can calculate a rating for each dish given only the dish name. This requires a scientifically based framework in order to give reliable ratings.

Finding a proper framework for evaluating dishes

Several free accessible online databases already provide scores about dishes such as the USDA database. Unfortunately, none of those databases was providing nutritional information that was based on dishes served as restaurants but rather based on typical homemade recipes. Thus, these databases are not accurate for evaluating restaurant dishes.

This led to the challenging condition that we had to fetch as much nutritional information from actual restaurant dishes as possible and to develop our own evaluation algorithm in order to rate the food.

After browsing through multiple scientific sources for optimal nutritional information we decided to use the reference values by the Deutsche Gesellschaft für Ernährung (DGE). The DGE provides a detailed, comprehensive database of a variety of criteria that dishes have to meet in order to be classified as healthy. These reference values include vitamins, minerals, macronutrients, proportion of macro nutrients, amount of fiber, etc.

From all databases with REST API access that we tested, the [nutritionix.com](https://api.nutritionix.com/) API was by far the most comprehensive database in terms of restaurant dish entries as well as the variety of nutritional information for each dish. Another benefit is that the database only lists nutritional information for restaurant dishes rather than home made recipes. This fits perfectly for our use case. Furthermore, the Nutritionix API provides information on nearly every value that is included in the DGE reference tables. This leads to a situation where we are able to evaluate macro-nutrients proportions, amount of sugar, amount of healthy fats (divided in mono-unsaturated, poly-unsaturated fats, saturated fats, trans-fats), amount of fiber, 10 different minerals and 13 different vitamins. This leads to 38 values for each dish that we can evaluate and aggregate into a final score.

We do not see a problem in using an US based database as the default language of the App is English, thus making the US arguably the biggest market for it. Furthermore, available literature on our topic is mostly from the US and also the most detailed databases have their origin in the US. We cannot verify in how far literature claims and database entries are representable for other countries.

A big advantage of our app is that one can use it in multiple application scenarios such as eating in a mensa, only eating a snack or dining out in restaurant.

The Evaluation Algorithm

Below is an example of all nutritional information our algorithm evaluates (with explanations). Score values will be explained in more detail later on:

Table 1: Nutritional information processed by our algorithm with explanations

Nutritional value	Explanation
Mealtype + DishName	Mealtype is either breakfast, lunch, dinner, or snack
Grams of protein	Grams of proteins contained in the dish. DGE recommends a certain proportion of protein in comparison to fat and carbohydrates.
Grams of fat	Grams of fat contained in the dish. DGE tolerates a certain proportion of fat in comparison to protein and carbohydrates.
Grams of carbohydrates	Grams of carbohydrates contained in the dish. DGE tolerates a certain proportion of carbohydrates in comparison to fat and proteins.
Grams of sugar	Grams of sugar contained in the dish. DGE tolerates a maximum of 10% sugar of total energy in the dish.
Grams of fiber	DGE recommends to consume 13g of fiber daily.
Grams of mono-unsaturated fats	DGE marks mono-unsaturated fats as healthy.
Grams of poly-unsaturated fats	DGE marks poly-saturated fats as very healthy.
Grams of saturated fats	DGE marks saturated fats as unhealthy.
Grams of trans fats	DGE marks trans fats as unhealthy.
Vitamins	
Vitamin A	DGE recommends 0.9mg daily.
Vitamin D	DGE recommends 20µg daily.
Vitamin E	DGE recommends 13mg daily.
Vitamin K	DGE recommends 65µg daily.
Vitamin B1	DGE recommends 1.1mg daily.
Vitamin B2	DGE recommends 1.3mg daily.
Niacin	DGE recommends 1.3mg daily.
Vitamin B6	DGE recommends 1.3mg daily.
Folat	DGE recommends 300µg daily.
Vitamin D	DGE recommends 6mg daily.
Pantothenic Acid	DGE recommends 45mg daily.
Vitamin B12	DGE recommends 3mg daily.
Vitamin C	DGE recommends 100mg daily.
Minerals	
Sodium	DGE recommends 1500mg daily. Higher intakes can cause negative health consequences.
Chloride	DGE recommends 2300mg daily.
Calcium	DGE recommends 1000mg daily.
Phosphorus	DGE recommends 700mg daily.
Magnesium	DGE recommends 325mg daily.
Iron	DGE recommends 13mg daily.
Zinc	DGE recommends 8mg daily.
Selenium	DGE recommends 65ng daily.
Iodine	DGE recommends 3.5µg daily.
Potassium	DGE recommends 4000mg daily.

Fluoride	DGE recommends 8mg daily.
Scores (Explained in more detail below)	
Macro Score	Input: Grams of carbohydrates, fats and proteins, proportion information according to DGE based on mealtype, user preferences Output: Evaluates in a score between 0-100+ if the proportion of nutrients can be considered healthy according to DGE.
Sugar Score	Input: Grams of sugar, user preferences Output: Evaluates in a score between 0-100 if the amount of sugar is tolerable according to DGE.
Fiber Score	Input: Grams of fiber, proportion information according to DGE based on mealtype, user preferences Output: Evaluates a score between 0-100+ if the amount of fiber contained in this dish meets or exceeds DGE standards.
Fat Score	Input: Grams of all 4 mentioned fat types, user preferences Output: Evaluates a score between 0-100+ if contained fats can overall be considered healthy according to DGE standards.
Vitamin Bonus	Input: All 13 mentioned vitamins Output: Evaluates a bonus starting at 100 that checks for each vitamin if a high content of that vitamin in the dish can be rewarded with a bonus. May not affect or higher total score.
Mineral Bonus	Input: All 10 mentioned minerals Output: Evaluates a bonus or negative Bonus starting at 100 and checks for each mineral if a high or too high content of that mineral in the dish can be rewarded or punished with a bonus. May not affect, higher, or lower total score
Total Score	Input: All previous scores that get calculated by our algorithm, weightings of each score, counter array that checks if all information is comprehensive, user preferences Output: Combines all scores with user preferences, with a weighting depending on information richness and certain dish attributes (explained later on). Normalizing the score makes it interpretable and allows assigning a color to the dish.

100+ means that the dish can receive bonus points for positive values that exceed DGE standards like exceptionally high amounts of vitamins, a large amount of healthy fats, a large amount of fiber, etc

Although a variety of minerals and vitamins can cause an overdose with negative health consequences, this overdose for most vitamins and minerals is not achievable by eating a single restaurant dish. For this reason, we do not include punishment scores for more vitamins and minerals than we currently do. We only include punishment scores for minerals like Sodium, where a typical restaurant dish can realistically cause an overdose.

How we developed the formula

According to the DGE, multiple factors are relevant when evaluating the health of the dish. However, each criteria category has different relevance when coming to a final conclusion. The challenging task for us was to translate sometimes-vague descriptions of criteria relevance state in the literature in a specific quantitative formula by giving criteria categories different weights. We split the different criteria in the following categories with following weights:

Macronutrient score: The DGE recommends consuming 30% of daily energy intake in lunch and more specifically 35% of daily protein intake, 30% of daily fat intake and 30% of daily carbohydrate intake. An optimal lunch should have exactly this proportion of macronutrients. For other types of meals (breakfast, dinner, snacks) these proportions are different. This is why our evaluation algorithm may give different scores to the same dish depending if the meal is a lunch or a dinner for example. A dish fulfilling exactly these nutrients receives a score of 100. Points get deducted for each percentage shift towards higher fat and higher carbohydrate proportion. Slight bonus points are given for each percentage shift towards higher protein. The DGE gives this proportion medium to high relevance, which is why we give it a weight of 2.

Sugar score: The DGE tolerates a maximum of 10% of sugar in a dish. Each percentage increase will result in negative points. A dish containing less than 10% sugar will receive a sugarscore of 100.

According to the DGE dishes that are very high in sugar exceeding a specific threshold (not further specified) have additional negative health consequences. Additionally, while the relevance of amount of sugar is important, we should not give dishes too much credit just for not containing a lot of sugar. To address these factors we define a Boolean variable that punishes a sugar score lower than the threshold of 75 to receive consequences with higher weighting. Thus the standard weighting for sugar is 1 (sugarscore between 76 to 100) to have a standard relevance for non-high sugar dishes. If, however, the sugarscore is 75 or below we give it a relevance of 4, thus highly increasing its negative impact to address the explanation given by the DGE.

Fiber score: The DGE recommends a 13g of fiber each day. To evaluate fiber in a dish we multiply the proportion of recommended energy intake for a meal (eg. 0.3 for lunch) with that recommended amount. With a standard score of 100, points get deducted for not meeting that amount, while points get added when exceeding this amount (since there are only positive impacts of eating more fiber to a natural degree). The DGE implies a medium relevance, which is why we give the fiber score a weighting of 1.

Fat score: There are different types of fats with following health implications. Please note that these health implications are highly simplified to keep the paragraph concise. Mono-saturated fats: Healthy. Poly-saturated fats: Very healthy. Unsaturated fats: Unhealthy. Trans-fats: Very unhealthy.

The standard score is 100. Each gram of healthy (and very healthy) fat adds (more) points, while every gram of unhealthy (and very unhealthy) fat deducts (more) points. Unhealthy fats are punished more than healthy fats are rewarded to maintain our intended strictness. The DGE implies a high relevance for the healthiness of fats in a meal. This is why we give it a weighting of 3.

Vitamin Score and Minerals Score: These scores play a special role as we cannot expect a meal to meet every recommended value for each vitamin and mineral. Rather, we give bonus points for vitamins and minerals that are contained above the reference value per day multiplied with percentage of energy intake for that meal. However, minerals like sodium can have a negative impact when eaten more than the reference value. For this reason, we deduct points for those vitamins and minerals that can be exceeded to an unhealthy amount in a diet. The standard score is 100. Each large amount of beneficial vitamin and mineral results in bonus points. Each too large amount of vitamin or mineral that can turn unhealthy results

in negative points. Due to their special role (23 different vitamins and minerals get evaluated in the algorithm) it is not reasonable to try to guess a weighting based on vague implications in the literature for each. This is why the overall score for vitamins and minerals do not receive a specific weighting but get added as bonus or negative points to the total score before it gets normalized. After normalization of the total score (explained in total score) these scores get a dynamic implicit weighting on calculation. This way the weighting for each value may vary between 1 and 3 based on information richness of the API response (explained later) and overall initial health assessment of the dish (high sugar or not, ...).

User preferences: We consider individual user preferences in our scores to offer a unique selling point. We can take allergies, diet preferences and goals into account. Users might be allergic to different ingredients, be vegetarian or on a diet and opting for less carbohydrates. Evaluating different scores depending on user preferences offers additional value to users. On a dedicated tab in our App, users can select their preferences. The standard option is standard preferences and no allergies. However, users can select for the above-mentioned scores if they want more or less impact for that particular score. This information is forwarded as the userpreferences array parameter to the main evaluation method. Assuming a user wants to punish high amounts of sugar harder than our algorithm does, he/she can change the option in diet preferences with the following impact in our formula when calculating the sugar score:

Current formular: $\text{sugarscore} = 100 - (100 * \text{sugarRatio}) * 3 * \text{preferences}$;

The standard option for preferences is 1. That means we punish each percentage of sugar that is above the 10% threshold proposed by DGE with 3 punishing points. If the user selects a stricter punishment for high amounts of sugar in his/her diet preferences tabs, the preferences value gets updated to 2. Thus, each exceeding percentage of sugar is now punished with 6 instead of 3 minus points. This way we can generate individual scores and an individual total score for each user by combining scientific health standards as well as considering personal preferences.

Total Score: To aggregate each of the six score value with their weighting to a meaningful total score, each score with their weighting has to get added up. Minerals and Vitamin Score with their special role are not weighted and just added with 100-VitaminScore and 100-Mineralscore to the total score at this point. The value however does not get normalized yet. Here one can find a standard output for “chocolate” (with standard user preferences):

Macroscore: 81, Sugarscore: 0, fiberscore: 62, fat score: 87 Vitamin Score: 100 Mineralscore: 100

Please note that the colors in this case are no evaluation at all but are just used to better visualize the calculation!

With this output, the dish is excessively high in sugar (sugarscore <75) which results in a weighting of 4 for the sugarscore. Adding up all other weightings (2+1+3) we end up with a sum of weightings of 10. It should be noted again that Vitamin and Mineralscore do not get weighted. Calculating the not normalized final score goes as follows: $81 * 2 + 0 * 4 + 62 * 1 + 87 * 3 + (100 - 100) + (100 - 100) = 485$

Each score is multiplied with its unique weighting to calculate a total score of 485. This total score is not normalized and does not give any information on the actual health of the dish. In order to be able to accurately interpret results we have to divide it by n (n is the sum of all weights). Summing up the weights: $4 + 2 + 1 + 3 = 10$. Dividing the total score by 10 gives us the normalized total score of 48.5 that can now be used to interpret dish health.

This result shows that chocolate for lunch is not a healthy dish. A healthy dish requires a normalized total score of over 100 to make sure there are no negative aspects of the dish (unless they get outweighed by

overly positive scores). As we can see chocolate does not provide a good proportion of macro-nutrients as it contains too much carbohydrates and fat. It also contains way too much sugar. Therefore, the sugarscore is 0. Furthermore it does not contain enough fiber and the healthiness of fats is not optimal (fatscore = 87). In addition, overall there are no outstanding vitamin or mineral benefits due to eating chocolate (standard score of 100 for vitamins and minerals). All these factors make it reasonable for chocolate to receive a bad total score. In the following paragraph, we want to look a bit closer on the interpretation of the totalized normal score.

Providing a quickly understandable rating for each dish

With the formulas shown in the last paragraphs, we achieved to aggregate all 38 nutritional information in one normalized total score. However, we can display this information even more user friendly. We can do the interpretation of the total score for the user by only displaying one intuitively understandable colour to the user. A traffic light scheme seemed the most natural to us. The colours have following meanings:

Green: This dish can be safely recommended to the user. It is certain to be healthy.

Yellow: This dish may be a valid option if the user dislikes every green marked dish. Yellow dishes however have minor critique points that results in them being not an optimal choice.

Red: The user should not order this dish. Red dishes have major critique points that results in them having negative health consequences.

We will now explain a way to accurately translate the normalized total score into the correct colour based on these definitions on how to interpret each colour.

Towards proper interpretation of the normalized total score

To repeat our objective when rating food: We only want to mark dishes as healthy when we are certain that they really are. That is why we want to be strict on our evaluation. The only logical consequence is to only give a recommendation for a dish and marking it green if the dish has a normalized total score of more than 100. This means that either a dish has no negative health benefits and at least one positive in order to exceed the score of 100. Alternatively, a dish can have numerous outstandingly positive attributes to make up for a negative side effect. It should be repeated again that negative health consequences always get evaluated a lot stricter than positive benefits are credited in order to match our strictness objective. This makes it hard for a dish to actually make up for bad scores. For a dish containing that much sugar that the sugar score results in 0 there is most likely no way for it to still get a total score of over 100 by containing a lot of vitamins, minerals and healthy fats. This ensures that only healthy dishes make it into that top tier and receive a green font size. Dishes can also receive a yellow font size. The reason is to indicate that the dish is not optimal yet there are some benefits over a red dish. Yellow dishes are either dishes that experienced only slight deduction in points, or dishes that have a combination of multiple health benefits to one regular deduction. Examples for this can be seen in the appended video. Yellow dishes require a normalized total score of at least 90 to ensure that health benefits can justify the points' deduction from other sources.

Finally, red dishes are the ones we cannot recommend to a user. As we had to develop our own framework based on information indicated by the literature, we have to be very confident in order to justify recommending a dish to the user. For that reason, most dishes will receive a red rating. However, this exactly matches findings from the literature that argued that the majority of restaurant dishes are unhealthy. Indeed, the majority of our rated dishes receives a red rating, while yellow and green ratings are always reasonable. This proves at least a basic level of accuracy in our rating algorithms if DGE standards are applied.

For a dish to receive a red rating, it has to gain a score of 90 or lower. This means that a dish which contains for example 20% of sugar and has no other health benefits or flaws already receive a red rating. This strict evaluation is supported by 1. our strictness objective 2. Numerous literature findings that rate the majority of restaurants dishes as unhealthy and 3. A reasonable assessment of DGE standards.

Going deeper: The formula explained

We already mentioned that our framework uses 38 different values for evaluating a dish. We will now explain how each value is weighted in our evaluation algorithm and how we came up with these weightings. This was a challenging part. In our literature review, we could not find a single article that suggests specific quantitative values for when a dish should be classified as unhealthy or healthy. We only found indications of which values may play a high role in a healthy dish (such as amount of sugar) or a low role (such as amount of Vitamin B5). To keep the evaluation reasonable we had to make a decision how strictly we want to rate each dish. We concluded that the only reasonable way would be to rate dishes as strict as possible. Thus, a dish receives already a low rating for one high impact value that is not met according to DGE standards. The goal was to be certain that dishes we recommend with a high rating are healthy without a doubt. Only then, users can trust the app and actually act on behalf on these recommendations.

The current formula is the following:

$$\text{totalscore} = \text{makroscore} * 2 * (1 - \text{counter}[0]) + \text{sugarscore} * (1 + (3 * \text{highSugar})) * (1 - \text{counter}[1]) + \text{fiberscore} * 1 * (1 - \text{counter}[2]) + \text{fatscore} * \text{counter}[3] + (\text{vitaminscore} - 100) + (\text{mineralScore} - 100);$$

One can ignore the counter array for now which is used to integrate information richness to the formula. The idea behind this is to check if the API response contains indeed all values for a particular dish. Assuming the API response would not contain information on every of the 4 kinds of fats we would need to give the fat score a lower rating due to missing information richness. While this is an integration according to best programming practices, there was not a single case in our tests where the API response contained incomplete information. Thus, we can safely ignore the counter array for following explanations array and simplify the formula:

$$\text{totalscore} = \text{makroscore} * 2 + \text{sugarscore} * (1 + (3 * \text{highSugar})) + \text{fiberscore} + \text{fatscore} + (\text{vitaminscore} - 100) + (\text{mineralScore} - 100);$$

As one can see, Macroscore has a weighting of 2, and fiberscore has a weighting of 1. Fatscore has a weighting of 3 (assuming that the API provided information on all 4 kinds of fats for a sophisticated evaluation). Sugarscore has a special role: It has a weighting of 1 if the dish is in a reasonable range of sugar content and a weighting of 4 if it is high in sugar. This way, high amounts of sugar can have a big negative impact on the score, without declaring a dish without no sugar for very healthy just for the reason that it is not high in sugar. Vitaminscore and mineralscore start at 100. Each contained vitamin and mineral with a healthy impact around the reference value gains positive points. Each mineral and vitamin that in high consumption can have negative impact (such as sodium) deducts points from the score. Thus, a vitaminscore or mineralscore of under 100 mark a negative effect of vitamins and minerals on a dish and a score over 100 marks a positive effect. These bonuses or negative bonuses are added to the total score.

In order to end up with a value between 0-100+ for the total score, we have to normalize it. For this reason we have divide each score by the number of total weights. Vitaminscore and Mineralscore are not normalized as they function as a negative or positive bonus. This leads to the following formula.

totalscore = score / (10 - (3 * (1 - highSugar)) - counter[0] * 2 - counter[1] * 1 - counter[2] - (3 - counter[3]));

Assuming information richness again, we are left with the following normalization formula:

totalscore = score / (10 - (3 * (1 - highSugar)));

Assuming a dish is not high in sugar, this formula will add up all the weighing: Macroscore: 2; Fiberscore 1; Sugarscore 1; Fatscore 3; Mineralscore and Vitaminscore: 0 (added as bonus or negative bonus at the end): Results in a Total score of 7. $10 - 3$ also results to 7. Thus, the total score will have the correct value in a range between 0-100+, just as the other scores, in order to make an interpretation.

Example dishes and corresponding ratings

In order to help better understand our evaluation algorithm and impact of user preferences it may be useful to look at one example where we evaluate different dishes with our algorithm and explain resulting scores step by step. One can find a result for cheese cake in the Appendix.

The following graph sums up the structure of our evaluation algorithm:

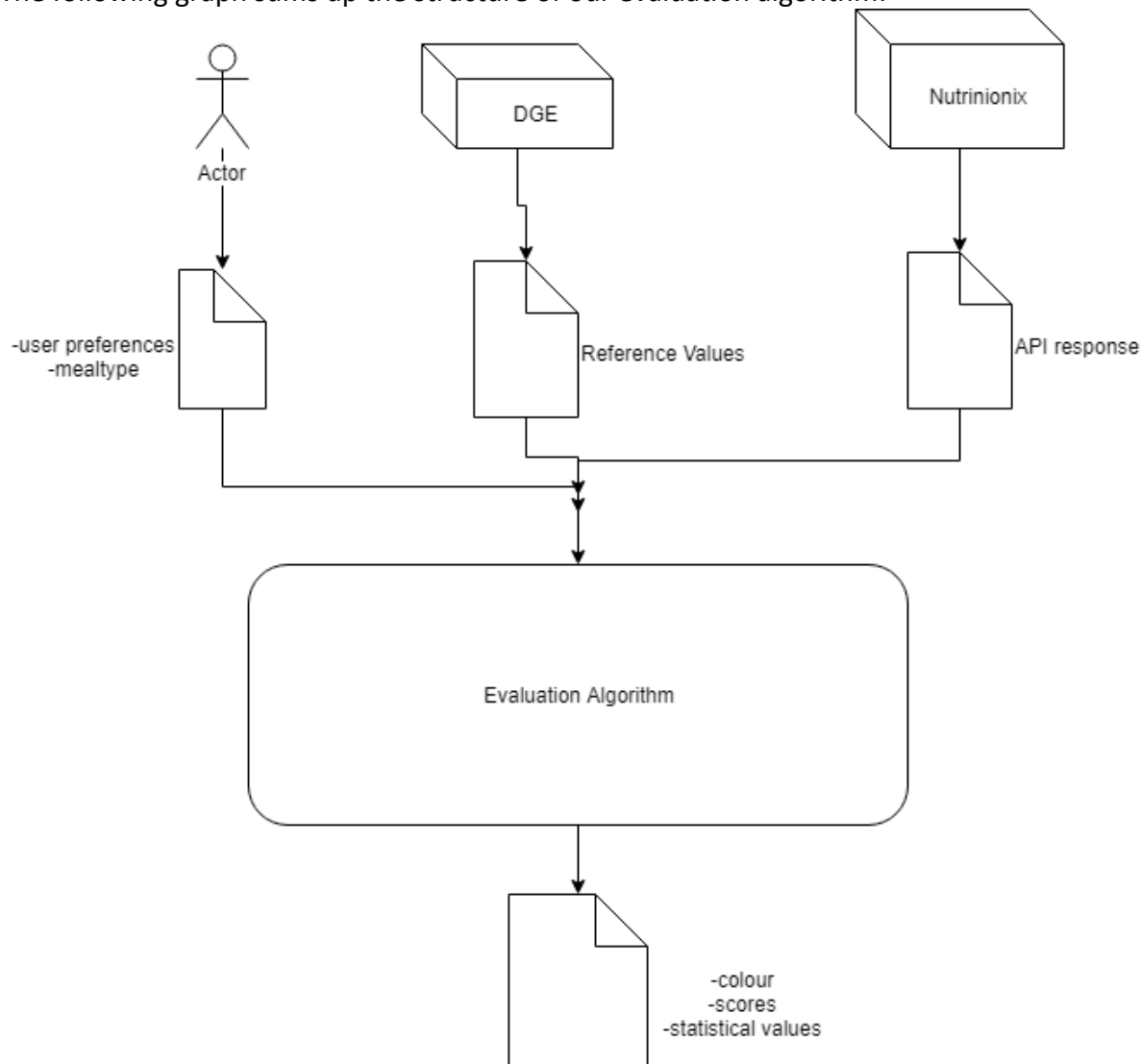


Figure 7: Visualized input/output structure of the evaluation algorithm

Further challenges: The API response

We experienced further challenges in the Nutriotionix API response. Lots of nutritional values the API provides for dish are encoded in an attribute id. Attribute id 309 represents Zinc for example. This required us to identify encoded values manually by crosschecking test food on other databases in order to match the attribute ids to the nutritional value they represent. Furthermore, the Nutrtionix API response contains different units than our evaluation algorithm in some cases. This required us to implement a method to transform units in the right input format for our evaluation algorithm.

2.2 Solution 2: Identify nearby, healthy restaurants and give users recommendations automatically

Keeping a restaurant and menu database

Finding a solution to this task required us to keep a database of restaurant that gets visited restaurant by any of our users with their scanned menu. This way, we can easily save restaurant location and dish ratings in order to give other users recommendations when they are nearby. One Requirement is that the user has to have GPS activated when scanning the menu. Problems may arise when multiple restaurants are too close to each other for us to identify the specific restaurant. In this case, we either ask for the restaurant name or do not add the menu to the recommendation database.

How users can receive recommendations

The manual way for our users to receive recommendations at any time is to open the corresponding tab on our app. This shows each nearby restaurant with their kitchen type, distance and an option to show the menu. Opening the menu shows the exact same page as if the users would sit in the restaurant and scan the menu manually with our App.

However, we also mentioned that we want to send the user recommendations automatically as we cannot expect him/her to open our app anytime he/she makes a decision on where to eat. Also, we assume that most of these decisions are not actively made anyway. Thus, we have to gain the users attention. We do this by sending push notification prior to the usual lunchtime of the user. We can easily get access to the typical lunchtime by analysing when the user usually scans a dish with our app. We can also consider the user's favourite kitchen type (such as Italian) to present a healthy dish that the user is very likely willing to order. We finally send a highly personalized push notification with a nearby restaurant and a picture of a green rated dish¹ that the user likes at the moment he/she is about to go to lunch. This ensures the highest possible impact of our push notification, giving users no excuse to not eat out healthy.

¹The dish is only recommended to the user if it is also marked as a green dish according to the individual diet preferences of this particular user. We only score dish names of restaurant menus. The scores get calculated again for each user considering their diet preferences. When opening the push notification the user will land on the restaurants recommendations tab.

The following chart gives an overview on how the push notifications work:

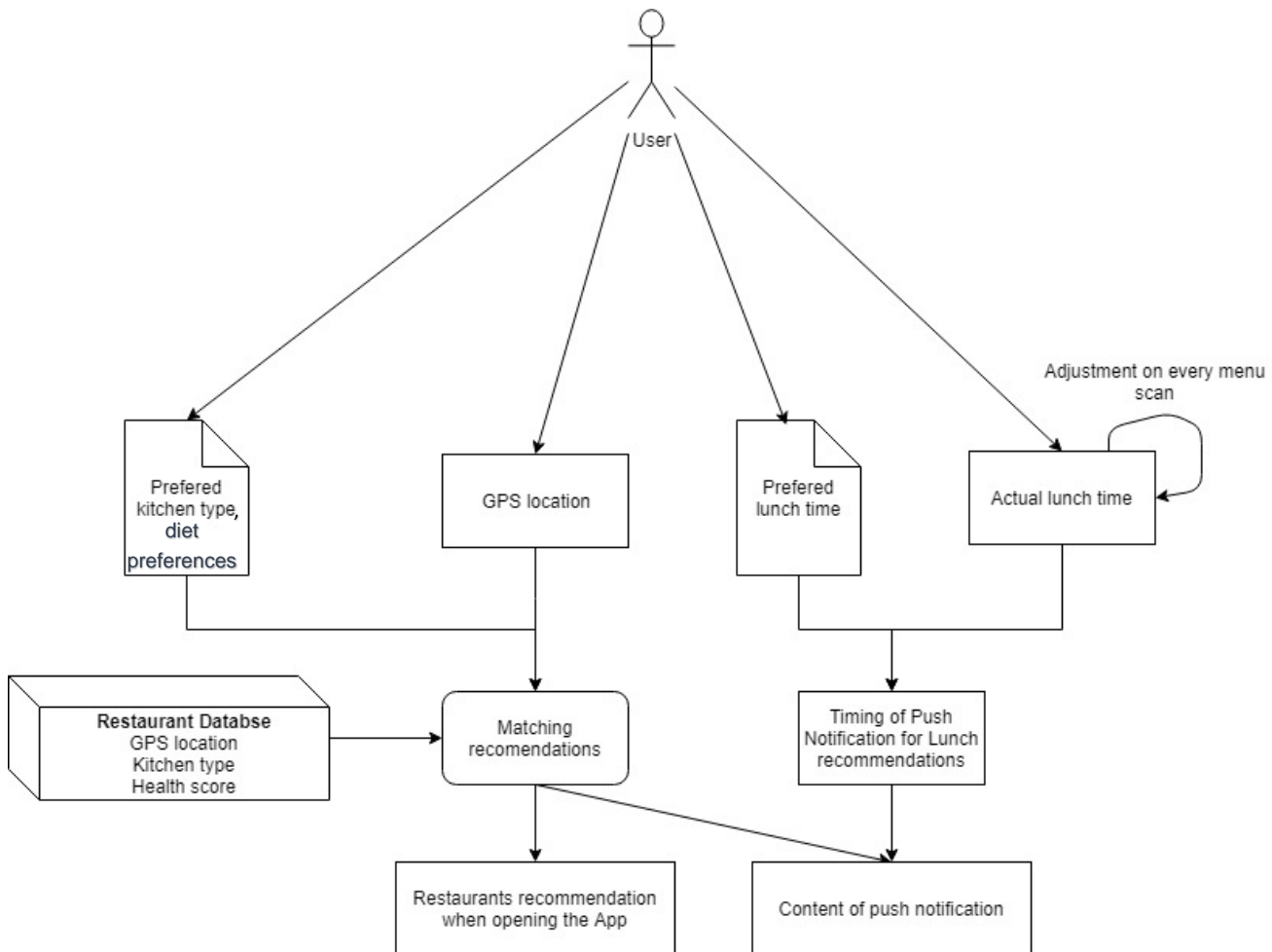


Figure 8: Visualized input structure of restaurant suggestions

2.3 Solution 3: Provide more detailed, optional information and education on each dish

Even though we evaluate 38 nutritional values for each dish, we initially only display the user one font colour according to the normalized total score. As discussed, we can help fixing current misconceptions by providing additional information and explanation on dish ratings. Some users are highly interested in more detailed information on each dish. A simple tab on the scanned dish reveals a detailed statistic page where once can take a look of every 38 values presented in a quickly understandable format. As this feature is more closely described in the Functionalities chapter, we will focus more on the educational aspect of the statistic page. On top of the statistic page, we present the most important reason which lead to a dish receiving a recommendation (green score) or no recommendation (yellow, red) score. This is a quick way to educate users and make them reflect on dish choices. The comment is based on the best or the worst sub-rating about the dish. If for example the fatscore was the highest score of a green rated dish, we display a comment, which informs the user that fats contained in this dish are exceptionally healthy. In order to display a positive comment the sub-score has to be over 110. This ensures a large health benefit over the standard score to justify a positive comment. Likewise, negative comments only get displayed for a sub-score below 87. On

top of this, users can navigate deeper in the statistic page, receiving information on each evaluation criteria such as each specific vitamin or mineral with their function explained as well as listing most common sources for that vitamin/mineral. This can be used by highly interested users to increase their knowledge on healthy dishes.

We will go in more detail on educational and informational features of our App in the end of this chapter and in the Functionality chapter.

Generating comments for a dish

In this paragraph, we are going in more detail on how the comment is generated. Internally, we generate up to two comments. As the UI would look too overloaded with two comments, we limit the visual representation to the most important comment on a dish.

Generating a positive comment

As previously described, an overall green rated dish has to contain at least one sub-score with a rating of 110 to receive a comment. After identifying the sub-score with the highest rating, we can already generate the comment:

Vitamin Score, Mineral Score, Fat Score, Fiber Score, Macro Score (can only be larger than 110 if there are high amounts of protein in a dish) is the highest:

Comment = "This dish contains a high amounts of [Vitamins, Minerals, Fiber, Proteins, Healthy fats]"

The following case cannot exist: Sugar score is capped at 100. Thus a dish cannot receive a positive comment for a low sugar score. This is intended, as a low sugar score is no exceptionally healthy attribute and is met by many dishes.

Generating a negative comment

As previously described, an overall yellow or red rated dish has to contain at least one sub-score with a rating of 87 or lower to receive a comment. After identifying the sub-score with the lowest rating, we can already generate the comment in the following cases:

Mineral Score (can only occur if dish contains too much sodium), Fat Score, Fiber Score, Sugar Score is the lowest:

Comment = "This dish contains too [much, many] [Sodium, unhealthy fats, fiber, sugar]."

The following case cannot exist: Vitamin score is capped at 100. Thus, a dish cannot receive a negative score for not containing a specific vitamin.

If Macro Score is the lowest sub-score, there are two possibilities: Either the bigger problem is a too high proportion of fat or the bigger problem is that the proportion of carbohydrates is too high. For this situation, we have to request more details by a specifically designed method for this case from our evaluation algorithm. The evaluation algorithm stores the negative points each dish receives for too high proportions of a nutrient. Using a get request for this array is necessary to generate the comment. Further, we need information on the meal type and recommended proportions to present a meaningful educational comment to the user.

Comment = This dish contains too much [carbohydrates,fat]. An optimal [breakfast, lunch, dinner, snack] should contain at most [X%] carbs, [X%] fats, [X%] proteins.

For the exact comment generation one can review the `getComment()` function of the `EvaluatorUtils` class. Multiple examples for generated comments can be seen in the appended video.

Providing educational information on any nutritional value

The user can also receive educational information if we display information to any nutritional value the user does not know. The idea is that the user just tabs on a nutritional value to get more information on it. A tab on fat score in the statistic tab of the UI could explain which types of fats are healthy, what health benefits or consequences different types of fat have and recommendations for food that contain healthy fats. A tab on any vitamin or mineral in the UI could explain the function of it and show sources that contain a lot of it. This feature is already implemented in the evaluator class but did not make it to the UI to a lack of time. The following example is from the `Evaluation` class and shows how the information will look like. For more detailed information, one can review the code of the `Evaluation` class.

```
explanation[0][0] = "Main Function: Vitamin A is important for vision. It helps the retina of the eye to function properly, particularly at night. Night blindness is an early sign of deficiency and blindness can result if preventative steps are not taken. Vitamin A also aids in maintaining healthy skin and it is sometimes used in the treatment of acne. It also plays a role in the growth of bones and it helps to regulate the immune system and fight infection.";
explanation[0][1] = "Source: Vitamin A is found in deep orange and dark green fruits and vegetables such as carrots, broccoli, kale and spinach. Eggs also contain vitamin A.";
```

Displaying a picture for each dish

Primarily to make the UI more appealing, we display a picture for each dish. The Nutritionix API also contains an image link for most dishes that we use to display the image next to the scanned dish. In case we do not find a picture for the dish, we are using a placeholder image.

Displaying a picture for the user can also have a slight benefit if the user does not know a certain dish. Looking at the picture and reviewing its statistic page can be enough to form an opinion on it.

2.4 Detecting dishes in a menu

Challenges of a typical menu

Ingredients should not be falsely detected as a dish (points to 'Coming with the dip of your choice' under Ham and cheese sandwich)

The prefix "Extra" should not affect the process of finding a match for "breadsticks" (points to 'Extra breadsticks')

"Salads" is a food category in this case and should not be listed as a dish in the output (points to the 'SALADS' header)

The two prefixes "Home" and "made" should not affect the process of finding a match for "quiche" (points to 'Home made quiche')

Noise after the dish name (12pcs) should not affect the process of finding a match for "Tuna sushi" (points to 'Tuna sushi (12pcs)')

Exotic dishes should also be detected (points to 'Nasi Goreng')

Also these ingredients should not be falsely detected as a dish (points to 'With egg, veggies and dip' under Fried Rice)

STARTERS	
Ham and cheese sandwich	\$4.00
Coming with the dip of your choice.	
Chicken soup	\$5.00
Seasoned with homemade herb mix of tomato, leaves and basil.	
Extra breadsticks	\$4.00
We always serve one plate of breadsticks as a starter for each of our customers.	

MAIN DISHES	
Home made quiche	\$9.00
Home made cheese & seasonal vegetables.	
Tuna sushi (12pcs)	\$18
We only serve fresh Tuna.	
Vegetable stew	\$6.00
Wholesome, hearty vegetable stew.	
Nasi Goreng	\$10.00
Made with our traditional Indonesian recipe.	
Fried Rice	\$8.00
With egg, veggies and dip.	

SALADS	
Fresh chicken salad	\$8.50
Delicious salad with iceberg lettuce and ranch dressing. Croutons included.	
Egg Salad	\$7.50
Delicious salad topped with tomatoes, romaine lettuce, green peppers and your choice of dressing.	

Figure 9: Challenges of a typical menu

In figure 9, one can see common difficulties in accurately identifying dishes and separating them from words that only appear to be dishes. In addition, character noise around the dishes should not lead to a missing or false detection.

To extract the text from a menu in the first place, we are using the Google Cloud Vision API. The text response we receive from the API is not analysed in any semantic way. This means we also had to develop an algorithm that consistently detects dishes in the API response and solves above visualized challenges. In the following paragraphs, we will show this procedure.

Explaining our dish detection methods



Figure 10: Bounding boxes recognized by the Google Cloud Vision API

Above, one can see visual example of a typical Google Cloud vision API response. Nearby text gets grouped in boxes. As one can see, there is not a lot of free space between “E-mail” and dwilson@brattinc.com, yet the texts are still put in a separate box. This is beneficial for our use case as restaurant dishes would have to be unrealistically close to each other to end up in the same box. Thus, we can assume that each row inside a box contains at most one dish. This is a crucial attribute, which made the Google Cloud Vision API our choice for text detection in restaurant menus. Each box is separated by a “\n” from the other boxes and each row inside each box is also separated with “\n” from its next row. By separating a whole output string of the Google cloud vision API with the regex “/n” we get an array of each row inside the image. Here is an example of the first lines of output from a restaurant menu: 12\nSOUPS\nAll our dishes are gluten free!\nAM\nmushrooms, tomato, cheese\nCrispy breadsticks\nCrispy breadsticks\n47. Sweet Corn Soup\nSweet Corn Veg Soup..\$4.20


Seperating by “/n” we receive an array that looks like this:

Table 2: extract of the first few characters from a Google API response, seperated by each line

Array index	String contained
[0]	12
[1]	SOUPS
[2]	All our dishes are gluten free!
[3]	AM
[4]	mushrooms, tomato, cheese
[5]	Crispy breadsticks
[6]	Crispy breadsticks
[5]	47. Sweet Corn Soup
[6]	Sweet Corn Veg Soup..\$4.20

In this array, we are only interested in actual dishes. Thus, we had to develop an algorithm, which extracts only dish names from the menu. In the following example, we explain the challenges of each String that our detection algorithm has to solve.

Table 3: Challenges of processing each candidate

String contained	Challenges
12	No dish contained. We do not want to waste network and computing power by making an API call for this string.
SOUPS	This is a category and not a dish. We need to erase categories like Appetizers, Salads, Soups, etc, to avoid displaying dish results to the user that are not orderable
All our dishes are gluten free!	Long text are highly unlikely to be dishes. In order to not make unnecessary API calls, all texts that exceed a certain length should be deleted.
AM	Likewise short single words are highly unlikely to be words and usually false detections by the Google API. In order to not query unnecessary API calls or let the autocomplete of the Nutritionix API show results for dishes that are not on the menu, those strings have to be detected and deleted.
mushrooms, tomato, cheese	<p>These are ingredients of a dish.</p> <p>The algorithm has to detect typical ingredients structure and do not include them in the results. We do not want the user to browse through tons of ingredients when the user only wants to receive dish ratings. This should not be a result of our menu scan:</p> 
Crispy breadsticks	The word "Crispy" might cause problems in the dish detection, as the API will most likely not have a match for "Crispy breadsticks". It will however have a result for "breadsticks". We need to detect typical Prefixes like fresh, spicy, crispy,... to maximize chances that we receive an output for each dish.
Crispy breadsticks	<p>This duplicate is likely caused by a smaller box inside a larger box. If ones looks at Figure 10 again, one can see that the word "Inc." is contained in the outer box as well as the inner box, thus appearing twice in the result string.</p> <p>In order to not query unnecessary API call we need to detect and delete duplicates</p>
47. Sweet Corn Soup	There can be a lot of unwanted characters in a row before the dish name actually starts. Very typical are numbers in front of each dish so that customers can order in restaurants by the number. We need to remove any character noise prior to the dish and start with the first actual word detected.
Sweet Corn Veg Soup..\$4.20	The same situation can happen at the end with price tags or additional information in brackets. We have to be able to remove any character noise at the end, which is not correlated to the dish.

We are able to remove all mentioned challenges in Table 3 with our dish detection algorithm. For simplicity purposes, we only explain all relevant methods for this particular example. The colour of each below listed method is matched with the colour of the challenge this method solves.

Table 4: Relevant function calls for this example explained

Function Call	Explanation
<code>String[] candidates = menuText.split("\n");</code>	Splits the long string result from the API into an array containing each box row.
<code>removeNoiseAtStart(candidates[i]);</code>	Removes character noise at the start of the dish. Transforms “47. Sweet Corn Soup” into “Sweet Corn Soup”.
<code>removeNonAlphabeticCharacters(candidates[i]);</code>	Removes non alphabetic characters in a string and checks how to proceed with it. 1) Detects ingredients: “mushrooms, tomato, cheese” by “,” and deletes the string. 2) Detects a price tag after “Sweet Corn Veg Soup..\$4.20” and transforms it to “Sweet Corn Veg Soup”. 3) Detects “12” and deletes the string.
<code>wordsWithMinLength(candidates[i], 3);</code>	Deletes words that are not long enough to contain a dish. Currently those are words with a length lower than 3. Deletes “AM”.
<code>removeCommonpreFixes(candidates[i]);</code>	Compares each word at the start recursively with a list of common preFixes like “Fresh”, “Crispy”, ... Transforms “Crispy breadsticks” to “breadsticks”.
<code>removeTooManyWordsInString(candidates[i], 4);</code>	Deletes a long text that is very likely to not contain dishes. Deletes “All our dishes are gluten free!”.
<code>removeCommonNonDishWords(candidates);</code>	Compares each word at the start recursively with a list of common not dish related words such as “Starters”, “Desserts”. Detects “SOUPS” as a food category and deletes it.
<code>removeDuplicates(candidates);</code>	Removes duplicates before making the API call with multiple candidates, to save network and computing power. Detects the second “breadsticks” (got transformed from “crispy breadsticks” to “breadsticks” already) and deletes it.

After running all of these matches, we receive the following final candidate array:

Table 5: Final candidates

Array index	String contained
[0]	Sweet Corn Soup
[1]	Sweet Corn Veg Soup

As one can see, we successfully extracted all dishes of the example. Only two strings of the seven initial strings contained dishes on the menu. Thus, it is highly necessary to use our implemented methods.

Handling non-dish strings that are not detected and deleted by our algorithm

Let us assume even with all our methods we would not have been able to detect a string that does not represent a dish. Let finalCandidates[2] be “April Fools”. This string has a not distinguishable structure from any other dish like “Spring Roll” for example. Only semantics differ. As our dish detection algorithm does not make API calls, it only detects syntactic attributes of dishes and non-dishes as well as a limited number of semantic attributes.

However, even this case is not a problem. Making an API request of a string that is not a dish can only result in two outputs: 1. No result was found 2. The autocorrect of the API transforms the string into a valid dish

The first output is not a problem, as we do not display empty results to the user. We do not however, want the second output possibility to occur and display the user a dish result that is not contained on the menu. Thus, we only include exact matches in the results. The query string “April fools” gets compared to the output string after our API call and gets deleted in any case. Table 6 illustrates all possible scenarios that could occur.

Table 6: Possible scenarios of API response

String contained in final candidates	Food name output of the API response
Sweet Corn Soup	Sweet Corn Soup
Sweet Corn Veg Soup	Sweet Corn Veg Soup
April fools	[Empty String]
Prizza	Pizza

Results are only displayed in the UI for “Sweet Corn Soup” and “Sweet Corn Veg Soup”. “April Fools” leads to an empty response and does not get displayed. “Prizza” gets autocorrected to “Pizza” by the API. With a string match, “Pizza” does not equal “Prizza” and thus does not get included in the final results.

Dish detection summary

The worst case would be to present many words to the user that are actually not dishes and let the user do the dish detection by himself/herself.

The combination of the mostly syntactical analysis with our developed algorithm and the semantical analysis executed after an API response it is guaranteed that we reduce API calls and only get results for actually existing dishes. There is no possibility that a dish result will end up as a result to the user without being an actual item listed on the menu.

There is only the low possibility that dishes are not detected due to

- 1) Failing detection of the Google API (does occur rarely, bad lighting, etc)
- 2) Very untypically boxing by the Google API (has not happened to us this far)
- 3) Dish gets mistaken for a Non-Dish by the algorithm and deleted (has not happened to us this far)
- 4) Dish is not contained in the Nurtitionix API (largest risk)

We want to emphasize, that in our tests nearly all dishes from typical English written menus were detected. Also, not detecting a few dishes is not a big problem. If the majority of dishes are detected, we are still able to display a healthy dish to the user that he/she may like with a high chance. Users can always use our manual search function to search manually for dishes that are not detected by us or not written on the menu.

3 Functionalities

(We recommend to also watching our appended video that explains App-flow and functions)

The highest priority development requirement according to our survey was that users do not want to spend a lot of time using a dietary assistant app. We developed all functionalities with keeping in mind that we have to present information at one glance as quickly as possible to the user. Thus, we need to make the process of selecting a dish and finding a healthy restaurant take minimal time.

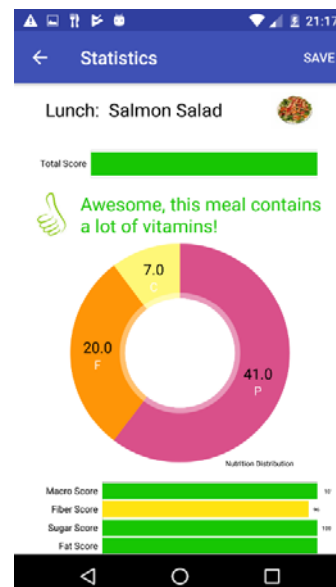
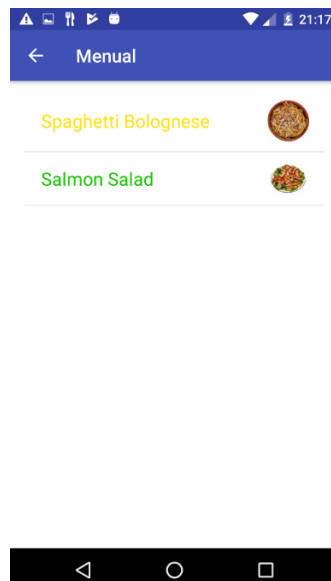
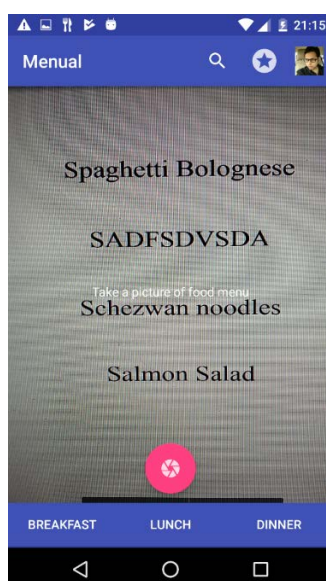
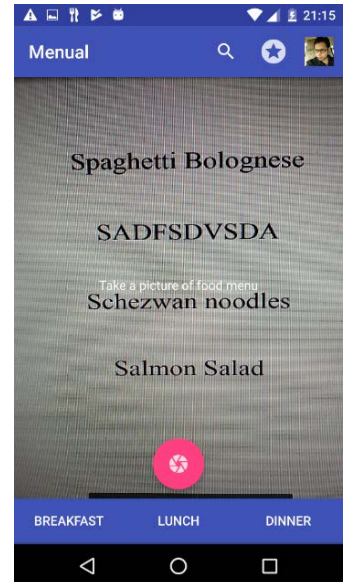
Main tab

For this reason, after the very first login of the user, the App immediately starts in the camera view. With only one tab on the camera button the user can receive ratings for each dish in a few seconds.

On the bottom one can see the selection for breakfast, lunch and dinner. By default the App will use the current time to anticipate which kind of meal the user is about to scan. The user can also specify the meal type manually by tapping on an option. By clicking on the star symbol at the top, the user will be directed to the restaurant suggestion and history tab. Tapping on the user's Google profile picture he/she can change diet preferences or switch account. The user can also search a meal by name manually by tapping on the search button.

Scanning a physical menu

Tapping on the camera button makes a picture of the menu and sends it to the Google Cloud Vision API. The response gets processed in our dish detection algorithm to detect each dish on the menu. Each dish name is then sent to the Nutritionix API to receive nutritional information. This information gets processed, structured and evaluated by our evaluation algorithm and rated by a final color (red/green/yellow). Finally, each detected dish with its rating and a picture of it is shown in a list. Due to this intuitive layout, the user can have an overview of the menu in seconds to form a decision. The user can tap on any dish to view advanced statistics and can proceed to tap on the save button to add the dish to his/her history. This way an interested user can optionally browse through detailed statistic of multiple scanned dishes before making the choice. A busy user can skip reviewing the statistic page and immediately save the dish to history.



Left Screenshot: Main tab

Middle Screenshot: After taking picture, menu is scanned. Shows a list of each detected dish with rating and picture.

Right screenshot: Tapping on any dish reveals statistics, total score, sub-score and up to one comment

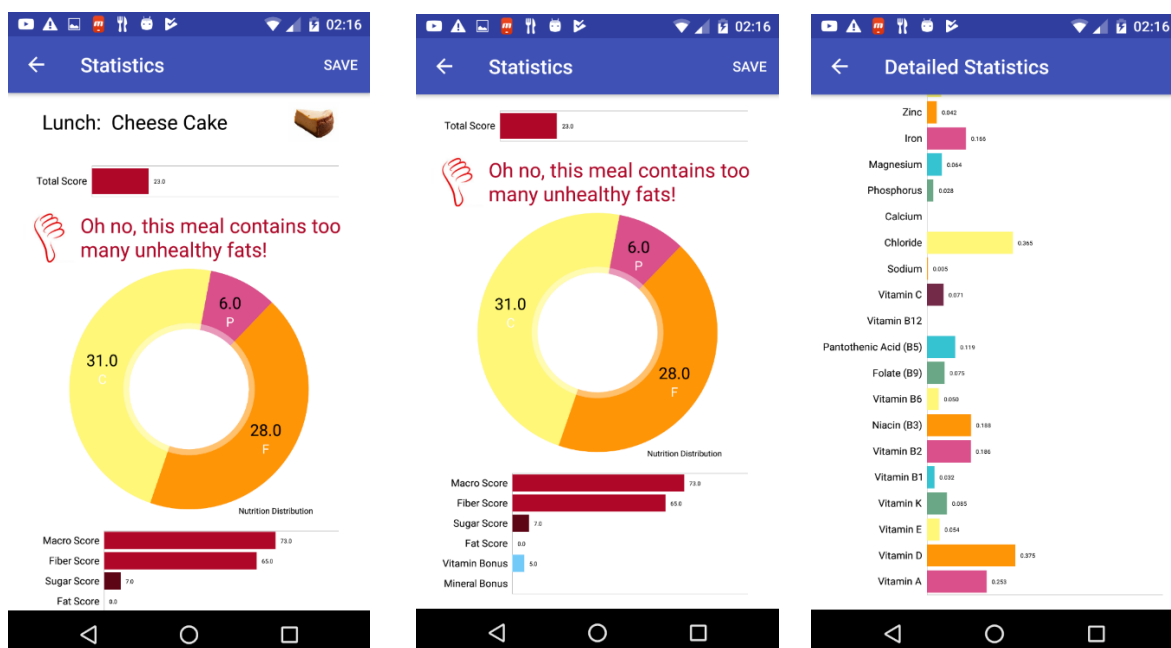
Food nutrition statistics

Our third development task that was based on our survey results and literature findings, demands to provide more detailed, optional information and education on each dish.

Informational Content: We provide information for the user by displaying information of each score and nutritional value that we evaluate with our evaluation algorithm. At the very top, we show the total score with its interpretation color (red in this case) as the most important score on the dish. This (normalized) total score combines every other score, user preferences and all national information evaluated in one score and color. The next informational content is the pie chart below that shows the proportion of each Macronutrient. Below, the user can find the corresponding macro score that is based on evaluating this proportion. Following are all other 5 sub-scores with a color and a value representing how the dish performed in each category. In this case, the Cheese Cake has too much sugar and fat, resulting in a red macronutrient score and not enough fiber. The sugar and healthy fat score are exceptionally bad at 7.5 and 0. By scrolling down one can reveal the other scores. Tapping on a score opens detailed information as is shown in the picture on the right. In this case, one can see the percentage of each vitamin and mineral that is contained in the dish, with the value representing their percentage of recommended daily intake.

Educational content: In order to educate the user it is important to explain why dishes are healthy or unhealthy and how certain attributes of dishes can affect the body. For a quick educational feedback, we generate and show a comment about the most important fact that made the dish receive a high or a low rating. In this case the Fat score (Value: 0) is the lowest score of all our sub-scores (Bonus scores start at 100, thus mineral score is 100 and bonus is 0). This way, the comment can tell the user the most important fact that lead to a bad rating.

Another educational feature would be providing information on each nutritional value, vitamin and mineral, its purpose and which food to eat that contains a lot of it. This information could pop up after tapping on a specific vitamin or mineral or a in the Detailed Statistics tab or any sub-score on the Statistic tab. All this information is already included and accessible in the evaluation class but could not be implemented into the UI due to a lack of time. Connecting the UI to the methods is the only step that is missing in order to implement this feature (see Methodology chapter for more information).



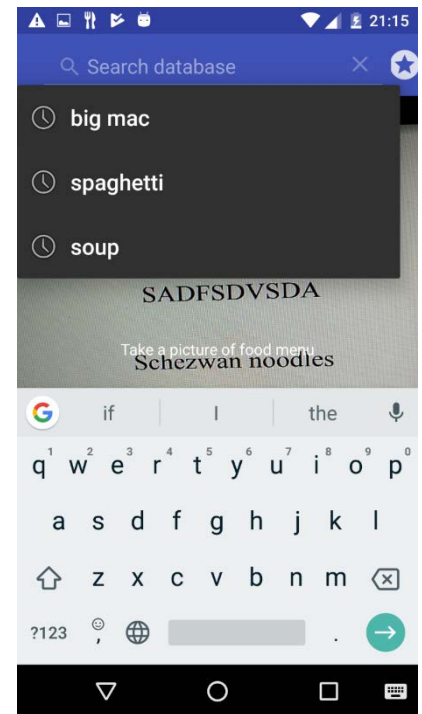
Left Screenshot: Statistic page pops up after tapping on a meal.

Middle Screenshot: Scrolling down to see more information.

Right screenshot: Tapping on any score or the pie chart reveals detailed statistics on that score.

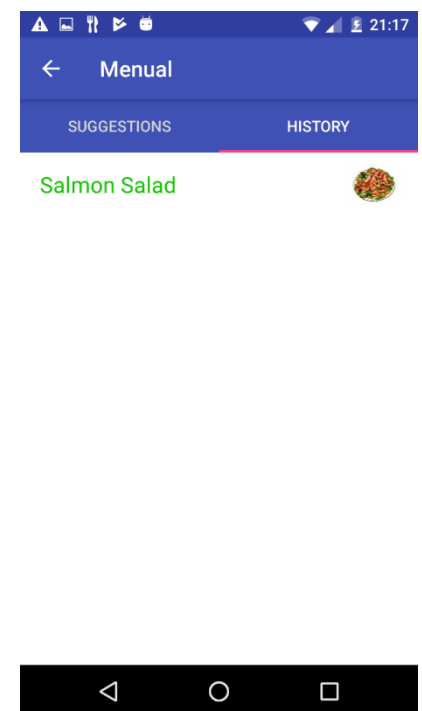
Search meal by name

In case a food item cannot be detected on the menu or a user just wants to look up nutritional information on an arbitrary dish, the user can use our search function. Via the search bar on top of the main screen the user can search a dish manually and receive nutritional information for a single dish. As always the dish can be saved to history and checked for detailed statistical values as shown in the “Food nutrition statistic” functionality.



History

The user can save any scanned food to his/her history in order to review the overall quality and decisions of previously ordered dishes. This way, the user can look up detailed information on previous dishes at any time or save dishes on restaurant menus for later reviewing.

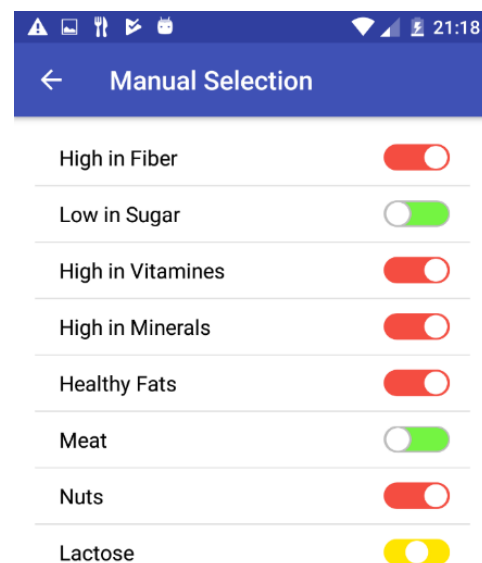
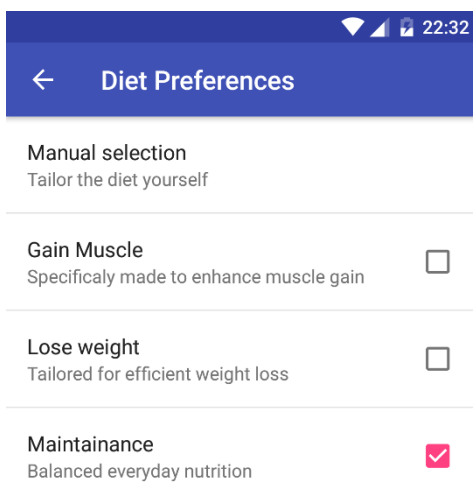


Diet preferences

Diet preferences differ among people. There may be intolerances for certain ingredients or personal restrictions. Also depending on different goals, users may want to over-prioritize or under-prioritize certain attributes of their dish. A user that is on a diet might be interested to punish high amounts of sugar harder than our algorithm does. Thus, a user can set his/her dietary preferences in the settings. The search results will then be customized accordingly.

The standard selection for each option is yellow, which means our algorithm is not affected by preferences. Green means over-prioritize for scores (Low in sugar), or explicitly allowed for restrictions (Meat). Red means under-prioritize for scores (High in fiber) or not allowed for restrictions (Nuts). For a restriction, there is no difference between yellow and green.

The user can currently also choose from two different goals with “losing weight” and “building muscle”. Tapping on one of those options automatically preselect each preference. We plan to add more profiles/goals in the future.



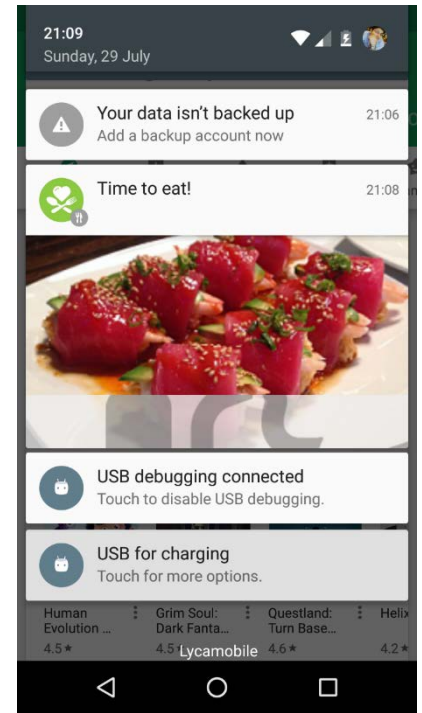
Opening diet preferences let the choose from the different goals, our standard option or to manually select preferences



After Tapping on Manual selection, shows this screen to let the user choose preferences manually.

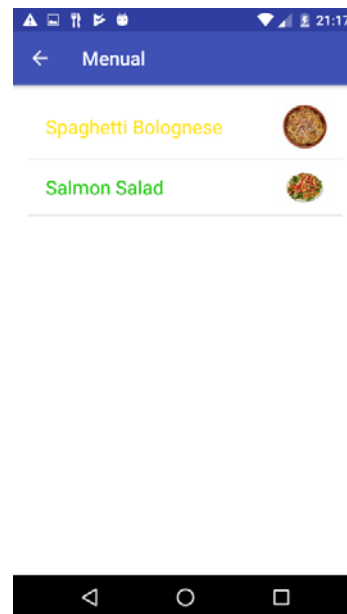
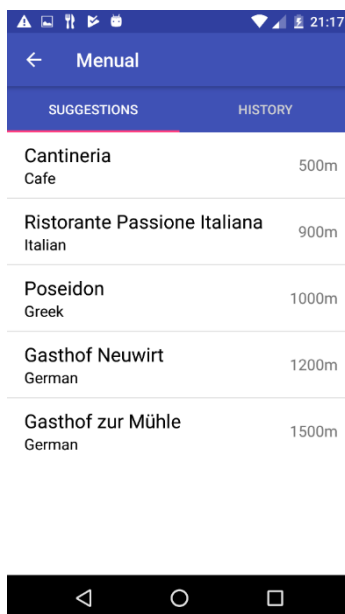
Meal time notification

Task 2 of the problem definition section required us to give users recommendation automatically. We cannot expect the user to open the app manually every time he/she wants to eat out. Thus, based on user location and before his/her usual dining time we send a push notification with a healthy dish the user is most likely willing to order (based on preferred kitchen type). In case during dining time, our app detects a nearby restaurant that offers a healthy dish the user is likely willing to order, the user receives a push notification. Tapping on the notification will open the suggestion tab, showing the correlated restaurant, its distance to the user as well as each dish on its menu with rating. As we did not initialize an actual user generated restaurant database up until this point, the picture and the corresponding dish is a mockup.



Restaurant suggestions

Our app suggests its users nearby healthy restaurants. Users can access this tab manually from the main page or by opening a push notification. The restaurants are sorted by distance to the current user location and also contain a kitchen type (Italian, Turkish, etc). When user taps on a restaurant, the restaurant's menu items are listed in the same tab as if the user would have scanned the menu of the restaurant by him/herself. The user can then make an informed choice whether he/she wants to visit the restaurant. As we did not initialize an actual user generated restaurant database up until this point, the restaurants and their corresponding menus are mockups.



Left Screenshot: Restaurant suggestions with kitchen type and distance to the user

Right Screenshot: After tapping on any restaurant, displays the restaurant menu with dish ratings

4 Architecture

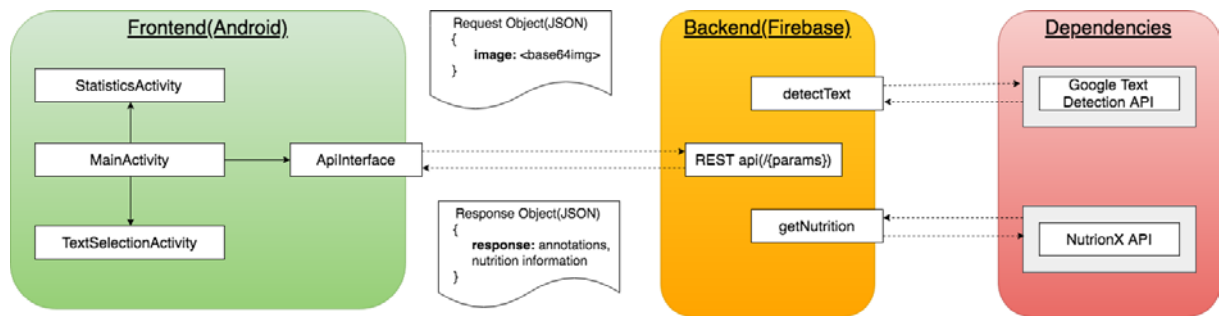


Figure 11: Architecture of our Android App

Our application consists of a 3 layered architecture with an android based frontend, a firebase server less backend and 3rd party APIs for text recognition and getting nutrition information based on food item names. The above diagram summarizes the entire architecture.

The frontend has a main activity where the user captures the picture of the menu to scan the food text from it. It also has a search bar on the top for typing the food name in case taking a photo was not possible. `TextSelectionActivity` displays the detected food items in a list form with each item marked as red, yellow and green based on their nutrition category. `StatisticsActivity` displays the summary and the detailed information about all the nutrients available in the food item. All the activities interact with the backend using an API interface, which holds the signature of all the backend endpoints.

We do not host a server on our end. Rather, we use a server less backend based on firebase with two endpoints. One for detecting the text from the picture and the other for getting the nutrients for each food item text. The request and response is a JSON object and is parsed on the client side before showing on the UI. Our backend interacts with Google Text API for receiving the bounding boxes around each text section of the user image with contained text. These text boxes are analyzed by our internal methods to extract dish names. These are then sent to another 3rd party API, Nutritionix, which then sends the nutrition details. Our internal methods evaluate the API response to generate ratings and structure nutritional information.

5 Key responsibilities

During the development of the App, we assigned the following key responsibilities:

Andrei: UI, design, adding features, coordination

Shayan: UI, setting up App architecture, doing API requests, firebase setup

Christopher: research, processing API responses, business logic: Development of dish evaluation algorithm, development of dish detection algorithm

6 Results of the evaluations

We had two evaluations during the project, which were very helpful to get feedback on our progress and receive recommendations on what to implement next.

First evaluation

The first version of our app demanded the user to tap on the dish on the picture of the menu to select it and get results only for that specific dish. We got the feedback that we should find a way to implement that taking a picture of the menu will automatically detect every dish of the menu to display all results at once. This was a very challenging task, as it required us to develop our own dish detection algorithm that recognizes every dish on the menu and only dishes on the menu. The full description of challenges and solutions to this task can be found in the Methodology chapter. Before the second evaluation we finished this task. Our dish recognition algorithms can detect dishes from any menu and filter out ingredients, non-dish related text and other noise.

Second evaluation

Before the second evaluation, we finally finished everything on the logic side. This was by far the greatest challenge of our project, as it required us to do a lot of literature research to develop a framework for evaluating a dish, making multiple API calls that can help us for database access and image recognition and finally developing our own dish recognition algorithm to detect all dishes inside a menu. Thus, the UI was not very developed until this point. We got the task to finish the UI with every functionality we planned to implement, as well as presenting images for each dish for a more appealing look of our app.

We managed to implement every functionality in the UI including push notifications, preferences, menu scans, results, recommendations, user login, manual search and history. We also provide images for each dish to display a picture next to the dish in the UI. Additionally, we implemented a detailed statistic page where a user can tap on a scanned dish to receive information on all of our 38 nutritional values we evaluate in our algorithm. This data gets sorted and displayed in an appealing way to give the user as much information as possible if required. In addition, we added a comment to each dish to educate the user and explain the most important reason why a dish scanned is healthy or unhealthy.

Thus, we implemented all features recommended at the evaluation in our app and also added mentioned additional features that we found to be useful on top.

7 Discussion of limitations

We consider our project rather ambitious, which was also confirmed by the realise-ability score at our idea presentation. This necessarily comes with limitations in the quality of our features. At this point, we rely on one API to give us accurate information and due to a lack of frameworks and specific information in the literature, we had to make a lot of assumptions in the evaluation process of a dish. Also, as we use a US databases, our App should be very accurate for dining out in the US. It remains to be verified whether this remains to be accurate for restaurants in other countries. In the following paragraphs, we discuss where different functionalities are limited in our App.

Limitations in the evaluation of dishes

There is a very straightforward limitation of evaluating a dish in our algorithm: It has to be contained in the Nutritionix food database. As soon as it is not contained in the database (or for the future in any other food database), we cannot evaluate it.

Another limitation is the completely scientific evaluation of dishes. Until now, the literature to this topic only contains tendencies but not specific weightings on how important different nutritional categories can be aggregated to a final score. This is the reason we had to develop the specific weightings our self by interpreting vague to more specific implications that are contained in the literature. For example we translated medium relevance for fiber in a weighting of 1, medium to high relevance for macronutrients proportions to a weighting of 2 and high relevance for healthy fats in a weighting of 3. This added interpretation requires us to be extremely strict on evaluated dishes in order to be certain to not recommend an unhealthy dish to a user.

At the end of our final presentation we were asked the question what we think of including ingredients to the evaluation. This can, if scientifically accurately done, definitely increase the accuracy of the evaluation. However, this is an extremely challenging task, as it would require us to develop an even more complex framework that matches ingredients to the food via text detection and finds a way to combine total scores of ingredients and the corresponding dish to one final score. At the same time, this final score should be equivalent to a menu scan where no ingredient information is available.

As further described in the future outlook chapter we would need scientific guidance and a lot more time to overcome these limitations. We are confident however, that our evaluation algorithms delivers reasonable and accurate evaluations already.

Limitations in the dish detection of menus

Again, a dish has to be contained in an online food database in order to recognise it. Otherwise, it will get deleted from our dish candidate list in the last step. As discussed at the end of our final presentation, we could also imagine a menu where our algorithm would not be able to detect a dish when fonts are that close to each other that two dishes would end up in the same “box” that is forwarded to us by the Google API. However, while this is a theoretic possibility, this situation does not occur on a menu, as fonts would have to be unrealistically close to each other. Thus, we do think that the dish detection is already very advanced and is practically only limited by the comprehensiveness of our food database. In our test, we regularly detected every dish on different menus, even on big menus with a lot of noise.

Limitations in the restaurant recommendation system

In order to always be able to recommend a healthy restaurant to the user, we need a large database of different resultants. Each time a user scans a dish on a restaurant menu and has GPS activated, our restaurant database can get larger. The limitation to this principle is that it only gets better over time and only comprehensive if a lot of users use it. Currently this method or asking restaurants to send us their menus is the only way however to acquire this data. We are naming this an initial scaling limitation. This functionality

scales very well with a lot of users, increasing its own value automatically but needs a minimum database size to be effective.

Limitations conclusion:

In conclusion, we believe that the functionalities of our app are already advanced and only bear a few limitations. We can divide all perceived limitations in three different categories:

1. An expertise limitation: As mentioned, available literature lacks specific weightings of nutritional information and does not provide specific frameworks for evaluating dishes. To overcome these limitations, we would have to work closely with nutritional experts to make evaluation results more accurate and possibly include ingredients written on the menu to the total score.
2. A database limitation: We rely heavily on accuracy and comprehensiveness of our used food databases. While the database we use turned out to be very comprehensive and accurate, we cannot rate dishes that are not included in a food database or that contain inaccurate information.
3. An initial scaling limitation: Our recommendation system can only achieve its full potential if lots of users in different locations use the App. Every time a user scans any menu, the database gets larger. Up until a minimum amount of content, the database has limited use.

8 Outlook on further development

This chapter offers some insight into the desired future strategy of the app developed during this project. It comprises a more general overview and a concrete presentation of to be developed features.

Already implemented features and the next development step

As already presented, Manual is an application developed to tackle two of the biggest problems regarding individual nutrition in modern society: lack of time and the desire to eat healthy. The solution to this situation is focused on a clearly identifiable tendency in the urban society, which is that more and more individuals choose to eat in restaurants both for lunch as also for dinner. This addresses the main constraint, which is time, but leaves many facing the daily questions of “what” and “where” to eat. These questions can also be further split into four main directions: taste, price, healthiness and proximity.

Manual sets out to address each of these. It is desired to be an app that encourages healthy eating by being there for the user every step of the way during this process and providing all the needed information to accomplish this goal. The app already supports the user even before the opening of the app. For this, individual notifications based on usual eating times, food interests, dietary preferences and available meals and restaurants haven been developed. Such notifications are a proactive measure on the application side to motivate the user daily. The user is offered all the needed information about the suggested meal, should he/she be interested on this particular possibility.

If the user chooses a different restaurant, the application can also support the user through the scanning feature. As different users have unique preferences, Manual allows the user to directly impact scores by restricting ingredients (no meat, no nuts, ...) or change weighting of the algorithm (punish high amounts of sugar harder,...). The user received quickly visible feedback on each dish and even detailed statistics on each dish of any unknown menu. The user can also further review the detailed statistics on each dish or review his/her dining history with informational and educational content. The scanning feature is not only an individual feature, as with the allowance of the user, the data can be saved in the restaurant database to enrich the list of suggestions to the entire user basis. This way each individual use of the app automatically increases the value for each other user. This user co-generated content leads to lower maintenance effort. The database is currently not initialized and relies on mockup data to offer a minimum database size. The next development step would be to set up the database to start collecting user data. As this is our only current mockup feature, this step would be most important to attend next. For the near future, we could also store age and sex of the user to use more accurate reference values. While this would only slightly change the accuracy of our evaluation algorithm, it is easy to implement. Due to a lack of time, we did not implement this feature already.

Future steps: Solving limitations

We identified three limitations in the previous chapter that can be solved with a long-term focus.

1. An expertise limitation: To further enhance accuracy of our evaluation algorithm we would have to work closely with nutritional experts to make evaluation results more accurate and possibly include ingredients written on the menu to the total score.

The first step would be to analyse each score, weighting and formula we use to calculate the score. Nutritional experts could correct values that we did not derive correctly from the literature. Alternatively, there could be literature in the future that provides a framework for our specific use case, thus providing exact formulas to calculate a total score by evaluating different attributes of a

dish. After that, we might be able to find or create a framework that also includes ingredients written on the menu to the total score.

2. A database limitation: Currently, we are only using the Nutritionix database. This means we are highly dependent on its comprehensiveness and accuracy. By utilizing more databases and comparing results, we could introduce a confidence score. If nutritional information of the different databases match within a tolerance interval we can give a dish a higher confidence score. We can decrease confidence level if information is conflicting. We could visualize the confidence score by using darker shade color font size for higher confidence and lighter shade color font size for lower confidence. A dark green dish font size would represent a healthy meal with a high confidence level. A light green dish font size would represent a healthy meal with low confidence level. Furthermore, using different databases can increase comprehensiveness of our database, especially if we use databases from different origins. These could increase the chance of containing regional specific values. The following flow-chart illustrates the evaluation process could look like in the future:

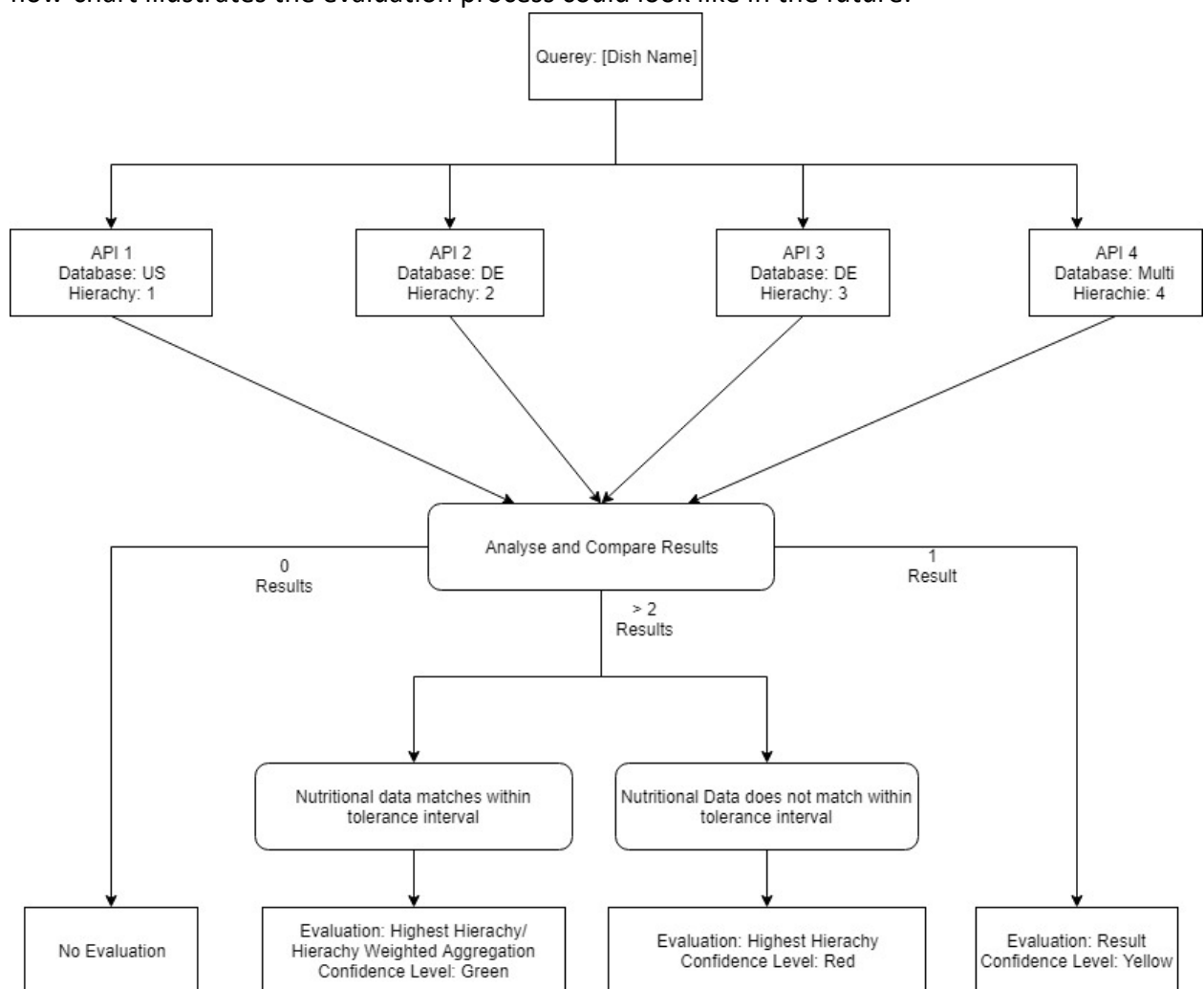


Figure 12: Adding confidence levels, evaluation process with multiple databases

3. An initial scaling limitation: Until a minimum amount of content, our restaurant recommendation database has limited use. A solution to this limitation could involve creating partner accounts for restaurants who can themselves populate the app on a daily basis. This way, healthy restaurants could receive free advertisement through our app by filling the database and being promoted by our app. A valid strategy might be to use the app only in a limited region that also a small database is

comprehensive enough to provide value to users and incentives for restaurants to make the effort and create a partner account. As our other main features also work without the restaurant recommendations, we could also consider deactivating the feature until we have a large database of user scanned menus and restaurants. The database automatically gets filled which each menu scan without requiring the recommendation feature to be available. After gaining popularity, restaurants would have more incentives to create partner accounts.

Monetizing our App

Finally, a monetization strategy could revolve around 3 major approaches. The first strategy involves billing partner restaurants a fixed amount per gained customer. This should be directly traceable by user input or through location tracking. This means that if the user is detected within the premise of a given restaurant and saves a dish of that restaurant to his/her history, we consider him/her to be a paying customer or customer with interest. The second strategy revolves around in-app advertising of products in the area of healthy nutrition. Finally, the third strategy could focus more on the user. Implementing the freemium model could offer additional functionality should the user choose to pay a monthly fee. One of these features could be individualized suggestions based on an actual nutrition expert. The expert could follow the user preferences and history to offer only those suggestions that best fit the given requirements.

Summary

If Manual will be developed into a fully functional product, all the above-mentioned features can be integrated to offer unique selling propositions to its users or possible restaurant partners. The next step would be to replace the restaurant recommendation database mockup with an actual running database saving scanned user dishes. After this step, all high priority long-term features would be implemented. One could then proceed to address current limitations by improving current features and adding new ones. Optionally, the monetization strategy could be implemented and lower priority features could get added. However, this optional step should be done carefully to not overload the app with features or harming user experience due to a conflict of interest.

References

European Journal of Clinical Nutrition volume 70, pages 97–103 (2016). Fast-food and full-service restaurant consumption and daily energy and nutrient intakes in US adults. Available from: <https://doi.org/10.1038/ejcn.2015.104>

The American Cancer Society, 1996, Advisory Committee on Diet, Nutrition and Cancer Prevention. Available from: <https://onlinelibrary.wiley.com/doi/full/10.3322/canjclin.46.6.325>

Kant AK, Graubard BI (2018) A prospective study of frequency of eating restaurant prepared meals and subsequent 9-year risk of all-cause and cardiometabolic mortality in US adults. PLOS ONE 13(1): e0191584. Available from: <https://doi.org/10.1371/journal.pone.0191584>

Maalouf J, Cogswell ME, Gunn JP, et al. Monitoring the Sodium Content of Restaurant Foods: Public Health Challenges and Opportunities. American Journal of Public Health. 2013;103(9):e21-e30. doi:10.2105/AJPH.2013.301442.

Cohen DA, Bhatia R. Nutrition Standards for Away-from-home Foods in the United States. Obesity Reviews. 2012;13(7):618-629. doi:10.1111/j.1467-789X.2012.00983.x. Available from: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3380140/#idm140609406547456aff-info>

Rajagopal Raghunathan, Rebecca Walker Naylor, and Wayne D. Hoyer (2006) The Unhealthy = Tasty Intuition and Its Effects on Taste Inferences, Enjoyment, and Choice of Food Products. Journal of Marketing: October 2006, Vol. 70, No. 4, pp. 170-184.

Institute of Medicine (US) Committee on Dietary Guidelines Implementation; Thomas PR, editor. Improving America's Diet and Health: From Recommendations to Action. Washington (DC): National Academies Press (US); 1991. 4, Interpretation and Application of the Recommendations in the Diet and Health Report. Available from: <https://www.ncbi.nlm.nih.gov/books/NBK235262/>

Reference Values derived from <https://www.dge.de/wissenschaft/referenzwerte/>

Appendix

Link to our GitHub repository: <https://github.com/shayansiddiqui/manual>

We appended a video to better illustrate our functionalities. Here one can see the menu we used in that video:



Example statistical values for Cheese cake:

Here one can see all 38 nutritional values of cheese cake and a brief explanation to resulted score.

Value after Vitamins and Minerals represents percentage of daily-recommended intake.

LowRes Photo: https://d2xdmhkmkbyw75.cloudfront.net/107_thumb.jpg

HighRes Photo: https://d2xdmhkmkbyw75.cloudfront.net/107_highres.jpg

[0] Lunch: Cheese Cake

[1] Total Score 23.0 Based on all sub-scores below with their weighting [2-7]

[2] Macro Score 73.0 // Low due to bad proportion [8-10]

[3] Sugar Score 7.0 // Low due to high amounts of sugar [11]

[4] Fiber Score 65.0 Low due to low amounts of Fiber

[5] Fat Score 0.0 //Low due to high amounts of unhealthy fats and low amounts of healthy [12-14]

[6] Vitamin Bonus 5.0 // One bonus due to large amount of Vitamin A (25%) [16]

[7] Mineral Bonus -5.0 //One punishment due to high amounts of sodium (36%) [29]

[8] Proteins: 6.0

[9] Fat: 28.0

[10] Carbs: 31.0

[11] Sugar : 27.0

[12] Saturated Fats (Unhealthy) 12.4

[13] Mono-unsaturated Fats (Healthy) 10.7925

[14] Poly-unsaturated Fats (Very Healthy) 2.0025

[16] Vitamin A 0.2532407407407407

[17] Vitamin D 0.375

[18] Vitamin E 0.05384615384615384

[19] Vitamin K 0.08461538461538462

[20] Vitamin B1 0.03181818181818182

[21] Vitamin B2 0.1856153846153846

[22] Niacin (B3) 0.18753846153846152

[23] Vitamin B6 0.05

[24] Folate (B9) 0.075

[25] Pantothenic Acid (B5) 0.11896666666666667

[27] Vitamin B12 0.07083333333333333

[28] Vitamin C 0.005

[29] Sodium 0.365

[31] Chloride 0.028125

[32] Calcium 0.06375

[33] Phosphorus 0.16607142857142856

[34] Magnesium 0.04230769230769231

[35] Iron 0.06057692307692308

[37] Zinc 0.0796875

[38] Selenium 0.1

The dish receives the following colour: red

Acknowledgements

Foremost, we would like to thank our supervisors for giving us the possibility and inspiration to work on this interesting topic. We particularly thank for their feedback and guidance during the team evaluation that allowed us to further increase the user experience of our app. Finally, we would like to thank our previous team members Sophia Christ and Patrick Lerch for their contribution to the concept of our app.